Abstract
In this work we explore a new approach to solving the problem of surface reconstruction from cross-section curves. Existing reconstruction methods focus on producing smooth interpolations and may require a large number of cross-sections to recreate feature-rich objects. In the context of medical image segmentation, we make the observation that anatomical structures usually share a common overall shape across subjects. This observation motivates taking a template-based approach that can better capture geometric features, instead of solving for the surface from the cross-sectional curves alone. We deform an existing template, whose shape represents the structure of interest, to fit a set of target curves. We describe our algorithm with the main focus on finding a correspondence between a mesh and a cross-section curve network. We show our results with real medical data and compare to current reconstruction methods.

Keywords: Surface reconstruction, Template deformation, Cross-section contours

1. Introduction
Reconstructing a surface from planar cross-sections is a well studied problem in geometric modeling. The problem arises frequently in the context of medical image segmentation: despite the advances in techniques for automatic segmentation of anatomical structures from 3D volumes (e.g., MRI, CT, ultrasound), and due to the inherent ambiguity and noise in the images, medical practitioners today still routinely and manually delineate 2D contours of the structures on slices of the volume. To reduce the manual effort of contour drawing, an algorithm that reconstructs the surface from these contours should require as few contours as possible in order to create a satisfactory result.

A variety of methods have been developed for surface reconstruction from cross-sections. Most of them treat reconstruction as an interpolation problem, and the shape of the surface is governed by a certain smoothness energy. While these methods can reasonably recreate the shape of smooth objects from a small number of contours, they have inherent difficulty in recreating objects that are rich in geometric features if such features are not completely captured by the contours. This is highlighted in the Ferret brain example in Figure 1: given just a few contours (a), existing methods create interpolating surfaces (b,c) that, although being smooth, fail to capture the “grooves” of the brain surface (e). Also, note that adding additional contours only yield small improvement in the reconstructed features, implying that a significant number of contours would be needed to achieve a satisfactory reconstruction.

Motivation. We would like to develop a reconstruction approach that can faithfully reproduce geometric features without requiring an excessive amount of contours. We focus on the application setting of interactive medical image segmentation, and we make two observations in this context. First, due to human bias and inherent ambiguities in images, the contours only approximate the actual object boundary, and hence exact interpolation is not necessary. Second, and more importantly, despite the variation among subjects, anatomical structures usually share a common overall shape between individuals in a group (e.g., those at the same developmental stage). These two observations motivate a new approach to reconstruction: instead of building surfaces directly from the contours, we deform an existing surface (called the template), whose shape is representative of the anatomical structures in the group, to fit the contours (which we shall call the target contours henceforth).

The key challenge in this template-based approach is establishing the mapping between the template and the target contours. This is a highly ill-posed problem, since the 1-dimensional contour curves carry very little geometric information to constrain its mapping to the surface. To make the problem more tractable, we assume that the template is equipped with a set of pre-defined cross-section planes (called source planes) that are in one-to-one correspondence with the supporting planes of the target contours (called target planes). In addition, the position and orientation of each source plane with respect to the template should be similar to the position and orientation of the corresponding target plane with respect to the to-be-reconstructed object. These source planes therefore serve as a constraint on how the target contours are mapped onto the template surface. An example template with source planes is shown in Figure 1 (e) top.

Our idea of using source planes is grounded in recent
Figure 1: Reconstructing a ferret brain from two sets of cross-section contours (a), containing six (top) and ten (bottom) contours respectively: when compared with existing interpolation-based methods such as [1] (b) and [2] (utilizing the underlying image volume) (c), our template-based method (d) does a much better job at recovering prominent geometric features of the brain (e.g., the grooves) even with a sparse set of contours. Our method utilizes a template equipped with a set of source planes (see (e) top). We show the template for the case of ten source planes and we use a subset of these planes for the six planes case. The ground truth shape is shown in (e) bottom.

studies on guided segmentation of biomedical images [3]. In this interaction paradigm, instead of having the users freely pick planes in the 3D image to draw contours, the system picks a set of pre-defined planes (i.e., source planes) and the users can adjust each plane before drawing contours on them (i.e., target planes). The system also displays, as a reference, the cross-sections of templates at each source plane. It was found that this guided paradigm allows novice users to draw contours as accurately as experts. In this paradigm, the source planes are defined only once on the template, and they are chosen by an expert to capture the major geometric features of the anatomical structure as well as regions whose shape vary significantly among individuals.

