**SurfCut: Surfaces of Minimal Paths From Topological Structures**

Marei Algarni and Ganesh Sundaramoorthi

**Abstract**—We present SurfCut, an algorithm for extracting a smooth, simple surface with an unknown 3D curve boundary from a noisy 3D image and a seed point. Our method is built on the novel observation that certain ridge curves of a function defined on a front propagated using the Fast Marching algorithm lie on the surface. Our method extracts and cuts these ridges to form the surface boundary. Our surface extraction algorithm is built on the novel observation that the surface lies in a valley of the distance from Fast Marching. We show that the resulting surface is a collection of minimal paths. Using the framework of cubical complexes and Morse theory, we design algorithms to extract these critical structures robustly. Experiments on three 3D datasets show the robustness of our method, and that it achieves higher accuracy with lower computational cost than state-of-the-art.

**Index Terms**—Segmentation, surface extraction, minimal paths, computational topology, cubical complex, Morse-Smale complex

1 INTRODUCTION

MINIMAL path methods [1], built on the Fast Marching algorithm [2] (see also [3]), have been widely used in computer vision. They provide a framework for extracting continuous curves from possibly noisy images. For instance, they have been used in edge detection [4] and object boundary detection [5], mainly in interactive settings as they typically require user defined seed points. Because of their ability to provide continuous curves, robust to clutter and noise in the image, generalizations of these techniques to extract the equivalent of edges in 3D images, which form surfaces, have been attempted [6], [7]. These methods apply to extracting a surface with a boundary that forms a curve, possibly in 3D, which we call a free-boundary. Extraction of surfaces with free-boundary is important because many edges form these surfaces, and edges are fundamental structures that are prevalent in images. Some applications include medical datasets (e.g., lung fissures, walls of heart ventricles) [8] and scientific imaging datasets (e.g., fault surfaces in seismic images, an important problem in the oil industry) [9]. In [8] an alternative method to extract such surfaces, based on the theory of minimal surfaces [10], is provided. However, existing approaches to surface extraction for surfaces with free-boundary have a limitation - they require the user to provide the boundary of the surface or other user laborious input.

In this paper, we use the Fast Marching algorithm and techniques from computational topology to create an algorithm for extracting the boundary of a surface from a 3D image and a single seed point, and an algorithm to extract the surface. Our main idea is to use Fast Marching to “smooth” a local (possibly noisy) likelihood map of the surface in a way that is guaranteed to preserve locations of critical structures, and then extract the structures with methods, built from computational topology, that guarantee correct topology. We show that the resulting structures correspond to the surface of interest, and the surface is a collection of minimal paths. Our method is applicable to any imaging modality, and can be used to extract any simple surface with boundary from an image that contains noisy local measurements (possibly an edge map) of the surface. We demonstrate the method on two applications - fault extraction from seismic images, and lung fissure extraction from CT.

Our contributions are: 1. We introduce the first algorithm, to the best of our knowledge, to extract a closed 3D space curve forming the boundary of a surface from a single seed point. It is based on extracting critical structures from a distance produced by Fast Marching; 2. We introduce a new algorithm, based on extracting a critical structure of the FM distance, to extract a surface given its boundary and a noisy image. It produces a topologically simple surface whose boundary is the given space curve. The surface is shown to be formed from minimal paths. Both boundary and surface extraction have $O(N \log N)$ complexity, where $N$ is the number of pixels. 3. We provide a fully automated algorithm using the algorithms above to extract all such surfaces from a 3D image. 4. We test our method on challenging datasets, and we quantitatively out-perform comparable state-of-the-art in free-boundary surface extraction.

1.1 Related Work

1.1.1 Surface Extraction

Active surface methods [11], [12], [13], based on level set methods [14], their convex counterparts [15], graph cut methods [16], [17], and other image segmentation methods partition the image into volumes and the surfaces enclose these volumes. These methods have been used widely in segmentation. However, they are not applicable to our problem since we seek a surface, whose boundary is a 3D curve, that does not enclose a volume nor partition the image.

Our method uses the Fast Marching (FM) Method [2]. This method propagates an initial surface (e.g., a seed point) in an image in the direction of the outward normal with speed proportional to a function defined at each pixel of

M. Algarni and G. Sundaramoorthi are with KAUST (King Abdullah University of Science & Technology), Thuwal, Saudi Arabia. Email: {marei.algarni, ganesh.sundaramoorthi}@kaust.edu.sa
the image. The end result is a distance function, which gives the shortest path length (measured as a path integral of the inverse speed) from any pixel to the initial surface. The method is known to have better accuracy than discrete algorithms based on Dijkstra’s algorithm. Shortest paths from any pixel to the initial surface can be obtained from the distance function [1] (see also [18]). This has been used in 2D images to compute edges in images. A limitation of this approach is that it requires the user to input two points - the initial and ending point of the edge. In [4], the ending point is automatically detected. These methods are not directly applicable to extracting a surface forming an edge in 3D.

Attempts have been made to use minimal paths to obtain edges that form a surface. In [7], [19], minimal paths are used to extract a surface edge with a cylindrical topology, a topology different from our problem. The user inputs the two boundary curves (in parallel planes) of the cylinder and minimal paths joining the two curves are computed conveniently using the solution of a regularized transport partial differential equation. Surface extraction with less intensive user input was attempted in [6]. There, a patch of a sheet-like surface is computed with a user provided seed point and a bounding box, with the assumption that the patch slices the box into two pieces. The algorithm extracts a curve that is the intersection of the surface patch with the bounding box using the distance function to the seed point obtained with Fast Marching. Once this boundary curve is obtained, the patch is computed using [19]. The obvious drawbacks of this method are that only a patch of the desired surface is obtained, and a bounding box, which may be cumbersome to obtain, must be given by the user.

Another approach to obtaining a surface along image edges from its boundary is minimal surfaces [8], [20]. The minimal weighted area simple surface interpolating the boundary is obtained by solving a linear program. Faster implementations for minimal surfaces are explored in [8], using algorithms for the minimum cost network flow problem (e.g., [21], [22], [23], [24]). This significantly speeds up the approach, although it requires an initial surface, and the algorithm is dependent on it. The main drawback of minimal surfaces is that the user must input the boundary of the surface, which our method addresses. It is also computationally expensive as we show in experiments.

An approach for surface extraction that does not require user input is [25]. There, a matrix based on the local smoothed Hessian matrix of the likelihood is used to generate a ridge in the image near the desired surface. Then surface normals based on the matrix are computed, which are used to generate several surfaces. This method is convenient since it is fully automated. This approach has been tailored to seismic images for extracting fault surfaces [9], and it is the state-of-the-art in that field. Our method also smooths the likelihood, but in a way that preserves locations of critical structures, resulting in a more accurate surface. Also, our extraction of the critical structures, by using tools from computational topology, guarantees a simple surface topology.

1.1.2 Computational Topology
Our method is a discrete algorithm and is based on the framework of cubical complexes [26], [27]. This framework allows for performing operations analogous to topological operations in the continuum. It has been used for thinning surfaces in 3D based on their geometry [28] to obtain skeletons (or medial representations [29], [30], [31]) of geometrical shapes. This theory guarantees correct topology of extracted structures. Our novel algorithms use concepts from cubical complex theory. In contrast to [28], our method is designed to robustly extract ridges of a function or data defined on a surface (defined by Fast Marching), rather than geometrical properties of a surface.

Our method uses a topological construction called the Morse complex [32] from Morse theory to extract ridges on a manifold. There is a large literature that aims to compute the Morse complex and an extension called the Morse-Smale Complex, from discrete data [33], [34], [35], [36]. Roughly, these Morse complexes describe the behavior of the gradient flow of a function within regions. We use cubical complexes to construct the Morse complex since they are naturally suited for image data, defined on grids. Conceptually, our algorithm for the Morse complex appears similar to [33, even though the technical details and notions of discrete topology are different. Our contribution is not to provide another algorithm for the Morse complex, but to use the Morse complex for the purpose of free-boundary surface extraction from images.

