Edge Detection by Adaptive Splitting II. The Three-Dimensional Case

Bernardo Llanas · Sagrario Lantarón

Abstract In Llanas and Lantarón, J. Sci. Comput. **46**, 485–518 (2011) we proposed an algorithm (EDAS-d) to approximate the jump discontinuity set of functions defined on subsets of $\mathbb{R}^d$. This procedure is based on adaptive splitting of the domain of the function guided by the value of an average integral. The above study was limited to the 1D and 2D versions of the algorithm. In this paper we address the three-dimensional problem. We prove an integral inequality (in the case $d = 3$) which constitutes the basis of EDAS-3. We have performed detailed computational experiments demonstrating effective edge detection in 3D function models with different interface topologies. EDAS-1 and EDAS-2 appealing properties are extensible to the 3D case.

Keywords 3D edge detection · Jump discontinuity set · Adaptive splitting · Implicit smoothing · 3D high-resolution imagery

1 Introduction

1.1 The Edge Detection Problem

The recent development of 3D acquisition technologies has emphasized the need of techniques for the understanding of volumetric data sets in several fields. For example, 3D medical imaging try to model organs and internal structures of the human body. These models are important in image guided surgery, diagnostics, etc. [8, 10, 33]. Other application areas include the determination of the interface which separates two or more different material media [12, 30] and the enhancement of some numerical algorithms [2].

The detection of objects in volumetric data sets is a challenging problem due to the size of the data and the variability of the features of interest. Depending on the concrete analysis task the accurate detection of edges can be very important. For example, the estimation of geometric and differential properties of reconstructed object boundaries such as volume, curvature or even higher order properties requires particularly accurate edge localization algorithms. Due to its technological importance, many procedures have been proposed to determine 3D edges (interfaces). Below, we give a (non-exhaustive) classification:

- Direct Methods. The aim of these methods is to determine the edges of a determined function.
  - 1D edge detection methods. These procedures can be extended to detect the size and position of discontinuities of a multi-dimensional function by holding all but one dimension fixed and determining the edges as a function of the fixed coordinates. A practical application of this method is described in [1, 5]. The concentration edge detection algorithm [14] and the Gegenbauer reconstruction method [15] are combined to preprocess the data.
  - Spatial difference filters [37]. These methods are used for detecting edges in digital 3D-images. An image is a discretely defined function. Its domain is a set of nodes in a regular grid. The local function of this type of filters contains the calculation of the difference between density values of an input image. Most 3D filters can be synthesized from 2D filters. The 3D difference filters were first reported in [22]. In some cases the computation cost of these methods may become excessive.
  - Polynomial fitting method [3, 4]. This method is based on a local polynomial annihilation property on a set of irregularly distributed points in a bounded domain of $\mathbb{R}^d$.
  - Deformable models. These methods consider a geometric object surrounding the interest area (active contour) depending on several parameters. They define an energy functional which consists of the sum of an internal and an external energy. When the functional in minimized, the internal energy constraints the shape of the active contour and the external energy locates the contour in desired image features.

    Deformable models can be broadly classified into explicit or implicit models.
    - Explicit methods (snakes) were introduced by Kass et al. [20] and generalized to 3D by Terzopoulos et al. [36]. They represent the object by means of meshes (curves, surfaces or solids) which shape evolves in the minimization of energy process. However, since the model topology is created before the deformations, explicit models are not generally able to segment complex shapes with genus higher than 0. In order to overcome this limitation, several methods have been developed in past years: T-snakes [26], active volume methods [7, 32], etc.
    - Implicit methods (level set formulation) In the late 1990s, Sethian et al. [31] proposed a new framework called level set. The basic idea was to embed the contour evolution into iso-value curves of a function with higher dimensionality. Such functions were called level set functions. This kind of representation of contours and surfaces provides topological flexibility in the segmentation process so that the topological changes are automatically handled. However, it makes difficult the user interaction and increases the computational cost. A recent formulation of this method is given in [27].

    Deformable models present some inconveniences such as sensibility to the initial contours, stopping at local minima of the energy, slowness, and computational cost.
- Indirect Methods. The aim of these methods is to divide the domain of the function into a collection of homogeneous subsets (segmentation). Edges appear as the boundaries of these subsets. These methods are less efficient than direct methods. Examples of these

procedures include region-based algorithms [21, 42] and registration based segmentation [35].

1.2 An Adaptive Splitting Algorithm for 3D Edge Detection

Adaptive splitting procedures have been used in approximation theory (see, for example [25], and the references therein). Adaptive meshing methods for edge detection have been studied mainly in the context of image processing (discretely defined functions). In image multiscale analysis, adaptive (nonuniform) grids for the finite element method applied to the Perona-Malik equation [29] with the modification suggested in [11] were studied in [6] (3D case). In image segmentation and representation (see [23] for a recent account), many algorithms are based on adaptive approaches. Most adaptive methods produce an image segmentation using a variant of the split and merge method [18, 41].

An adaptive splitting approximation algorithm was proposed in [25]. In [24] it was proved (for $d = 1$ and $d = 2$) that the average integral used in its splitting criterion is an effective jump detector. These results imply that the algorithm is divergent for functions with jump discontinuities, but they allow us to modify it to obtain an efficient edge detection method.

In this paper we extend the above results to the three-dimensional case. We study the capability of EDAS-3 to detect 3D edges. The parameters of the original approximation algorithm have a new additional meaning. The local error gives the detection threshold. The stopping criterion gives the precision of the obtained jump points. We introduce new parameters to optimize the search for jumps and the numerical computations.

EDAS-3 is an algorithm to detect jump discontinuities of functions defined by data in the physical space. The domain of the function is supposed to be the difference of a convex set and a set of Lebesgue measure zero (continuously defined functions). Its output consists of an approximation of the jump discontinuity set of the function (see Sect. 2). It is based on an integral inequality fulfilled by the sets containing a jump discontinuity.

The algorithm builds a piecewise affine function which provides a rough approximation of f away from the discontinuities. In the neighbourhood of an edge, this function is a good approximant. The integrals whose evaluation is necessary to obtain the piecewise affine function are computed by numerical methods which are exact for polynomials of a certain degree. This provides an "implicit smoothing" of the target function, that accelerates the convergence away from the discontinuities and makes the algorithm robust in the presence of noise.

The choice of the local error is guided by the magnitude of the searched jumps rather than by the degree of approximation away from the discontinuities. This allows to obtain good results with a poor approximation (homogeneity property) when the magnitude of the jumps is large enough.

When we apply EDAS-3 to determine the edges of a 3D image (discrete function), we extend it to a function defined on a cube containing the discrete domain (see Sect. 4.4). The resulting function is piecewise constant (each voxel is represented by a cube and the function is constant on it). EDAS-3 does not try to segmentate the image, because it is not concerned about its accurate approximation. EDAS-3 determines a set of tetrahedra containing jump discontinuity points. The approximate result is a flexible set of tetrahedra instead of a rigid set of surfaces. The numerical cubature process is faster than operations involving large amounts of voxels. The average integral criterion, the implicit smoothing and a suitable choice of the minimum magnitude of jump reported can reduce the distortion due to the noise of the image.

The search procedure of EDAS-3 and its flexible way of representing interfaces allow to handle automatically topological changes. In addition, EDAS-3 overcomes some of the limitations of deformable models:

Initial conditions (contours), are not necessary.
It is not a variational approach. This avoids the problem of stopping at local minima of the energy.
It does not need to use complicated data structures. The implicit smoothing implies a relatively fast processing.

Finally, in the case of complex problems, we can partition the domain and apply EDAS-3 independently to each subdomain. This allows parallel processing.

This paper is organized as follows. Section 2 gives some mathematical preliminary results and states the main result of the article. In Sect. 3 we recall the EDAS-d algorithm and detail its implementation for $d = 3$. Section 4 presents 3D computational experiments with synthetic and real models. Section 5 provides some concluding remarks. The paper is closed with an Appendix containing the proof of Theorem 1.