Problem and method. The geometric problem addressed in this paper can be stated as follows: given a template, a set of possibly intersecting source planes, and a corresponding set of target planes containing target contours, deform the template so that the neighborhood of the source planes on the template fit the target contours. Note that we cannot ask the cross-sections of the template at the source planes to fit the target contours, since the network topology of the target contours can be different from the cross-sections of the template at the source planes.

We solve the problem using the classical alternative optimization approach that alternates between computing the deformation and finding the mapping. The key step in our method is mapping a possibly complicated network of target contours onto the template surface, so that the mapped network preserves the shape of the target network while staying close to the source planes. This is achieved by a novel algorithm that combines dynamic programming with a greedy combinatorial optimization.

We tested our method with a number of real-world data set involving anatomical structures from humans and animals. The results demonstrated significantly improved geometric features over traditional, interpolation-based approaches for surface reconstruction from curves, even with only a few target contours (e.g., Figure 1 (d)).

Contributions. To the best of our knowledge, we propose the first template-based reconstruction method from spatial curves. We make the following technical contributions:

1. We formulate the problem of template-based reconstruction from cross-section curves in the context of biomedical applications (Section 3). We make the problem tractable by leveraging the availability of source planes and their correspondences with the supporting planes of the cross-section curves.

2. We propose an alternating optimization solution for the above-mentioned problem (Section 4). In particular, we introduce a novel algorithm for mapping a network of spatial curves onto a surface.

2. Related Work

2.1. Surface Reconstruction from Curves

Building surfaces from cross-section contours has been extensively studied in the graphics community since the 70s [4, 5]. We briefly summarize the works in this area, and refer readers to more detailed reviews in recent papers such as [6, 2].

Most methods for cross-section reconstruction fall into three general classes. The first class is based on the Delaunay triangulation of cross-section curves and/or planes [7, 8, 9]. The second class obtains the surface by projecting the curves onto a set of auxiliary sheets (e.g., medial axis or straight skeleton of a cell in the plane arrangement) [10, 11, 12, 13, 14]. The last class defines and extracts
the iso-surface of an implicit function, such as radial basis functions [15, 1], mean-value interpolants [6], and harmonic functions [2]. While earlier methods focus on the simple input setting of a stack of parallel cross-sections containing disjoint contour loops, recent methods can handle non-parallel cross-sections and contour networks (for multiply-labelled domains). More recently, the topology of the output surface can also be controlled [2].

Although capable of generating closed and interpolating surfaces, existing methods all rely on some smoothness models to constrain the shape of the surface that connects the input curves. Reconstruction methods based on implicit functions rely on that function’s underlying smoothness model. For example, radial basis functions [15, 1] minimize the thin-spline energy. Other reconstruction methods typically require a post-process fairing on the initial mesh surface, and the shape of the final surface depend on the fairing technique (e.g., [14, 2] uses surface-diffusion flow [16]). Note that these smoothness models inherently prevent existing methods from creating interesting geometric features that are not well characterized by the curves themselves (e.g., the grooves in the Ferret brain in Figure 1).

2.2. Template-based Reconstruction

Templates are often used for surface reconstruction from incomplete data, particular point clouds (see the survey article [17]). Template deformation is often formulated as a minimization problem where the objective function consists of a fitting term (i.e., how well the template fits the data) and a shape term (i.e., how much distortion is introduced to the template shape). For non-rigid deformations, a variety of ways exist to formulate the shape term, such as using differential coordinates [18, 19], local affine transformations [20], thin-plate spline [21], and statistical shape models [22].

Formulating the fitting terms requires establishing the mapping between the template and the data. Correspondences between two (complete or partial) surfaces have been well-studied (see the survey article [23]). The simplest approach is based on proximity in the Euclidean space, as used in the classical ICP (Iterative Closest Point), but the application is limited to templates in close range of the input data. To deal with large deformations, additional constraints have been adopted to regularize the mapping. Common constraints are based on either surface-based geometric features, such as spin images [24], integral descriptors [25], and SHOT features [26], or the deformation space of the surface, such as isometry [27, 28] and conformality [29, 30]. However, these constraints are designed for input data with a 2-dimensional structure (e.g., point cloud covering a surface area). To the best of our knowledge, no correspondence methods have been developed that map between a template surface and a target spatial curve network.

