SurfCut: Free-Boundary Surface Extraction

Marei Algarni and Ganesh Sundaramoorthi

King Abdullah University of Science & Technology (KAUST)
{marei.algarni, ganesh.sundaramoorthi}@kaust.edu.sa

Abstract. We present *SurfCut*, an algorithm for extracting a smooth simple surface with unknown boundary from a noisy 3D image and a seed point. In contrast to existing approaches that extract smooth simple surfaces with boundary, our method requires less user input, i.e., a seed point, rather than a 3D boundary curve. Our method is built on the novel observation that certain ridge curves of a front propagated using the Fast Marching algorithm are likely to lie on the surface. Using the framework of cubical complexes, we design a novel algorithm to robustly extract such ridge curves and form the surface of interest. Our algorithm automatically cuts these ridge curves to form the surface boundary, and then extracts the surface. Experiments show the robustness of our method to errors in the data, and that we achieve higher accuracy with lower computational cost than comparable methods.

Keywords: Segmentation, surface extraction, Fast Marching methods, minimal path methods, cubicle complexes

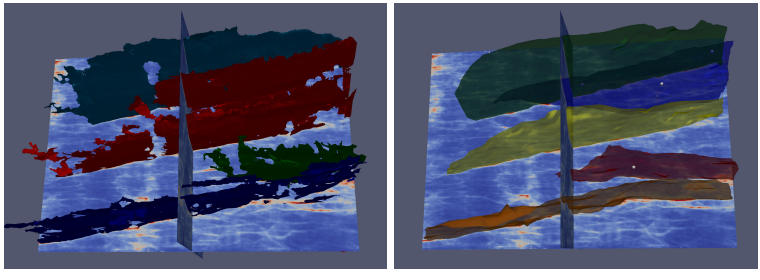

Fig. 1. SurfCut determines a surface whose boundary is a 3D curve from a noisy image. [Left]: The result of a competing method [1] contains holes and inaccurate surface boundary caused by noise. [Right]: SurfCut results in accurate surfaces without holes.

1 Introduction

Minimal path methods [2], built on the Fast Marching algorithm [3], have been widely used in computer vision. They provide a framework for extracting continuous curves from possibly noisy images. They have been used for instance 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 in various applications, including medical (e.g., the outer wall of ventricles forms a surface with boundary) [8] and scientific imaging (e.g., fault surfaces in seismic images) [9]. In [8] an alternative method to extract such surfaces, based on the theory of minimal surfaces, 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 build on Fast Marching algorithms to create an algorithm for extracting the boundary of a surface from a 3D image and a single seed point, and a corresponding algorithm to extract the surface. We validate our algorithm on seismic images for extracting fault surfaces, which form surfaces with free-boundaries. This has wide ranging applications in the oil industry [9]. Although we validate our method with such images, our method is general and can be used to extract any simple surface with boundary from an image that contains noisy local measurements (possibly from an edge map) of the surface.

The contributions of this work are: **1.** We introduce the first algorithm, to the best of our knowledge, to extract a closed space curve in 3D forming the boundary of a surface from a single seed point based on Fast Marching. **2.** We introduce a new algorithm to extract a surface given its boundary and a noisy image that produces a topologically simple surface whose boundary is the given space curve. Both curve 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

Surface Extraction: Active surface methods [10–12], based on level set methods [13], their convex counterparts [14], graph cut methods [15, 16], 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 builds on the Fast Marching (FM) Method [3]. This method propagates an initial surface (for example, a seed point) within 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 of the method 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 [2] (see also [17]). This has been used in 2D images to compute edges of images when derivatives of the image are noisy. 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. However, 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], minimal paths are used to extract a surface edge topologically equivalent to a cylinder. The user inputs the two boundary curves of the cylinder and minimal paths joining the two curves are computed conveniently using the solution of a 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 [7]. 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.

An alternative approach to obtaining a surface along image edges from its boundary is with the use of minimal surface theory [18, 8]. It is argued that minimal surfaces are more natural extensions of the 2D shortest path problem to 3D. The minimal weighted area surface interpolating the boundary is obtained by solving a linear program. The drawback of this method is that the user must input the boundary of the surface, which our method addresses, and it is computationally expensive as we show in experiments.