1.1.3 Extensions to Conference Paper
A preliminary version of this manuscript has appeared in [37]. In this version, we have derived the theoretical foundations: 1) we provide analytical arguments to show that our algorithms correctly capture the surface of interest by relating it to constructions in Morse theory, and 2) we relate the surface extracted to minimal paths by showing the surface is formed by collections of minimal paths, thus inheriting known regularity properties from such paths. We extended our ridge extraction algorithm to better deal with extraneous structures. We have also extended our experiments to more datasets, including medical data, and extended the quantitative comparison to minimal surface approaches.

1.2 Overview of Method
Our algorithm consists of the following steps (see Figure [1]): i) Weighted Distance to Seed Point Computation: From a given seed point on the surface, the Fast Marching algorithm is used to propagate a front to compute shortest path distance from any point in the image to the seed point (Section 3.1). ii) Ridge Curve Extraction: At samples of the propagating front, the ridge curves of the Euclidean path distance of minimal paths to the seed point are computed (Section 3.2). These curves lie on the surface of interest. iii) Surface Boundary Detection: At snapshots, a graph is formulated from curves from the previous step, and is cut along locations where the Euclidean distance between points on adjacent curves are small, resulting in the outer boundary of the surface (Section 3.3). iv) Surface Extraction: Finally, the desired surface with boundary obtained from the last step is computed (Section 4). Our method requires notions from topology, which we review next. We then proceed to our algorithm.
2 Topological Preliminaries

In this section, we present theory and notions from topology and computational topology that will be relevant in subsequent sections in designing and justifying our novel algorithms for surface extraction.

2.1 Topological Structures

Our algorithms extract topological structures from functions defined on the image domain and manifolds embedded in the image. We give formal definitions for these topological structures, ridges and valleys, and then the Morse complex.

2.1.1 Critical Structures

Intuitively, ridge points of a function defined on a manifold correspond to local maxima when restricted to subspaces of directions rather than the whole space of possible directions. Similarly, valley points correspond to local minima of a function when restricted to subspaces of directions. We now give more formal definitions. We consider functions $h: M \subset \mathbb{R}^n \to \mathbb{R}$, defined on a $n-1$ dimensional manifold. For a point $x \in M$, we denote $T_x M$ to be the tangent space of $M$ at $x$, which consists of all valid directions at the point $x$ on $M$. We first define the critical points of $h$ as the points $p$ on $M$ where the gradient vanishes, i.e., $\nabla h(p) = 0$. Note that the gradient refers to the intrinsic gradient $\nabla h(x) \in T_x M$, i.e., it is defined by the relation $dh(x) \cdot v = \nabla h(x) \cdot v$ for all $v \in T_x M$ where $dh(x) \cdot v$ denotes the directional derivative of $h$ at $x$ and the right hand side is the usual Euclidean dot product. Ridges and valleys are formally defined by [38] as follows.

Definition 1 (Ridge and Valley). Let $h: M \subset \mathbb{R}^n \to \mathbb{R}$ where $M$ is an $n-1$ dimensional manifold. Let $\lambda_1 \leq \cdots \leq \lambda_{n-1}$ and $e_1, \ldots, e_{n-1} \in T_x M$, be eigenvalues and eigenvectors of the Hessian $H h(x) |_{x \in M}$. Let $k < n - 1$.

- A point $x \in M$ is a $n-1-k$ dimensional ridge point of $h$ if $\lambda_k < 0$ and $\nabla h(x) \cdot e_m = 0$ for $m = 1, \ldots, k$.

- A point $x \in M$ is a $n-1-k$ dimensional valley point of $h$ if $\lambda_{n-k} > 0$ and $\nabla h(x) \cdot e_m = 0$ for $m = n-k, \ldots, n-1$.

The conditions above ensure zero derivatives in a subspace of directions, and the conditions on the Hessian ensure the function is concave (for ridges) and convex (for valleys) in the appropriate subspace. Differentiability in the definition is not needed, and there are more generic conditions for continuous functions, e.g., that the function value is higher at the ridge point than other points in a subneighborhood corresponding to a subspace of directions.

2.1.2 Morse Complex

Our algorithms for extracting the previous structures do not directly use the differential definitions above, as they are not robust to noise in the image. We will design our algorithms based on topological constructions in Morse theory [39]. We introduce basic notions from that literature, and the exact relation to ridges and valleys will be left to subsequent sections when we specify our algorithms. We define the ascending and descending manifolds of a critical point as all points on a path along the negative (positive, respectively) gradient direction that leads to the given critical point. A path on a manifold $M$ is a mapping $\gamma: [0, \infty) \to M$. A gradient path is specified by the differential equation $\gamma'(t) = \pm \nabla h(\gamma(t))$, where $h$ is some function defined on $M$. Formally, the ascending and descending manifolds of a critical point $p$ of $h$ are defined as follows [32].

**Definition 2** (Ascending and Descending Manifolds). Let $h: M \to \mathbb{R}$ be a function and $p$ be a critical point of $h$. The ascending manifold at $p$ is

$$A(p) = \{ x \in M : \text{there exists } \gamma: [0, \infty) \to M \text{ such that } \gamma(0) = x, \gamma(\infty) = p, \gamma'(t) = -\nabla h(\gamma(t)) \}. \quad (1)$$

The descending manifold at $p$ is

$$D(p) = \{ x \in M : \text{there exists } \gamma: [0, \infty) \to M \text{ such that } \gamma(0) = x, \gamma(\infty) = p, \gamma'(t) = \nabla h(\gamma(t)) \}. \quad (2)$$

For instance, consider the function $h: \mathbb{R}^2 \to \mathbb{R}$ defined by $h(x, y) = x^2 + y^2$. Its ascending manifold at the critical point $0$ is $A(0) = \mathbb{R}^2$ as all negative gradient paths lead to the origin. Note also that $D(0) = 0$. See Figure 2 for a visualization in the one-dimensional case.
A logical operation called a deformation retraction, i.e., the operation of continuously shrinking a topological space to a subset. For example, a punctured disk can be continuously shrunken to its boundary circle. Therefore, the boundary circle is a deformation retraction of the punctured disk, and the two are said to be homotopy equivalent. We are interested in an analogous discrete operation, whereby faces of the cubical complex can be removed while preserving homotopy equivalence. Free faces (see Fig. 3), defined in cubical complex theory, can be removed simplifying the cubical complex, while preserving a discrete notion of homotopy equivalence. These are defined formally as:

**Definition 4.** Let \( X \) be a cubical complex, and let \( f, g \subseteq X \).

- \( g \) is a **proper face** of \( f \) if \( g \neq f \) and \( g \) is a sub-face of \( f \).
- \( g \) is **free** for \( X \), and the pair \((g, f)\) is a **free pair** for \( X \) if \( f \) is the only face of \( X \) such that \( g \) is a proper face of \( f \). If \( g \) is not free, it is called **isthmus**.

The definition provides a constant-time operation to check whether a face is free. For example, if a cubical complex \( X \) is a subset of the 3-dim complex formed from a 3D image grid, a 2-face is known to be free by only checking whether only one 3-face containing the 2-face is contained in \( X \).

In the next section, we construct cubical complexes for the evolving front produced from the Fast Marching algorithm, and retract this front by removing free faces to obtain a lower dimensional ridge curve that lies on the surface that we wish to obtain. We also retract a volume to obtain a valley, which forms the surface of interest.

## 3 Surface Boundary Extraction

In this section, we present our algorithm for extracting the boundary curve of a free-boundary surface from a possibly noisy local likelihood map of the surface defined in a 3D image. The algorithm consists of retracting the fronts (closed surfaces) generated by the Fast Marching algorithm to obtain ridge curves on the surface of interest. We therefore review Fast Marching in the first sub-section before defining our novel algorithms for surface extraction.