2 Mathematical Preliminaries

Let $R \subset \mathbb{R}^d$ be a compact d-interval. We say that a function $g : R \to \mathbb{R}$ is *quasi-continuous* if the set of points where g is not continuous has zero Lebesgue measure.

Consider a finite collection $\{C_i\}_{i=1}^n$ of connected sets with pairwise disjoint nonempty interiors such that

$$R = \bigcup_{i=1}^{n} C_i .$$

Let $\{f_i : i = 1, \ldots, n\}$ be a set of continuous functions on R. Define the function

$$f(\mathbf{x}) \equiv f_i(\mathbf{x}) \quad \text{if} \ \ \mathbf{x} \in \overset{\circ}{C}_i,$$

where $\overset{\circ}{C}$ denotes the interior of C. We say that f is a *general piecewise continuous function*. If $C_i,\ \ i = 1, \ldots, n$, are closed and convex sets, we say that f is a *piecewise continuous function*. Since the boundary of a convex set has zero Lebesgue measure, all piecewise continuous functions are quasi-continuous.

In the above definition, if the functions $f_i,\ \ i = 1, \ldots, n$, are constant, we say that f is a *general piecewise constant function* and a *piecewise constant function*, respectively.

Let f be a general piecewise continuous function and let Γ_i be the boundary of C_i, $i = 1, \ldots, n$. Define $\Gamma \equiv \bigcup_{i=1}^n \Gamma_i$. Let $\mathbf{x} \in \Gamma$, then for some $m \leq n$, $\mathbf{x} \in \Gamma_{i_j}$, $j = 1, \ldots, m$, where $i_j \in \{1, 2, \ldots, n\}$. Define

$$A \equiv \max_{j=1,\ldots,m} \{f_{i_j}(\mathbf{x})\}, \quad \text{and} \quad B \equiv \min_{j=1,\ldots,m} \{f_{i_j}(\mathbf{x})\}.$$

If $A \neq B$ we say that f has a *jump discontinuity* at $\mathbf{x}$ (we also say that $\mathbf{x}$ is a *jump point*). We call $|A - B|$ magnitude of the *jump* of f at $\mathbf{x}$. The set of points in Γ with jump discontinuities is called *jump discontinuity set* and denoted by Γ^J. We call *edge* any subset of the jump discontinuity set. The set of points in Γ^J with magnitude of jump greater than l is denoted by Γ_l^J. The set of points in Γ^J with magnitude of jump greater than l and lower than u is denoted by Γ_{lu}^J. Our aim is to obtain a good approximation of these sets for a given function. We use the term *interface* as an abbreviation for jump discontinuity set.

The *d-simplex* T generated by the set of affinely independent vectors $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$, denoted by $\langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$, is defined to be the convex hull of the vectors $\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d$. We denote by $|T|$ the diameter of a d-simplex T.

Consider a function $f : T \subset \mathbb{R}^d \to \mathbb{R}$. We denote by $L_T f$, the unique affine map such that

$$L_T f(\mathbf{v}_j) = f(\mathbf{v}_j), \quad j = 0, 1, \ldots, d.$$

In this section we study the average integral

$$AI_T(f) \equiv \frac{\int_T |f(\mathbf{x}) - L_T f(\mathbf{x})| d\mathbf{x}}{\mathrm{v}(T)},$$

where $\mathrm{v}(T)$ denotes the Lebesgue measure of T.

As we prove below, the behavior of $AI_T(f)$ depends on the continuity or lack of continuity of f on T. This fact lays the foundation for the algorithm proposed in the next section. The comportment of $AI_T(f)$ when f is continuous is given by the following result.

Proposition 1 ([13]) *Let $S \subset \mathbb{R}^d$ be compact and let $f : S \to \mathbb{R}$ be continuous. Then given $\epsilon > 0$, there exists $\delta > 0$ such that*

$$|f(\mathbf{x}) - L_T f(\mathbf{x})| < \epsilon$$

for all $\mathbf{x} \in T$, where T is a d-simplex contained into S with $|T| < \delta$.

Corollary 1 *Let $S \subset \mathbb{R}^d$ be compact and let $f : S \to \mathbb{R}$ be continuous. Then given $\epsilon > 0$, we have that $AI_T(f) < \epsilon$ for any small enough d-simplex $T \subset S$.*

If f is smooth the above result can be sharpened, see [24].

From the above result, we can conclude, in the case of continuous functions, that $AI_T(f) \to 0$ as $|T|$ approaches zero. We study below the case in that f presents jump discontinuities on T, when $d = 3$.

From now on, we limit ourselves to the case where $f(x)$ is a function of Heaviside type. This suffices to study image processing applications and functions with smooth Γ^J.

Consider the Heaviside function

$$H(x) \equiv \begin{cases} 0, & \text{if } x \le 0, \\ 1, & \text{if } x > 0. \end{cases}$$

We have

Theorem 1 *Let $h : \mathbb{R}^3 \to \mathbb{R}$ be the function defined by*

$$h(\mathbf{x}) \equiv JH(ax_1 + bx_2 + cx_3 + d),$$

where a, b, c, d, and J are real numbers such that either a or b or c are distinct from zero and $J > 0$. Define $r \equiv \{\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3 : ax_1 + bx_2 + cx_3 + d = 0\}$ and let T be an *arbitrary tetrahedron in* $\mathbb{R}^3$ *such that* $r \cap \mathring{T} \neq \emptyset$. *Then*

$$\frac{7J}{32} \leq \frac{\int_T |h(\mathbf{x}) - L_T h(\mathbf{x})| d\mathbf{x}}{\mathrm{v}(T)} \leq \frac{3J}{4}, \tag{1}$$

where $\mathrm{v}(T)$ *is the volume of* T.

The proof of this theorem is given in the Appendix.

3 Edge Detection by Adaptive Splitting Algorithm

3.1 Statement of the Algorithm

We state the algorithm in the d-dimensional case. Consider the d-interval $R = [a, b]$, where $a, b \in \mathbb{R}^d$. We can find n (closed) simplices T_i such that

$$\mathring{T}_i \cap \mathring{T}_j = \varnothing, \quad i \neq j,$$
$$\bigcup_{i=1}^{n} T_i = R,$$

(for example, see [9, 28]). We call $P \equiv \{T_i\}_{i=1}^n$ a partition of R. The set of partitions of R is denoted by $\mathcal{P}(R)$. Given a d-simplex T we denote by $\mathcal{V}(T)$ the set of its vertices.

The proposed algorithm builds a partition $P \in \mathcal{P}(R)$ and the associated piecewise affine approximant $L(\mathbf{x})$ defined by

$$L(\mathbf{x}) \equiv L_T f(\mathbf{x}) \text{ if } \mathbf{x} \in T \subset P.$$

The average integral

$$AI_T(f) \equiv \frac{\int_T |f(\mathbf{x}) - L_T f(\mathbf{x})| d\mathbf{x}}{\mathrm{v}(T)},$$

can be considered as the local error of the approximant L.

Given a function $f : R \to \mathbb{R}$ and an initial partition P_1 of R, denote by E_1 the maximum local error of the approximant L. In the cases where Theorem 1 can be applied, E_1 also provides a detection threshold, because the algorithm will detect the jumps with magnitude greater than $32E_1/7$. Denote by E_2 the approximation error of the points in Γ^J. Let E_4 be a positive real parameter.

If $(AI_T(f) \leq E_1$ or $|T| \leq E_2)$ and $|T| \leq E_4$ we call T a *good* simplex, otherwise T is called *bad*.

We call E_4 exploration parameter (in difficult problems, it is necessary to make E_4 small. This is due to the fact that an inaccurate numerical evaluation of $AI_T(f)$ can eliminate simplices containing points of Γ^J).

Edge Detection by Adaptive Splitting Algorithm (EDAS-d)

Step 1. The good simplices in the initial partition P_1 are put into the set $\mathcal{G}_1$ and the bad simplices are put into the set $\mathcal{B}_1$.