Another approach for surface extraction, which does not require user input, is the approach by [1]. There, a local differential operator (based on the smoothed Hessian matrix) is used to compute the likelihood of a pixel belonging to the desired surface. Then, connected components of the maxima of this likelihood are computed, obtaining several surfaces within the image. This method is convenient since it is fully automated. However, it is sensitive to noise, and estimates the boundary of the surface inaccurately, as we show in experiments. This approach has been applied to seismic images for extracting fault surfaces [9], and it is regarded as the state-of-the-art in that field.

Cubical Complexes for Thinning: Our method is a discrete algorithm and is based on the framework of cubical complexes [19]. 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 [20] to obtain skeletons (or medial representations [21]) of geometrical shapes. This is proven to be robust to noise or fine topological features. Our novel algorithms use concepts from cubical complex theory. In contrast to [20], 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.

1.2 Overview of Method

Our algorithm consists of the following steps (see Figure 2): **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 points of the Euclidean path distance of minimal paths to the seed point are computed by removing points on the front from least to greatest distance while preserving topology (Section 3.2). This results in a closed curve that lies 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 when a cost threshold is exceeded (Section 3.3). **iv) Surface Extraction**: Finally, points in the image excluding the cut curve are removed from highest to lowest based on weighted distance to the seed point while preserving topology - resulting in the desired surface (Section 3.4).

Our method requires notions of topology preservation, which we review from cubical complex theory in the next section. We then proceed to our algorithm.

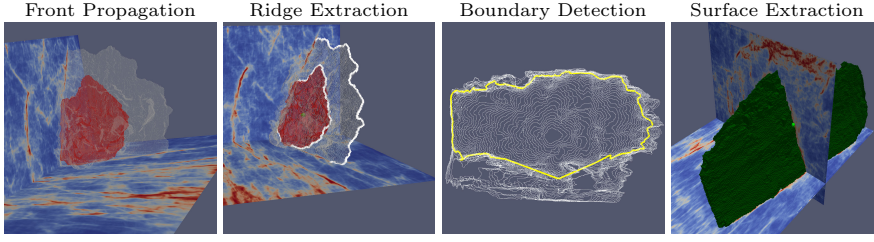

Fig. 2. Overview of SurfCut. Starting from a user specified seed point on the surface, a front is propagated (left), curves are extracted (middle left), a cut of these curves is performed forming the boundary (middle right), and the surface is extracted (right).

2 Cubical Complexes Theory

In this section, we review notions from cubical complex theory. 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 in the next sections require the extraction of lower dimensional structures (curves from surfaces, and surfaces from volumes), it is important that the algorithms are guaranteed to produce lower dimensional structures. The theory of cubicle

complexes [20] guarantees such lower dimensional structures are generated while having 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:

Definition 1 *A d-dimensional* **cubical complex** *is a finite collection of faces of d-dimensions and lower such that every sub-face of a face in the collection is contained in the collection.*

Our algorithms will consist of simplifying cubicle 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. 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 cubicle complex can be removed while preserving homotopy equivalence. *Free faces*, defined in cubical complex theory, can be removed simplifying the cubicle complex, while preserving a discrete notion of homotopy equivalence. These are defined formally as:

Definition 2 *Let X be a cubicle complex, and let $f, g \subset X$ be faces.*
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.*

The definition directly provides a constant-time operation to check whether a face is free. For example, if a cubicle 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 cubicle complexes for the evolving front produced from the Fast Marching algorithm, and retract this front by removing free faces to obtain a lower dimensional curve that lies on the surface that we wish to obtain. We also retract a volume to obtain the surface of interest.