### 3.1 Fronts Localized to the Surface With Fast Marching

We use the Fast Marching Method [2] to generate a collection of fronts that grow from a seed point and are localized to the surface of interest. We denote by \( \phi : \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R}^+ \), a possibly noisy function defined on each pixel of the given image grid. It has the property that (in the noiseless situation) a small value of \( \phi(x) \) indicates a high likelihood of the pixel \( x \) belonging to the surface of interest.

Fast Marching solves, with complexity \( O(N \log N) \) where \( N \) is the number of pixels, a discrete approximation to \( U : \Omega \subset \mathbb{R}^3 \rightarrow \mathbb{R}^+ \), the solution of the eikonal equation:

\[
\begin{aligned}
|\nabla U(x)| &= \phi(x) & x \in \Omega \setminus \{p\} \\
U(p) &= 0
\end{aligned}
\]  

(3)

where \( \nabla \) denotes the spatial gradient (partials in all coordinate directions), and \( p \in \Omega \) denotes an initial seed point. For our situation, \( p \) will be required to lie somewhere on the surface of interest. The function \( U \) at a pixel \( x \) is the weighted minimum path length along any path from \( x \) to \( p \), with weight defined by \( \phi \). \( U \) is called the weighted distance. Minimal paths can be recovered from \( U \) by following the gradient descent of \( U \) from any \( x \) to \( p \). A front (a closed surface, which we hereafter refer to as a front to avoid confusion with the free-boundary surface) evolving from the seed point at each time instant is equidistant (in terms of \( U \)) to the seed point and is iteratively approximated by Fast Marching. As noted by [1], a positive constant added to the right hand side of (3) may be used to induce smoothness.
of paths. The front, evolving in time, moves in the outward normal direction with a speed proportional to $1/\phi(x)$. Fronts can be alternatively obtained by thresholding $U$ at the end of Fast Marching. The solution of (3) is continuous, and can be approximated as smooth since the solution is a viscosity solution [40], and so a limit of smooth functions.

3.2 Contours on the Surface from Front Ridges

If we choose the seed point $p$ to be on the free-boundary surface of interest, the front generated by Fast Marching will travel the fastest when $\phi$ is small (i.e., along the surface) and travel slower away from the surface, and thus the front is elongated along the surface at each time instant (see Figure 4). Our algorithm is based on the following observation: points along the front at a time instant that have traveled the furthest (with respect to Euclidean path length), i.e., traveled the longest time, compared to nearby points, lie on the surface of interest. This is because points traveling along locations where $\phi$ is low (on surface) travel the fastest, tracing out paths that have large arc-length.

This property can be more easily seen in the 2D case (see Figure 4): suppose that we wish to extract a curve rather than a surface from a seed point, using Fast Marching to propagate a front. At each time, the points on the front that travel the furthest with respect to Euclidean path length lie on the 2D curve of interest. This has been noted in 2D by [4]. In 3D (see Figure 4), we note this generalizes to ridge points of Euclidean minimal path length $U_E$ (defined next) are on the surface of interest. The Euclidean minimal path length $U_E$ is defined as follows. Define a front $F = \partial \{ x \in \Omega : U(x) \leq D \}$ where $\partial$ denotes the boundary operator. The function $U_E : F \rightarrow \mathbb{R}^+$ is such that $U_E(x)$ is the Euclidean path length of the minimal weighted path (w.r.t to the distance $U$) from $x$ to $p$.

Computationally, $U_E$ is easy to obtain by keeping track of another function $U_E : \Omega \rightarrow \mathbb{R}^+$ in Fast Marching for $U$. One follows the ordered traversal of points according to Fast Marching in solving for $U$, and simultaneously updates the value of $U_E$ based on a discretization of (3) with $\phi$ chosen equal to 1. This gives the Euclidean length of minimal paths determined from $U$.

The fact that ridge points lie on the surface is visualized in the right of Figure 4. Points on the intersection of the surface and the front are such that in the direction orthogonal to the surface, the minimal paths have Euclidean lengths that decrease. This is because $\phi$ becomes large in this direction, thus minimal paths travel slower in this region, so they have lower Euclidean path length. Along the surface, at the points of intersection of the surface and front, the path length may increase or decrease, depending on the uniformity of $\phi$ on the surface. This implies points on the intersection of the front and surface are ridge points of $U_E[F]$.

We now give an analytic argument that common points to the surface and the front are ridge points.

**Proposition 1.** Suppose $S \subset \Omega$ is a smooth surface and $p \in S$. Consider the front $F = \{ U = D \}$ and suppose $x \in S \cap F$ then $x$ is a ridge point of $U_E : F \rightarrow \mathbb{R}$, where $U_E(y)$ is defined as the Euclidean length of the minimal path from $y$ to $p$. We assume that locally $\phi$ is larger on $S$ than points not on $S$.

**Proof.** Let $x \in S \cap F$ and let $N$ be a normal vector to $S$ at $x$. We choose a neighborhood $V_x \subset \Omega$ around $x$ so that $S$ is approximately flat and $\phi$ is approximated as

$$
\phi(x) = \begin{cases} 
K_1 & x \notin S \cap V_x \\
K_2 & x \in S \cap V_x 
\end{cases},
$$

where $K_1 > K_2 > 0$, which are constants. Let us consider a point $y = x + \varepsilon N$, where $\varepsilon > 0$ is small, and the minimal path from $y$ to $p$ (see Figure 5). We note that minimal paths within $V_x \setminus S$ will be straight lines as $\phi$ is uniform in that region. For $\varepsilon > 0$ small enough, we can find $q \in S$ on the minimal path from $x$ to $p$ so that the minimal path from $y$ to $p$ is the straight line path from $y$ to $q$ appended to the minimal path from $q$ to $p$. We note that if we let $\ell = |x - q|$ then

$$
U(x) = U(q) + K_2 \ell.
$$

Also,

$$
U(y) = U(q) + K_1 \sqrt{\ell^2 + \varepsilon^2},
$$

Fig. 4: [Top left]: The evolving Fast Marching (FM) front at two different time instances in orange and white. The function $1/\phi$ evaluated at $x$ is the likelihood of surface passing through $x$, and is visualized (red - high values, and blue - low values). The fronts are localized near the surface of interest. Ridge points of $U_E$, the Euclidean path length of minimal weighted paths, lie on the surface of interest. [Top right]: This is more easily seen in 2D where the local maxima of the Euclidean path length (red balls) of minimal paths (dashed) are seen to lie on the curve of interest. The green contour is a snapshot of the front. [Bottom]: Schematic in 3D with front (blue), surface (green), and several minimal paths (orange). Orthogonal to the surface where the surface intersects the front, the Euclidean path length decreases. Along the surface, the path lengths may increase or decrease. This indicates ridge points.

Fig. 5: Schematic of quantities in the proof of Proposition 1.
and any point \( z \) on the line between \( y \) and \( q \) will have
\[
U(z) = U(q) + tK_1 \sqrt{\ell^2 + \varepsilon^2}
\]
where \( t \in (0, 1) \). If we search for the point \( z \) on the line between \( q \) and \( y \) on the front \( F \), which has \( U(z) = U(x) \), we find that
\[
t = \frac{K_2}{K_1} \frac{\ell}{\sqrt{\ell^2 + \varepsilon^2}} < 1.
\]
Therefore, the Euclidean length of the minimal path from \( z \) to \( p \) is
\[
U_E(z) = \frac{K_2}{K_1} \ell + \text{len}(\gamma_{q,p})
\]
where \( \text{len}(\gamma_{q,p}) \) is the length of the minimal path from \( q \) to \( p \). Notice this has less length than the path from \( x \) to \( p \), which is \( U_E(x) = \ell + \text{len}(\gamma_{q,p}) \). Therefore, \( U_E(z) < U_E(x) \). So moving in the direction \( N \) along \( F \) reduces the Euclidean length of minimal paths. This same argument holds for any \( z \) within \( V_x \) along the direction \( -N \) from \( x \). This implies that \( x \in F \cap S \) is a one-dimensional ridge point of \( U_E \).

