EUROGRAPHICS 2018 / D. Gutierrez and A. Sheffer
(Guest Editors)

Volume 37 (2018), Number 2

Repairing Inconsistent Curve Networks on Non-parallel Cross-sections

Z. Y. Huang[1] and M. Holloway[1] and N. Carr[†2] and T. Ju[‡1]

[1]Washington University in St. Louis, USA
[2]Adobe Inc., USA

Abstract

In this work we present the first algorithm for restoring consistency between curve networks on non-parallel cross-sections. Our method addresses a critical but overlooked challenge in the reconstruction process from cross-sections that stems from the fact that cross-sectional slices are often generated independently of one another, such as in interactive volume segmentation. As a result, the curve networks on two non-parallel slices may disagree where the slices intersect, which makes these cross-sections an invalid input for surfacing. We propose a method that takes as input an arbitrary number of non-parallel slices, each partitioned into two or more labels by a curve network, and outputs a modified set of curve networks on these slices that are guaranteed to be consistent. We formulate the task of restoring consistency while preserving the shape of input curves as a constrained optimization problem, and we propose an effective solution framework. We demonstrate our method on a data-set of complex multi-labeled input cross-sections. Our technique efficiently produces consistent curve networks even in the presence of large errors.

CCS Concepts

•***Computing methodologies*** → ***Mesh models; Volumetric models;***

1. Introduction

1.1. Motivation

Surface reconstruction from cross-sectional curves has been extensively studied in geometric processing for the past few decades. One of the primary applications of this problem is in interactive image segmentation, particularly for 3D volumes that arise from biomedical imaging (e.g., MRI or CT scans). In a typical session, an expert user would delineate boundaries of a region of interest (e.g., an organ) on selected 2D slices of the 3D volume, and the computer would reconstruct a surface that interpolates those cross-sectional curves. Even with the advances in automatic segmentation methods, interactive segmentation remains a standard practice since it is difficult for existing automated methods to perform accurately and consistently on real-world imaging data.

The majority of reconstruction algorithms are specialized for parallel cross-sections, in part due to the natural choice of axial planes of 3D volumes for boundary delineation. A key limitation of using parallel slices, however, is the restriction of the slice orientation. Allowing a user to choose planes whose orientations are adapted to the 3D shape has the potential to lower the number of planes needed to segment the shape, and thereby reducing human delineation time. This hypothesis has motivated the development of several reconstruction algorithms that are capable of handling arbitrarily oriented cross-sections [BM07, LBD*08, BV09, BVG11, HKHP11, ZHCJ15, HGJ16, HZCJ17]. Some of these algorithms are further capable of reconstructing a surface network that partitions the space into multiple labels (e.g., air, bone, muscle, etc.) given multi-labelled slices delineated by curve networks [LBD*08, BV09, BVG11, HZCJ17].

Despite the availability of these algorithms for surfacing non-parallel slices, a critical but overlooked challenge in deploying these algorithms in practice is making sure that the input slices to these algorithms are *consistent*. A set of slices is said to be consistent if, for any two slices that intersect at a line l, the labelling induced by the curve networks on each slice agrees on l (see Figure 1). The majority of surfacing algorithms *require* a consistent input. Unfortunately, many applications that produce slices for surfacing do not guarantee consistency among the slices. For example, in most off-the-shelf software for interactive 3D volume segmentation, an expert delineates boundaries on each slice independently, and there is no mechanism in the software to enforce consistency among what the expert draws on intersecting slices.

For inputs consisting of only few slices with simple shapes (such as the one in Figure 1), manual corrections may be possible to re-

† ncarr@adobe.com
‡ taoju@wustl.edu

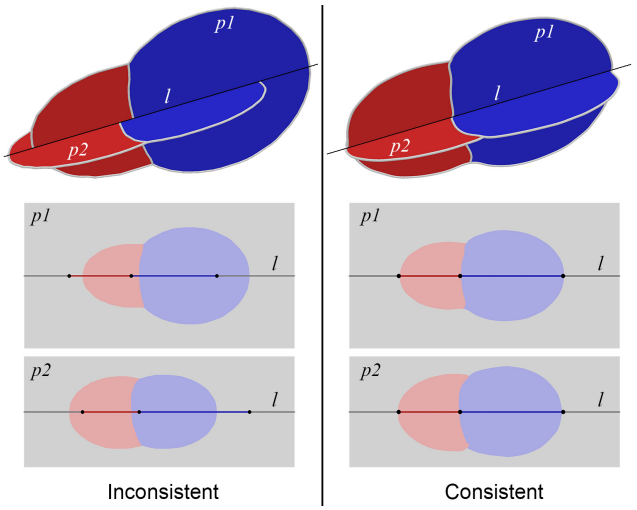

Figure 1: *Curve networks on two intersecting planes (p1,p2) with inconsistent (left) and consistent (right) labeling. The pictures at the bottom show the labeling on each plane as well as the labeling from the other plane on the intersection line (l).*

store the consistency. However, the task can become intractable even for modest size data sets (see Figures 6, 7, 8, 9, 10). We therefore propose an algorithm to automatically restore consistency. Our algorithm fills the gap between real-world data (which are often inconsistent) and existing surfacing algorithms (which typically require consistency), hence allowing these algorithms to be more easily adopted in practice.

1.2. Problem statement

Restoring label consistency can be stated as a constrained curve deformation problem. We are given a set of planes, each divided by a curve network into regions equipped with labels $1,\ldots,n$ ($n \geq 2$). We wish to deform the curve network on each plane in a minimal way to guarantee that the labels on any pair of intersecting planes are consistent along their intersection line.

While geometric deformation has been extensively studied in the past (see a brief review in Section 2), our problem is unique in that the constraints are not fully determined in advance: while the labeling of the planes needs to agree at their intersection lines, the finally agreed labels along the intersection lines are not known *a priori*, especially where the labels on the planes differ. To tackle the challenge, we propose a novel consistency-constrained implicit deformation method. Representing the curve networks on each cross-section by implicit functions, we formulate the deformation task as a constrained optimization problem on the function values with a quadratic energy objective. As our optimization formulation involves both real-valued and integer variables, obtaining a solution efficiently can be challenging. To this end, we propose a solution strategy tailored to our specific problem. When tested on real-world inputs, our method runs significantly faster then off-the-shelf solvers while achieving similar energy values.

Contributions We present the first algorithm for restoring consistency to non-parallel cross-sections. Our algorithm would allow surfacing algorithm for non-parallel cross-sections to be able to process a much wider range of inputs that are often generated by practical applications. Technically, we make two main contributions:

1. We formulate restoration of label consistency as a constrained non-linear optimization problem using a multi-labelled implicit representation.

2. We propose a novel solution strategy of this challenging optimization problem. The solution is shown to be efficient and effective on real world data sets.

2. Relate work

We briefly review two bodies of work that are related to ours, namely reconstruction from cross-sections and curve deformations.

Reconstruction from cross-sections Since the 70's, extensive research has been conducted on reconstructing surfaces from cross-sectional curves. Earlier method focus on handling parallel cross-sections that are partitioned by closed curve loops into inside and outside regions [Kep75, FKU77, Boi88, BS96, OPC96, TO99, BGLSS04, BV07]. The key idea in these methods is to divide the space (or more practically, a bounding box) by the cross-sectional planes into "cells" and build surface pieces within each cell. When all planes are parallel, all cells have a uniform and simple shape (a slab) that is bounded by two planes. If the planes are arbitrarily oriented, each cell may be a general convex polytope bounded by an arbitrary number of planes, which makes the surfacing task more challenging.

Over the past decade, research on cross-section-based reconstruction has focused on handling non-parallel inputs. A number of methodologies have emerged including Delaunay meshing [BM07], projecting curves onto a medial structure [LBD*08, BV09], solving implicit functions [BVG11, HKHP11, ZHCJ15, HZCJ17], and template fitting [HGJ16]. Some of these algorithms are capable of handling even more general inputs such as multi-labelled cross-sections [LBD*08, BV09, BVG11, HZCJ17], partial planes [BV09], and unknown regions [BVG11].

