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 ridge curves of the Euclidean length of minimal paths ending on a level set of the solution of the eikonal equation lie on the surface. Our method extracts these ridges and cuts them to form the surface boundary. Our surface extraction algorithm is built on the novel observation that the surface lies in a valley of the eikonal equation solution. The resulting surface is a collection of minimal paths. Using the framework of cubical complexes and Morse theory, we design algorithms to extract ridges and valleys 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], [3] that solves the eikonal equation, 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 (intensity discontinuities) in 3D images, that form surfaces, have been attempted [6], [7]. These methods apply to extracting a surface whose boundary forms a curve, possibly in 3D. We call the boundary a *free-boundary*. Extraction of surfaces with free-boundary is useful 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 require the user to provide the boundary of the surface or other laborious input.

In this paper, we use the Fast Marching (FM) algorithm and techniques from computational topology to create an algorithm for extracting the boundary of a surface from a 3D image using a single seed point, and an algorithm to extract the surface from the boundary. We use Fast Marching to "smooth" a local (possibly noisy) likelihood map of the surface in a way that is guaranteed to preserve locations of local extrema of the likelihood under the smoothing. We then extract closed curves, from the propagating front generated by FM from the seed point, containing these local extrema. This is done by extracting generalized local maxima, which we call ridges, of the Euclidean minimal path length function computed from FM. The resulting ridge curves are shown to lie on the surface of interest. This is true since the front travels fastest along the surface, generating paths with extremal length on the surface. Ridges are extracted efficiently using computational topology, guaranteeing that they are closed curves. The boundary of the surface is shown to be points on the curves with a minimum distance between such curves, and is computed with a simple graph cut. The surface is a collection of minimal paths containing the boundary and is shown to be an extrema of the distance computed by FM, which is extracted using computational topology to guarantee a simple topology. See Figure 1. Our method is applicable to any imaging modality. It extracts any simple surface with a boundary from an image of noisy local measurements (e.g., an edge map) of the surface.

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 curves from a topological construct, called the Morse-Complex, from a distance produced by Fast Marching. **2.** We introduce a new algorithm, based on extracting extrema 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 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

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

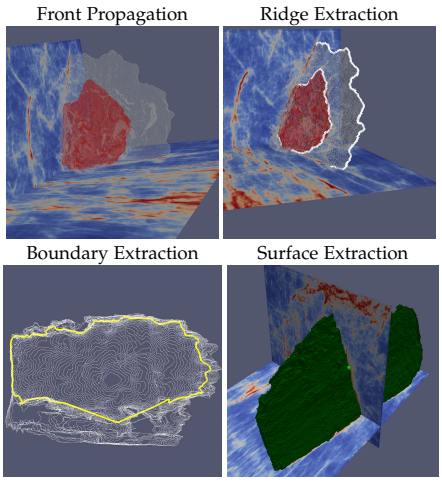

Fig. 1: **Overview of SurfCut**. [Top, left]: From a seed point on the surface, a front is propagated, [Top, right]: generalized local extrema, i.e., ridges, of a function of FM distance are extracted, [Bottom, left]: locations with minimum Euclidean distance between curves are extracted forming the boundary, [Bottom, right]: the surface is extracted as generalized local minima, i.e., valley, of the FM distance.

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 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, which 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. The 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. 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 extrema, 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 that the resulting algorithms produce structures that match the true topology of the desired structures (e.g., curve, surface or volume). Our novel algorithms use concepts from cubical complex theory. In contrast to [28], our method is designed to robustly extract topological structures of a *function* or data defined on a surface (defined by Fast Marching), rather than geometrical properties of a surface. Specifically, we extract generalized local maxima of a function derived from FM to generate curves, which are used to generate the surface boundary. We extract generalized local minima of the FM distance to extract the surface from boundary. These generalizations will be made precise in Sections 3 and 4.

Our method uses a topological construction called the *Morse complex* [32] from Morse theory to extract such generalized extrema 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, 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 [35], 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 surface extraction from images.

1.1.3 Extensions to Conference Paper

A preliminary version of this manuscript has appeared in [37]. In this version, the theoretical foundations are provided: 1) we provide analytical arguments to show that our algorithms extract the desired surface by relating it to constructions in Morse theory, and 2) we show that the surface computed from our algorithm is formed by collections of minimal paths, thus inheriting known regularity properties from such paths. We also extended our ridge extraction algorithm to better deal with extraneous structures. See [38] for more technical and details beyond this paper.

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 sub-spaces of directions rather than the whole space of possible directions. Similarly, valley points correspond to local minima of a function when restricted to sub-spaces 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_xM 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 $\nabla h(x) \in T_xM$, i.e., it is defined by the relation $dh(x) \cdot v = \nabla h(x) \cdot v$ for all $v \in T_xM$ where $dh(x) \cdot v$ is a notation that 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 [39] 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 $e_1, \ldots, e_{n-1} \in T_xM$, be eigenvalues and eigenvectors of the Hessian $Hh(x)$ at $x \in M$. 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, and has greatest curvature in this subspace. The surface in our algorithm is a valley of the Fast Marching distance, and the boundary of the surface will be obtained by computing ridges of the Euclidean minimal path distance function to the seed point, as we will show. We will extract these structures via the Morse complex, defined in the next section. The formal definitions are provided to understand why the Morse complex is computed. The curvature property in the definitions will not be needed for our algorithms, and our proofs will not show this property. Technically, these structures should be called generalized maxima and minima rather than ridges and valleys, but we use this terminology for brevity.