3 Free-Boundary Surface Extraction

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

3.1 Fronts Localized to the Surface Using Fast Marching

We use the Fast Marching Method [3] to generate a collection of fronts that grow from a seed point and are localized to the surface of interest. We denote by $\phi : \mathbb{Z}_n^3 \to \mathbb{R}^+$, where $\mathbb{Z}_n = \{0, 1, \ldots, n-1\}$, a 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 $U : \mathbb{Z}_n^3 \to \mathbb{R}^+$ to the eikonal equation:

$$\begin{cases} |\nabla U(x)| = \phi(x) & x \in \mathbb{Z}_n^3 \backslash \{p\} \\ U(p) = 0 \end{cases} \tag{1}$$

where ∇ denotes the spatial gradient, and $p \in \mathbb{Z}_n^3$ 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. 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 [2], a positive constant added to the right hand side of (1) may be used to induce smoothness of paths. The front, evolving in time, moves in the outward normal direction with speed proportional to $1/\phi(x)$. Fronts can be alternatively obtained by thresholding U at the end of Fast Marching.

3.2 Retracting Fronts for Curves on the Surface

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 3). 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 tend to lie on the surface of interest. This is because points traveling along locations where ϕ is low (on the surface) travel the fastest, tracing out paths that have large arc-length.

This property can be more readily seen in the 2D case (see Figure 3): suppose that we wish to extract a curve from a seed point, and we do so by 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 the 2D case by [4]. In the 3D case (see Figure 3), we note this generalizes to *ridge points*[1] of the Euclidean path length d_E (defined

[1] A one-dimensional ridge point of a function is such that all eigenvectors of the Hessian (except one) are negative and the derivative of the function in the direction of eigenvectors corresponding to negative eigenvalues are zero [22]. Intuitively, this means a local maximum in one direction.

next) likely lie on the surface of interest. To define Euclidean path length d_E, define a front $F = \{x \in \mathbb{Z}_n^3 : U(x) \in [D, D+\varepsilon)\}$ where $\varepsilon > 0$ is small. The function $d_E : F \to \mathbb{R}^+$ is such that $d_E(x)$ is the Euclidean path length of the minimal weighted path (w.r.t to the distance U) from x to p. Note that d_E is easy to obtain by keeping track of another distance U_E that solves the eikonal equation with the right hand side of the first equation in (1) equal to 1, while propagating the front to compute U.

The fact that ridge points likely lie on the surface is visualized in the right of Figure 3: 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 since ϕ 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 ϕ on the surface. This implies points on the intersection of the front and surface are ridge points of d_E.

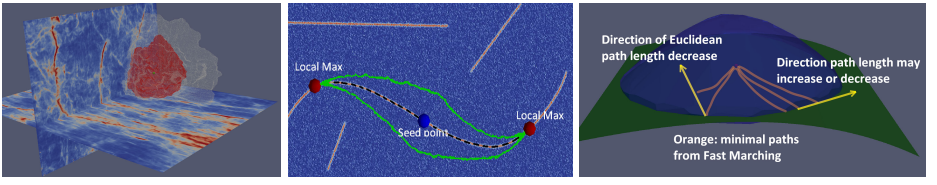

Fig. 3. [Left]: The evolving Fast Marching (FM) front at two different time instances in orange and white. The function $1/\phi$ evaluated at x is the likelihood of surface passing through x, and is visualized (red - high values, and blue - low values). Notice the fronts are localized near the surface of interest. Ridge points of d_E, the Euclidean path length of minimal weighted paths, lie on the surface of interest. [Middle]: 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. [Right]: 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 of the FM front lie on the surface.

Since ridge detection computed directly from its definition is sensitive to noise, scale spaces [23, 24] are often used. However, this 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 retracting the front to the ridge curve by an ordered removal of free faces (based on lowest to highest ordering based on d_E). The two dimensional cubicle complex C_F of the front at a time instant is constructed as follows:

- 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.

– C_F removes from C'_F face f with minimum cost and the smallest local minima with distance (determined by the seed point and f) away from f.

The last operation punctures the front at two locations (on both sides, with respect to the ridge, of the front) so that it can be retracted to the ridge curve. The ridge curve can be computed by removing free pairs until no free faces are left. This is described in Algorithm 1. We note the computational complexity of

Algorithm 1 Ridge Curve Extraction

Input: 2D cubical complex C_F of FM front and Euclidean distance U_E
Sort C_F 1-faces from min to max based on U_E
repeat
    Let g be the minimum value 1-face in C_F
    if (f, g) or (g, f) is a free pair in C_F for some f **then**
        Remove f and g from C_F
    end if
until no free pairs in C_F

this extraction is $O(N \log N)$ where N is the number of pixels.