3. Problem Formulation

We are given a template mesh \( M \) with vertices \( \{v_1, \ldots, v_n\} \) and target contours \( C \) representing the cross-sections of an unknown object \( O \) on target planes \( B = \{b_1, \ldots, b_l\} \). We allow target planes to be non-parallel with each other, and hence \( C \) in general is a curve network composed of planar segments meeting at junctions. We denote the vertices of \( C \) as \( \{p_1, \ldots, p_m\} \). Each vertex is associated with one or multiple planes of \( B \). To further constrain the problem, we assume the knowledge of source planes \( A = \{a_1, \ldots, a_l\} \), such that the position and orientation of each \( a_i \) \((i = 1, \ldots, l)\) with respect to \( M \) is similar to those of the corresponding target plane \( b_i \) with respect to \( O \) (Figure 2 far-left). Our goal is to compute a deformed template \( M' \) with new vertex locations \( \{v'_1, \ldots, v'_n\} \), so that a neighborhood of the source planes on the un-deformed template fits the target contours after deformation. To formulate the problem, we will also look for a mapping \( \phi : \{1, \ldots, m\} \rightarrow \{1, \ldots, n\} \) that maps each target contour vertex \( p_i \) to some template vertex \( v_{\phi(i)} \). Intuitively, we would like deformed vertices \( v'_{\phi(i)} \) to be as close to \( p_i \) as possible.

It is important that we do not limit the image of \( \phi \) to be only vertices of the template that are on the source planes. Since the network topology of \( C \) can be different from the topology of the curve network obtained by intersecting \( M \) with source planes \( A \), a mapping from \( C \) to the latter network may not exist. In the example of Figure 2 far-left, the intersection between \( M \) and \( A \) is a network with 4 junctions and 8 segments, while the target contours \( C \) contains 6 junctions and 12 segments. The other choice that we could have made is computing a separate mapping for each target plane \( b_i \) that maps the target contours on \( b_i \) to the points on the corresponding source plane \( a_i \). However, when the topology of \( C \) differs from that of the intersection of \( M \) and \( A \), these per-plane mappings may disagree at a global level and create highly distorted deformations (see Results section).

Formally, we seek the pair \( \{M', \phi\} \) that minimizes the following 3-part energy:

\[
E(M', \phi) = w_{fit} E_{fit}(M', \phi) + w_{dis} E_{dis}(M') + w_{map} E_{map}(\phi) \tag{1}
\]

The three energy terms are explained as follows:

- The fitting term, \( E_{fit}(M', \phi) \), measures the deviation of the target contour vertices from their mapped locations on the deformed template:

\[
E_{fit}(M', \phi) = \sum_{i=1}^{m} ||p_i - v'_{\phi(i)}||^2 \tag{2}
\]

- The distortion term, \( E_{dis} \), measures the distortion of the template shape due to deformation. The literature has suggested many ways to quantify such distortion. For computational efficiency, we adopt the linear formulation in [19] that approximately measures
non-rigid motions, 
\[ E_{\text{dis}}(M') = \sum_{i=1}^{n} \sum_{j \in N(i)} w_{ij} \| (v'_i - v'_j) - R_i(v_i - v_j) \|^2 \] 
where \( N(i) \) denotes the indices of the 1-ring neighbors of vertex \( v_i \), \( R_i \) is an estimated local rotation matrix that best aligns \( v_i \) and its 1-ring neighborhood with \( v'_i \) and its 1-ring neighborhood, and \( w_{ij} \) is the cotangent weight. Note that replacing this term with alternative choices of distortion measures (e.g., discrete Laplacian [18] or local affine transformations [20]) will not affect the rest of our algorithm.

- The mapping term, \( E_{\text{map}} \), measures the irregularity of the mapping \( \phi \). Intuitively, we favor mappings whose image on the template is a curve network that lies in the neighborhood of the source planes and bears similar shape as the target network \( C \). To measure dissimilarity in curve shapes, we adopt the first-order differences that are commonly used for matching planar curves [31]. In our case, the differences are computed after transforming the target planes to align with the source planes. Our definition of \( E_{\text{map}} \) captures these two goals, deviation from source planes and shape dissimilarity, as follows:

\[
E_{\text{map}}(\phi) = w_{\text{src}} \sum_{i=1}^{m} \sum_{j \in H(i)} d(u_{ij}, v_{\phi(i)}) + \\
\sum_{\{i,j\} \in E(C)} \| [T_{H(i,j)}(p_i - p_j) - (v_{\phi(i)} - v_{\phi(j)})] \|^2 
\]