With the exception of [BVG11], all of the above methods attempt to exactly interpolate the input cross-sections, and hence they require a consistent input. The method of [BVG11] constructs and contours an implicit function on a cubical lattice, resulting in a surface that approximates the input curves. While the method is able to process inconsistent inputs, it creates notable artifacts (e.g., unnatural pinching) where the cross-sections disagree along their intersections [VMB*15]. As we shall demonstrate later (Figure 6), restoring consistency to the cross-sections can significantly improve the smoothness of surfaces created by [BVG11].

Geometric deformations Deformation of geometry, being 2D curves or 3D surfaces, is a major research topic in computer graphics and related domains. Broadly speaking, deformation methods can be classified into explicit or implicit ones based on how the deformation is represented. *Explicit* methods either deform the geometry itself, by computing displacements of points on the geom-

etry [BS08], or deform the embedding space around the geometry, using some control structure such as lattices [SP86], cages [JSW05], skeletons [MTLT88], and points [SMW06]. On the other hand, *implicit* methods represent the geometry as the level set of an implicit function in 2D (for curves) or 3D (for surfaces) and indirectly deform the curve by modifying the values of the functions [BB97]. Implicit representations are particularly attractive for interactive editing due to the ease in manipulating the functions [BW90, SWG05, SWSJ07]. Other popular ways for manipulating the functions include interpolating between given functions (as in shape metamorphosis [Hug92, COSL98, BW01, TO05]), associating the function with a particle system [DG95], evolving the function with a user-specified speed function (as in the powerful level-sets method [Set99, TO*03]), and interpolating functional or positional constraints using a variational framework [TO02, KHR02, HKHP11, YYW12].

Deformation methods aim at fulfilling well-defined objectives specified by the user or by the application. Examples of such objectives are handles or sketches (as in interactive editing), a target shape (as in shape metamorphosis or registration), or intensity features in an image (as in image segmentation). We are not aware of any deformation method that is designed for consistency objectives like those in our problem. Our method falls into the class of variational implicit methods [TO02, KHR02, HKHP11, YYW12]. Unlike existing methods that formulate deterministic constraints as simple equalities or inequalities, our method uses additional integer variables to encode the uncertainty of labelling along intersection lines, which results in a more challenging optimization problem.

3. Problem formulation

A perhaps tempting strategy for restoring consistency is to directly deform the input curve networks. Such strategy would find some minimum-energy displacement of the curve vertices constrained by the requirement that, for any two slices intersecting at line l, the curve networks on both slices should intersect with l at the same set of locations. However, formulating such constraint is difficult, since we do not know the number or the location of these intersection points on l. Furthermore, the topology of the curve networks remains fixed in this strategy, which reduces the flexibility of the deformation.

We adopt a different strategy that makes it easy to formulate the consistency constraints and also permits topological changes of the curve networks. The curve network on each plane is implicitly represented by functions over the plane, and curve deformation is indirectly achieved by modifying the functions. In this representation, consistency among slices can be formulated by linear inequality constraints along the intersection lines after introducing additional integer variables (i.e., the labels of vertices on the intersection lines). By expressing the amount of curve deformation as an energy term on the functions, we can cast the task of restoring consistency as a constrained mixed-integer optimization problem.

We first discuss the implicit representation of curve networks on each plane in Section 3.1. After introducing our deformation energy using this representation in Section 3.2, we present our optimization formulation in Section 3.3.

3.1. Implicit representation

Curves that partition the plane into two labels can be implicitly represented as the level set of a scalar function. To represent a curve network that partitions the plane into $n > 2$ labels, we consider a commonly used implicit representation that utilizes a vector function [LSSF06, FJW10, YYW12, HZCJ17]. We first briefly review such representation, and then discuss our choice of the initial vector function and how it is discretized on input cross-sections.

Implicit representation To represent an n-labelled domain, we consider a vector function $\vec{f}(\vec{x}) = \{f_1(\vec{x}), \ldots, f_n(\vec{x})\}$ where each f_i is a continuous scalar function and $\vec{x}$ is a point in the domain. The value of $f_i(\vec{x})$ can be intuitively understood as the "strength" of label i at $\vec{x}$. We assign each point $\vec{x}$ the label that has the maximal strength,

$$Label(\vec{x}) = \arg\max_{i=1,\ldots,n} f_i(\vec{x}). \quad (1)$$

We are interested in the boundary between regions of different labels, known as the *interface set* [HZCJ17]. More precisely, $\vec{x}$ is on the interface set if $Label(\vec{x})$ contains more than one label. Interface sets are natural generalizations of level sets, since the interface set for $n = 2$ labels are equivalent to the level set of some scalar function [HZCJ17]. For $n > 2$ labels, an interface set in the 2D domain can be made up of curve segments meeting at non-manifold junctions (where three or more labelled regions meet).

Initial vector function We seek a vector function $\vec{f}$ whose interface set reproduces an input curve network. This is equivalent to asking that the labelling defined by $\vec{f}$ (Equation 1) coincides with the labelling of regions partitioned by the network. Our definition of $\vec{f}$ is based on signed distance functions. Specifically, consider a point $\vec{x}$ lying inside a region with input label i. We set $f_j(\vec{x})$ to be positive only if $j = i$ and negative for all other labels j. The magnitude of $f_j(\vec{x})$ is set as the Euclidean distance from $\vec{x}$ to the nearest curves that bound regions with label j. If label j is absent from the plane, we set $f_j(\vec{x}) = -\infty$. It is easy to see that this definition of $\vec{f}$ ensures that $Label(\vec{x}) = i$. Figure 2 illustrates the vector function over a three-labelled 1D domain.

While one could define the vector function in more sophisticated

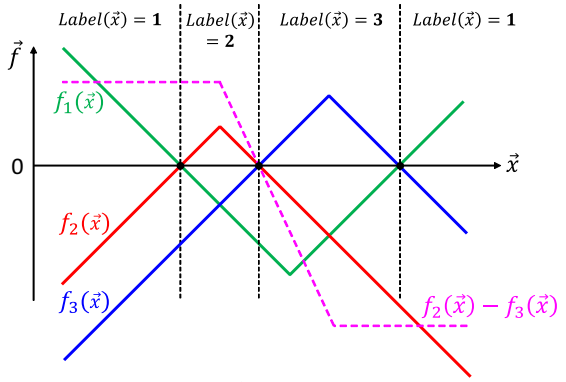

Figure 2: *Vector function $\vec{f} = \{f_1, f_2, f_3\}$ defined as signed distance functions over a three-labelled 1D domain. Note that the difference function $f_2 - f_3$ (magenta dotted graph) is a distance-like function locally at the interface between labels 2 and 3.*