Step 2. At each step j we have a set of good simplices $\mathcal{G}_j$ and a set of bad simplices $\mathcal{B}_j$. Divide each bad simplex into two simplices, by splitting its largest edge. Test whether these children are good or bad to obtain the sets $\mathcal{G}_{j+1}$ and $\mathcal{B}_{j+1}$.

Step 3. The algorithm stops if $\mathcal{B}_j = \varnothing$. Then $G = \mathcal{G}_j$ is the searched partition. If the stopping criterion is not satisfied, go to Step 2.

Step 4. Obtain the following subset of G

$$A\Gamma^J_{E_3} \equiv \{T \in G : AI_T(f) > E_1 \text{ and } j^a \equiv \max_{v_i \in \mathcal{V}(T)} \{f(v_i)\} - \min_{v_i \in \mathcal{V}(T)} \{f(v_i)\} > E_3\}.$$

where E_3 is the minimum magnitude of jump reported. E_2 is also a stopping criterion. The convergence of this algorithm is guaranteed by the definition of good simplex. $A\Gamma^J_{E_3}$ is a set of simplices containing points of $\Gamma^J_{E_3}$. This constitutes an approximation of $\Gamma^J_{E_3}$. In the cases considered by Theorem 1, we have that $A\Gamma^J_{E_3} \supset \Gamma^J_{E_3}$. In Sect. 4, instead of describing $A\Gamma^J_{E_3}$, we have considered the set of barycenters of $T \in A\Gamma^J_{E_3}$ to better visualize the result.

3.2 Practical Implementation of EDAS-3

In practice we have implemented the above algorithm using a tree whose leaves correspond to good simplices. Tree construction follows a technique similar to that detailed in [25]. The integrals $\int_T |f(\mathbf{x}) - L_T f(\mathbf{x})| d\mathbf{x}$ have been computed using the Grundmann and Möller's formula [17].

We have considered a positive real parameter E_5 (adaptivity of cubature formulas parameter). This factor allows to apply higher degree cubature formulas in large regions and low degree cubature formulas in small regions. In this way the computational cost is optimized. The adaptive cubature procedure is described below.

– If $|T| < E_5$, the computations have been done using the Grundmann and Möller's rules of degree $p = 5, 7, 9, 11, 13$, and 15 with $d = 3$. Due to the lack of error estimates for the Grundmann and Möller's rules, we have followed the usual practice of comparing successive values of the integral corresponding to an increasing number of points and taking the value obtained when the sequence becomes stationary. We denote by $\widehat{AI}^p_T(f)$ the value of $AI_T(f)$ computed by the cubature rule of degree p. The pseudocode is described by the following statements.

```
**Set** $p = 5$;
**Compute** $\widehat{AI}^5_T(f)$ and $\widehat{AI}^7_T(f)$;
**while** ($|\widehat{AI}^p_T(f) - \widehat{AI}^{p+2}_T(f)| \geq E_1/10$ && $p < 13$)
{
    $p = p + 2$;
}
```

```
if $(|\widehat{AI}_T^p(f) - \widehat{AI}_T^{p+2}(f)| < E_1/10)$      $\widehat{AI}_T(f) = \widehat{AI}_T^{p+2}(f)$;
else                                                               $\widehat{AI}_T(f) = 10^6$;
```

- If $|T| \geq E_5$, the computations have been done using the Grundmann and Möller's rules of degree $p = 13$ and 15 with $d = 3$. The corresponding pseudocode is listed below.

```
if $(|\widehat{AI}_T^{13}(f) - \widehat{AI}_T^{15}(f)| < E_1/10)$      $\widehat{AI}_T(f) = \widehat{AI}_T^{15}$;
else                                                              $\widehat{AI}_T(f) = 10^6$;
```

To sum up, the algorithm requires the following positive real parameters

- E_1: Maximum local error of the approximant L and detection threshold.
- E_2: Approximation error of the points in Γ^J and stopping criterion.
- E_3: Minimum magnitude of jump reported.
- E_4: Exploration parameter.
- E_5: Adaptivity of cubature formulas parameter.

3.3 Theoretical Foundation of EDAS-3

Results in Sect. 2 guarantee the convergence away from the discontinuities and the suitable local behavior of EDAS-3. We can consider three cases:

- Piecewise constant functions whose Γ^J consists of a set of planes. Theorem 1 can be applied to most tetrahedra in a partition. 3D images are an example of this type of functions (Sect. 4.4).
- General piecewise constant functions with curved Γ^J. If the interface is smooth enough, then Theorem 1 is approximately valid when the tetrahedra become very small (Sects. 4.1, 4.2 and 4.3).
- General piecewise continuous functions. In this case, Theorem 1 is not directly applicable. In spite of this, EDAS-3 has given good results in some particular cases (Sect. 4.5).

4 Computational Experiments

The experimental environment has been the following:

- CPU QuadCore Intel Core 2 Quad 6700, 2666 MHz.
- RAM 4GB.
- Running software Microsoft Visual C.

We have considered different functions $f : R \to \mathbb{R}$, where R is a cube in $\mathbb{R}^3$. Each face of R has been divided into four triangles by means of its diagonals. The vertices of these triangles and the center of the cube yield a division of R into 24 disjoint tetrahedra. This resulting set has been used as the initial partition P_1 of EDAS-3.

Given a function and the parameters $\{E_i, i = 1, 5\}$, we use the following point-based representation of the approximated jump discontinuity set to show the results

$$PA\Gamma_{E_3}^J \equiv \{x \in \mathbb{R}^3 : x \text{ is the barycenter of some } T \in A\Gamma_{E_3}^J\}.$$

Let l, u be real numbers such that $l < u$. Define

$$[|l, u|] \equiv \{x = (x_1, x_2, x_3) \in PA\Gamma_{E_3}^J : l \leq x_3 \leq u\}.$$

Table 1 Dependence of E_1 on J

J	E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
$5. \times 10^{-1}$	$1. \times 10^{-1}$	0.1	2.5×10^{-1}	0.5	10.0	154704	33984	4.2
$2. \times 10^{-1}$	$4. \times 10^{-2}$	0.1	1.0×10^{-1}	0.5	10.0	154704	33984	4.2
$1. \times 10^{-1}$	$2. \times 10^{-2}$	0.1	5.0×10^{-1}	0.5	10.0	154704	33984	4.2
$5. \times 10^{-2}$	$1. \times 10^{-2}$	0.1	2.0×10^{-2}	0.5	10.0	154704	33984	4.2
$1. \times 10^{-2}$	$2. \times 10^{-3}$	0.1	5.0×10^{-3}	0.5	10.0	154704	33984	4.2
$1. \times 10^{-3}$	$2. \times 10^{-4}$	0.1	5.0×10^{-4}	0.5	10.0	154704	33984	4.2

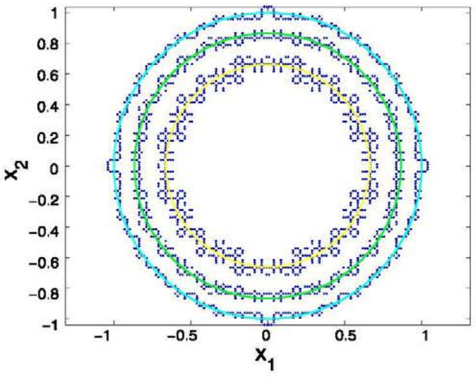

Fig. 1 (Color online) Performance of EDAS-3 ($J = 10^{-3}$): Slice $\langle -0.03, 0.03\rangle$, exact solution for $x_3 = 0$ (*light blue line*); slice $\langle 0.47, 0.53\rangle$, exact solution for $x_3 = 0.5$ (*green line*); slice $\langle 0.72, 0.78\rangle$, exact solution for $x_3 = 0.75$ (*yellow line*)

The slice $\langle l, u\rangle$ is defined as the projection of $[|l, u|]$ on the plane $x_1 - x_2$. All the accompanying graphs show $\langle l, u\rangle$ for different values of l and u. We have represented them as a cloud of dark blue points. The intersections of Γ^J and planes perpendicular to the x_3-axis (exact solutions) have been drawn as continuous colored lines.