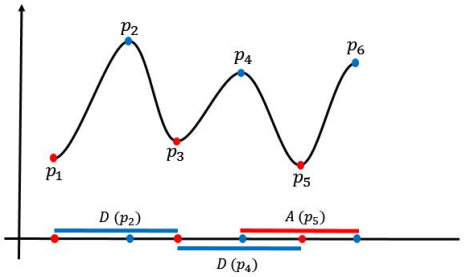

Fig. 2: Illustration of some ascending and descending manifolds of a one-dimensional function.

2.1.2 Morse Complex

Our algorithms for extracting the previous structures do not use the differential conditions above, as they are not robust to noise. We will design our algorithms based on constructions in *Morse theory* [40]. We introduce the Morse complex, and in a later section we show how the Morse complex can be used to obtain ridges and valleys. 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

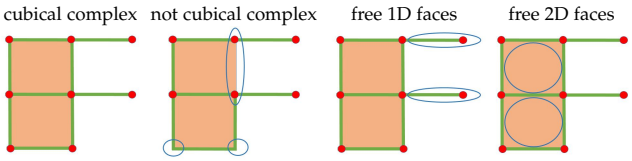

Fig. 3: [Left two images]: Illustration of faces that form a cubical complex (left) and faces that do not form a cubical complex (0,1,2-faces are marked in red, green and orange). The missing 1-face and 0-faces circled in blue on the right are not in the complex, but they are sub-faces of other faces in the set. [Right two images]: Example of 1-face, 0-face free pairs, and 2-face, 1-face free pairs (circled in blue).

visualization in the one-dimensional case. A reader familiar with the Watershed tranform would notice that the ascending manifolds are catchment basins.

The ascending manifolds of local minima decompose the manifold M into disjoint open sets. Similarly, the descending manifolds of all local maxima decomposes the manifold M into disjoint open sets. The latter decomposition forms the *Morse complex* of h, and the former is the Morse complex of $-h$.

2.2 Cubical Complexes Theory

We now introduce notions from cubical complex theory, which is the basis for our algorithms in future sections. This theory defines topological notions (and computational methods) for discrete data that are analogous to topological notions in the continuum. The notion of *free pairs*, i.e., those parts of the data that can be removed without changing topology of the data, is pertinent to our algorithms. Since the algorithms we define require the extraction of lower dimensional structures (ridge curves from surfaces, and valley surfaces from volumes), it is important that the algorithms are guaranteed to produce lower dimensional structures with correct topology. The theory of cubical complexes (e.g., [27], [28]) guarantees such lower dimensional structures are generated with homotopy equivalence to the original data.

Our data (either a curve, surface or volume) will be represented discretely by a cubical complex. A cubical complex consists of basic elements, called *faces*, of d-dimensions, e.g., points (0-faces), edges (1-faces), squares (2-faces) and cubes (3-faces). Formally, a *d-face* is the cartesian product of d intervals of the form $(a, a+1)$ where a is an integer. We can now define a cubical complex (see Fig. 3) as follows.

Definition 3. *A d-dimensional* **cubical complex** *is a finite set of faces of d-dimensions and lower such that every sub-face of a face in the set is contained in the set.*

Our algorithms consist of simplifying cubical complexes by an operation that is analogous to the continuous topological 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 shrunk 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 $f, g \subset X$.*

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 by 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:

$$\begin{cases} |\nabla U(x)| = \phi(x) & x \in \Omega \backslash \{p\} \\ U(p) = 0 \end{cases} \tag{3}$$

where ∇ 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 ϕ. 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 [41], and so a limit of smooth functions.

Following, we will make use of distance functions where ϕ is non-uniform as well as uniform. Minimal paths will refer to minimal paths from the former.

3.2 Contours on the Surface from Front Ridges

We give the intuition behind the algorithm, then this will be made precise in the propositions. 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 ϕ 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), equivalently, traveled the longest time [1], compared to nearby points, lie on the surface of interest. This is because points consistently traveling along locations where ϕ 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. The Euclidean minimal path length U_E is defined as follows. Define a front $F = \partial\{x \in \Omega : U(x) \leq D\}$ where ∂ denotes the boundary operator. The function $U_E : F \to \mathbb{R}^+$ is such that $U_E(x)$ is the Euclidean length of the minimal path (w.r.t to the weight ϕ) from x to p.

Computationally, U_E is obtained by keeping track of another function $U_E : \Omega \to \mathbb{R}^+$ 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 $\phi = 1$.

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 ϕ becomes large in this direction, thus minimal paths travel slower in this region, so they have lower Euclidean path length. Along the intersection of the surface and front, the path length may increase or decrease, depending on the uniformity of ϕ on the surface. This implies points on the intersection of the front and surface are ridge points of $U_E|F$:

Proposition 1. *Let $S \subset \Omega$ be 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 $U_E : F \to \mathbb{R}$, where $U_E(y)$ is the Euclidean length of the minimal path from y to p. We assume that locally ϕ is larger on S than points not on S (i.e., $\phi(x + \varepsilon e) \geq \phi(x)$ for all e normal to S and all ε sufficiently small).*

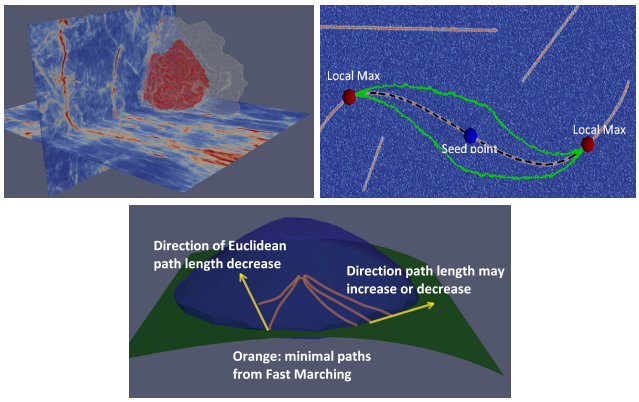

Fig. 4: [Top left]: The evolving Fast Marching (FM) front at two different time instances in orange and white. The function $1/\phi$ is the likelihood of surface, and is visualized (red - high values, and blue - low values). Ridge points of U_E, the Euclidean length of minimal paths, lie on the surface 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 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.