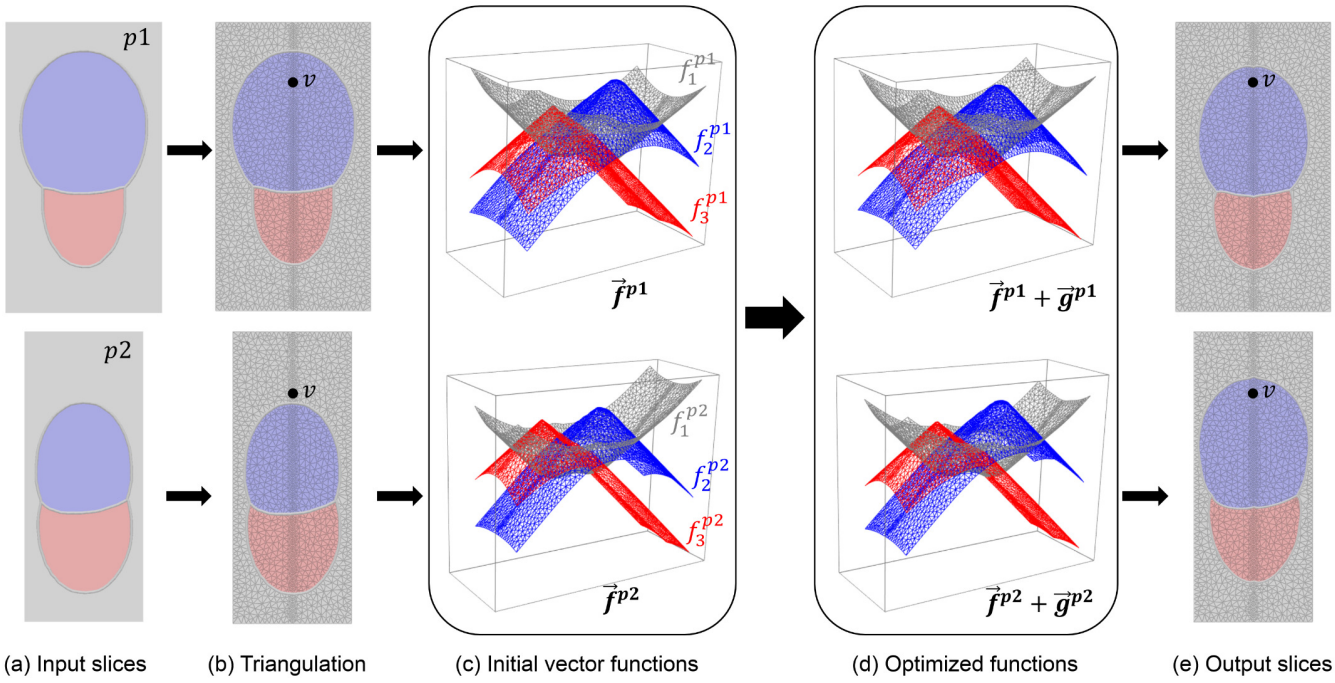

Figure 3: *Overview of our algorithm. Each input slice (a) is first triangulated (b), and a vector function is computed per slice to reproduce the input labels (c). Then functions on all slices are optimized together to enforce label consistency while minimizing deformations (d). Finally, the output curve networks are extracted as the interface sets of the optimized functions (e).*

ways (e.g., using higher degree polynomials [YYW12]), we chose signed distance functions not only due to its simplicity but also because of the convenience it offers for formulating the deformation energy (Section 3.2). A key observation is that a Euclidean distance function has a constant gradient magnitude, hence a small change to the function values leads to a bounded spatial displacement of its level set. Also, adding a constant function to a distance function leads to level sets with similar shapes (i.e., offset curves). As a result, the magnitude and variation of the change to a distance function correlates with the amount of deformation of the level set. This is not true for general smooth functions. Note that the interface set between two labels i, j coincides with the zero-level set of their difference function, $f_i - f_j$. In our signed-distance-based definition of $\vec{f}$ above, $f_i - f_j$ is similar to a signed distance function (with a constant gradient magnitude of 2) in the vicinity of the interface set between labels i and j (see Figure 2). As a result, we can approximately measure the deformation of the interface set between labels i, j by the magnitude and variation in the change to the difference function $f_i - f_j$, as we shall elaborate in Section 3.2.

Discretization We use piecewise linear vector functions encoded by values on vertices of a triangulation. A cross-section plane is discretized by a constrained Delaunay triangulation where the constrained set includes edges and vertices of the curve network as well as the intersection lines with other planes. For consistency between cross-sections, all planes that share one intersection line l use a common set of vertices and edges on that line. This set is created by first uniformly sampling l and then adding new vertices where l intersects with the curve networks or with other intersection lines (e.g., where three or more planes meet at a point). The sampling density along l is chosen to be sufficiently high to capture the inconsistency among the planes. We then use Shewchuk's Triangle package [She96] to compute the triangulation. Figure 3 (b) shows the triangulation of the two slices on the left of Figure 1.

After triangulation, we compute a vector $\vec{f}(v) = \{f_1(v), \ldots, f_n(v)\}$ at each vertex v as defined above. To obtain the region label at each vertex, and assuming that each input curve is equipped with labels on its two sides, we use a flooding process over the triangulation to obtain the labels of vertices that are not on the curve network. To avoid numerical difficulties, for each vertex v on the input curve network that bounds regions with labels $\Phi \subseteq \{1, \ldots, n\}$, we assign v the label with the lowest index in Φ and set the magnitude $\|f_i(v)\|$ for all $i \in \Phi$ to be a small positive constant ε. Figure 3 (c) visualizes, for each slice in (b), the three scalar functions corresponding to the three labels.

Interface set reconstruction Given the initial vector functions, the core of our method (Section 4) is an optimization process that modifies these functions to ensure label consistency between slices. After optimization, we need to extract the interface set of the modified vector function on each slice as the output curve network. To do so, we use a dual scheme akin to that in previous works [FJW10] but over a triangulation. We create two types of points on the output curve network, either on triangle edges (called edge points) or inside triangle faces (called face points). For each triangle edge between two vertices with different labels i, j, we locate the edge point as the zero-crossing of the function $f_i - f_j$ along that edge. For each triangle whose vertices do not have a common

label, we locate the face point as the centroid of the edge points. Finally, to create the network, we connect each face point inside a triangle to the edge points on the triangle's edges. To produce smoother results while conforming to vertex labelling, we fair the curve networks using the non-shrinking centroid-averaging scheme of Taubin [Tau95] while constraining each edge (resp. face) point to lie on the respective triangle edge (resp. face). Figure 3 (e) shows the interface sets from the modified functions in (d).

3.2. Deformation energy

While there are many ways to modify the input to achieve consistency, we are looking for a way that best preserves both the location and shape of the curve networks. As mentioned earlier, our implicit representation makes it possible to capture the amount of curve deformation by the magnitude and variation in the changes to (the difference of) the implicit functions.

Consider an initial vector function $\vec{f}$ in any domain, and let the modified function be $\vec{f}+\vec{g}$ where $\vec{g}$ is another vector function. In the continuous setting, our deformation energy has the integral form

$$E(\vec{g}) = \int \lambda \sum_{i\in R} \|g_i(\vec{x})\|^2 + \sum_{i,j\in R} \|\nabla(g_i(\vec{x}) - g_j(\vec{x}))\|^2 d\vec{x} \qquad (2)$$

where $R \in \{1,\ldots,n\}$ is the set of labels where $f_i \neq -\infty$ for any $i \in R$. The first term inside the integral measures the magnitude of the change to each scalar function, and the second term measures the variation (as Dirichlet energy) of the change to the difference between scalar functions. The two terms are balanced by a user-specified constant λ. Note that penalizing the first term has the indirect effect of penalizing the change to the difference between scalar functions, but this term additionally acts as a regularizer to prevent spurious solutions of $\vec{g}$ that differ by a constant function.

For a piecewise linear vector function $\vec{g}$ over a triangulation of the plane, the integral energy in Equation 2 can be equivalently expressed in the following matrix form

$$E(\mathbf{g}) = \mathbf{g}^T M \mathbf{g} \qquad (3)$$

where $\mathbf{g}$ is a flattened list of values $g_i(v)$ for all vertices v in the triangulation and labels $i \in R$. The matrix M encodes the geometric structure of the triangulation and can be derived from Equation 2 using integrals of barycentric coordinates over triangles. Specifically, let $M_{a,b}$ denote the $|R|\times|R|$ submatrix whose top-left position in M is $\{(a-1)|R|+1,(b-1)|R|+1\}$. The submatrix is non-zero only if the a-th and b-th vertices share a triangle edge, in which case it has the form

$$M_{a,b} = \begin{cases} \lambda \sum_{t\in T_a} \sigma_t I_{|R|} + \sum_{c\in V_a} \omega_{a,c} H_{|R|}, & \text{if } a = b \\ \frac{\lambda}{2} \sum_{t\in T_{a,b}} \sigma_t I_{|R|} - \omega_{a,b} H_{|R|}, & \text{if } a \neq b \end{cases} \qquad (4)$$

where T_a, V_a are the list of triangles and vertices in the 1-ring neighborhood of the a-th vertex, $T_{a,b}$ is the list of triangles containing both the a-th and b-th vertices, σ_t is the area of triangle t, $\omega_{a,b}$ is the cotangent weights [PP93] for the edge between the a-th and b-th vertices, I_m is the identify matrix of size m, and H_m is the Laplacian matrix of a complete graph with m nodes.

3.3. Optimization formulation

We wish to modify the implicit functions on all planes to achieve two goals. First, any vertex v shared by multiple planes should have the same label on those planes. That is, on each of those planes, the corresponding scalar function of that label should be no smaller than the function of any other label at v. Second, the sum of the deformation energy (E in Equation 3) over all planes should be minimized. Hence we have an optimization problem with a quadratic objective function (the second goal) and linear inequality constraints (the first goal). Note that formulating the inequality constraints requires the knowledge of the final label of each vertex on the intersection lines between slices. As a result, we formulate a mixed integer problem that solves for both function values at all vertices (which are real-valued) and labels at those vertices on intersection lines (which are integers).