Numerical data tables include the total number of tetrahedra generated by EDAS-3 (TNT), and the number of tetrahedra containing jump points inside them ($TNJT$).

In all experiments, CPU time is expressed in seconds and denoted by CPU (s).

4.1 General Properties of EDAS-3

In this section we study experimentally the following general properties of EDAS-3:

- Dependence of the parameter E_1 on the magnitude of the jump (J).
- Dependence of the precision on the parameter E_2.
- Robustness against continuous and discontinuous random perturbations.

To study the dependence of the parameter E_1 on J, we have considered the general piecewise constant function with spherical interface

$$f_1(x_1, x_2, x_3) = \begin{cases} 1, & \text{if } (x_1, x_2, x_3) \in B_1, \\ 1 + J, & \text{if } (x_1, x_2, x_3) \in R - B_1, \end{cases}$$

where $R = [-4, 4]^3$ and $B_1 \equiv \{(x_1, x_2, x_3) : x_1^2 + x_2^2 + x_3^2 \leq 1\}$. We have determined the threshold value of E_1 (to obtain acceptable results) corresponding to different values of J. The results are reported in Table 1. Figure 1 shows a comparison between the results obtained with EDAS-3 and the exact ones (the graph is the same for all values of J in Table 1).

Table 2 Dependence of the precision on the parameter E_2

J	E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
0.5	0.1	0.10	0.25	0.5	10.0	154704	33984	4.2
0.5	0.1	0.05	0.25	0.5	10.0	325824	134592	12.3
0.5	0.1	0.01	0.25	0.5	10.0	5782896	3269856	272.2

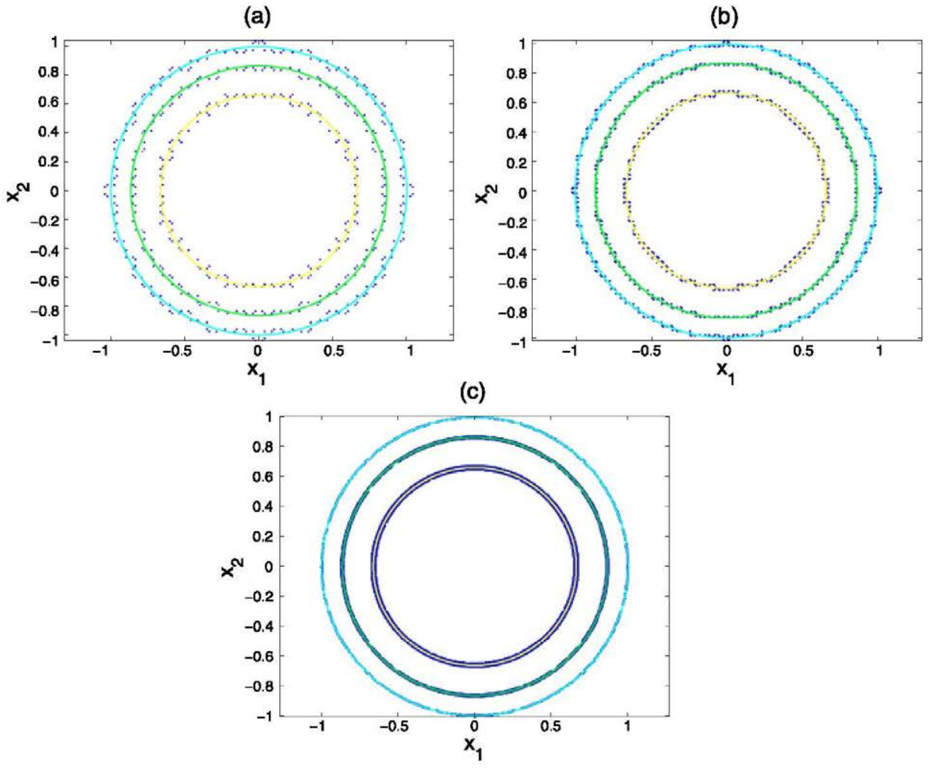

Fig. 2 Precision versus E_2. Slice $\langle -0.01, 0.01\rangle$, exact solution for $x_3 = 0$ (*light blue line*); slice $\langle 0.49, 0.51\rangle$, exact solution for $x_3 = 0.5$ (*green line*); slice $\langle 0.74, 0.76\rangle$, exact solution for $x_3 = 0.75$ (*yellow line*): (**a**) $E_2 = 0.1$, (**b**) $E_2 = 0.05$, (**c**) $E_2 = 0.01$

Since $7/32 \simeq 0.22$, Table 1 shows a good agreement between the experimental results and those predicted by Theorem 1.

To find the relationship between the precision and the parameter E_2, we have considered the function f_1 with $J = 0.5$ and $E_1 = 10^{-1}$. We have varied E_2. The results are given in Table 2. Figure 2 shows how the precision of EDAS-3 increases as E_2 decreases.

To study the effect of continuous noise on the performance of EDAS-3, we have used the continuously perturbed function

$$f_1^p(x_1, x_2, x_3) \equiv \begin{cases} 1 + \omega \sin(100(x_1 + x_2 + x_3)), & \text{if } (x_1, x_2, x_3) \in B_1, \\ 1.5 + \omega \sin(100(x_1 + x_2 + x_3)), & \text{if } (x_1, x_2, x_3) \in R - B_1. \end{cases}$$

Table 3 Performance of EDAS-3 on the continuously perturbed function

ω	E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
0.00	0.05	0.1	0.25	0.5	10.0	155616	34896	4.5
0.05	0.05	0.1	0.20	0.5	10.0	11343840	33906	1042.9
0.10	0.05	0.1	0.20	0.5	10.0	24149916	34326	2543.3

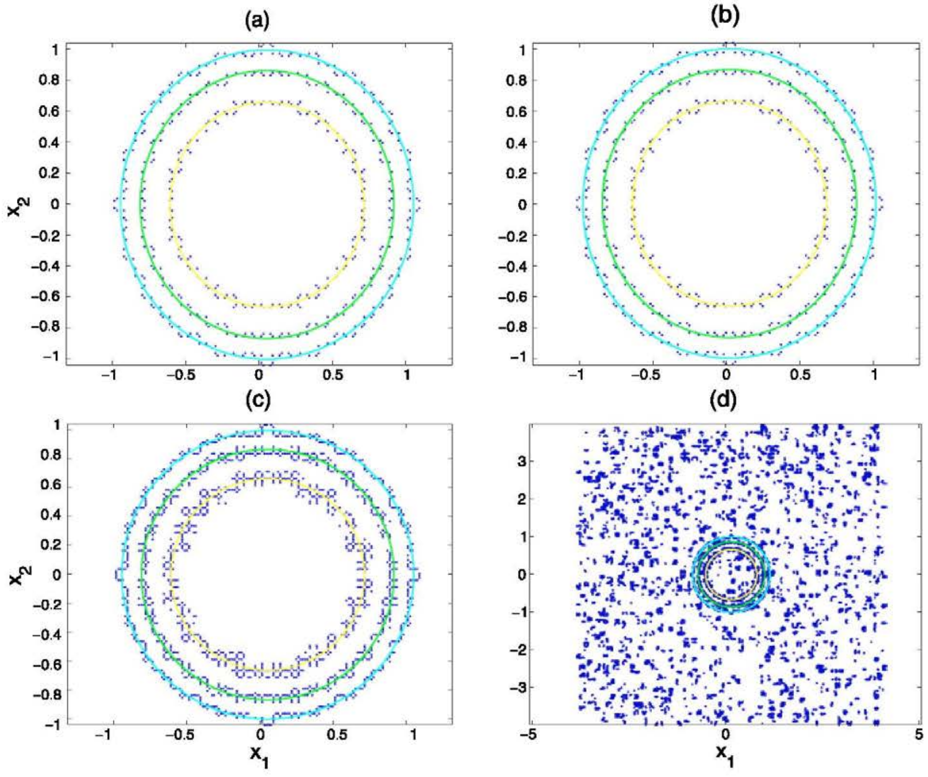

Fig. 3 Performance of EDAS-3 on the perturbed f_1 (the endpoints of the slices are the same as those in Fig. 2): (**a**) f_1^p, $E1 = 0.05$, $\omega = 0$, (**b**) f_1^p, $E1 = 0.05$, $\omega = 0.1$, (**c**) f_1^{rp}, $E1 = 0.1$, $\omega = 0.4$, (**d**) f_1^{rp}, $E1 = 0.1$, $\omega = 0.5$

The results obtained with several values of ω are reported in Table 3. Figure 3(b) shows that a noise with magnitude 20 percent of the jump, does not change the result obtained with $\omega = 0$; see Fig. 3(a).