Here, \( w_{\text{src}} \) is a balancing weight, \( d(a, v) \) measures the shortest geodesic distance from point \( v \) to any point on the cross-section of the template at plane \( a \), \( H(i) \) gives the list of indices of the target planes that contain the contour vertex \( p_i \), \( H(i,j) \) is the unique index of the target plane that contains two edge-adjacent contour vertices \( p_i, p_j \) (we assume the generic situation that no edge of the target contours lies on the intersection of two target planes), \( E(C) \) is the list of polygon edges on \( C \), and \( T_i \) is the transformation that aligns target plane \( b_i \) with the source plane \( a_i \). We obtain \( T_i \) by first translating the centroid of those contour vertices on \( b_i \) to the centroid of the intersection of \( M \) and \( a_i \) and then rotating \( b_i \) around the cross-product of the normal vectors of \( a_i \) and \( b_i \).

The scalars \( w_{\text{fit}}, w_{\text{dis}}, w_{\text{map}} \) are balancing weights of the three energy terms. We use the setting \( w_{\text{dis}} = 10, w_{\text{map}} = 1, w_{\text{src}} = 0.25 \) for all our experiments. We shall justify the choice of these values with examples in Section 5.

4. Optimization

4.1. Overview

Minimizing the energy \( E(M', \phi) \) (Equation 1) is a mixed continuous-discrete optimization problem. We perform alternative optimization that breaks the problem down into a continuous optimization and a discrete optimization (see Figure 2). Note that this is a common strategy for constrained geometry optimization [32].

After an initial deformation, we alternate between finding the mapping \( \phi \) while keeping \( M' \) fixed (i.e., the discrete problem) and computing the deformation \( M' \) while keeping \( \phi \) fixed (i.e., the continuous problem). The continuous problem can be readily solved using existing mesh deformation techniques. Since we borrowed the distortion term \( E_{\text{dis}}(M') \) from [19], we use their As-Rigid-As-Possible technique to solve the continuous problem. Our key contribution is solving the discrete mapping problem, which will be discussed in the next section.

To initialize the iterative process, we first compute an initial, gross mapping \( \phi \), which then gives rise to the initial deformed mesh \( M' \) by solving the continuous problem given \( \phi \). To get the mapping, we utilize the transformation \( T_i \) that aligns a target plane \( b_i \) with its corresponding source plane \( a_i \), and take the vertex on the template mesh that is closest to the transformed location of a target contour vertex. If the contour vertex lies on multiple target

---

**Figure 2:** Work flow of our algorithm. The input includes a template mesh \( M \), equipped with source planes \( A \), and target contours \( C \) lying on target planes \( B \). Note that \( C \) can have a different network topology than the intersection of \( M \) with \( A \). After computing an initial deformation \( M' \), we repeatedly alternate between solving for the mapping \( \phi \) while fixing \( M' \) and updating the deformation \( M' \) given \( \phi \) for a fixed number of iterations.
planes (i.e., it is a junction of the contour network $C$), then we will consider the average location after applying the transformation on each of those planes. Specifically, for each target contour vertex $p_i$, we compute

$$\overline{p_i} = \frac{\sum_{j \in H(i)} T_j p_i}{|H(i)|},$$

where $H(i)$, as explained earlier, is the list of target planes that contain $p_i$, and transformations $T_j$ are computed as discussed in Section 3. Then we set $\phi(i)$ as the index of the vertex on $M$ closest to $\overline{p_i}$.

The two steps, solving the continuous and discrete problems, are iterated until either a fixed number of iterations is reached or the change in the template shape is smaller than a threshold. We use the former strategy with typically 6 iterations. Since the quality of mapping improves over time, we initially set $w_{fsl} = 1.25$ and increment it by 2 in each iteration.