An example of ridge curves detected is shown in Figure 4. This procedure of retracting the Fast Marching front 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 sub-section describes the stopping criteria.

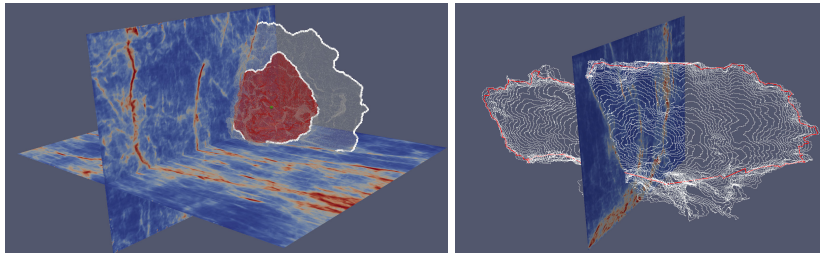

Fig. 4. [Left]: Ridge curve extraction by retracting the Fast Marching front at two instants. [Right]: An example cut (red) of ridge curves, forming the surface boundary. Notice that the cut matches with the end of high surface likelihood (bright areas).

3.3 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 [15].

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 4 shows an example of a cut that is obtained. Figure 5 shows a synthetic example.

3.4 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 cubicle complex framework and has complexity $O(N \log N)$. Although there is another algorithm, [8], for this task, it is computationally expensive as we show in Section 4.

We retract the cubicle complex of the image with the constraint that the boundary curve and faces joining to it cannot be removed. We accomplish this retraction by an ordered removal of free faces based on weighted path length U determined from Fast Marching to form the surface of interest. This results in fronts that have large distance U from the seed point being removed first. By the constraint, only the parts of the fronts that do not touch the boundary can be removed. As the removal progresses, faces are removed on either side of the surface. This creates a "wrapping" effect around the surface of interest, which have small values of U. Near the end of the algorithm, points on the surface cannot be removed without creating a hole, so no faces are free, and thus the algorithm stops. The algorithm is described in Algorithm 2. Figure 5 shows a synthetic example of the evolution of this algorithm. Figure 6 additionally shows the result of surface extracted from the data used in Figure 4.

Algorithm 2 Surface Extraction from Boundary of Surface

Input: C_I - 3D cubicle complex of image
Input: ∂S - boundary of surface (1D cubicle complex)
Input: U - weighted distance to seed point p
Sort faces of C_I based on U in decreasing order
repeat
    Let g be the 2-face with largest weight
    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 no free faces in C_I without intersection to ∂S

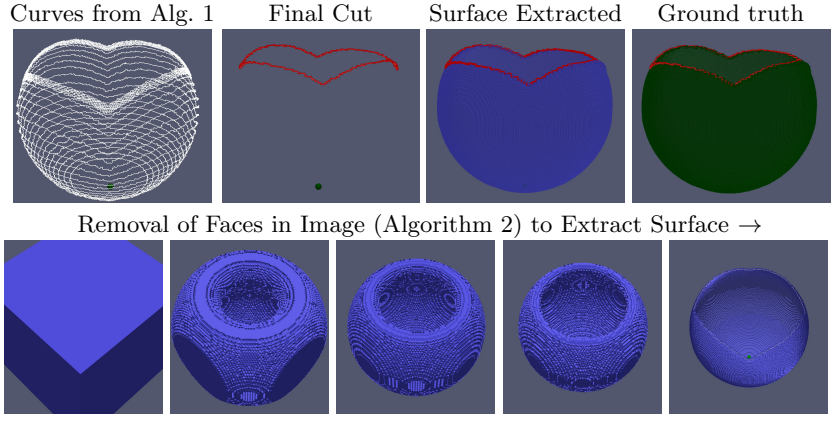

Fig. 5. Synthetic example of extracting a sphere with top cut such that the boundary is four arcs. The image (not shown) is a noisy image of the cut sphere with holes. Ridge curves are extracted via Algorithm 1 (top left). The final cut of ridge curves (top, middle left), the final surface extracted via Algorithm 2 (top, middle right), and the ground truth (top, right) are shown. Various snapshots of the removal of faces in Alg. 2 is shown (bottom), and the final result is the surface of interest.