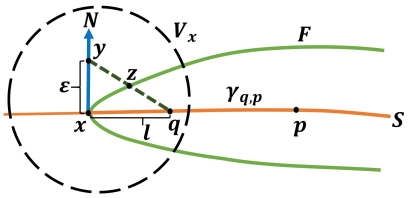

Fig. 5: Schematic of quantities in the proof of Proposition 1.

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 ϕ 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. This is an approximation by step functions valid by Weierstrass's Theorem. 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 $V_x \backslash S$ will be straight lines as ϕ 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,

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

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 [42], [43] are often used. However, that approach, while being more robust to noise, may distort the data, and it is often difficult to obtain a connected curve as the ridge. Therefore, we derive a robust method by making use of the Morse complex and cubical complex theory to extract the ridge of interest from the data U_E. Cubical complex theory guarantees the correct topology of the desired ridge (as a 1-dimensional closed curve).

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_x sufficiently small around x, we have that $\partial A(p_1) \cap V_x$ divides V_x, i.e., $V_x = [V_x \cap A(p_1)] \cup [V_x \cap A(p_2)]$ ($p_1 \neq p_2$) for the case when V_x intersects two ascending manifolds. Note that $-\nabla h(y) \cdot N_2 > 0$ for $y \in V_x \cap A(p_2)$ where N_2 is the inward normal to $\partial A(p_2)$ when V_x 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 $y \in V_x \cap A(p_1)$. 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 $V_x \backslash \{x\}$ 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. □

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 of $U_E|F$.

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. 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 ϕ, the *function* U_E defined on the front would have a rather simple topology, indeed a volcano structure (see left image in Fig. 6), 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 1 Morse Complex Extraction

procedure MORSE COMPLEX(C_F, U_E)
  ▷ C_F = *cubical 2-complex,* U_E = *cost on 1-faces in* C_F
  id $\leftarrow 0$
  Create heap of 1-faces ordered by U_E (min at top)
  repeat
    Remove 1-face g from heap
    if g is a subset of two faces f_1 and f_2 in C_F **then**
      Remove g, f_1, f_2 from C_F
      $l(f_1) \leftarrow l(f_2) \leftarrow$ id, id $\leftarrow$ id $+ 1$
      ▷ *new id for ascending manifold; hole at local min*
    else if (g, f) is a free pair in C_F **then**
      Remove g, f from C_F
      $l(f) \leftarrow l(f_{adj})$ where $f_{adj} \supset g$ and $f_{adj} \notin C_F$
      ▷ *labels face same as adjacent face containing* g
    else if (f, g) is a free pair in C_F **then**
      Remove g, f from C_F
    else if g is isthmus **then**
      $l(g) = \{l(f_1), l(f_2)\}$ where $f_1, f_2 \supset g$
      ▷ *label is unordered list*
    end if
  until heap is empty
  return C_F, l ▷ *Ridges, labels for 2-faces, ridges*
end procedure

Algorithm 2 Highest Ridge Curve Extraction

procedure HIGHESTRIDGE(C_F, U_E)
  ▷ C_F *cubical 2-complex of Fast Marching front*
  ▷ *Euclidean trajectory length* U_E *defined on 1-faces*
  repeat
    (C'_F, l) = MORSE COMPLEX (C_F, U_E)
    Create 2-cubical complex C''_F with
      a 2-face f for each unique 2-face id in l
      a 1-face g for each unique 1-face id in l
      g joins f_1 and f_2 if $l(g) = \{l(f_1), l(f_2)\}$
    for g each 1-face in C''_F **do**
      $R = \{g' \in C'_F : l(g') = l(g)\}$ ▷ *a ridge*
      $U'_E(g) \leftarrow$ average of U_E along R
    end for
    $C_F \leftarrow C''_F, U_E \leftarrow U'_E$
  until no degree three 1-faces in C'_F
  return C'_F
end procedure

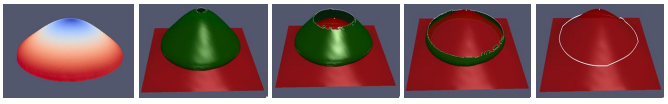

Fig. 6: [1st image]: Front color coded with Euclidean path length U_E (top view). Red indicates high values. The bottom view (not shown) is a symmetric flip. Topologically, U_E forms a volcano structure (ridge, i.e., top of volcano, is darkest red), and inside the volcano is blue. [Subsequent images]: 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).

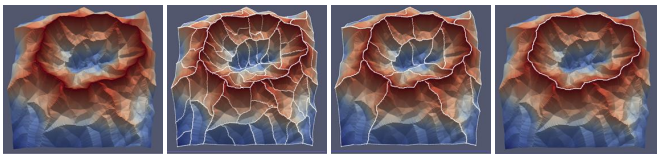

Fig. 7: Illustration of Algorithm 2 operating on noisy data to obtain the highest ridge.