4.2. Curve Network Mapping

By fixing the deformation $M'$, the non-constant portion of the energy of Equation 1 be can rewritten as follows:

$$E(\phi) = \sum_{i=1}^{m} \alpha(i, \phi(i)) + \sum_{(i,j) \in E_C} \beta(i, j, \phi(i), \phi(j))$$

(5)

where $\alpha(i, i^*)$ is the cost of mapping a single contour vertex $p_i$ to a template vertex $v_{i^*}$,

$$\alpha(i, i^*) = w_{fsl} \|p_i - v_{i^*}\|^2 + w_{map} w_{src} \sum_{j \in H(i)} d(a_j, v_{i^*})$$

and $\beta(i, j, i^*, j^*)$ is the cost of mapping two edge-adjacent contour vertices $\{p_i, p_j\}$ to a pair of template vertices $\{v_{i^*}, v_{j^*}\}$,

$$\beta(i, j, i^*, j^*) = w_{map} \|T_{H(i,j)}(p_i - p_j) - (v_{i^*} - v_{j^*})\|^2$$

Minimizing $E(\phi)$ becomes a quadratic assignment problem, which is known to be NP-hard. While approximate solutions can be computed efficiently, for example using spectral techniques [33], the results are often far from being optimal.

Our key observation is that, since the domain of the mapping, $C$, has a 1-dimensional structure, the problem is not as difficult as the general assignment problem. In fact, the optimal mapping can be found in polynomial time for connected components of $C$ that are free of junctions (i.e., disjoint loops). Building on top of this optimal algorithm, we devise a greedy combinatorial search algorithm to find an approximate solution for the rest of $C$ that contains junctions (i.e., networks).

To reduce the computational cost, we limit the possible choices of $\phi(i)$ to a list of candidate template vertices $\Phi_i \subset \{1, \ldots, n\}$ for each contour vertex $i$. That is, we want to find the mapping $\phi$ that minimizes $E(\phi)$ under the restriction that $\phi(i) \in \Phi_i$ for all $i = 1, \ldots, m$. In our implementation, we create the list $\Phi_i$ by computing the cost $\alpha(i, j)$ for each vertex $v_j$ of $M$ and taking the $k$ vertices with the least cost. In all of our experiments, we find choosing $k = 50$ gives satisfactory results. Figure 3 (a) illustrates the candidate list for two junction vertices (green) and vertices on a contour curve segment (red). Figure 3 (b) shows the optimal mapping for two junctions on the target contours (green dots) and interior vertices on a contour curve segment (red dots). (c): Three iterations in the greedy algorithm, showing the mapped curve network in each iteration (as connected dots on the surface) and highlighting (in yellow circle) the mapped junction point that is adjusted in the iteration.

4.2.1. Mapping Loops

To motivate the algorithm for mapping closed loops, we first consider the (rather impractical) case where $C$ is a single, open curve segment connecting consecutive vertices in the list $\{p_1, \ldots, p_m\}$. The energy-minimizing mapping $\phi$ in this case can be formulated as the shortest path in a weighted graph constructed as follows. We create nodes $n_{i,s}$ for all $i = 1, \ldots, m$ and $s = 1, \ldots, k$, each representing a possible mapping $\phi(i) = \Phi_i(s)$, the latter being the $s$-th candidate in the list $\Phi_i$. Each node is given a weight $\alpha(i, \Phi_i(s))$. It is helpful to imagine the nodes as points on a grid, so that $n_{i,s}$ lies on column $i$ and row $s$. We then connect an arc between every pair of nodes on two consecutive columns, or $n_{i,s}, n_{i+1,t}$ for any $s, t$ and $i = 1, \ldots, m - 1$. The weight of this arc is $\beta(i, i + 1, \Phi_i(s), \Phi_{i+1}(t))$. Define the length of a path in this graph as the sum of node weights and arc weights along the path. The optimal mapping $\phi$ is thus encoded by the nodes along the shortest path from any node in the first column, $n_{1,s}$, to any node in the last column, $n_{m,t}$, for any $s, t$. Using dynamic programming, this path can be found in $O(k^2m)$ time.

The algorithm can be easily extended, with the same time complexity, to the case when $C$ is a single closed
loop. Assume that the order of vertices on the loop is still \( \{p_1, \ldots, p_{n+1}\} \) such that \( p_{n+1} = p_1 \). We construct the graph as above. To ensure that we obtain a unique mapping for \( p_1 \), we seek the shortest path between any two nodes in the first and last column that lie on the same row, that is, between \( n_1, s \) and \( n_{m+1}, s \) for any \( s = 1, \ldots, k \).