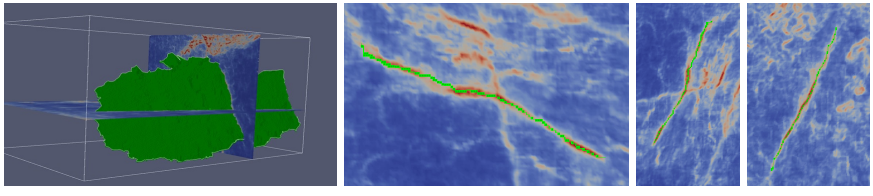

Fig. 6. [1st image]: The surface extracted using the boundary curve shown in Figure 4. [Images 2-4]: Validation on slices that intersect the surface computed with SurfCut (green) passes through locations of high likelihood of the true surface (red regions).

4 Experiments

Supplementary and executables are available[2]. We quantitatively assess our method by presenting an experimental protocol and comparing against a competing algorithm. To the best of our knowledge, there is no other algorithm that extracts both the boundary of the free-surface and the surface given a seed point. Existing methods with user interaction require user input of the surface boundary. Therefore, we compare our method in an interactive setting and automated setting (with seed points automatically initialized) to [1]. [1] returns all surfaces by detecting connected components of maxima of the local surface likelihood map. In an interactive setting, we choose the surface returned by [1] 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.

4.1 Dataset and Parameters

We evaluate our method on fault surface extraction from 3D seismic images, which are cluttered and have subtle edges. Faults are edges that form surfaces with boundary, which typically have curvature. Several faults may exist within the volume. We test on two separate datasets with image sizes $463 \times 951 \times 651$. We have obtained ground truth segmentations (human annotated) of two faults within each image. We compute $\phi(x)$ (a local edge map) by computing the eigenvalues of the smoothed Hessian and choosing $\phi(x)$ to be small when one eigenvalue is small compared to the other two, as described in [1].

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).

4.2 Evaluation Protocol

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

$$P_S = \frac{|\{v \in S_r \,:\, d_{S_{gt}}(v) < \varepsilon\}|}{|S_r|},\ R_S = \frac{|\{v \in S_{gt} \,:\, d_{S_r}(v) < \varepsilon\}|}{|S_{gt}|},\ F_S = \frac{2P_S R_S}{P_S + R_S}$$

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

[2] https://sites.google.com/site/surfacecut/

matches to the surface. The F-measure provides a single quantity summarizing both precision and recall. All quantities are between 0 and 1 (higher is more accurate). 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.

4.3 Evaluation

Robustness to Smoothing Degradations: The local surface likelihood typically contains parameters, 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 for the Hessian computation. The smoothing parameter is varied from $\sigma = 0, 2, 3, \ldots, 14$. We initialize our algorithm with a user specified seed point. Results are shown in Figure 7, where we plot the F-measure versus the smoothing amount both in terms of surface and boundary measures. Notice our method degrades only gradually and maintains consistently high accuracy in both measures in contrast to [1].

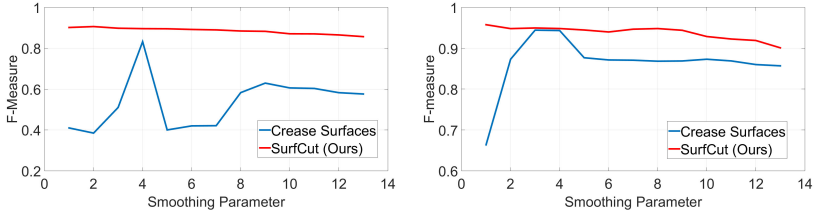

Fig. 7. Quantitative Analysis of Smoothing Degradations Boundary (left) and surface (right) F-measure versus smoothing degradations for our method and [1].

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 Hessian computation to the one with highest F-measure in the previous experiment. We choose noise levels as follows: $\sigma^2 = 0, 0.05, \ldots, 0.5$. Results are shown in Figure 8. Results show that our method consistently returns an accurate result in both measures, and degrades only slightly.