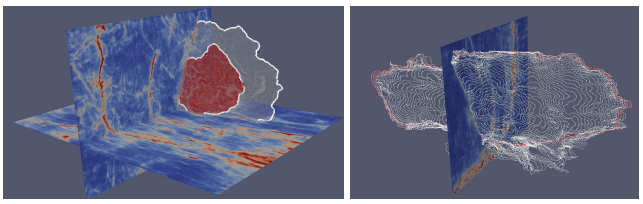

Fig. 8: [Left]: Ridge curve (white) extraction by retracting the Fast Marching front at two instants. [Right]: An example cut (red) of ridge curves, forming the surface boundary. Notice that the cut matches with the end of high $1/\phi$ (bright areas).

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.

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].

The graph G is formed as follows. Each of the curves is resampled so that all curves have the same number of nodes as the final curve. This is done to avoid the graph cut favoring a cut of small length near the seed point. Then

- 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 forms the 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 ϕ 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 increases 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 ϕ 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 $K_1 >> K_2 > 0$. 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 $y = x \pm \varepsilon N$ 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 ϕ 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 (see Figure 9). 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

$$\begin{aligned} U(y) &= LK_2 - \frac{\varepsilon}{\sqrt{1-r^2}}K_2 + \varepsilon\frac{\sqrt{2-r^2}}{\sqrt{1-r^2}}K_1 \\ &= LK_2 + \frac{\varepsilon}{\sqrt{1-r^2}}[K_1\sqrt{2-r^2} - K_2] > LK_2, \end{aligned}$$

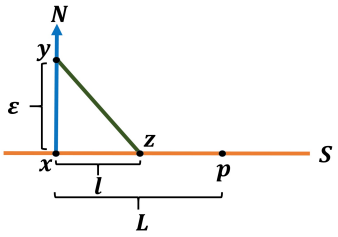

Fig. 9: Quantities defined in the proof of Proposition 3.

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 $L_t = \{U = T(t)\}$ that 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 x disconnects V_{t-}, not preserving homotopy equivalence. Therefore, $V_t = S_t \cup \{U < T(t)\}$ where S_t contains all points on ∂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 10 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 γ determined by the gradient descent of h with initial condition x lies on the valley of h containing x.*

Algorithm 3 Surface Extraction from Boundary of Surface

procedure VALLEYEXTRACT(C_I, U, ∂S)
  ▷ C_I = *cubical 3-complex of image*, U = *FM distance*
  ▷ ∂S = *boundary of surface (1-complex)*
  Create heap of 2-faces ordered by U (max at top)
  repeat
    Remove 2-face g from heap
    if (g, f) is a free pair in C_I for some f **then**
      Remove f and g from C_I
    else if (f, g) is a free pair in C_I for some f and $g \cap \partial S = \emptyset$ **then**
      Remove f and g from C_I
    end if
  until heap is empty
  return C_I ▷ *2-cubical complex of Valley*
end procedure

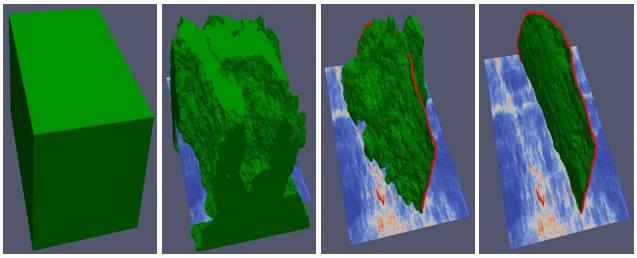

Fig. 10: Illustration of valley extraction by Algorithm 3, which retracts the volume while preserving 1-faces on the surface boundary (red). This gives the surface of interest.

Proof. For simplicity, we assume $M = \mathbb{R}^3$ and that the valley is two-dimensional. By definition of a 2D valley in $\mathbb{R}^3$, we have that $\nabla h(x) \cdot N_x = 0$ and $N_x^T Hh(x) \cdot N_x > 0$ for some unit direction $N_x \in \mathbb{R}^3$ where Hh denotes the Hessian. For every neighborhood V_x of x sufficiently small, there exists $y \in V_x$ such that $\nabla h(y) \cdot N_y = 0$ and $N_y^T Hh(y) \cdot N_y > 0$ for some N_y. If that were not the case, then x would be an isolated critical point, which is not the case. By smoothness of h, N is a smooth function. Let S be the points the satisfy the conditions on the gradient and Hessian in V_x.

We consider the path γ defined by the gradient descent of h starting from x. Then by definition of γ and Taylor expansion of h,

$$\nabla h[\gamma(\Delta t)] \approx \nabla h[x - \Delta t \nabla h(x)] \approx \nabla h(x) - \Delta t Hh(x) \cdot \nabla h(x).$$

Taking the dot product of the above with $N_{\gamma(\Delta t)} \approx N_x$, by a Taylor expansion, we have

$$\nabla h[\gamma(\Delta t)] \cdot N_{\gamma(\Delta t)} \approx -\Delta t N_x^T Hh(x) \cdot \nabla h(x).$$

Note that $N_x^T Hh(x) = \lambda N_x^T$ with $\lambda > 0$ since N_x is an eigenvector of $Hh(x)$ by definition of valley. Since $N_x^T \nabla h(x) = 0$ by the definition of valley, we have that $\nabla h[\gamma(\Delta t)] \cdot N_{\gamma(\Delta t)} \approx 0$. Also, $N_{\gamma(\Delta t)}^T Hh[\gamma(\Delta t)] \cdot N_{\gamma(\Delta t)} > 0$ as $\gamma(\Delta t) \in V_x$. Therefore, $\gamma(\Delta t)$ is also in the valley, and thus continuing this way, we can show that the path γ formed from the gradient descent is also in the valley. □

Using the last property, we can show the surface extracted by our algorithm is a collection of minimal paths to p.