In general, \( C \) can be made up of several connected components, and each component is either a single loop or a network. The algorithm above can be applied to finding the optimal \( \phi \) over any component of \( C \) that is a closed loop.

4.2.2. Mapping Networks

To find the mapping for a network \( C \) (or a network component of \( C \)), it is not enough to find a mapping for each individual curve segment (e.g., using the algorithm described above), since the mapping on different segments meeting at a junction may disagree for that junction point. On the other hand, if we fix the choice of mapping at each junction point of \( C \), the optimal mapping for the rest of \( C \) can be found by applying the algorithm above to each curve segment. The only adaptation needed is that the shortest path should be found between two fixed nodes in the first and last columns of the graph that correspond to the fixed mapping of the two junction points.

This observation motivates the following greedy strategy. We first make a best guess at the mapping \( \phi(i) \) for each junction \( p_i \) and compute the optimal mapping for the remaining vertices of \( C \). Then we iteratively alternate between adjusting the mapping of one junction and updating the optimal mapping for curve segments incident to the junction. To choose that junction, we pick the one whose adjusted mapping would result in the greatest drop in total energy \( E(\phi) \) after updating the mapping of its incident segments. In essence, we are taking a steepest-descent route in the space of mappings of the junctions.

To implement the strategy efficiently, we need to be able to quickly evaluate the energy of the optimal mapping of a curve segment, bounded by junctions \( p_i, p_j \), under different choices of \( \phi(i) \) and \( \phi(j) \). We do so by pre-computing, for all pairs of junction mappings \( \{\phi(i) = i^*, \phi(j) = j^*\} \) where \( i^* \in \Phi_i \) and \( j^* \in \Phi_j \), the energy of optimally mapping the rest of the vertices on the curve segment to the template. This can be done in \( O(k^3m) \) time using a similar dynamic-programming pass as described above. The resulting pairwise energy values are stored in a matrix \( D_{i,j} \), such that \( D_{i,j}(i^*, j^*) \) gives the minimal mapping energy of the curve segment when \( \{\phi(i) = i^*, \phi(j) = j^*\} \). As an example, Figure 3 (b) shows two optimal mapping of the same curve segment (red) for two pairs of junction mappings.

The complete algorithm proceeds as follows:

1. Initialize: Set \( \phi(i) = \Phi_i(1) \) for each junction \( p_i \). For each junction \( p_i \) and every candidate mapping \( i^* \in \Phi_i \), let \( \gamma(i^*) = \sum_{j \in N(i)} D_{i,j}(i^*, \phi(j)) \) be the total mapping energy of all incident curve segments to \( p_i \) when \( \phi(i) = i^* \) (here \( N(i) \) is the list of indices of junctions that share a common curve segment with \( p_i \)). Let \( \psi(i) = \arg \min_{\phi(i)} \gamma(i^*) \) and \( \delta(i) = \gamma_i(\phi(i)) - \gamma_i(i^*) \).

   Intuitively, setting \( \phi(i) = \psi(i) \) would result in the lowest total mapping energy for all incident curve segments (while fixing the mapping at all other junctions), and \( \delta(i) \) is the decrease in energy resulted from changing the current choice of mapping \( \phi(i) \) to \( \psi(i) \).

2. Repeat: Pick the junction \( p_i \) with the greatest \( \delta(i) \) among all junction vertices. If \( \delta(i) < 0 \), the algorithm terminates, since no more adjustment can be made to junction mappings that would lower the energy (i.e., a local minimum is reached). Otherwise, set \( \phi(i) = \psi(i) \), \( \delta(i) = 0 \), and update \( \psi(j), \delta(j) \) for all neighboring junctions \( j \in N(i) \).

Note that the algorithm always terminates, because the energy \( E(\phi) \) decreases monotonically in each iteration and the space of all mappings is finite. The number of iterations is proportional to the number of junctions and also depends on how far \( M' \) is from \( C \). In our experiments, we found that the algorithm typically iterates for about half the number of junctions or less. A few iterations of the algorithm are illustrated using the ellipsoid example in Figure 3 (c).

Figure 4: Deformation results for the ellipsoid example (top, same as in Figure 2) under different parameter settings. Each row varies a single parameter (bold text), and the remaining parameters assume the default values (\( k = 50, w_{\text{dis}} = 10, w_{\text{src}} = 0.25 \)).