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 9. They show our algorithms consistently returns a boundary and surface of similar accuracy regardless of the seed point location.

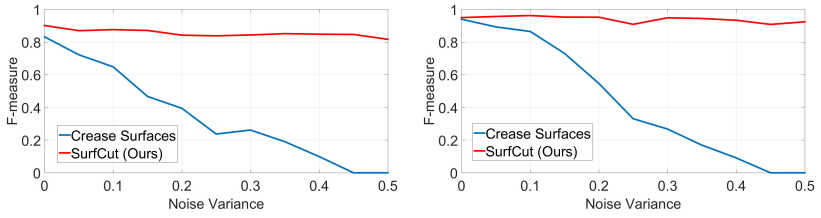

Fig. 8. Quantitative Analysis of Noise Degradations Boundary (left) and surface (right) F-measure versus the noise degradation plots for our method and [1].

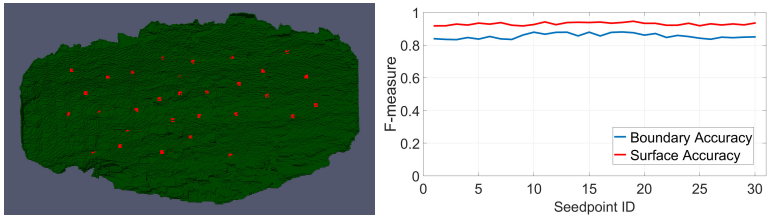

Fig. 9. Robustness to Seed Point Choice: [Left]: A visualization of the seed points chosen. [Right]: Boundary F-measure versus various seed point indices. The same boundary and surface accuracy is maintained no matter the seed point location.

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 [25] 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 [1]. There are 6 ground truth surfaces in this dataset. Our algorithm correctly extracts 6 surfaces, while [1] extracts 4 surfaces (2 pairs of faults are merged together each as a single connected component). The results are visualized in Figure 1 (each connected component in different color). Another dataset is shown in Figure 10.

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, [1]

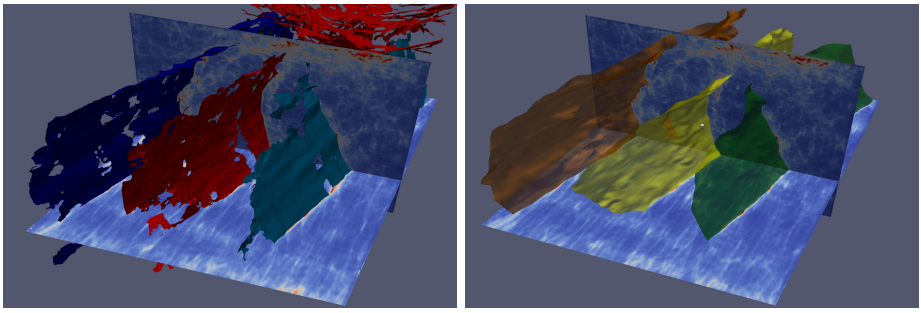

Fig. 10. Final results in an automated setting. [Left]: The final result by [1], which contains holes and detects clutter due to noise in the data. [Right]: The results of our method extracts the correct number of surfaces and produces smooth simple surfaces.

takes about 2.5 hours on the same dataset. Even though the method [8] requires manual input of the boundary curve of the surface, we state the time of [8] for surface extraction. Using Gurobi's state-of-the-art linear programming implementation, the method takes over 10 hours for a *single* surface (and the time grows drastically with increasing image sizes). Ours takes 1 minute given the boundary (both achieve similar accuracies). Our solution may not achieve the minimal surface as in [8], but it does achieve a surface with high fidelity to the surface of interest. Speeds are reported on a single Pentium 2.3 GHz processor.

5 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 experiments on noisy and corrupted data with possible interruptions that our method accurately determines both the boundary and the surface, and the method is robust to seed point choice. In comparison to an approach which extracts 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 is in extracting multiple intersecting surfaces. Our boundary extraction method may extract boundaries of one or both parts of the intersecting 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 of our boundary extraction is the subject of future work. This is important in seismic images, where surfaces can intersect.