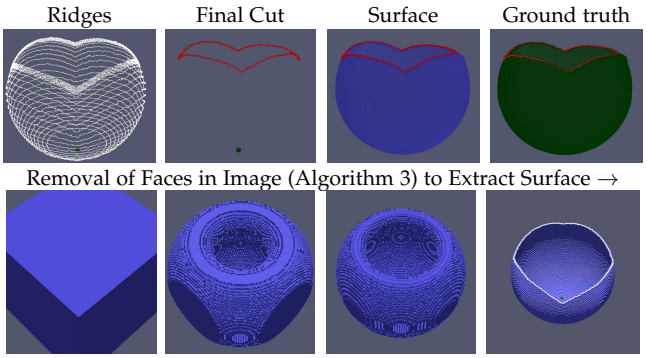

Fig. 11: Synthetic example of extracting a sphere with top cut such that the boundary is four arcs. The image (not shown) is a noisy image of the cut sphere with holes. Ridge curves are extracted via Algorithm 1 (top left). The final cut of ridge curves (top, middle left), the final surface extracted 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 γ_x formed from the gradient descent of U starting from x stays in V by Proposition 4. The path γ_x is also a minimal path since gradient paths of U are minimal paths. Note that γ_x ends at p. Therefore, we see that V is the union of γ_x over all x. □

5 Experiments

Supplementary video are available[1]. We qualitatively and quantitatively assess our method by comparing against competing algorithms.

5.1 Datasets and Parameters

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

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. 12. 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 downsampled for the lower resolution images. Noise with level $\sigma = 0.1$ is then added to the images.

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

1. https://sites.google.com/site/surfacecut/pami

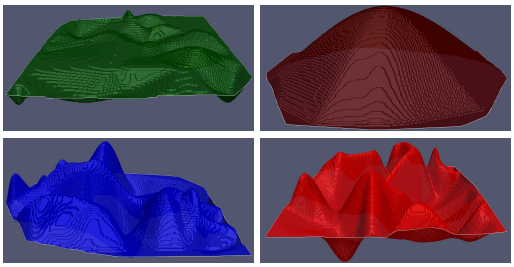

Fig. 12: Example surfaces in our synthetic dataset. Each surface has a different boundary curve, and the surfaces are of different shape, exhibiting various degrees of randomness.

noisy, and faults can be found by locating discontinuities, which is difficult due to subtle edges. Each image consists of multiple faults. We have obtained ground truth segmentations (human annotated) of two faults within each image for each slice.

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) [44]. 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., [45], [46]), 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 ϕ, 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 ∂S_r and ∂S_{gt} the respective boundaries. We define precision, recall and F-measure as

$$\begin{aligned} N(r \to gt) &= |\{v \in S_r \,:\, d_{S_{gt}}(v) < \varepsilon\}| \\ N(gt \to r) &= |\{v \in S_{gt} \,:\, d_{S_r}(v) < \varepsilon\}| \\ P_S &= N(r \to gt)/|S_r|, \\ R_S &= N(gt \to r)/|S_{gt}|, \\ F_S &= 2P_S R_S/(P_S + R_S) \\ \text{GT Cov.} &= (N(r \to gt) + N(gt \to r))/(|S_r| + |S_{gt}|) \end{aligned}$$

where $d_S(v)$ denotes the distance between v and the closet 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 [47]. We similarly define precision $P_{\partial S}$, recall $R_{\partial S}$ and F-measure for ∂S_r and ∂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.

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.

$100 \times 100 \times 100$ pixel images

Method	Time	F	GT-Cov.	P	R
LP	1167	0.93±0.01	0.94±0.01	0.91±0.02	0.96±0.01
MCNF	12.75	0.92±0.01	0.90±0.01	0.93±0.01	0.92±0.02
Surfcut	**1.87**	**0.95±0.02**	**0.95±0.02**	**0.96±0.02**	**0.94±0.03**

$500 \times 500 \times 500$ pixel images

Method	Time	F	GT-Cov.	P	R
LP	>24hr	NA	NA	NA	NA
MCNF	35614	0.94±0.01	0.92±0.01	0.94±0.01	0.93±0.01
Surfcut	**421**	**0.96±0.01**	**0.96±0.01**	**0.97±0.01**	**0.94±0.01**

$800 \times 800 \times 800$ pixel images

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

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 [48] to implement MCNF. There are no other methods that solve this problem. We choose ϕ 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 images 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.

5.3.2 *Seismic Data: Surface and Boundary Extraction*

We now compare against the competing method for free boundary surface extraction. To the best of our knowledge,

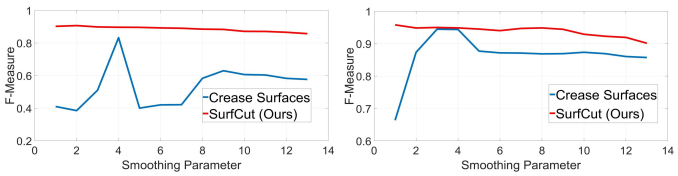

Fig. 13: **Quantitative Analysis of Smoothing Parameter** Boundary (left) and surface (right) F-measure versus smoothing degradations for our method and [25].

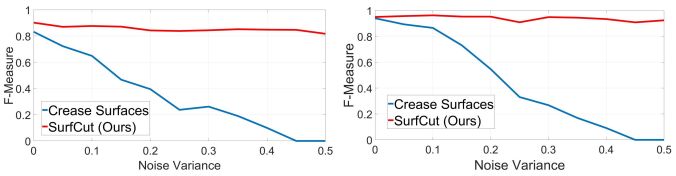

Fig. 15: **Quantitative Analysis of Noise Degradations** Boundary (left) and surface (right) F-measure versus the noise degradation plots for our method and [25].