This tells us that points of the front that are on the surface must be ridge points, and so we restrict our attention to ridge points on the front as possible points on the surface.

### 3.3 Ridge Curve Extraction Using the Morse Complex

Since computing ridges directly from Definition 1 using differential operators, is sensitive to noise, scale spaces \[41], \[42] are often used. However, that approach, while being more robust to noise, may distort the data, and it does not guarantee connectedness. A heap is used to keep track of the faces in order. The computational complexity of this extraction is therefore \( O(N \log N) \) where \( N \) is the number of pixels, an over-estimate since the faces in the complex are significantly lower than the number of pixels.

Ideally, in the case of clean data \( \phi \), the function \( U_E \) defined on the front would have a rather simple topology, indeed a volcano structure (see left image in Fig. 8), where the ridge separates the inside of the volcano from the outside. The two minimum of \( U_E \) on each side of the ridge would correspond to points away from the surface in the direction of the surface normal. In this case, the previous algorithm would produce the inside of the volcano, and the outside as two components of the complex, and the boundary between them as the ridge, as desired. However, due to noise other ridge structures besides the main ridge of interest can be extracted.

Fortunately, we can simplify the extracted collection of ridges from the previous algorithm by applying the algorithm iteratively. We construct a new complex with a 2-face for each ascending manifold computed, and a 1-face connecting 2-faces if two corresponding ascending manifolds have intersecting boundaries. Each 1-face in this new complex is assigned a value to be the average of 1-faces in the common boundary between ascending manifolds. The Morse complex of this simplified complex is then computed, and the process is repeated until only one loop remains. The algorithm is given in Algorithm 2. Figure 7 shows an example run through this algorithm.

**Algorithm for Ridges via Morse Complex:** Next, we specify a discrete algorithm to determine the Morse complex of \( -U_E[F] \). The boundaries of ascending manifolds can then be used to extract the relevant ridge. We retract the front to the ridge curve by an ordered removal of free faces based on lowest to highest ordering based on \( U_E[F] \).

Given a front \( F \), obtained by thresholding the distance \( U \), the two-dimensional cubical complex \( C_F \) of the front is constructed as follows. Let \( \mathbb{Z}_n = \{0, 1, \ldots, n - 1\} \) be a sampling of a coordinate direction of the image. Then

- \( C_F \) contains all 2-faces \( f \) in \( \mathbb{Z}_n^3 \), between any 3-faces \( g_1, g_2 \) with the property that one of \( g_1, g_2 \) has all its 0-sub-faces with \( U < D \) and one does not.
- Each face \( f \) of \( C_F \) has cost equal to the average of \( U_E \) over 0-sub-faces of \( f \).

Our algorithm for Morse complex extraction and boundaries of the ascending manifolds is given in Algorithm 1.

The algorithm creates holes at local minima of the function \( U_E[F] \) defined on 1-faces by removing the adjacent 2-faces. It then removes free faces in increasing order of \( U_E[F] \) so as to preserve homotopy equivalence. The removed points associated with a local minimum form the ascending manifold for the local minimum. The faces that cannot be removed without breaking homotopy equivalence, i.e., the isthmus faces, form the boundaries of the ascending manifolds. The algorithm removes all 2-faces and preserves only isthmus 1-faces, and hence the remaining structure of \( C_F \) is one dimensional. Further, since the algorithm preserves homotopy equivalence, the remaining structure at the end of the algorithm is connected. This is a clear advantage over computation of ridges from differential operators, which does not guarantee connectedness.

**Relation Between Ridges and Morse Complex:** In the following proposition, we note that certain ridges of a smooth function can be computed by computing ascending manifolds. We assume that \( M \) is a 2-manifold.

**Proposition 2.** Boundaries of ascending manifolds of \( h \) are ridges of \( h \).

**Proof.** Suppose that \( x \in \partial A(p_1) \) then for any neighborhood \( V_z \) sufficiently small around \( x \), we have that \( \partial A(p_1) \cap V_z \) divides \( V_z \), i.e., \( V_z = [V_z \cap A(p_1)] \cup [V_z \cap A(p_2)] \) \((p_1 \neq p_2)\) for the case when \( V_z \) intersects two ascending manifolds. Note that \( -\nabla h(y) \cdot N_2 > 0 \) for \( y \in V_z \cap A(p_2) \) where \( N_2 \) is the inward normal to \( \partial A(p_2) \) when \( V_z \) is small enough. If this were not the case, then paths following the negative gradient would intersect the boundary \( \partial A(p_2) \), which is not the case since they flow into \( p_2 \). By a similar argument, \( -\nabla h(y) \cdot N_2 < 0 \) for \( y \in V_z \cap A(p_1) \). Since the function \( h \) is assumed smooth and thus the gradient is continuous, we must have that \( \nabla h(x) \cdot N_2 = 0 \). Further, the function is decreasing away from \( x \) along the directions \( \pm N_2 \) as points in \( V_z \{x\} \) belong to ascending manifolds. Therefore, the point \( x \) is a local maximum in the direction \( N_2 \). Ridges satisfy this property. Hence, boundaries of the ascending manifolds are ridges. \( \square \)
Algorithm 1 Morse Complex Extraction
1: procedure Morse Complex($C_F$, $U_E$)
2:   $C_F = \text{cubical } 2\text{-complex}, \ U_E = \text{cost on 1-faces in } C_F$
3:   id $\leftarrow 0$
4:   Create heap of 1-faces ordered by $U_E$ (min at top)
5: repeat
6:   Remove 1-face $g$ from heap
7:   if $g$ is a subset of two faces $f_1$ and $f_2$ in $C_F$ then
8:     Remove $g$, $f_1$, $f_2$ from $C_F$
9:     $l(f_1) \leftarrow l(f_2) \leftarrow \text{id}, \ \text{id} \leftarrow \text{id} + 1$
10:     $\triangleright$ new id for ascending manifold; hole at local min
11:   else if $(g, f)$ is a free pair in $C_F$ then
12:     Remove $g$, $f$ from $C_F$
13:     $l(f) \leftarrow l(f_{\text{adj}})$ where $f_{\text{adj}} \supset g \text{ and } f_{\text{adj}} \notin C_F$
14:     $\triangleright$ labels face same as adjacent face containing $g$
15:   else if $(f, g)$ is a free pair in $C_F$ then
16:     Remove $g$, $f$ from $C_F$
17:   else if $g$ is isthmus then
18:     $l(g) = \{l(f_1), l(f_2)\}$ where $f_1, f_2 \supset g$
19:     $\triangleright$ label is unordered list
20:   until heap is empty
21: return $C_F, l$  $\triangleright$ Ridges, labels for 2-faces, ridges
22: end procedure

Algorithm 2 Highest Ridge Curve Extraction
1: procedure HighestRidge($C_F$, $U_E$)
2:   $C_F = \text{cubical } 2\text{-complex of Fast Marching front}$
3:   $U_E = \text{Euclidean trajectory length}$
4:   defined on 1-faces
5: repeat
6:   $C_F, l = \text{Morse Complex} (C_F, U_E)$
7:   Create 2-cubical complex $C'_F$ with
8:   a 2-face $f$ for each unique 2-face id in $l$
9:   a 1-face $g$ for each unique 1-face id in $l$
10:   $g$ joins $f_1$ and $f_2$ if $l(g) = \{l(f_1), l(f_2)\}$
11: for $g$ each 1-face in $C'_F$ do
12:   $R = \{g' \in C'_F : l(g') = l(g)\}$
13:   $U'_E(g) \leftarrow \text{average of } U_E$ along $R$
14: end for
15: $C_F \leftarrow C''_F, U_E \leftarrow U'_E$
16: until no degree three 1-faces in $C_F$
17: return $C_F$

Fig. 6: [1st image]: Front color coded with Euclidean path length $U_E$ (top view). Red indicates high values. The bottom view (not shown) is a symmetric flip. Topologically, $U_E$ forms a volcano structure (ridge, i.e., top of volcano, is darkest red), and inside the volcano is blue. [Subsequent images]: Illustration of iterations (from left to right) of Algorithm 1 on noise-less data to obtain the ridge curve (white) on the Fast Marching front (green) by computing the Morse complex of $U_E$. The ridge curve lies on the surface of interest (red).