We now detail the formulation, starting with some notations. Let P be the set of triangulated planes and $\vec{f}^p$ be the initial vector function on each plane $p \in P$. Let I be the set of all vertices on the intersection lines between planes, and denote by P_v the planes on which v lies. Note that we need to store at a vertex $v \in I$ multiple vectors $\vec{f}^p(v)$, one for each plane $p \in P_v$. Since not all labels are present on every plane, we denote by R_p the set of labels present on the plane p and R_v the set of labels present on all planes $p \in P_v$. In the example of Figure 3, the input planes are $P = \{p1, p2\}$, and $P_v = P$ for any vertex v on the intersection line. As both planes contain three labels, we have $R_{p1} = R_{p2} = R_v = \{1,2,3\}$.

We solve for the change in the vector function $\vec{f}^p$ on each plane p, denoted by the vector function $\vec{g}^p$ (see Figure 3 (c,d)), as well as one integer label $L(v) \in R_v$ at each vertex $v \in I$. Rewriting $\vec{g}^p$ as a flat list $\mathbf{g}_p$ that consists of $g_i^p(v)$ for each vertex v on p and each label $i \in R_p$, the optimization objectives are:

$$\text{Minimize:} \quad \sum_{p\in P} \mathbf{g}_p^T M_p \mathbf{g}_p \qquad (5)$$

$$\text{Subject to:} \quad g^p_{L(v)}(v) + f^p_{L(v)}(v) \geq g_i^p(v) + f_i^p(v) + \varepsilon, \quad \forall v \in I, p \in P_v, i \in R_p, i \neq L(v) \qquad (6)$$

This formulation minimizes the sum of deformation energy over all planes (Equation 5) while enforcing consistency of labelling over all vertices on the plane intersections (Equation 6). Here, M_p is the matrix used in Equation 3 over the triangulated plane p.

As the number of vertices on all planes can be large (typically thousands), solving the optimization problem as formulated above can be prohibitively expensive. To reduce the problem size, we make two observations. First, the linear inequalities in (6) only involve function values at vertices on the intersection lines (I). Second, since the deformation energy is quadratic, the minimal energy after fixing the values at a subset of the vertices (e.g., I) can be expressed as a quadratic function of these values. As a result, we can re-formulate the optimization problem to solve for both function values and labels *only* at vertices in I.

Specifically, let $\mathbf{g}_p = \{\mathbf{g}_{p,U}, \mathbf{g}_{p,I}\}$, where $\mathbf{g}_{p,I}$ are values at vertices on the intersection lines between p and other planes, and $\mathbf{g}_{p,U}$ are values at the remaining vertices on p. In this ordering, the ma-

trix M_p has the form

$$M_p = \begin{vmatrix} A & B \\ B^T & C \end{vmatrix} \tag{7}$$

where A is a square matrix of size $|\mathbf{g}_{p,U}| \times |\mathbf{g}_{p,U}|$. Let $N_p = C - B^{-1}A^{-1}B$ (i.e., the Schur complement of the block A of matrix M_p), the minimal energy of (5) can be re-written using only variables on I as

$$\text{Minimize: } \sum_{p \in P} \mathbf{g}_{p,I}^T \, N_p \, \mathbf{g}_{p,I} \tag{8}$$

Note that, unlike the sparse matrix M_p, N_p is dense. However, we have observed in our experiments that the number of vertices on the intersections ($|I|$) are usually 1 to 2 orders of magnitude fewer than the toal number of vertices, and hence using the objective of Equation 8 still yields a significant speed-up over the original form in Equation 5.

To summarize, the reduced formulation solves for values $\vec{g}^p(v)$ and labels $L(v)$ only for vertices $v \in I$, with the objective of (8) and constraints of (6).

4. Optimization

The optimization problem can be understood intuitively as searching for the minimum of a convex energy (Equation 8) over a set of disjoint polytopes in the solution space. To see this, consider a higher dimensional space $\mathcal{G}$ where each point $\mathbf{g} \in \mathcal{G}$ represents the vector of real variables over all planes, that is, $\mathbf{g} = \cup_{p \in P} \mathbf{g}_{p,I}$. Each set of labels $L(v)$ for all vertices $v \in I$ yields a set of linear inequalities in (6), which in turn defines a convex polytope of feasible region in $\mathcal{G}$. The goal is to find a label set L such that the minimal energy within the corresponding polytope in $\mathcal{G}$ is smallest among all polytopes.

A standard trick to solve such problems is to cast it as a mixed-integer program (MIP). To do so, we first replace the integer variable $L(v)$ by an array of binary variables $m_i(v)$, one for each label $i \in R_v$, so that $m_i(v) = 1$ only if $i = L(v)$. Keeping the energy goal as in (8), an equivalent mixed-integer linear program (MILP) can be constructed by replacing the linear equalities in (6) with

$$\begin{aligned} (1 - m_i(v))C + g_i^p(v) + f_i^p(v) \geq g_j^p(v) + f_j^p(v) + \varepsilon, \\ \forall v \in I, p \in P_v, i, j \in R_p, i \neq j, \end{aligned} \tag{9}$$

where C is a large constant, and by adding new constraints including $m_i(v) \geq 0$ for all i and $\sum_{i \in R_v} m_i(v) = 1$. Alternatively, we can construct a mixed-integer non-linear program (MINLP) by replacing (9) with

$$\begin{aligned} m_i(v)(g_i^p(v) + f_i^p(v) - g_j^p(v) - f_j^p(v) - \varepsilon) \geq 0, \\ \forall v \in I, p \in P_v, i, j \in R_p, i \neq j, \end{aligned} \tag{10}$$

However, these MIP formulations have their own drawbacks. The large constant C used in the MILP formulation may lead to weak linear programming relaxation and numerical issues, whereas the constraints in the MINLP formulation are non-convex. These limitations result in low efficiency in the solution process. When testing on our data sets, where the number of variables can be on the order of hundreds, we found that state-of-the-art MIP solvers (e.g., Gurobi) fail to return a solution even after running for hours.

We propose an efficient method for solving our optimization problem without converting it to MIP. The key observation is that, given a label set L, minimizing the energy within the polytope of L is a quadratic programming (QP) problem, which can be solved much more efficiently than MIP. Using the QP as a building block, we follow a greedy strategy to search for the optimal label set. It starts with a initial labeling obtained from the signed distance functions along the intersection lines. It then incrementally changes the labelling, one vertex at a time, to decrease the QP energy. These two stages are detailed next.

4.1. Initial labels

A straight-forward scheme to initialize the label of a vertex v on an intersection line is to average the signed distance vectors over all planes containing v and take the label with the maximum value in the averaged vector. Specifically, recall that P_v is the set of planes containing v and R_v is the set of labels present on all these planes, this scheme initializes label $L(v)$ for all $v \in I$ as:

$$L(v) = \arg\max_{i \in R_v} \sum_{p \in P_v} f_i^p(v)/|P_v|. \tag{11}$$

This simple scheme, however, may produce "jumps" in the labels along an intersection line. In particular, the labelling along an intersection line l between two planes p_1, p_2 may suddenly change at a vertex v where l meets another intersection line l' between planes p_1, p_3. The jump is caused by the fact that $L(v)$ considers the function values on all three planes p_1, p_2, p_3 while the labels at the remaining vertices of l consider function values on only two planes p_1, p_2.

To create a smoother set of labels, we proceed in two steps. In the first step, we use Equation 11 to determine labels at those vertices that lie on multiple intersection lines. We call these vertices *junctions*, denoted by J. In the second step, we modify the function along each intersection line to match the labels at the junctions while maintaining the smoothness of the original function. The modified functions are then used to obtain the labels of the non-junction vertices using the simple averaging scheme above.