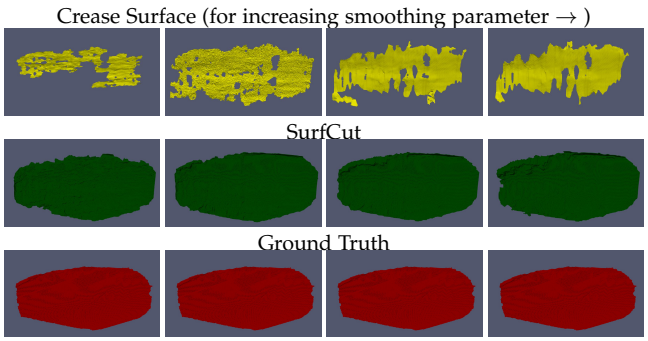

Fig. 14: **Qualitative Analysis of Smoothing Parameter**. Results displayed by varying the parameter in ϕ (larger towards the right). Surfaces extracted by Crease surfaces and our method are displayed with the ground truth.

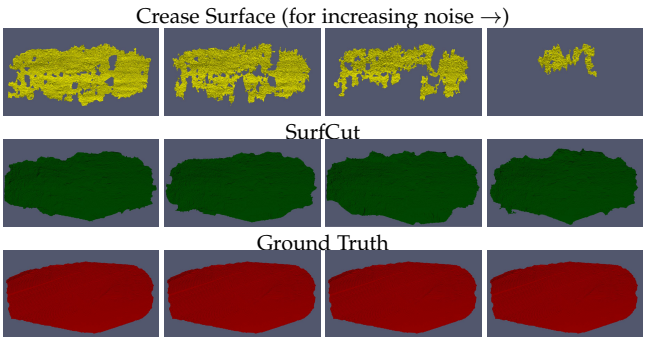

Fig. 16: **Qualitative Analysis of Noise Degradations**. Results displayed by varying the additive noise to ϕ (larger towards the right). Surfaces extracted by Crease surfaces and our method are displayed with the ground truth.

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 ϕ, 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 ϕ 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 ϕ 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 [49] 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

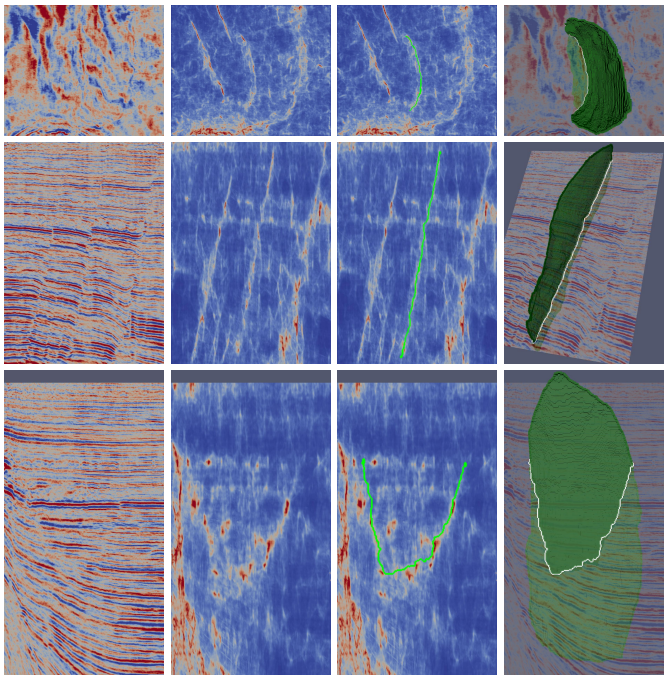

Fig. 17: **Slice-wise Validation on Seismic Data**: [Left column]: Slices corresponding to x-y, x-z, and y-z planes, [Middle, left]: Local surface likelihood ($1/\phi$), [Middle right]: Intersection of SurfCut result (green) with slice, [Right column]: surface from SurfCut at certain viewpoint.

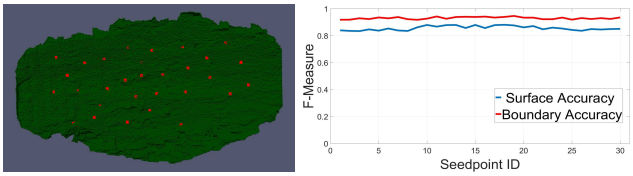

Fig. 18: **Robustness to Seed Point Choice**: [Left]: A visualization of the seed points chosen. [Right]: Boundary F-measure versus various seed point indices. The same boundary and surface accuracy is maintained no matter the seed point location.

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 2 hours on the same dataset. Speeds are reported on a single Pentium 2.3 GHz processor.

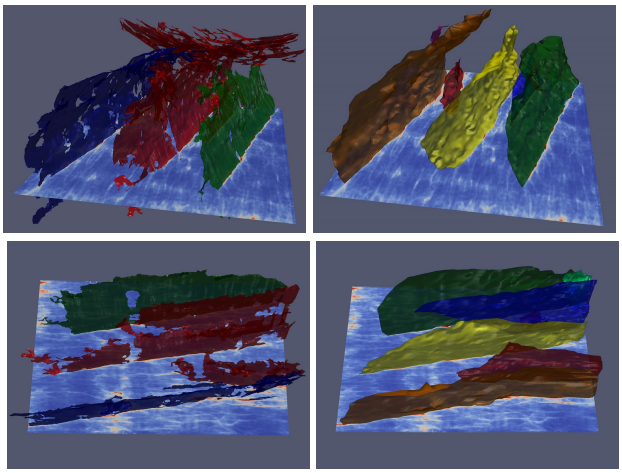

Fig. 19: Example result in an automated setting. [Left]: Result by Crease Surfaces, which contains holes and incorrectly detects clutter (top, red) due to noise in the data. [Right]: Results of SurfCut, which extracts the correct number of surfaces and produces smooth simple surfaces.