Tables 2 and 3 show an increase in the number of tetrahedra that is due to different reasons. In Table 2 we increase the precision of EDAS-3 by decreasing the value of the parameter E_2. As a result, a large increase of tetrahedra arises around the discontinuity surface; see Fig. 13 in [24]. In Table 3 the increase of ω makes f_1^p difficult to approximate by a piecewise affine function away from the discontinuities. In this case the increase of tetrahedra takes place on the whole domain because the approximant requires many tetrahedra to approximate f_1^p.

Table 4 Performance of EDAS-3 on the randomly perturbed function

ω	E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
0.10	0.1	0.1	0.45	0.5	10.0	661030	27237	95.1
0.20	0.1	0.1	0.45	0.5	10.0	2538203	26220	403.7
0.30	0.1	0.1	0.45	0.5	10.0	5946536	26221	976.4
0.40	0.1	0.1	0.45	0.5	10.0	9866541	26839	1652.9
0.50	0.1	0.1	0.45	0.5	10.0	13071176	485582	2220.4

To study the effect of discontinuous random noise on the performance of EDAS-3, we have used the randomly perturbed function $f_1^{rp} \equiv f_1 + \omega\ rp$, where

$$rp(x_1,x_2,x_3) \equiv r(i_1,i_2,i_3), \quad i_j = 0,\dots,79,$$

$$i_j \equiv \begin{cases} floor(A_j), & \text{if } A_j \neq 80, \\ 79, & \text{if } A_j = 80, \end{cases}$$

$$A_j \equiv 10\cdot(4+x_j), \quad j=1,2,3.$$

r is a random matrix with 512000 entries between 0 and 1 defined in MATLAB by $r = rand[80][80][80]$. $floor(x)$ is a MATLAB function that gives the largest integer not greater than x. The resulting function rp is a piecewise constant random function.

The results obtained by EDAS-3 for several values of the parameter ω are given in Table 4. If ω is less than 0.5, thresholding ($E_3 = 0.45$) avoids the detection of noise jumps; see Fig. 3(c). If $\omega = 0.5$ the magnitude of many noise jumps equals 0.5 (jump of true edge points) and the algorithm detects them; see Fig. 3(d). Since by Theorem 1, the jumps with magnitude less than $4E_1/3$ are not detected, we can state that implicit smoothing, thresholding and the value of the parameter E_1 are the procedures of EDAS-3 to avoid noise effects.

4.2 General Piecewise Constant Function with Tubular Tree-like Interface

Blood vessels and airways of the human body form dense tubular tree-like structures. The segmentation of these structures in volumetric datasets is of vital interest for many medical applications [8, 38]. In the following example we define a function with a multiply connected interface which models a vascular/bronchial tube. We need some previous definitions

$$e_1 \equiv (x_1-25)^2/16+(x_2-25)^2/4, \qquad e_2 \equiv (x_1-25)^2+(x_2-25)^2,$$

$$e_3 \equiv (x_1-10)^2/16+(x_2-10)^2/4, \qquad e_4 \equiv (x_1-10)^2+(x_2-10)^2,$$

$$e_5 \equiv (x_1-40)^2/16+(x_2-40)^2/4, \qquad e_6 \equiv (x_1-40)^2+(x_2-40)^2,$$

$$h_1 \equiv (x_3-15)/10, \qquad h_2 \equiv (25-x_3)/10.$$

$$T_1 \equiv \{(x_1,x_2,x_3) : e_1 \leq 1 \quad \text{and} \quad x_3 \in [25,30]\},$$

$$T_2 \equiv \{(x_1,x_2,x_3) : e_2 \geq 1 \quad \text{and} \quad x_3 \in [25,30]\},$$

$$T_3 \equiv \{(x_1,x_2,x_3) : h_1e_1 + h_2e_3 \leq 1 \quad \text{and} \quad x_3 \in [15,25)\},$$

$$T_4 \equiv \{(x_1,x_2,x_3) : h_1e_2 + h_2e_4 \geq 1 \quad \text{and} \quad x_3 \in [15,25)\},$$

$$T_5 \equiv \{(x_1,x_2,x_3) : h_1e_1 + h_2e_5 \leq 1 \quad \text{and} \quad x_3 \in [15,25)\},$$

$$T_6 \equiv \{(x_1,x_2,x_3) : h_1e_2 + h_2e_6 \geq 1 \quad \text{and} \quad x_3 \in [15,25)\},$$

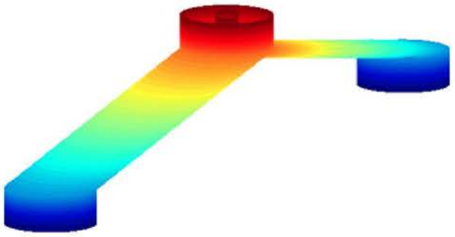
Fig. 4 Jump discontinuity set of f_2

Table 5 Performance of EDAS-3 on f_2

E_1	E_2	E_3	E_4	E_5	TNT	$TNTJ$	CPU (s)
0.1	0.1	0.5	1.0	10.0	10185592	2317886	722.8

$$T_7 \equiv \{(x_1, x_2, x_3) : e_3 \leq 1 \quad \text{and} \quad x_3 \in [10, 15)\},$$
$$T_8 \equiv \{(x_1, x_2, x_3) : e_4 \geq 1 \quad \text{and} \quad x_3 \in [10, 15)\},$$
$$T_9 \equiv \{(x_1, x_2, x_3) : e_5 \leq 1 \quad \text{and} \quad x_3 \in [10, 15)\},$$
$$T_{10} \equiv \{(x_1, x_2, x_3) : e_6 \geq 1 \quad \text{and} \quad x_3 \in [10, 15)\}.$$

Define

$$T \equiv \bigcup_{i=1}^{5} (T_{2i-1} \cap T_{2i}),$$

and

$$f_2(x_1, x_2, x_3) \equiv \begin{cases} 1, & \text{if } (x_1, x_2, x_3) \in T, \\ 2.5, & \text{if } (x_1, x_2, x_3) \in R - T, \end{cases}$$

where $R = [0, 50]^3$.

Figure 4 shows the interface (Γ^J) of f_2. The results obtained are reported in Table 5. Figure 5 shows a comparison between the slices of the cloud of points generated by EDAS-3 and the intersections of the interface with planes perpendicular to the x_3-axis.

4.3 General Piecewise Constant Function with Disconnected Interface

This experiment shows the capability of EDAS-3 for dealing with functions having interfaces with a complex topological structure.