Fig. 7: Illustration of Algorithm 2 operating on noisy data to obtain the highest ridge.

Fig. 8: [Left]: Ridge curve (white) extraction by retracting the Fast Marching front at two instants. [Right]: An example cut (red) of ridge curves, forming the surface boundary. Notice that the cut matches with the end of high $1/\phi$ (bright areas).

An example of ridge curves detected for multiple fronts is shown in Figure 8. This procedure of retracting the Fast Marching front to form the main ridge is continued for different fronts of the form $\{U < D\}$ with increasing $D$. This forms many curves on the surface of interest. In practice, in our experiments, $D$ is chosen in increments of $\Delta D = 20$, until the stopping condition is achieved, and this typically results in $10 - 20$ ridge curves extracted. The next subsection describes the stopping criteria.

3.4 Stopping Criteria and Surface Boundary Extraction
To determine when to stop the process of extracting ridge curves, and thus obtain the outer boundary of the surface of interest, we make the following observation. Parts of the curves generated from the previous section move slowly, i.e., become close together with respect to Euclidean distance at the boundary of the surface. This is because the speed function $1/\phi$ becomes small outside the surface. Hence, for the curves $c_i$ generated, we aim to detect the locations where the distance between points on adjacent curves becomes small. To formulate an algorithm robust to noise, we formulate this as a graph cut problem [16].

We define the graph $G$ as follows:
- vertices $V$ are 0-faces in all the 1-complexes $c_i$ formed from ridge extraction
- edges $E$ are $(v_1, v_2)$ where $v_1, v_2 \in V$ are such that $v_1, v_2$ are connected by a 1-face in some $c_i$ or $v_1$ is a 0-face in $c_i$ and $v_2$ is the closest (in terms of Euclidean distance) 0-face in $c_{i+1}$ to $v_1$
- a cost $|v_j - v_k|$ is assigned to each edge $(v_j, v_k)$ where $v_j$ and $v_k$ belong to different $c_i$ (so that the min cut will be where adjacent curves are close)
- for edges $(v_j, v_k)$ such that $v_j$ and $v_k$ belong to the same $c_i$, the cost is the minimum Euclidean distance between segment $(v_j, v_k)$ and segments on $c_{i+1}$
- the source is the seed point $p$, and the sink is the last ridge curve $c_l$

We wish to obtain a cut of $G$ (separating $G$ into two disjoint sets) with minimum total cost defined as the sum of
all costs along the cut. In this way, we obtain a cut of the ridge curves along locations where the distance between adjacent ridge curves is small. The process of obtaining ridge curves from the Fast Marching front is stopped when the cost divided by the cut size is less than a pre-specified threshold. This cut then forms the outer boundary of the surface. The computational cost of the cut (compared to other parts of the algorithm) is negligible as the graph size is typically less than 0.5% of the image. Figure 8 shows an example of a cut that is obtained. Figure 11 shows a synthetic example.

4 Surface Extraction

We now present our algorithm for surface extraction. Given the surface boundary curve determined from the previous section, we provide an algorithm that determines a surface going through locations of small \( \phi \) and whose boundary is the given curve. Our algorithm uses the cubical complex framework and has complexity \( O(N \log N) \).

4.1 Valley Extraction Algorithm and Rationale

We show now that the surface of interest lies in a valley of \( U : \Omega \to \mathbb{R}^+ \), the weighted minimal path length.

**Proposition 3.** Suppose \( S \subset \mathbb{R}^3 \) is a smooth surface and \( p \in S \). Let \( \phi : \Omega \subset \mathbb{R}^3 \to \mathbb{R}^+ \) be a function with low values on \( S \) and higher values outside (locally). Then \( S \) is a valley of \( U \), where \( U \) is the solution of the eikonal equation with \( U(p) = 0 \).

**Proof.** We show that for \( x \in S \), \( U \) decreases away from \( x \) in the direction \( \pm N \), the normals to the surface at \( x \). For a small enough neighborhood \( V_x \) around \( x \), we may assume that \( S \) is flat and that \( \phi \) is approximated by

\[
\phi(x) = \begin{cases} 
K_1 & x \notin S \cap V_x \\
K_2 & x \in S \cap V_x 
\end{cases}
\]

where \( K_1 \gg K_2 > 0 \). We also assume (for now) that \( x \) close enough to \( p \) so that \( p \) lies in \( V_x \). In this case, we see that

\[
U(x) \approx U(p) + K_2 |x - p| = LK_2,
\]

as the minimal path from \( p \) to \( x \) is approximately a straight line path on the surface, as the surface is nearly flat in \( V_x \).

Let \( y = x + \varepsilon N \) for \( \varepsilon > 0 \) sufficiently small. We now consider the minimal path from \( y \) to \( p \). Note outside the surface, the path must be nearly a straight line as \( \phi \) is constant. Similarly, on the surface the minimal path must be a straight line. We see that the minimal path is a straight line between \( y \) and some point \( z \) on the line joining \( x \) to \( p \) and then the straight line between \( z \) and \( p \). Therefore,

\[
U(y) = \min_{\ell} K_2 (L - \ell) + K_1 \sqrt{\ell^2 + \varepsilon^2}
\]

where \( L \) is the length of the segment between \( x \) and \( z \). The minimizer is \( \ell = \varepsilon / \sqrt{1 - r^2} \), where \( r = K_2 / K_1 < 1 \). This yields that

\[
U(y) = LK_2 - \frac{\varepsilon}{\sqrt{1 - r^2}} K_2 + \frac{\varepsilon \sqrt{2 - r^2}}{\sqrt{1 - r^2}} K_1
\]

\[
= LK_2 + \frac{\varepsilon}{\sqrt{1 - r^2}} [K_1 \sqrt{2 - r^2} - K_2] > LK_2,
\]

where the last inequality follows from the fact that \( \sqrt{2 - r^2} > 1 \) and \( K_1 > K_2 \). Therefore, \( U(y) > U(x) \) and so we see a local minimum in the direction \( N \) which implies \( x \) lies in a 2-d valley of \( U \). We may now apply the same argument using \( x \) to play the role of \( p \), and show that all points in a neighborhood of \( x \) on the surface are on a valley. We may continue in this way to show all points on the surface are on the valley.

**Algorithm:** We can use the above fact to design an algorithm for extracting the surface. We may perform a deformation retraction of \( V_0 = \{ U \leq T(0) \} \) where \( T(0) \) is chosen to enclose the entire surface, and \( T(t) \) is a decreasing function of \( t \). At each time, the points of the level set \( L_t = \{ U = T(t) \} \) retain the homotopy equivalence to \( V_t \) are removed from \( V_t \). We further impose that the boundary of the surface must not be removed from \( V_t \). This way, all points that are on the surface are retained. One can show this with an inductive argument. Assume for a given time \( t \), the union of all retained sets is a 2-dim set \( S_t \) (\( t \) is just before \( t \) that is on the surface, and so \( V_t = S_t \) \( \cup \) \( \{ U \leq T(t) \} \). Note that the latter set in the union is a volume. A point \( x \in \partial S_t \) with \( U(x) = T(t) \) cannot be removed. Since \( x \) is on the surface, which by the proposition is a valley point, the normal to the surface at \( x \) is tangent to \( L_t \) and \( U \) is strictly increasing along the normal. Therefore, removing point \( z \) disconnects \( V_t \), not preserving homotopy equivalence. Therefore, \( V_t = S_t \) \( \cup \) \{ \( U < T(t) \} \) where \( S_t \) contains all points on \( \partial S_t \).