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 [46], [50], [51]. State-of-the-art methods in fissure extraction use a method similar to Crease surfaces to extract the surface. We choose ϕ 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. 20. Some visualizations of the surface results are shown in Fig. 21. 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.

Surface accuracy

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

Boundary accuracy

Method	F	GT-Cov.	P	R
Crease Surfaces	0.70±0.11	0.72±0.08	0.69±0.10	0.71±0.12
Surfcut	**0.86±0.04**	**0.86±0.06**	**0.85±0.06**	**0.87±0.05**

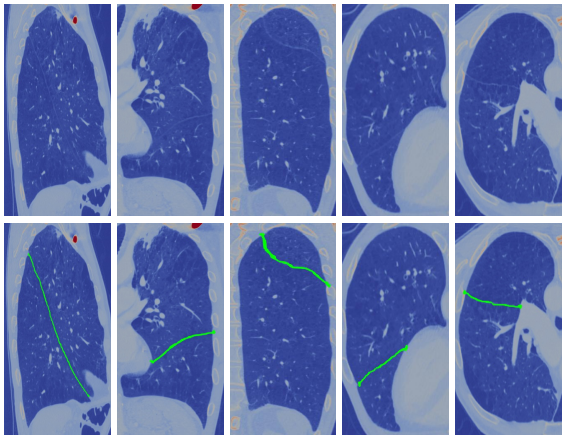

Fig. 20: **Slice-wise Validation in Lung CT Dataset**. [Top]: Various slices of an image of a patient, [Bottom]: Surface generated with SurfCut intersected with the slice above (green) superimposed on the slice. Notice the structure of interest is a subtle thin lines in the slices (top).

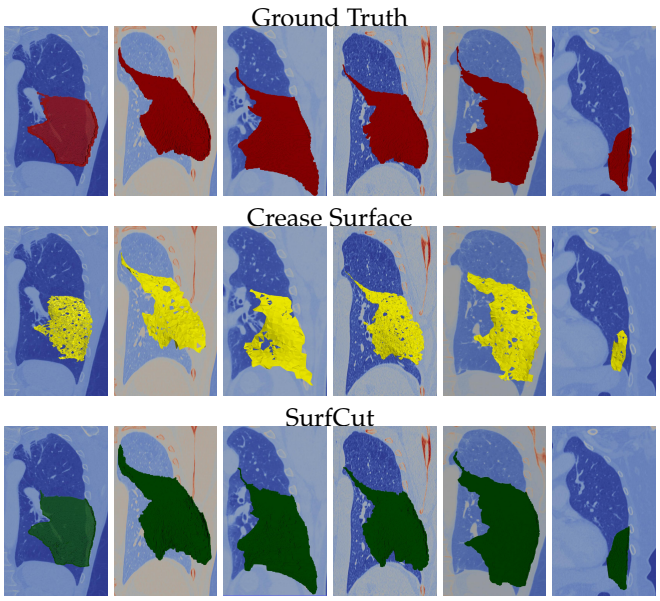

Fig. 21: **Qualitative Results on Lung CT**. Columns show the surfaces on the same slice on the same patient for various methods. Moving through a row shows the surface for different patients, and a slice of the image is shown for various different slices. SurfCut extracts more of the fine structure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

experiments on noisy and corrupted data with possible interruptions that our method accurately determines both the boundary and the surface, and it is robust to seed point choice. In comparison to extracting connected components of edges in 3D images, our method is more accurate in both surface and boundary measures. The computational cost of our algorithm is less than competing approaches.

A limitation of our method (as with competing methods) is extracting intersecting surfaces. Our boundary extraction method may extract boundaries of one or both of the surfaces depending on the data. However, if given the correct boundary of one of the surfaces, our surface extraction produces the relevant surface. This limitation is the subject of future work, which is relevant to faults in seismic images.

Acknowledgments

Funded by KAUST OCRF-2014-CRG3-62140401 and VCC.

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] R. Ardon, L. D. Cohen, and A. Yezzi, "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. Grady, "Computing exact discrete minimal surfaces: Extending and solving the shortest path problem in 3d with application to segmentation," in *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 69–78.

[21] A. V. Goldberg, "An efficient implementation of a scaling minimum-cost flow algorithm," *Journal of algorithms*, vol. 22, no. 1, pp. 1–29, 1997.

[22] P. Kovács, "Minimum-cost flow algorithms: an experimental evaluation," *Optimization Methods and Software*, vol. 30, no. 1, pp. 94–127, 2015.

[23] T. Brunsch, K. Cornelissen, B. Manthey, H. Roglin, and C. Rosner, "Smoothed analysis of the successive shortest path algorithm," *SIAM Journal on Computing*, vol. 44, no. 6, pp. 1798–1819, 2015.

[24] L. R. Ford Jr and D. R. Fulkerson, *Flows in networks*. Princeton university press, 2015.

[25] T. Schultz, H. Theisel, and H.-P. Seidel, "Crease surfaces: From theory to extraction and application to diffusion tensor mri," *Visualization and Computer Graphics, IEEE Transactions on*, vol. 16, no. 1, pp. 109–119, 2010.

[26] V. A. Kovalevsky, "Finite topology as applied to image analysis," *Computer vision, graphics, and image processing*, vol. 46, no. 2, pp. 141–161, 1989.

[27] T. Kaczynski, K. Mischaikow, and M. Mrozek, *Computational homology*. Springer Science & Business Media, 2006, vol. 157.