Specifically, for the second step, we represent the change of the original function $\vec{f}^p$ along an intersection line l on a plane p as another vector function $\vec{h}^{p,l}$. Note that there will be one function $\vec{h}^{p,l}$ for each plane p that contains l. We wish to find $\vec{h}^{p,l}$ such that the modified function $\vec{f}^p + \vec{h}^{p,l}$ along l is as similar to $\vec{f}^p$ as possible while matching the label at each junction vertex on l. To measure similarity, we consider the same energy as Equation 2 but over a 1-dimensional line, which penalizes the integral of the squared magnitude and gradient of $\vec{h}^{p,l}$ along the line. Using this energy, we need to solve a quadratic program for each plane p and intersection line l. The variables are $h_i^{p,l}(v)$ for each vertex $v \in l$ (including junction vertices) and each label $i \in R_p$ (labels present on p). The objective and constraints are:

$$\text{Minimize: } \mathbf{h}_{p,l}^T \, M_l \, \mathbf{h}_{p,l} \tag{12}$$

$$\text{Subject to:} \quad h^{p,l}_{L(v)}(v) + f^{p}_{L(v)}(v) \geq h^{p,l}_{i}(v) + f^{p}_{i}(v) + \varepsilon, \quad \forall v \in J \cap l, i \in R_p, i \neq L(v), \tag{13}$$

where $\mathbf{h}_{p,l}$ is a flattened list of $h^{p,l}_i(v)$. Matrix M_l has a similar structure as M in Equation 3. Specifically, the $|R_p| \times |R_p|$ submatrix of M_l whose top-left position is $\{(a-1)|R_p|+1, (b-1)|R_p|+1\}$ is non-zero only if the a-th and b-th vertices share a common edge on l. Let the edge between the a-th and b-th vertices be e, each non-zero submatrix has the same form as Equation 4 except that now T_a is the list of edges sharing the a-th vertex, $T_{a,b} = \{e\}$, σ_t is the length of edge t, and $\omega_{a,b} = 1/\sigma_e$.

After solving for $\vec{h}^{p,l}$ for all planes p containing an intersection line l, and denoting this set of planes by P_l, we obtain the label for all non-junction vertices v on l by averaging functions over these planes:

$$L(v) = \arg\max_{i \in R_v} \sum_{p \in P_l} (f^p_i(v) + h^{p,l}_i(v))/|P_l|. \tag{14}$$

4.2. Updating labels

As mentioned before, fixing the vertex labels L reduces our optimization problem (8,6) to a quadratic program (QP). We consider the minimal energy of this QP as a function of L, denoted by $E(L)$. Starting from an initial label set L_0, we will create a sequence of label sets $L_1, L_2, \ldots$ such that $E(L_{k+1}) < E(L_k)$ for $k \geq 0$.

Our approach is guided by the observation that $E(L_k)$ is always achieved by some point on the boundary of the polytope of L_k. To see this, observe that the convex energy E has a unique local minimum in the solution space $\mathcal{G}$ that corresponds to no change to the initial implicit functions. Assuming the input is inconsistent, the minimal-energy solution does not lie inside any feasible regions. As a result, the minimizer in the polotype of L_k has to lie on one or more facets of the polytope. Intuitively, the polytopes that are "on the other side" of these facets are likely to have even lower energy. We therefore enumerate these polytopes and pick one with the lowest energy as our next label set L_{k+1}.

More specifically, each facet of the polytope of L_k corresponds to an equality in the constraint set (6), or

$$g^p_{L_k(v)}(v) + f^p_{L_k(v)}(v) = g^p_i(v) + f^p_i(v) + \varepsilon$$

for some vertex v, plane p, and label i. The polytope "on the other side" of this facet corresponds to values of $g^p_{L_k(v)}(v), g^p_i(v)$ that make the left-hand side smaller than the right-hand side. With ε being a small constant, the inequality would change the label of v from $L_k(v)$ to i. This leads to the following simple algorithm. First, we identify all vertex-label pairs $\{v^*, i^*\}$ that satisfy the above equality on some plane. For each such pair, we create a new label set L^* such that $L^*(v) = L(v)$ for all $v \neq v^*$ and $L^*(v^*) = i^*$, and then compute $E(L^*)$ by solving QP (8,6). We then choose the next label set L_{k+1} to be the L^* with minimal $E(L^*)$, if such minimal energy is smaller than $E(L_k)$. Otherwise, the process terminates and outputs L_k as the solution.

The optimization process is illustrated in Figure 4 on the simple example from 1. Observe that the vertices whose labels change during the updates are located close to the interface set. In practice, we have observed that the initial labels obtained by our method are usually fairly close to the final labels (see next section). As a result, iterative updates can converge quickly to a locally optimal solution.

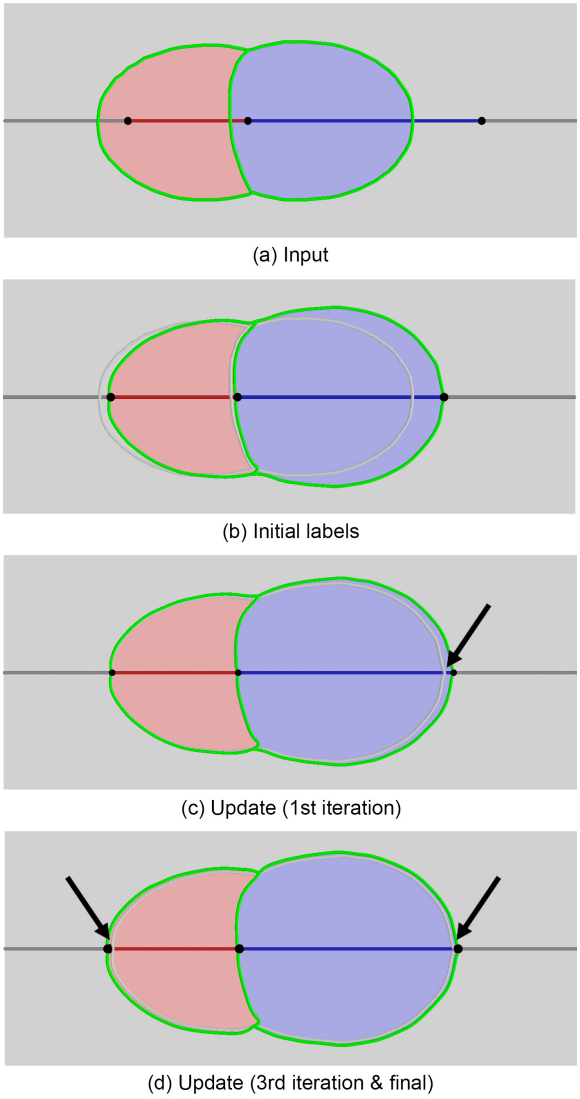

Figure 4: *Optimization process on the input in Figure 1 (left, plane p2), showing the labeling on the plane (as red, blue, gray colors) and interface set (green curves) in the input (a), after initializing the labels on the intersection lines (b) (see Section 4.1), and after the first (c) and final (d) iterations of label updates (see Section 4.2). Interface sets in previous steps are shown in white curves in subsequent steps in (b,c,d) for comparison, and locations where vertices change labels are indicated by arrows.*

5. Experimental results

We test our algorithm on simple synthetic inputs as well as a few non-trivial examples describing anatomical structures. Our imple-

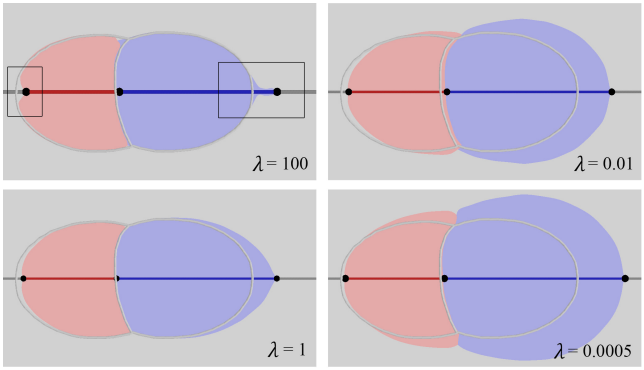

Figure 5: *Results of our method (on input in Figure 1 left, showing plane p2) for different values of λ in Equation 2. The labeling is shown as colored regions, and the input curve network is shown as gray curves for reference.*