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

References

1. Schultz, T., Theisel, H., Seidel, H.P.: Crease surfaces: From theory to extraction and application to diffusion tensor mri. Visualization and Computer Graphics, IEEE Transactions on **16**(1) (2010) 109–119
2. Cohen, L.D., Kimmel, R.: Global minimum for active contour models: A minimal path approach. International journal of computer vision **24**(1) (1997) 57–78
3. Sethian, J.A.: A fast marching level set method for monotonically advancing fronts. Proceedings of the National Academy of Sciences **93**(4) (1996) 1591–1595
4. Kaul, V., Yezzi, A., Tsai, Y.: Detecting curves with unknown endpoints and arbitrary topology using minimal paths. Pattern Analysis and Machine Intelligence, IEEE Transactions on **34**(10) (2012) 1952–1965
5. Mille, J., Bougleux, S., Cohen, L.D.: Combination of piecewise-geodesic paths for interactive segmentation. International Journal of Computer Vision **112**(1) (2015) 1–22
6. Benmansour, F., Cohen, L.D.: From a single point to a surface patch by growing minimal paths. In: Scale Space and Variational Methods in Computer Vision. Springer (2009) 648–659
7. Ardon, R., Cohen, L.D., Yezzi, A.: 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) 520–535
8. Grady, L.: Minimal surfaces extend shortest path segmentation methods to 3d. Pattern Analysis and Machine Intelligence, IEEE Transactions on **32**(2) (2010) 321–334
9. Hale, D., et al.: Fault surfaces and fault throws from 3d seismic images. In: 2012 SEG Annual Meeting, Society of Exploration Geophysicists (2012)
10. Caselles, V., Kimmel, R., Sapiro, G.: Geodesic active contours. International journal of computer vision **22**(1) (1997) 61–79
11. Yezzi Jr, A., Kichenassamy, S., Kumar, A., Olver, P., Tannenbaum, A.: A geometric snake model for segmentation of medical imagery. Medical Imaging, IEEE Transactions on **16**(2) (1997) 199–209
12. Chan, T.F., Vese, L.A.: Active contours without edges. Image processing, IEEE transactions on **10**(2) (2001) 266–277
13. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on hamilton-jacobi formulations. Journal of computational physics **79**(1) (1988) 12–49
14. Pock, T., Schoenemann, T., Graber, G., Bischof, H., Cremers, D.: A convex formulation of continuous multi-label problems. In: Computer Vision–ECCV 2008. Springer (2008) 792–805
15. Boykov, Y.Y., Jolly, M.P.: 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. Volume 1., IEEE (2001) 105–112
16. Rother, C., Kolmogorov, V., Blake, A.: Grabcut: Interactive foreground extraction using iterated graph cuts. In: ACM transactions on graphics (TOG). Volume 23., ACM (2004) 309–314
17. Ulen, J., Strandmark, P., Kahl, F.: Shortest paths with higher-order regularization. Pattern Analysis and Machine Intelligence, IEEE Transactions on **37**(12) (2015) 2588–2600

18. Grady, L.: 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. Volume 1., IEEE (2006) 69–78
19. Kovalevsky, V.A.: Finite topology as applied to image analysis. Computer vision, graphics, and image processing **46**(2) (1989) 141–161
20. Chaussard, J., Couprie, M.: Surface thinning in 3d cubical complexes. In: Combinatorial Image Analysis. Springer (2009) 135–148
21. Siddiqi, K., Pizer, S.: Medial representations: mathematics, algorithms and applications. Volume 37. Springer Science & Business Media (2008)
22. Eberly, D., Gardner, R., Morse, B., Pizer, S., Scharlach, C.: Ridges for image analysis. Journal of Mathematical Imaging and Vision **4**(4) (1994) 353–373
23. Lindeberg, T.: Edge detection and ridge detection with automatic scale selection. International Journal of Computer Vision **30**(2) (1998) 117–156
24. Kolomenkin, M., Shimshoni, I., Tal, A.: Multi-scale curve detection on surfaces. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. (2013) 225–232
25. Rusu, R.B., Cousins, S.: 3d is here: Point cloud library (pcl). In: Robotics and Automation (ICRA), 2011 IEEE International Conference on, IEEE (2011) 1–4