In this case $R = [-10, 10]^3$. The equation of a torus with axis x_3, inner radius r_i and outer radius r_o is given by $(r_o - \sqrt{x_1^2 + x_2^2})^2 + x_3^2 = r_i^2$. Let $r_o = 5$ and $D(x_1, x_2, x_3) = (5 - \sqrt{x_1^2 + x_2^2})^2 + x_3^2$. Define

$$f_3(x_1, x_2, x_3) \equiv \begin{cases} 1, & \text{if } D \leq 1, \\ 3, & \text{if } D > 1 \text{ and } D \leq 4, \\ 6, & \text{if } D > 4 \text{ and } D \leq 9, \\ 10, & \text{if } D > 9 \text{ and } D \leq 16, \\ 25, & \text{otherwise.} \end{cases}$$

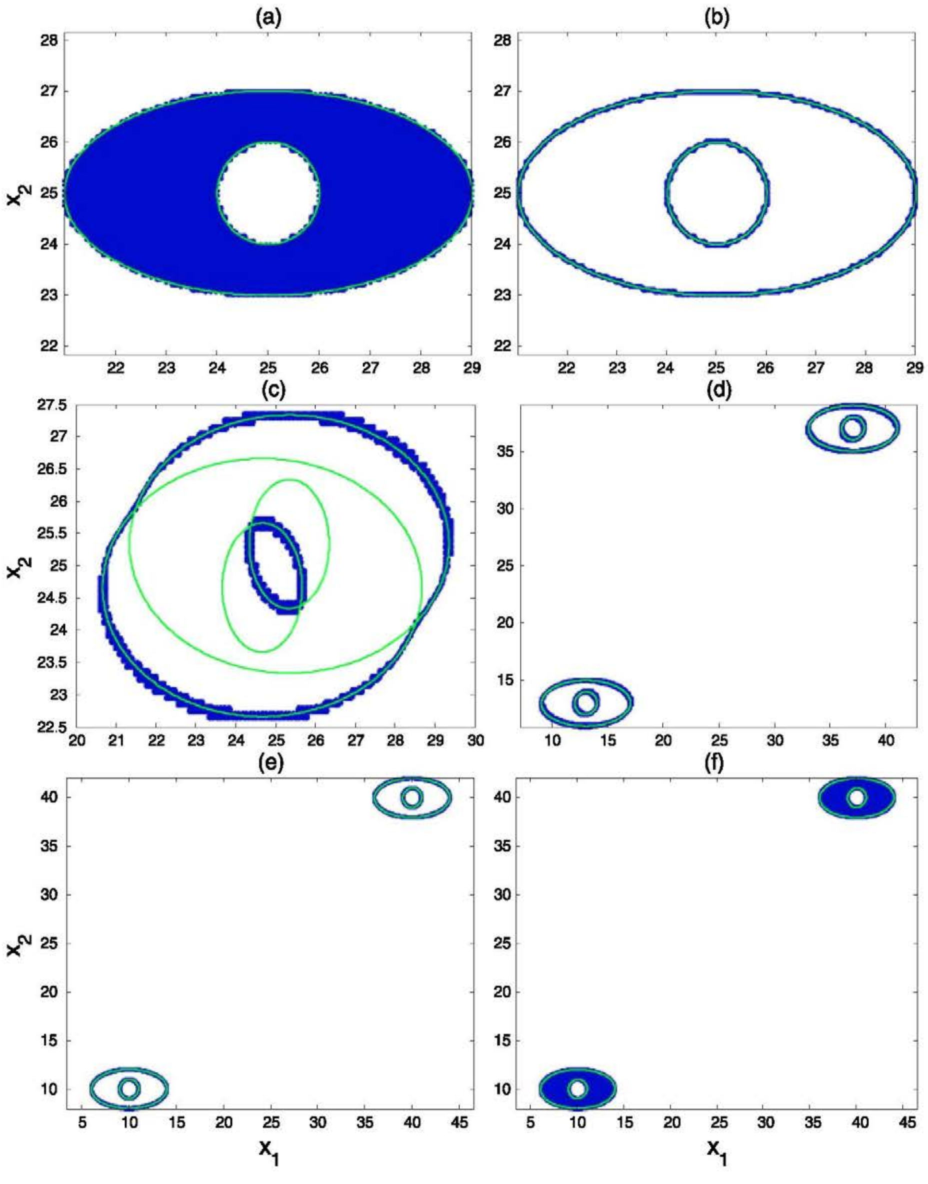

Fig. 5 (Color online) Performance of EDAS-3 on f_2. The exact solutions corresponding to $x_3 = (l + u)/2$ for each slice $\langle l, u\rangle$ are the continuous *green lines*: (**a**) slice $\langle 29.95, 30.05\rangle$, (**b**) slice $\langle 26.95, 27.05\rangle$, (**c**) slice $\langle 24.75, 24.80\rangle$ (the exact solution is the part of the *green lines* contained in the cloud of points), (**d**) slice $\langle 16.95, 17.05\rangle$, (**e**) slice $\langle 11.95, 12.05\rangle$, (**f**) slice $\langle 9.95, 10.05\rangle$

Figure 6 shows the graph of f_3. The results obtained with EDAS-3 are reported in Table 6. Figure 7 shows a comparison between the slices of the cloud of points generated by EDAS-3 and the intersections of the interface with planes perpendicular to the x_3-axis.

Fig. 6 Function f_3

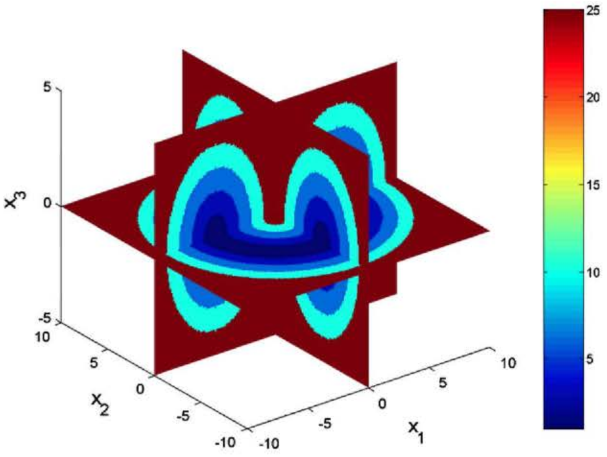

Table 6 Performance of EDAS-3 on f_3

E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
0.1	0.1	1.0	2.0	10.0	9455552	4972752	671.1

4.4 Performance on Real 3D Images and Comparison with other Methods

In this section we study the performance of EDAS-3 on 3D images. These images are discrete functions given by 3D-arrays. They can be extended to piecewise constant functions. The interface of these extended functions consists of a set of planes. Therefore, Theorem 1 can be applied to most tetrahedra in a partition. In the experiments we have used a 3D image obtained by linear interpolation $(At + (1 - t)B)$ of two CT section images (A and B) of a real brain (Radiology, Uppsala University Hospital) [19]; see Fig. 8. Interpolation from serial cross sections is a customary practice when the sections are not closely spaced [16]. In this way the appearence of the embedded 3D object is recaptured.

The test 3D image exhibits a complex structure containing several three dimensional objects. In complex problems, it may be difficult to use deformable model algorithms. Therefore, we have compared EDAS-3 with 3D difference filters (Sobel, Prewitt) which are suitable for arbitrary (unstructured) images. In the experiments with the 3D-Sobel method, the following mask to obtain the partial derivative of the image intensity with respect to the variable x_1, was adopted:

$$MX(:,:,1) = [-2\,0\,2;\ -3\,0\,3;\ -2\,0\,2],$$
$$MX(:,:,2) = [-3\,0\,3;\ -6\,0\,6;\ -3\,0\,3],$$
$$MX(:,:,3) = [-2\,0\,2;\ -3\,0\,3;\ -2\,0\,2].$$

The corresponding mask used for the 3D-Prewitt method is:

$$MX(:,:,1) = [-1\,0\,1;\ -1\,0\,1;\ -1\,0\,1],$$
$$MX(:,:,2) = [-1\,0\,1;\ -1\,0\,1;\ -1\,0\,1],$$
$$MX(:,:,3) = [-1\,0\,1;\ -1\,0\,1;\ -1\,0\,1].$$