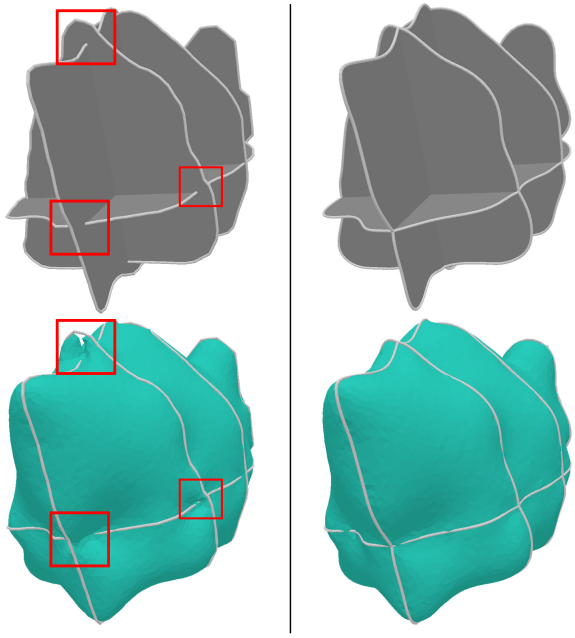

Figure 6: *The result (top-right) of repairing an inconsistent two-labelled Atrium data set (top-left, several inconsistencies are highlighted), and surfaces reconstructed from these two sets of slices using [BVG11] (bottom; observe the artifacts in bottom-left).*

mentation uses Gurobi for solving the quadratic program (8,6) given a fixed label set L.

Choice of parameter We first evaluate the effect of the parameter λ in our deformation energy (Equation 2). Figure 5 shows the results of our method on the same input in Figure 1 (left) for different values of λ. Observe that λ controls the trade off between two competing goals: maintaining the location of the input curves and preserving their shape. If the value is too large (e.g., $\lambda = 100$), most of the input curve network is kept in place but at the cost of severe shape distortion around the intersection lines (highlighted by the boxes) to satisfy consistency. If the value is too small (e.g., $\lambda = 0.0005$), our method will strive to maintain the overall shape of the curves but may produce a significant amount of scaling. Nevertheless, we found that there is a reasonably large range of values of λ (e.g., between 0.001 and 0.1) for which our method produces plausible results for our test examples. These parameters are reported in Table 1.

Complex examples We test our method on several non-trivial biological data sets containing multiple planes (5 or more) that intersect with each other in complex ways (Figures 6, 7, 8, 9, 10). All of the examples exhibit a large number of inconsistencies. In the case of multi-labelled data (Figures 8, 9, 10), observe that the interaction between multiple labels would make it very difficult to rectify manually while simultaneously preserving the shape of the input contour. Our method is capable of restoring consistency on all planes in each data. Note that in some cases the topology of the curve network changes in the output (Figure 8, plane p2, cyan region). The flexibility of allowing topological changes without additional effort is another benefit of using an implicit representation.

The output of our algorithm can be utilized by any existing method for surfacing non-parallel cross-sections. While the surfacing method of [BVG11] can be applied to an inconsistent input, the surface often contains artifacts near inconsistency between the cross-sections. An example is shown in the bottom-left of Figure 6 using the inconsistent input in the top-left (the artifacts are highlighted). Applying the same surfacing method to the consistent cross-sections produced by our algorithm results in an artifact-free surface (Figure 6 bottom-right).

To make the solution process more efficient for these complex examples, we further simplify the problem size by reducing the set of vertices I whose values and labels that we solve for in the optimization formulation (8, 6). We observed that vertices that change labels in the optimization process are either inconsistent to start with (i.e., having different labels on different planes in the input) or close to these inconsistent vertices on the intersection lines. We therefore restrict the set I to inconsistent vertices plus a fraction η of all vertices on the intersection lines ranked in descending order by their distances to inconsistence vertices. We use $\eta = 0.1$ in all three examples. Theoretically, it is possible for some vertices that are on the intersection lines but excluded from I to become inconsistent, since there are no label constraints imposed on these vertices. In this case, one could re-run the optimization again by including the newly inconsistent vertices in I, and repeat the process until no more inconsistency is present. However, we have not had any need to run optimization more than once on our data set.

Performance The core of our algorithm solves a non-linear constrained optimization problem which can also be formulated as a mixed integer program (MIP). Such problems are notoriously challenging to solve even using carefully engineered general solvers. For example, we tried to use Gurobi to solve the MILP formulation as described in Section 4, and it failed to converge even after hours of running on all our complex examples (even after we restrict the set I). Our proposed approach is made efficient by our careful choice of initialization which places the starting guess close

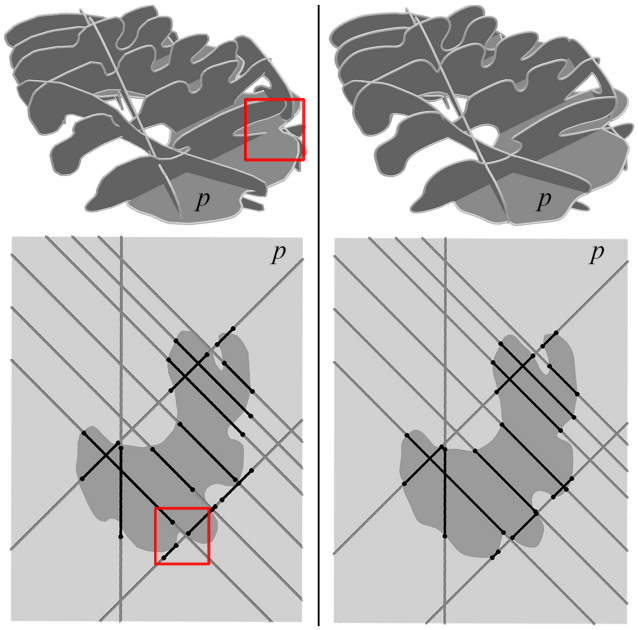

Figure 7: *The result (right) of repairing a two-labelled ferret brain data set that is highly inconsistent (left, one inconsistency is highlighted). The bottom pictures show the labeling on one of the planes (p) as well as labelling from other planes on intersection lines.*

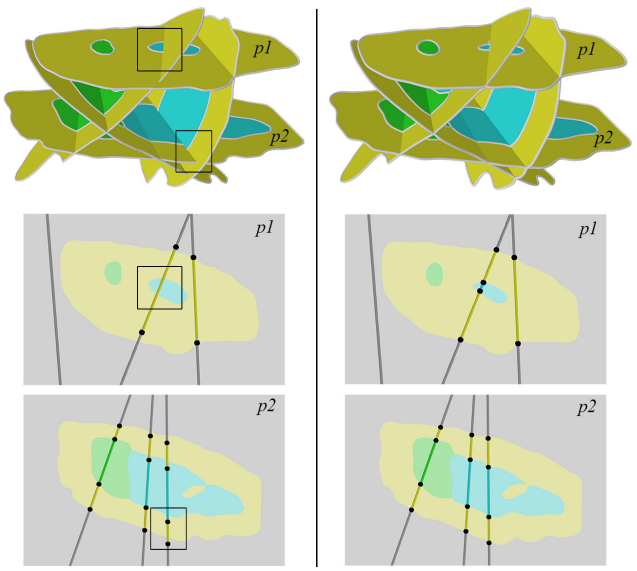

Figure 8: *The result (right) of repairing a 4-labelled liver data set (left, two inconsistencies are highlighted), showing the labeling on two planes (p1,p2) at the bottom.*

	# Planes (Labels)	# Total vertices	$\|I\|$	λ	Pre-proc time	Initial time	Update time
Atrium	5 (2)	5740	109	0.01	0.6398	0.0258	0.448619
Ferret Brain	10 (2)	13131	300	0.01	3.1514	0.703	62.945
Liver (Fig 8)	5 (4)	8222	95	0.1	10.943	0.5324	13.4681
Liver (Fig 9)	6 (4)	20799	125	0.005	60.7131	0.628	29.1373
Mouse Brain	6 (7)	14159	168	0.025	127.661	2.394	291.436

Table 1: *Data size and running time for the examples in Figures 6, 7, 8, 9, 10, showing the number of planes, number of labels, total number of vertices in the triangulations, number of vertices in the reduced intersection set I, λ value, and timing (in seconds) for each of the three stages of our method.*

to the final solution (Section 4.1), and leveraging the efficiency of solving smaller quadratic programs (QP) (Section 4.2).