5. Results

5.1. Parameters

Our method utilizes several parameters, some offering balance among various terms in our definition of energy \( E \)
in Equation 1 (i.e., $w_{fit}, w_{dis}, w_{map}, w_{src}$) and others providing trade-off between speed and optimality during curve network mapping (i.e., the number $k$ of nearest neighbors). We use the following set of parameter values for all our real-world examples: $w_{dis} = 10$, $w_{map} = 1$, $w_{src} = 0.25$, $k = 50$, and $w_{fit} = 1.25$ initially and incremented by 2 in each iteration.

We demonstrate the results under different values of several key parameters on the synthetic ellipsoid example in Figure 4. The observations for each parameter are summarized as follows:

$w_{dis}$ (second row): This parameter balances the two goals of fitting and shape preservation in deformation. If the parameter value is too small, large distortion of the shape would occur (as seen in the insert for $w_{dis} = 1$). On the other hand, setting the parameter too high (e.g., $w_{dis} = 20$) would prevent fitting to the target contours (see the insert). We found that $w_{dis} = 10$ offers a good trade off between the two goals.

$w_{src}$ (third row): This parameter controls the strength of the source plane constraint in the mapping term of the energy (Equation 4). The parameter is particularly important when the source planes and target planes are aligned respectively with geometric features of the template and the unknown object. For example, one of the source-target plane pairs in the ellipsoid example are aligned with the outer ridge of the ellipsoid (red curves in Figure 4 (top)). Insufficient constraint (e.g., $w_{src} = 0.05$) would fail to deform the ridge of the ellipsoid to the corresponding target contour. On the other hand, too much constraint (e.g., $w_{src} = 0.65$) may create shape distortions (see the lower-right part) when the configuration of the target planes differ significantly with that of the source planes.

$k$ (last row): A small $k$ would allow faster computation of mapping, but it offers only limited number of candidates that could be insufficient for mapping (e.g., see result of $k = 5$). We found that our choice of $k = 50$ achieves similar results as larger values (e.g., $k = 100$).

5.2. Real-world Examples

We tested our method on two medical data sets: a ferret brain and a liver. To construct the template in each test, we asked experts to pick a subject that is considered “standard” and delineate 2D contours on a dense set of planes. The template surface was then created from these dense contours using the method of [1]. The same experts also chose the position and orientation of a sparse set of source planes. Given a test subject, a novice was asked to pick a set of target planes in correspondence with the source planes (we use the interface in [3]). In addition, we followed the same approach for creating the templates to create a ground truth mesh for each test subject. This ground truth mesh serves two purposes. First, to avoid potential errors made by novices when drawing contours, our target contours are set to be the cross-sections of the ground truth mesh on planes chosen by novices. Second, the ground truth is used to validate the accuracy of different reconstruction methods.

For each data set, we compared with existing methods and alternative algorithmic choices. The findings are summarized below.

**Ferret brain**: The main results are shown in Figure 1. We can see that our approach, compared to two other representative interpolation-based reconstruction methods [1, 2], achieves results much closer to the ground truth even when using a sparser set of contours.

In Figure 5, we further demonstrate the importance of the mapping term ($E_{map}$) in our energy formulation: not only does it constrain the network mapping to be close to the source planes, it also considers similarity of curve shapes. Enforcing similarity avoids matching from one contiguous target contour to disjoint parts of the template surface. In the result produced without the mapping term (by setting $w_{map} = 0$) during iterative optimization, shown in Fig 5 (b), the gap space between two nearby grooves highlighted in the box is significantly reduced, due to the incorrect correspondence between the groove and the orange contour (which should be under the groove). In contrast, the gap space is preserved with the mapping term.

**Liver**: The main results are shown in Figure 6 from two viewpoints. Even though the liver is not as rich in features as the ferret brain, our result in Figure 6 (d) is still closer to the ground truth than the interpolation-based approach in Figure 6 (b).

As mentioned in Section 3, an alternative to finding a mapping for the entire target contour network is to find...
Figure 6: Two views (top and bottom) of the results using different reconstruction approaches for a liver. We show the input target contours and template (a), the results using [1] (b), the results of mapping each plane independently (c), our deformation result (d), and the ground truth (e). Make note of the different layout of the source planes and target planes in the bottom view and how this affects the result in (c).