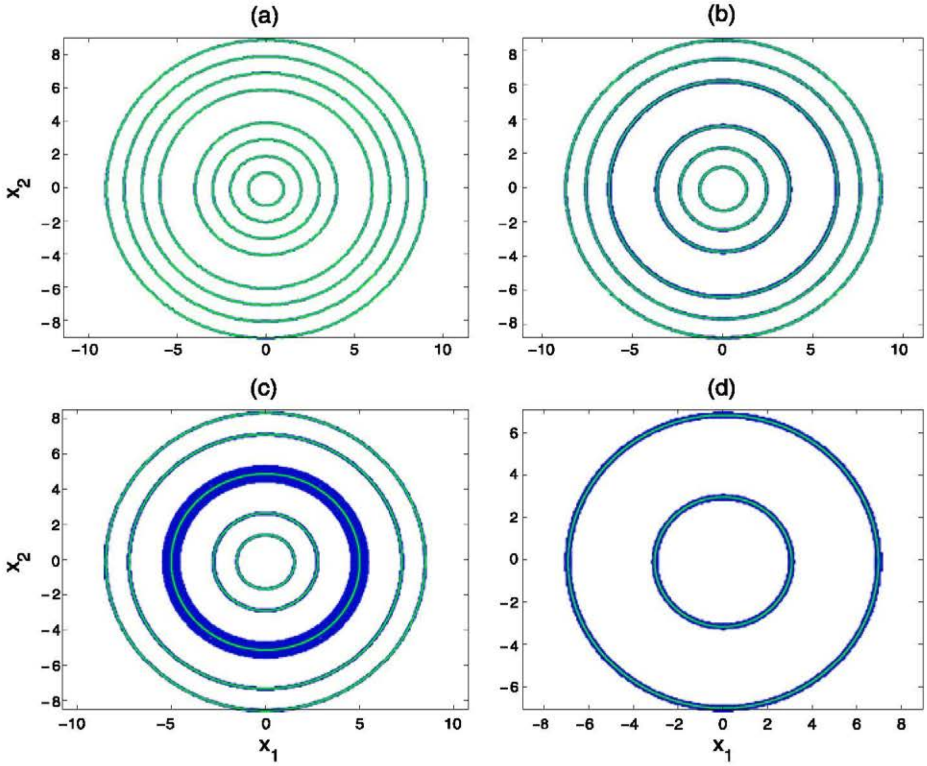

Fig. 7 Performance of EDAS-3 on f_3. The exact solutions corresponding to $x_3 = (l + u)/2$ for each slice $\langle l, u\rangle$ are the continuous *green lines*: (**a**) slice $\langle -0.05, 0.05\rangle$, (**b**) slice $\langle 1.45, 1.55\rangle$, (**c**) slice $\langle 1.95, 2.05\rangle$, (**d**) slice $\langle 3.45, 3.55\rangle$

We have used the MATLAB notation for matrices. The masks for the derivatives with respect to x_2 and x_3 can be obtained from the above ones [37, 39].

From each original 2D-image (A and B) of size 512×512 we have obtained two images of size 256×256 and 128×128 using the function *imresize* of MATLAB. The x_3 axis has been divided into h intervals. The value of the image on the x_3-interval $[k, k+1]$, $k = 0, \dots, h-1$ has been $(1 - (k/(h-1))A + (k/(h-1))B$. In this way we have generated 3D images with different resolution.

3D-Sobel and 3D-Prewitt algorithms have been programmed in MATLAB and their results are reported in Tables 7 and 8 respectively. Figure 9 shows the performance of these algorithms.

The results obtained by EDAS-3 are reported in Table 9. Figures 10(a) and 10(b) show its performance on the 3D image of size $512 \times 512 \times 21$.

We have applied EDAS-2 to the 2D images shown in Figs. 8(c) and 8(d). The results are reported in Table 10. Figures 10(c) and 10(d) show its performance on these images.

Since difference filters need to consider all the voxels, their time behavior is $O(nmh)$. This fact is confirmed by the experimental results. Therefore, difference filters are not suitable for high resolution images.

From the results obtained on 3D images of size $n \times m \times h$, we observe that EDAS-3 shows a time behavior compatible with $O((n + m + h)^\alpha)$ ($\alpha < 1$). In this case we have

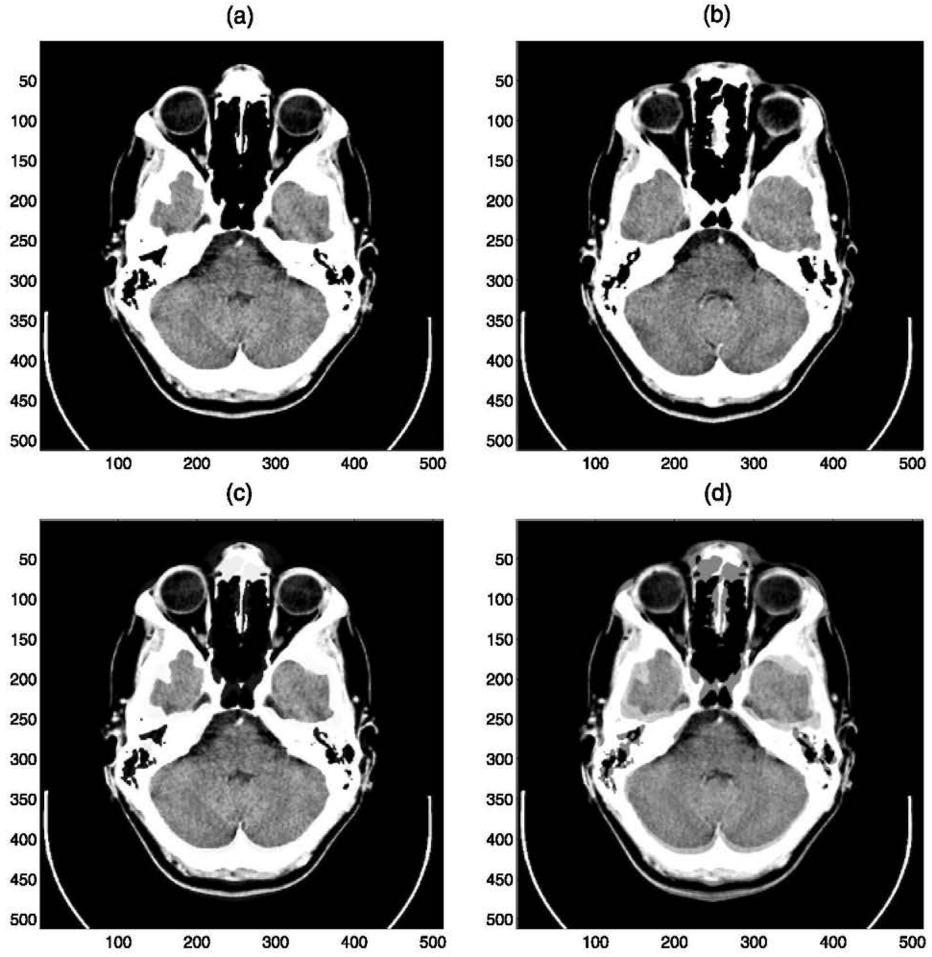

Fig. 8 3D image of size 512 × 512 × 21 obtained by linear interpolation of two CT sections of a real brain. (**a**) A (real image), (**b**) B (real image), (**c**) Interpolated image ($t = 0.9$), (**d**) Interpolated image ($t = 0.5$)

Table 7 Performance of 3D-Sobel on 3D images with different resolution

Size (Voxels)	Threshold	CPU (s)
128 × 128 × 6	1000	2.1
256 × 256 × 11	1000	20.2
512 × 512 × 21	1000	186.2

Table 8 Performance of 3D-Prewitt on 3D images with different resolution

Size (Voxels)	Threshold	CPU (s)
128 × 128 × 6	400	2.2
256 × 256 × 11	400	20.3
512 × 512 × 21	400	186.3

Table 9 Performance of EDAS-3 on 3D images with different resolution

Size (Voxels)	E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
$128 \times 128 \times 6$	0.1	0.5	55	32	10	18992772	3569601	3939.3
$256 \times 256 \times 11$	0.1	1	45	32	10	24746155	4835039	7349.6
$512 \times 512 \times 21$	0.1	2	30	32	10	25235117	6763826	14081.3