To better understand the performance of our method, we divide our processing time into three stages: a preprocessing stage that discretizes the planes and prepares data structures for optimization, initialization of labels, and iterative updates of labels. The preprocessing stage is dominated by the matrix inverse operation for obtaining the coefficient matrix N_p in the reduced energy objective (Equation 8). The inversion is done per plane p and the complexity depends on both the total number of vertices on p and the number of labels on that plane. The complexity of both the second and third stages depends on the size of the QP, which scales with the number of vertices and labels in the intersection set I, and the number of times QP is solved. For label initialization, QP is solved only twice per intersection line, one for each plane containing the line, and hence is usually very efficient. For label updates, QP needs to be solved for possibly many times depending on the number of iterations required to terminate and the number of vertices that need to be checked in each iteration.

We report the performance of the stages of our algorithm in Table 1. Our algorithm was implemented in C++ and runs on a laptop with 2.5 GHz Intel Core i7 and 16GB RAM. Overall, our method finishes within minutes for all examples. The dominating stage is updating labels, followed by the preprocessing stage.

Optimality To evaluate the optimality of our solution, we compare our results to those obtained by Gurobi using the MILP formulation. Since Gurobi fails to run on any of our complex examples, we design the following experiment. We take a subset of k planes from the ferret brain data (Figure 7), for $k = 2,\ldots,5$, and run both Gurobi (solving MILP) and our method (Sections 4.1,4.2) for each k. We stopped at $k = 5$, beyond which Gurobi could not return an answer after running for two hours. We report the energy of the solution found by both methods in Table 2, as well as the running time of each method. Observe that our method is able to achieve the same energy as the general MIP solver for all experiments, yet in significantly less time. Also note that our label initialization stage achieves close-to-optimal energy levels, which is an important factor for the fast convergence of our method.

6. Conclusion and discussion

In this paper, we consider the problem of solving label inconsistencies given contour networks on multi-labeled domains consisting of planar slices. We formulated the solution as a constrained optimization problem using an implicit representation where we carefully construct the energy function to preserve the shape of the contour while eliminating inconsistencies. We presented a targeted solver which exceeds the performance of tuned general solvers for this same problem. Our algorithm solves a critical step in the re-

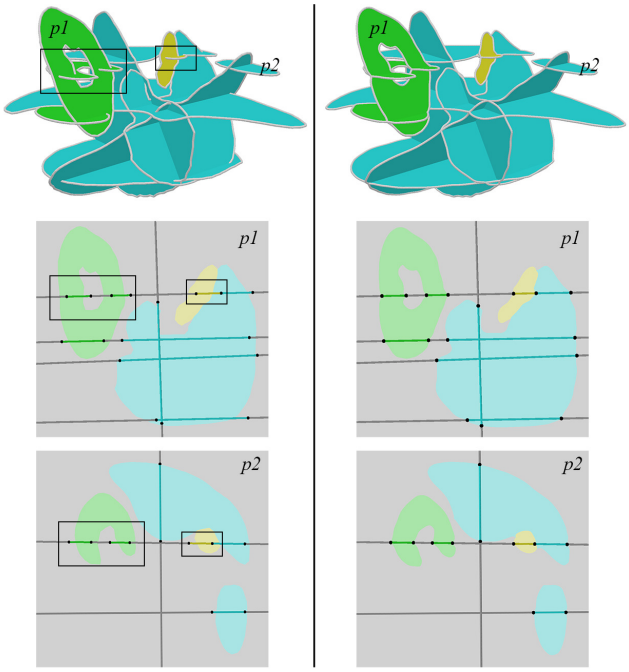

Figure 9: *The result (right) of repairing another 4-labelled liver data set (left, two inconsistencies are highlighted), showing labelling on two planes (p1,p2) at the bottom.*

k	Initialization energy	Final energy	Gurobi energy	Our time	Gurobi time
2	16.97	16.65	16.65	0.274	1.05
3	26.49	24.95	24.95	0.354	11.28
4	26.46	25.02	25.03	0.531	33.16
5	36.14	29.55	29.55	1.471	619.91

Table 2: *Comparing our optimization method and the MIP solver in Gurobi on a subset of k planes in the ferret brain data, in terms of minimal energy and time (in seconds).*

construction pipeline from cross-sections, and it is our hope that this method will pave the way for existing surfacing algorithms to reach a wide spread use in the scientific, medical, and design communities.

Limitations and future work Our work can be improved and extended in several ways. First, the deformation energy used in our optimization formulation captures the distortion to the input curve network in terms of its location and tangents. However, it does not explicitly preserve the smoothness of, or any sharp features on, the input curves. Augmenting the energy with higher-order terms has the potential to more faithfully retain the curve shape. Second, while label consistency is sufficient for reconstructing a continuous surface, reconstructing a *smooth* surface places stronger requirement on the input curve network, such as the differential properties where curves on different planes intersect. The precise consistency condition for smooth reconstruction, and how to enforce them, invite further investigation. Lastly, we would like to explore strategies to further speed up the optimization process, so that it may be used within an interactive volume segmentation program to give immediate feedback to the user as she delineates the boundary curves.

Acknowledgements We thank Noura Faraj and Tamy Boubekeur for providing the liver data sets, and Amit Bermano for the atrium data set and code [BVG11]. This work is supported by National Science Foundation grant IIS-1302200 and a gift from Adobe Inc.

References

[BB97] BLOOMENTHAL J., BAJAJ C.: *Introduction to implicit surfaces*. Morgan Kaufmann, 1997. 3

[BGLSS04] BAREQUET G., GOODRICH M. T., LEVI-STEINER A., STEINER D.: Straight-skeleton based contour interpolation. *Graph. Models 65* (2004), 323–350. 2

[BM07] BOISSONNAT J.-D., MEMARI P.: Shape reconstruction from unorganized cross-sections. In *SGP '07: Proceedings of the fifth Eurographics symposium on Geometry processing* (2007), pp. 89–98. 1, 2

[Boi88] BOISSONNAT J.-D.: Shape reconstruction from planar cross sections. *Comput. Vision Graph. Image Process. 44*, 1 (1988), 1–29. 2

[BS96] BAREQUET G., SHARIR M.: Piecewise-linear interpolation between polygonal slices. *Computer Vision and Image Understanding 63* (1996), 251–272. 2

[BS08] BOTSCH M., SORKINE O.: On linear variational surface deformation methods. *IEEE transactions on visualization and computer graphics 14*, 1 (2008), 213–230. 3

[BV07] BAREQUET G., VAXMAN A.: Nonlinear interpolation between slices. In *SPM '07: Proceedings of the 2007 ACM symposium on Solid and physical modeling* (2007), pp. 97–107. 2

[BV09] BAREQUET G., VAXMAN A.: Reconstruction of multi-label domains from partial planar cross-sections. *Comput. Graph. Forum 28*, 5 (2009), 1327–1337. 1, 2

[BVG11] BERMANO A., VAXMAN A., GOTSMAN C.: Online reconstruction of 3d objects from arbitrary cross-sections. *ACM Trans. Graph. 30*, 5 (Oct. 2011), 113:1–113:11. 1, 2, 8, 10

[BW90] BLOOMENTHAL J., WYVILL B.: Interactive techniques for implicit modeling. *ACM SIGGRAPH Computer Graphics 24*, 2 (1990), 109–116. 3

[BW01] BREEN D. E., WHITAKER R. T.: A level-set approach for the metamorphosis of solid models. *IEEE Transactions on Visualization and Computer Graphics 7*, 2 (2001), 173–192. 3

[COSL98] COHEN-OR D., SOLOMOVIC A., LEVIN D.: Three-dimensional distance field metamorphosis. *ACM Transactions on Graphics (TOG) 17*, 2 (1998), 116–141. 3

[DG95] DESBRUN M., GASCUEL M.-P.: Animating soft substances with implicit surfaces. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 287–290. 3