This procedure can be accomplished with an analogous algorithm in the discrete case. We retract the cubical complex of the image with the constraint that the boundary curve 1-faces cannot be removed. We accomplish this retraction by an ordered removal of free faces based on weighted path length \( U \). The algorithm is described in Algorithm 3. Figure 9 shows the evolution from Algorithm 3 to extract the surface from the data used in Figure 8. Figure 11 shows a synthetic example of the evolution of this algorithm.

4.2 Valley: Surface of Minimal Paths

We now relate the valley that is extracted by our algorithm to minimal paths. We show that the valley, and thus the surface extracted, is a surface formed from a collection of minimal paths to \( p \). First, we show that the gradient path starting from a point in the valley stays in the valley.

**Proposition 4.** Suppose \( x \in M \) is a valley point of \( h : M \to \mathbb{R} \), then the path \( \gamma \) determined by the gradient descent of \( h \) with initial condition \( x \) lies on the valley of \( h \) containing \( x \).
formed from the gradient descent is also in the valley. Thus continuing this way, we can show that the path \( \gamma \) as \( N \) is two-dimensional. By definition of a 2D valley in \( U \) is a valley of \( V \). The solution of the eikonal equation, containing the seed point \( p \) used to define \( U \). Then \( V \) is a union of minimal paths to \( p \).

\begin{proof}
Let \( x \in V \) then the path \( \gamma_x \) formed from the gradient descent of \( U \) starting from \( x \) stays in \( V \) by Proposition 5. The path \( \gamma_x \) is also a minimal path since gradient paths of \( U \) are minimal paths. Note that \( \gamma_x \) ends at \( p \). Therefore, we see that \( V \) is the union of \( \gamma_x \) over all \( x \).
\end{proof}

\section{Experiments}

Supplementary video are available. We qualitatively and quantitatively assess our method by comparing against competing algorithms.

\subsection{Datasets and Parameters}

We evaluate our method on three datasets of 3D images.

\textbf{Synthetic Dataset:} We construct a synthetic dataset consisting of 20 different surfaces with boundary at three different image resolutions, 100 \( \times \) 100 \( \times \) 100, 500 \( \times \) 500 \( \times \) 500 and 800 \( \times \) 800 \( \times \) 800. Each of the surfaces have different 3D boundary curves of different shape, and surfaces that have various degrees of coarse and fine features. Example surfaces are shown in Fig. 11. The images are formed by setting pixels not within distance 1 to the surface to 1 and all other pixels to 0. The surfaces meshes are downsampling for the lower resolution images. Noise with level \( \sigma = 0.1 \) is then added to the images.

\textbf{Seismic Dataset:} Seismic images are formed from measurements of seismic pulses reflected back from the earth’s sub-surface. They are 3D images, and are used to measure geological structures. We have a dataset of three volumes with dimensions 463 \( \times \) 951 \( \times \) 651. The goal is to extract fault surfaces, which form free-boundary surfaces within the volume. Faults may have significant curvature, and the boundaries are non-planar. The images are cluttered and

\begin{algorithm}[H]
\caption{Surface Extraction from Boundary of Surface}
\begin{algorithmic}[1]
\Procedure{ValleyExtract}{$C_I$, $U$, $\partial S$}
\Comment{$C_I$ = cubical 3-complex of image, $U$ = FM distance}
\Comment{$\partial S$ = boundary of surface (1-complex)}
\State Create heap of 2-faces ordered by $U$ (max at top)
\Repeat
\State Remove 2-face $g$ from heap
\If{$(g, f)$ is a free pair in $C_I$ for some $f$} Remove $f$ and $g$ from $C_I$
\EndIf
\Until heap is empty
\State return $C_I$
\EndProcedure
\end{algorithmic}
\end{algorithm}

Fig. 11: Synthetic example of extracting a sphere with top cut such that the boundary is four arcs. The image (not shown) is a noisy image of the cut sphere with holes. Ridge curves are extracted via Algorithm 1 (top left). The final cut of ridge curves (top, middle left), the final surface extracted via Algorithm 2 (top, middle right), and the ground truth (top, right) are shown. Snapshots in the removal of faces in Alg. 2 are shown (bottom), resulting in the surface (right).

Proposition 5. Suppose \( V \) is a valley of \( U \), the solution of the eikonal equation, containing the seed point \( p \) used to define \( U \). Then \( V \) is a union of minimal paths to \( p \).

Proof. Let \( x \in V \) then the path \( \gamma_x \) formed from the gradient descent of \( U \) starting from \( x \) stays in \( V \) by Proposition 5. The path \( \gamma_x \) is also a minimal path since gradient paths of \( U \) are minimal paths. Note that \( \gamma_x \) ends at \( p \). Therefore, we see that \( V \) is the union of \( \gamma_x \) over all \( x \).

5.1 Datasets and Parameters

We evaluate our method on three datasets of 3D images.

\textbf{Synthetic Dataset:} We construct a synthetic dataset consisting of 20 different surfaces with boundary at three different image resolutions, 100 \( \times \) 100 \( \times \) 100, 500 \( \times \) 500 \( \times \) 500 and 800 \( \times \) 800 \( \times \) 800. Each of the surfaces have different 3D boundary curves of different shape, and surfaces that have various degrees of coarse and fine features. Example surfaces are shown in Fig. 11. The images are formed by setting pixels not within distance 1 to the surface to 1 and all other pixels to 0. The surfaces meshes are downsampling for the lower resolution images. Noise with level \( \sigma = 0.1 \) is then added to the images.

\textbf{Seismic Dataset:} Seismic images are formed from measurements of seismic pulses reflected back from the earth’s sub-surface. They are 3D images, and are used to measure geological structures. We have a dataset of three volumes with dimensions 463 \( \times \) 951 \( \times \) 651. The goal is to extract fault surfaces, which form free-boundary surfaces within the volume. Faults may have significant curvature, and the boundaries are non-planar. The images are cluttered and
Lung CT Dataset: We use a dataset of 10 3D computed tomography (CT) of the lung of cancer patients from the Cancer Imaging Archive (TCIA) [43]. Each image has size $512 \times 512 \times Z$, where $Z$ varies between 300 and 700, depending on the patient. Our goal is to segment lung fissures (e.g., [44], [45]), which are the boundaries between sections of the lung. They are very thin, subtle structures, and form free-boundary surfaces. Each of the lung fissures in each image is human annotated, for every slice.

Parameters: Our algorithm, given the local surface likelihood $\phi$, requires only one parameter, the threshold on the cut cost. In all experiments, we choose this to be $T = 5$. This is not sensitive to the data (see Supplementary).