[28] J. Chaussard and M. Couprie, "Surface thinning in 3d cubical complexes," in *Combinatorial Image Analysis*. Springer, 2009, pp. 135–148.

[29] K. Siddiqi, A. Shokoufandeh, S. J. Dickinson, and S. W. Zucker, "Shock graphs and shape matching," *International Journal of Computer Vision*, vol. 35, no. 1, pp. 13–32, 1999.

[30] T. B. Sebastian, P. N. Klein, and B. B. Kimia, "Recognition of shapes by editing their shock graphs," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 26, no. 5, pp. 550–571, 2004.

[31] K. Siddiqi and S. Pizer, *Medial representations: mathematics, algorithms and applications*. Springer Science & Business Media, 2008, vol. 37.

[32] A. J. Zomorodian, *Topology for computing*. Cambridge university press, 2009, vol. 16.

[33] H. Edelsbrunner, J. Harer, and A. Zomorodian, "Hierarchical morse complexes for piecewise linear 2-manifolds," in *Proceedings of the seventeenth annual symposium on Computational geometry*. ACM, 2001, pp. 70–79.

[34] H. Edelsbrunner, J. Harer, V. Natarajan, and V. Pascucci, "Morse-smale complexes for piecewise linear 3-manifolds," in *Proceedings of the nineteenth annual symposium on Computational geometry*. ACM, 2003, pp. 361–370.

[35] A. Gyulassy, P.-T. Bremer, B. Hamann, and V. Pascucci, "A practical approach to morse-smale complex computation: Scalability and generality," *IEEE Transactions on Visualization and Computer Graphics*, vol. 14, no. 6, 2008.

[36] A. Gyulassy, P.-T. Bremer, and V. Pascucci, "Computing morse-smale complexes with accurate geometry," *IEEE transactions on visualization and computer graphics*, vol. 18, no. 12, pp. 2014–2022, 2012.

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

[38] M. Algarni, "Surfaces of minimal paths from topological structures and applications to 3d object segmentation," Ph.D. dissertation, KAUST, 2017.

[39] 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.

[40] J. Milnor, *Morse Theory.(AM-51)*. Princeton university press, 2016, vol. 51.

[41] M. G. Crandall and P.-L. Lions, "Viscosity solutions of hamilton-jacobi equations," *Transactions of the American Mathematical Society*, vol. 277, no. 1, pp. 1–42, 1983.

[42] T. Lindeberg, "Edge detection and ridge detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 117–156, 1998.

[43] M. Kolomenkin, I. Shimshoni, and A. Tal, "Multi-scale curve detection on surfaces," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 225–232.

[44] G. D. Hugo, E. Weiss, W. C. B. Sleeman, K. Salim, L. Paul J., Jun, and J. F. Williamson, "Data from 4d lung imaging of nsclc patients," The Cancer Imaging Archive, http://doi.org/10.7937/K9/TCIA.2016.ELN8YGLE.

[45] B. Lassen, E. M. van Rikxoort, M. Schmidt, S. Kerkstra, B. van Ginneken, and J.-M. Kuhnigk, "Automatic segmentation of the pulmonary lobes from chest ct scans based on fissures, vessels, and bronchi," *IEEE transactions on medical imaging*, vol. 32, no. 2, pp. 210–222, 2013.

[46] 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.

[47] S. M. Seitz, B. Curless, J. Diebel, D. Scharstein, and R. Szeliski, "A comparison and evaluation of multi-view stereo reconstruction algorithms," in *Computer vision and pattern recognition, 2006 IEEE Computer Society Conference on*, vol. 1. IEEE, 2006, pp. 519–528.

[48] B. Dezső, A. Jüttner, and P. Kovács, "Lemon–an open source c++ graph template library," *Electronic Notes in Theoretical Computer Science*, vol. 264, no. 5, pp. 23–45, 2011.

[49] R. B. Rusu and S. Cousins, "3d is here: Point cloud library (pcl)," in *Robotics and Automation (ICRA), 2011 IEEE International Conference on*. IEEE, 2011, pp. 1–4.

[50] R. Wiemker, T. Bülow, and T. Blaffert, "Unsupervised extraction of the pulmonary interlobar fissures from high resolution thoracic ct data," in *International Congress Series*, vol. 1281. Elsevier, 2005, pp. 1121–1126.

[51] E. M. van Rikxoort, B. van Ginneken, M. Klik, and M. Prokop, "Supervised enhancement filters: Application to fissure detection in chest ct scans," *IEEE Transactions on Medical Imaging*, vol. 27, no. 1, pp. 1–10, 2008.

Marei Algarni received the BS degree in Computer Science from KAU, Saudi Arabia, an MSc with Merit from the University of Bradford/UK, 2008. He then worked at Saudi Aramco. He completed his PhD degree in 2017 in Computer Science at KAUST (King Abdullah University of Science and Technology), and is currently a researcher at Saudi Aramco. His research interests is computer vision with interest in segmentation of 3D scientific datasets.

Ganesh Sundaramoorthi received the PhD in Electrical and Computer Engineering from Georgia Institute of Technology, Atlanta, USA. He was then a postdoctoral researcher in the Computer Science Department at the University of California, Los Angeles between 2008 and 2010. In 2011, he was appointed Assistant Professor of Electrical Engineering and Assistant Professor of Applied Mathematics and Computational Science at King Abdullah University of Science and Technology (KAUST). His research interests include computer vision and its mathematical foundations with recent interest in shape and motion analysis, video analysis, invariant representations for visual tasks, and applications in medical and scientific imaging. He was an Area Chair for ICCV 2017, and CVPR 2018.