Figure 7: The results of our algorithm using two different templates and one set of target contours. We show the source on the left, our result in the middle, and the Hausdorff distance from the ground truth on the right. The small insert image shows the Hausdorff distance of the result using [1] from the ground truth. We can achieve similar results using different templates with a few spots of variation and have much less error than the reconstruction method.

5.3. Performance

We implemented our algorithm in C++ on a Mac Pro (Late 2013), with a 3.7Ghz Quad-Core Intel Xeon E5 processor, and 16GB DDR3 RAM. We utilized the ANN library for k-nearest neighbors calculations. The template mesh and target contour information as well as the timing results (broken down into mapping and deformation) are shown in Table 1.

<table>
<thead>
<tr>
<th>Model</th>
<th>M Vert #</th>
<th>C #</th>
<th>C Vert #</th>
<th>Time per Iter</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ellipsoid</td>
<td>2162</td>
<td>3</td>
<td>109</td>
<td>0.197+0.091s</td>
</tr>
<tr>
<td>Torus</td>
<td>4416</td>
<td>3</td>
<td>330</td>
<td>0.604+0.228s</td>
</tr>
<tr>
<td>Ferret</td>
<td>5002</td>
<td>4</td>
<td>522</td>
<td>0.926+0.238s</td>
</tr>
<tr>
<td>Ferret</td>
<td>5002</td>
<td>6</td>
<td>749</td>
<td>1.306+0.238s</td>
</tr>
<tr>
<td>Ferret</td>
<td>5002</td>
<td>8</td>
<td>945</td>
<td>1.622+0.237s</td>
</tr>
<tr>
<td>Ferret</td>
<td>5002</td>
<td>10</td>
<td>1157</td>
<td>1.974+0.236s</td>
</tr>
<tr>
<td>LiverA</td>
<td>10036</td>
<td>8</td>
<td>761</td>
<td>1.366+0.565s</td>
</tr>
<tr>
<td>LiverB</td>
<td>7502</td>
<td>8</td>
<td>761</td>
<td>1.337+0.399s</td>
</tr>
</tbody>
</table>

Table 1: Specifications for different timing trials. For the ferret we ran tests using one template and subsets of the ten target contours. For the liver we ran two tests using two different templates and the same target contours. The first column shows the template name, followed by the number of vertices in the template mesh. Then the number of target contour planes followed by the number of target contour vertices. Lastly, we show run time as the mapping time plus the deformation time for each iteration (each test was run with six iterations).

6. Conclusions

We have detailed a new template-based surface reconstruction method from cross-section curves. We formulate the problem with the help of additional planes on the template surface that are in correspondence with the target contour planes. We then use alternative optimization that includes a novel algorithm for mapping a network of spatial curves onto a surface. We test our algorithm on both synthetic and real world data to illustrate our algorithmic choices and the algorithm’s effectiveness.

In the future, we would like to test our method on a wider range of real-world biomedical data sets, particularly
those with non-trivial topology. While contours of non-trivial models are available through previous works such as [2], our method is designed to work with contours of (deformed) variants of a common template structure. We expect to collect more of these data with the availability of tools such as [3].

Note that our algorithm is not limited to genus-zero shapes, as shown in Figure 8 (a). Observe that a reasonable deformation is produced even when the number of contour curves on the source and target planes differs. However, our method may not produce a satisfactory deformation when the source and target contours differ significantly and an insufficient number of planes are provided, as shown in Figure 8 (b).

Our method requires a set of source planes in addition to the template. While these planes are specified by an expert, the choice of planes could vary between individuals, and different choices would lead to different reconstruction results give a specific target. To remove human bias, we wish to explore automated methods for creating these planes on the template by analyzing the geometric features of one surface as well as the variation among a group of surfaces.

The curve network mapping algorithm currently does not enforce the topology of the mapped curve network on the template. A single target curve segment may be mapped to a self-intersecting curve on the template, and disjoint target segments may be mapped to intersecting curves on the template. We have observed that such intersections often appear during the first few iterations of optimization, but they tend to disappear as optimization progresses and the template surface gets closer to the target contours. It would be interesting to explore topology equivalence as an additional constraint in mapping, although it would make the optimization task more challenging.

Acknowledgements
The authors would like to thank Dr. Daniel Low, formerly of the Washington University School of Medicine in St. Louis for the liver data sets, and Dr. Philip Bayly of Washington University in St. Louis for the ferret brain data sets. This work is supported in part by NSF grants IIS-1302200 and IIS-0846072.

References