5.2 Evaluation Methodology

We validate our results with quantification measures for both the accuracy of the surface boundary and the surface using quantities analogous to the precision, recall and F-measure. We represent the surface and its boundary as voxels. Let $S_r$ denote the surface returned by an algorithm and let $S_{gt}$ be the ground truth surface. Denote by $\partial S_r$ and $\partial S_{gt}$ the respective boundaries. We define

$$N(r \rightarrow gt) = |\{v \in S_r : d_{S_{gt}}(v) < \varepsilon\}|$$

$$N(gt \rightarrow r) = |\{v \in S_{gt} : d_{S_r}(v) < \varepsilon\}|$$

$$P_S = \frac{N(r \rightarrow gt)}{|S_r|},$$

$$R_S = \frac{N(gt \rightarrow r)}{|S_{gt}|},$$

$$F_S = 2P_S R_S / (P_S + R_S)$$

$$\text{GT Cov.} = \left( N(r \rightarrow gt) + N(gt \rightarrow r) \right) / (|S_r| + |S_{gt}|)$$

where $d_S(v)$ denotes the distance between $v$ and the closest point to $S$ using Euclidean distance, $|\cdot|$ denotes the number of elements of the set, and $\varepsilon > 0$. The precision measures how close the returned surface matches to the ground truth surface. The recall defined above measures how close the ground truth matches to the surface. The F-measure provides a single quantity summarizing both precision and recall. GT-Cov. is another metric summarizing both the precision and recall. All quantities are between 0 and 1 (higher is more accurate). The precision and recall are similar to accuracy and completeness for closed surfaces in evaluating stereo reconstruction algorithms [46]. We similarly define precision $P_S$, recall $R_S$ and F-measure for $\partial S_r$ and $\partial S_{gt}$ using the same formulas but with the surfaces replaced with their boundaries. We set $\varepsilon = 3$ to account for inaccuracies in the human annotation.

5.3 Evaluation

5.3.1 Synthetic Data: Surface Extraction Given Boundary

We first evaluate three methods for surface extraction given a 3D-boundary curve of the surface, discrete-minimal surface computed with linear programming (LP) [8], discrete-minimal surface approximated with Minimum-Cost Network Flow (MCNF) [8], [21], [23], [24], and our surface extraction, described in Section 4. We use Gurobi’s state-of-the-art linear programming implementation, to implement LP. We use the Lemon library [47] to implement MCNF. There are no other methods that solve this problem. We choose $\phi$ to be the image. All methods are provided the ground truth 3D boundary curves. We evaluate the methods in terms of computational time, and in terms of surface accuracy. A summary of results are provided in Table 1. Average of results over all the images are provided. Our method is computationally faster than all other methods at all resolutions. LP is unable to perform in a reasonable time frame for image sizes above 100$^3$, and MCNF is unable to perform for image sizes above 500$^3$. At all resolutions, our method is faster. Speeds are reported on a single Pentium 2.3 GHz processor. The accuracy of our method is also the highest on all measures, but all have similar accuracies. The advantage of our method is clearly speed, and ability to deal with high resolution images. Note that the analysis was not extended to the real datasets as they have high resolution, making it too computationally expensive to test, and downsampling the images destroys the structures to be extracted.

| Method   | Time | $F$ | GT-Cov. | $P$ | $R$ |
|----------|------|-----|---------|-----|-----|
| LP       | >24hr| NA  | NA      | NA  | NA  |
| LP       | 35614| 0.94±0.01 | 0.92±0.01 | 0.94±0.01 | 0.93±0.01 |
| Surfcut  | 2227 | 0.97±0.01 | 0.96±0.01 | 0.97±0.01 | 0.94±0.01 |

| Method   | Time | $F$ | GT-Cov. | $P$ | $R$ |
|----------|------|-----|---------|-----|-----|
| LP       | >24hr| NA  | NA      | NA  | NA  |
| LP       | 2227 | 0.97±0.01 | 0.96±0.01 | 0.98±0.01 | 0.95±0.02 |

Table 1: Comparison of methods for surface extraction given the surface boundary on the synthetic dataset. Speed (in seconds), surface precision (P), recall (R), F-measure (F), and ground truth covering (GT-cov) are reported. Higher P, R, F, GT-Cov. indicate better fidelity to the ground truth.
there is no other general algorithm that extracts both the boundary of the free-surface and the surface given a seed point. Therefore, we compare our method in an interactive setting and automated setting (with seed points automatically initialized) to Crease Surfaces [25]. It computes the smoothed Hessian of $\phi$, and computes a modified matrix based on the relative difference in the first and second highest eigenvalues. It then forms the surface by determining locations where the eigenvector aligns with the gradient, and constructs connected surfaces. In an interactive setting, we choose the surface returned by [25] that is near to the user provided seed point (and best fits ground truth) to provide comparison to our method. In an automated setting, we use a seed point extraction algorithm (described later) to initialize our surface extraction.

We choose $\phi(x)$ to be the semblance measure in [9]; this along with [25] is state-of-the-art for seismic data.

**Robustness to Smoothing Degradations:** The semblance $\phi$ contains a smoothing parameter, which must be tuned to achieve a desirable segmentation. Therefore, it is important that the surface extraction algorithm be robust to changes in the parameter of the likelihood. Thus, we evaluate our algorithm as we vary the smoothing parameter. The smoothing parameter is varied from $\sigma = 0, 2, 3, \ldots, 14$. We initialize our algorithm with a user specified seed point. Quantitative results are shown in Figure 13 where we plot the F-measure versus the smoothing amount both in terms of surface and boundary measures. Some visual results of the surfaces are shown in Fig. 14. Notice our method degrades only gradually and maintains consistently high accuracy in both measures in contrast to [25].

**Robustness to Noise:** In applications, the image may be distorted by noise (this is the case in seismic images where the SNR may be low), and thus we evaluate our algorithm as we add noise to the image, and we fix the smoothing parameter of the semblance $\phi(x)$ to the one with highest F-measure in the previous experiment. We choose noise levels as follows: $\sigma^2 = 0, 0.05, \ldots, 0.5$. Quantitative results are shown in Figure 15, and some visualizations of the surfaces are shown in Fig. 16. Results show that our method consistently returns an accurate result in both measures, and degrades only slightly.

**Slice-wise Validation:** We now show some visual validation of our method by showing that the surface intersects with slices of the image in locations where there is a fault, and thus the value of $\phi$ is low. This is shown in Figure 17.

**Robustness to Seed-Point Location:** We demonstrate that our surface extraction method is robust to the choice of the seed point location. To this end, we randomly sample 30 points (with high local likelihood) from the ground truth surface. We use each of the points as seed points to initialize our algorithm. We measure the boundary and surface accuracy for each of the extracted surfaces. Results are displayed in Figure 18. They show our algorithms consistently returns a boundary and surface of similar accuracy regardless of the seed point location.

**Analysis of Automated Algorithm:** Even though our contribution is in the surface and boundary extraction from a seed point, we show with a seed point initialization, our method can be automated. We initialize our algorithm with a simple automated detection of seeds points. We extract seed points by finding extrema of the Hessian and then running a piece-wise planar segmentation of these points using RANSAC [48] successively; the point on each of the segments located closest to other points on the segment are seed points. This operates under the assumption that the
surfaces are roughly planar. If not, there could possibly be redundant seed points on the same surface, which would result in repetitions in surfaces in our final output. This could easily be filtered out. We run our boundary curve extraction followed by surface extraction for each of the seed points on the original datasets. We compare to [25]. There are 6 ground truth surfaces in this dataset. Our algorithm correctly extracts 6 surfaces, while [25] extracts 4 surfaces (2 pairs of faults are merged together each as a single connected component). Results on a dataset are visualized in Figure 19 (each connected component in different color). Notice that Crease Surfaces has holes, captures clutter, and connects separate faults.

Computational Cost: We analyze run-times on a dataset of size $463 \times 951 \times 651$. The run-time of our algorithm depends on the size of the surface. To extract one surface, our algorithm takes on average 10 minutes (9 minutes for the boundary extraction and 1 minute for the surface extraction). Automated seed point extraction takes about 3 minutes. Therefore, the total cost of our algorithm for extracting 6 faults is about 1 hour. We note that after seed point extraction, the computation of surfaces can be parallelized.