[FJW10] FENG P., JU T., WARREN J. D.: Piecewise tri-linear contouring for multi-material volumes. In *Advances in Geometric Modeling and Processing, 6th International Conference, GMP 2010, Castro Urdiales, Spain, June 16-18, 2010. Proceedings* (2010), pp. 43–56. 3, 4

[FKU77] FUCHS H., KEDEM Z. M., USELTON S. P.: Optimal surface reconstruction from planar contours. *Commun. ACM 20*, 10 (1977), 693–702. 2

[HGJ16] HOLLOWAY M., GRIMM C., JU T.: Template-based surface reconstruction from cross-sections. *Computers & Graphics 58* (2016), 84–91. 1, 2

[HKHP11] HECKEL F., KONRAD O., HAHN H. K., PEITGEN H.-O.: Interactive 3d medical image segmentation with energy-minimizing implicit functions. *Computers & Graphics 35*, 2 (2011), 275–287. 1, 2, 3

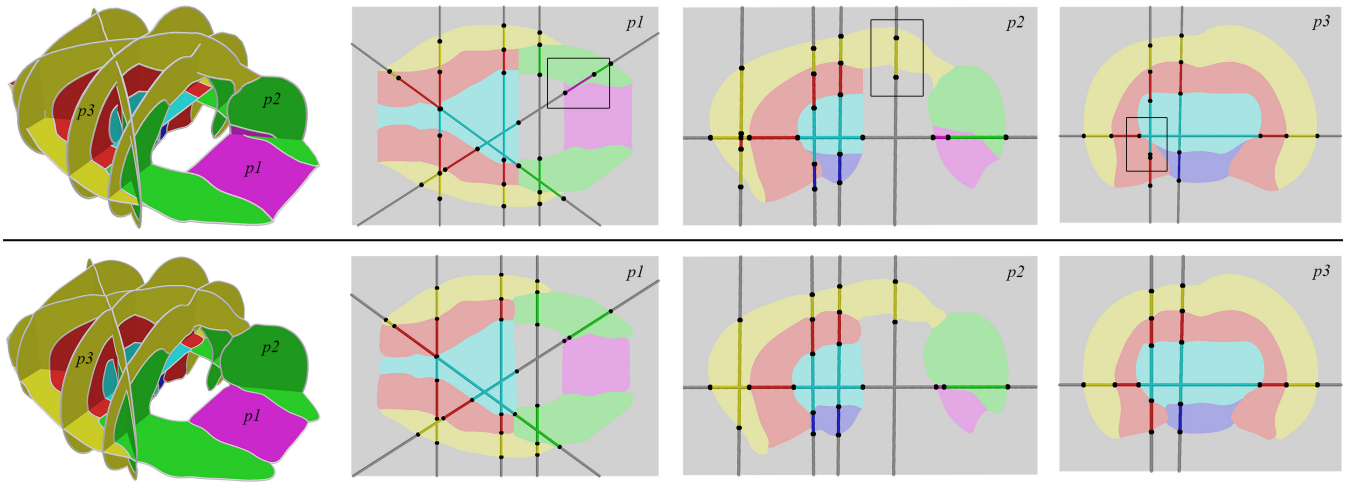

Figure 10: *Result (bottom) of repairing an inconsistent 7-labelled mouse brain set (top), showing the labeling on three planes (p1,p2,p3) and labeling on other planes along the intersection lines. A few inconsistencies are highlighted in black boxes.*

[Hug92] HUGHES J. F.: Scheduled fourier volume morphing. In *ACM SIGGRAPH Computer Graphics* (1992), vol. 26, ACM, pp. 43–46. 3

[HZCJ17] HUANG Z., ZOU M., CARR N., JU T.: Topology-controlled reconstruction of multi-labelled domains from cross-sections. *ACM Transactions on Graphics (TOG) 36*, 4 (2017), 76. 1, 2, 3

[JSW05] JU T., SCHAEFER S., WARREN J.: Mean value coordinates for closed triangular meshes. In *ACM Transactions on Graphics (TOG)* (2005), vol. 24, ACM, pp. 561–566. 3

[Kep75] KEPPEL E.: Approximating complex surfaces by triangulation of contour lines. *IBM Journal of Research and Development 19*, 1 (1975), 2–11. 2

[KHR02] KARPENKO O., HUGHES J. F., RASKAR R.: Free-form sketching with variational implicit surfaces. In *Computer Graphics Forum* (2002), vol. 21, Wiley Online Library, pp. 585–594. 3

[LBD*08] LIU L., BAJAJ C., DEASY J., LOW D. A., JU T.: Surface reconstruction from non-parallel curve networks. *Comput. Graph. Forum 27*, 2 (2008), 155–163. 1, 2

[LSSF06] LOSASSO F., SHINAR T., SELLE A., FEDKIW R.: Multiple interacting liquids. *ACM Trans. Graph. 25*, 3 (July 2006), 812–819. 3

[MTLT88] MAGNENAT-THALMANN N., LAPERRIRE R., THALMANN D.: Joint-dependent local deformations for hand animation and object grasping. In *In Proceedings on Graphics interfaceŠ88* (1988), Citeseer. 3

[OPC96] OLIVA J.-M., PERRIN M., COQUILLART S.: 3d reconstruction of complex polyhedral shapes from contours using a simplified generalized voronoi diagram. *Computer Graphics Forum 15*, 3 (1996), 397–408. 2

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *EXPERIMENTAL MATHEMATICS 2* (1993), 15–36. 5

[Set99] SETHIAN J. A.: *Level set methods and fast marching methods: evolving interfaces in computational geometry, fluid mechanics, computer vision, and materials science*, vol. 3. Cambridge university press, 1999. 3

[She96] SHEWCHUK J. R.: Triangle: Engineering a 2d quality mesh generator and delaunay triangulator. In *Applied computational geometry towards geometric engineering*. Springer, 1996, pp. 203–222. 4

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. In *ACM transactions on graphics (TOG)* (2006), vol. 25, ACM, pp. 533–540. 3

[SP86] SEDERBERG T. W., PARRY S. R.: Free-form deformation of solid geometric models. *ACM SIGGRAPH computer graphics 20*, 4 (1986), 151–160. 3

[SWG05] SCHMIDT R., WYVILL B., GALIN E.: Interactive implicit modeling with hierarchical spatial caching. In *Shape Modeling and Applications, 2005 International Conference* (2005), IEEE, pp. 104–113. 3

[SWSJ07] SCHMIDT R., WYVILL B., SOUSA M. C., JORGE J. A.: Shapeshop: Sketch-based solid modeling with blobtrees. In *ACM SIGGRAPH 2007 courses* (2007), ACM, p. 43. 3

[Tau95] TAUBIN G.: A signal processing approach to fair surface design. In *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques* (1995), ACM, pp. 351–358. 5

[TO99] TURK G., O'BRIEN J. F.: Shape transformation using variational implicit functions. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques* (1999), ACM Press/Addison-Wesley Publishing Co., pp. 335–342. 2

[TO02] TURK G., O'BRIEN J. F.: Modelling with implicit surfaces that interpolate. *ACM Transactions on Graphics (TOG) 21*, 4 (2002), 855–873. 3

[TO*03] TSAI R., OSHER S., ET AL.: Level set methods and their applications in image science. *Communications in Mathematical Sciences 1*, 4 (2003), 1–20. 3

[TO05] TURK G., O'BRIEN J. F.: Shape transformation using variational implicit functions. In *ACM SIGGRAPH 2005 Courses* (2005), ACM, p. 13. 3

[VMB*15] VARDOULIS O., MONNEY P., BERMANO A., VAXMAN A., GOTSMAN C., SCHWITTER J., STUBER M., STERGIOPULOS N., SCHWITTER J.: Single breath-hold 3d measurement of left atrial volume using compressed sensing cardiovascular magnetic resonance and a non-model-based reconstruction approach. *Journal of Cardiovascular Magnetic Resonance 17*, 1 (2015), 47. 2

[YYW12] YUAN Z., YU Y., WANG W.: Object-space multiphase implicit functions. *ACM Trans. Graph. 31*, 4 (July 2012), 114:1–114:10. 3, 4

[ZHCJ15] ZOU M., HOLLOWAY M., CARR N., JU T.: Topology-constrained surface reconstruction from cross-sections. *ACM Trans. Graph. 34*, 4 (2015), 128. 1, 2