In comparison, [25] takes about Pentium 2.3 GHz processor 2 hours on the same dataset. Speeds are reported on a single

5.3.3 Lung CT Data: Surface and Boundary Extraction

We now compare to Crease Surfaces for the Lung CT dataset. We compare the methods under the settings described in the previous section. For medical data, we modify the matrix based on the Hessian in Crease Surfaces to another matrix based on closeness to a plate-like structure as common in lung fissure detection [45], [49], [50]. State-of-the-art methods in fissure extraction use a method similar to Crease surfaces to extract the surface. We choose $\phi$ to be the plate-ness measure in our method. Quantitative results on the entire dataset are summarized in Table 2. Both in terms of surface and boundary accuracy, our method is more accurate with respect to all measures. Visual validation of our method on slice-wise views of the surface and image is shown in Fig. 19. Some visualizations of the surface results are shown in Fig. 20. Various slices are shown to help visualize features of the image. Crease surface generates surfaces with incorrect holes and many times cannot capture the entire fissure, hence low recall and precision on the boundary metrics. SurfCut does not contain any holes and accurately captures very fine and thin structures near the boundaries of the fissures.

6 Conclusion

We have provided a general method for extracting a smooth simple (without holes) surface with unknown boundary in a 3D image with noisy local measurements of the surface, e.g., edges. Our novel method takes as input a single seed point, and extracts the unknown boundary that may lie in 3D. It then uses this boundary curve to determine the entire surface efficiently. We have demonstrated with extensive
TABLE 2: Quantitative Evaluation on Lung Dataset. Comparison of methods in terms of surface and boundary accuracy. Precision (P), recall (R), F-measure (F), and ground truth covering (GT-cov) are reported. Higher P, R, F, GT-Cov. indicate better fidelity to the ground truth.

| Method       | Surface accuracy | Boundary accuracy |
|--------------|------------------|-------------------|
|              | F                | GT-Cov.            | P     | R     | F    | GT-Cov. | P     | R     |
| Crease Surfaces | 0.76±0.08        | 0.70±0.10         | 0.67±0.11 | 0.91±0.06 | 0.70±0.04 | 0.87±0.06 | 0.86±0.06 | 0.95±0.02 |
| Surfcut      | 0.91±0.04        | 0.87±0.06         | 0.86±0.06 | 0.95±0.02 | 0.91±0.04 | 0.87±0.06 | 0.86±0.06 | 0.95±0.02 |

Fig. 20: Slice-wise Validation in Lung CT Dataset. [Top]: Various slices of an image of a patient. [Bottom]: Surface generated with Surfcut intersected with the slice above (green) superimposed on the slice. Notice the structure of interest is a subtle thin lines in the slices (top).

Fig. 21: Qualitative Results on Lung CT. Columns show the surfaces on the same slice on the same patient for various methods. Moving through a row shows the surface for different patients, and a slice of the image is shown for various different slices. Surfcut extracts more of the fine structure of the fissures, better estimates the boundary, and recovers more of the surfaces than Crease surfaces.

ACKNOWLEDGMENTS
This work was supported by KAUST OCRF-2014-CRG3-62140401, and the Visual Computing Center at KAUST.

REFERENCES
[1] L. D. Cohen and R. Kimmel, “Global minimum for active contour models: A minimal path approach,” International journal of computer vision, vol. 24, no. 1, pp. 57–78, 1997.
[2] J. A. Sethian, “A fast marching level set method for monotonically advancing fronts,” Proceedings of the National Academy of Sciences, vol. 93, no. 4, pp. 1591–1595, 1996.
[3] J. N. Tsitsiklis, “Efficient algorithms for globally optimal trajectories,” Automatic Control, IEEE Transactions on, vol. 40, no. 9, pp. 1528–1538, 1995.
[4] V. Kaul, A. Yezzi, and Y. Tsai, “Detecting curves with unknown endpoints and arbitrary topology using minimal paths,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 34, no. 10, pp. 1952–1965, 2012.
[5] J. Mille, S. Bougleux, and L. D. Cohen, “Combination of piecewise-geodesic paths for interactive segmentation,” International Journal of Computer Vision, vol. 112, no. 1, pp. 1–22, 2015.
[6] F. Benmansour and L. D. Cohen, “From a single point to a surface patch by growing minimal paths,” in Scale Space and Variational Methods in Computer Vision. Springer, 2009, pp. 648–659.
[7] R. Ardon, L. D. Cohen, and A. Yezzi, “A new implicit method for surface segmentation by minimal paths: Applications in 3d medical images,” in Energy Minimization Methods in Computer Vision and Pattern Recognition. Springer, 2005, pp. 520–535.
[8] L. Grady, “Minimal surfaces extend shortest path segmentation methods to 3d,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 32, no. 2, pp. 321–334, 2010.
[9] D. Hale, “Methods to compute fault images, extract fault surfaces, and estimate fault throws from 3d seismic images,” Geophysics, vol. 78, no. 2, pp. O33–O43, 2013.
[10] J. M. Sullivan, “A crystalline approximation theorem for hypersurfaces,” Princeton Ph. D, 1990.
[11] V. Caselles, R. Kimmel, and G. Sapiro, “Geodesic active contours,” International journal of computer vision, vol. 22, no. 1, pp. 61–79, 1997.
[12] A. Yezzi Jr, S. Kichenassamy, A. Kumar, P. Olver, and A. Tannenbaum, “A geometric snake model for segmentation of medical imagery,” Medical Imaging, IEEE Transactions on, vol. 16, no. 2, pp. 199–209, 1997.
[13] T. F. Chan and L. A. Vese, “Active contours without edges,” Image processing, IEEE transactions on, vol. 10, no. 2, pp. 266–277, 2001.
[14] S. Osher and J. A. Sethian, “Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations,” Journal of computational physics, vol. 79, no. 1, pp. 12–49, 1988.
[15] T. Pock, T. Schoenemann, G. Graber, H. Bischof, and D. Cremers, “A convex formulation of continuous multi-label problems,” in Computer Vision–ECCV 2008. Springer, 2008, pp. 792–805.

[16] Y. Y. Boykov and M.-P. Jolly, “Interactive graph cuts for optimal boundary & region segmentation of objects in nd images,” in Computer Vision, 2001. ICCV 2001. Proceedings. Eighth IEEE International Conference on, vol. 1. IEEE, 2001, pp. 105–112.

[17] C. Rother, V. Kolmogorov, and A. Blake, “Grabcut: Interactive foreground extraction using iterated graph cuts,” in ACM transactions on graphics (TOG), vol. 23, no. 3. ACM, 2004, pp. 309–314.

[18] J. Ulen, P. Strandmark, and F. Kahl, “Shortest paths with higher-order regularization,” Pattern Analysis and Machine Intelligence, IEEE Transactions on, vol. 37, no. 12, pp. 2588–2600, 2015.

[19] L. Grady, “Computing exact discrete minimal surfaces: Extending a new implicit method for surface segmentation by minimal paths in 3d images,” Applied Mathematics & Optimization, vol. 55, no. 2, pp. 127–144, 2007.

[20] L. R. Ford Jr and D. R. Fulkerson, “An efficient implementation of a scaling algorithm for finding minimum-cost flows,” in Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on, vol. 1. IEEE, 2006, pp. 69–78.

[21] T. Schultz, H. Theisel, and H.-P. Seidel, “Crease surfaces: From ridges and bronchi,” IEEE transactions on medical imaging, vol. 32, no. 2, pp. 210–222, 2013.

[22] C. Xiao, B. C. Stoel, M. E. Bakker, Y. Peng, J. Stolk, and M. Staring, “Pulmonary fissure detection in ct images using a derivative of stick filter,” IEEE transactions on medical imaging, vol. 35, no. 6, pp. 1488–1500, 2016.

[23] M. Algarni and G. Sundaramoorthi, “Surfcut: Free-boundary surface extraction,” in European Conference on Computer Vision. Springer, 2016, pp. 171–186.

[24] D. Eberly, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, “Ridges for image analysis,” Journal of Mathematical Imaging and Vision, vol. 4, no. 4, pp. 353–373, 1994.