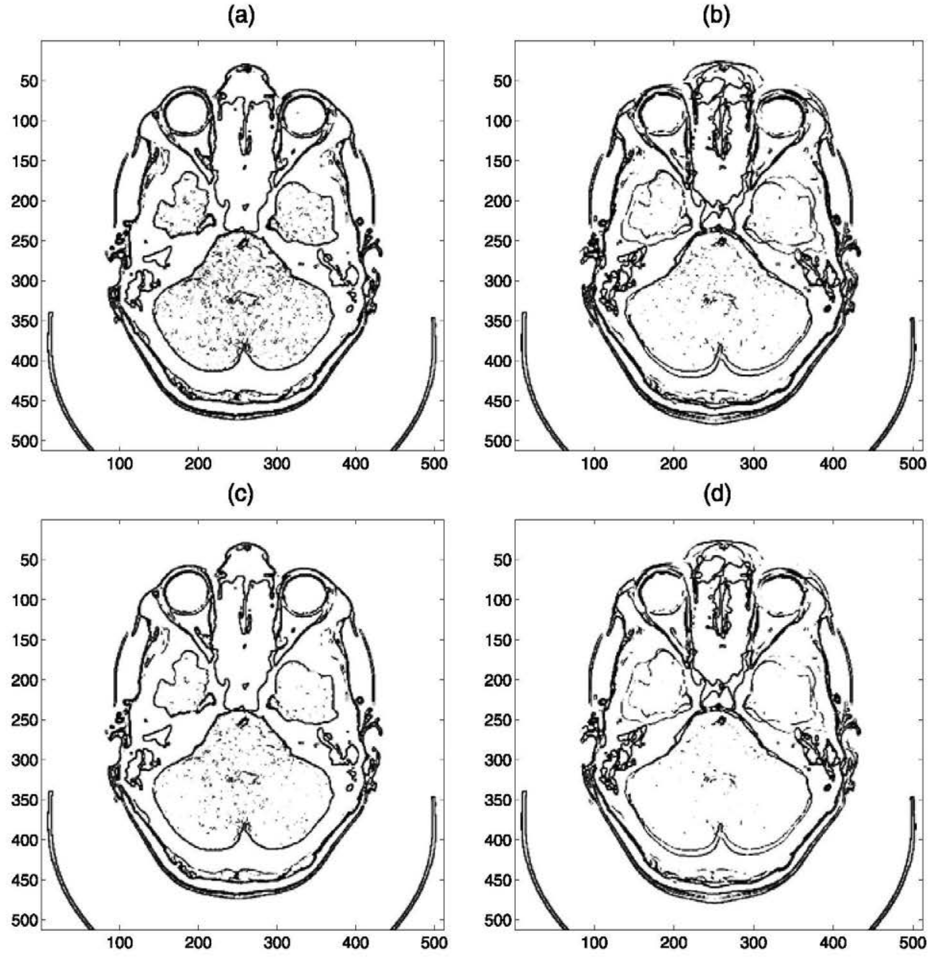

Fig. 9 Performance of 3D-Sobel and 3D-Prewitt methods on the image of size $512 \times 512 \times 21$: (**a**) Sobel ($t = 0.9$), (**b**) Sobel ($t = 0.5$), (**c**) Prewitt ($t = 0.9$), (**d**) Prewitt ($t = 0.5$)

considered the relative error $(E2/(n+m+h))$ constant, but this suffices to get better results as the resolution of the image increases, for example, compare Fig. 11 obtained from the image of size $128 \times 128 \times 6$ with Fig. 10(b). EDAS-3 can handle large size data sets. In fact, the examples studied in other sections of this paper have infinite resolution.

EDAS-3 is slow even with moderate sized images. This feature is shared with deformable model algorithms. For example, in the case of active volume methods, thousands of CPU

Table 10 Performance of EDAS-2 on images of size 512 × 512

Image	E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
0.9 A + 0.1 B	0.1	2	36	10	10	102587	24558	12.2
0.5 A + 0.5 B	0.1	2	36	10	10	102773	25399	12.3

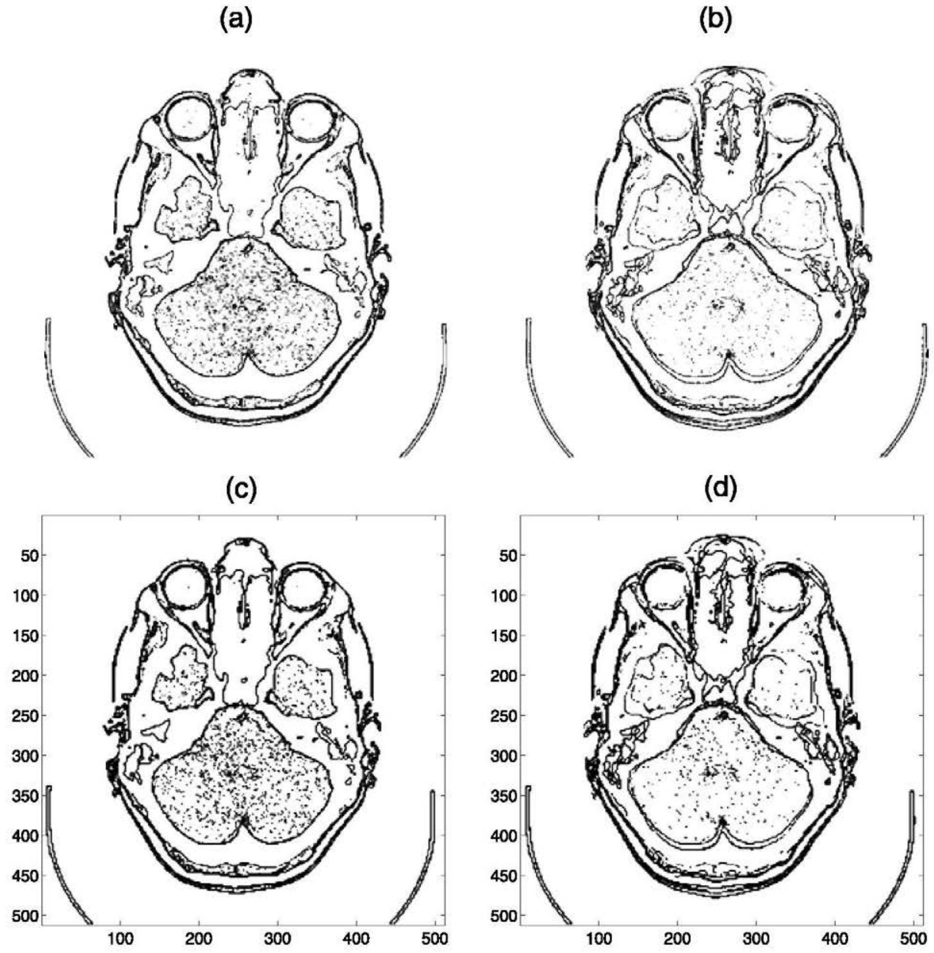

Fig. 10 Performance of EDAS-3 on the image of size 512 × 512 × 21: (**a**) slice ⟨2.3, 2.7⟩, (**b**) slice ⟨10.3, 10.7⟩. Performance of EDAS-2 on 2D images of size 512 × 512: (**c**) $t = 0.9$, (**d**) $t = 0.5$

seconds have been reported [32]. The major reason behind this is the different output of the algorithms. Difference filters provide a point based representation of the discontinuity set of the images. This representation is difficult to visualize and manipulate. It requires algorithms to reconstruct the 3D bodies contained in an image. On the contrary, EDAS-3 and some deformable models [7, 32] provide triangulated descriptions of the discontinuity sets that can be used directly in 3D visualization, virtual reality models [40], etc.

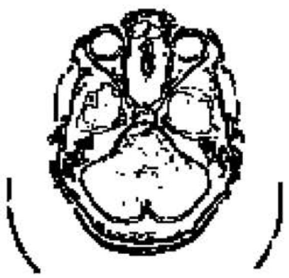
Fig. 11 Performance of EDAS-3 on the $128 \times 128 \times 6$ image: slice $\langle 2.95, 3.05 \rangle$

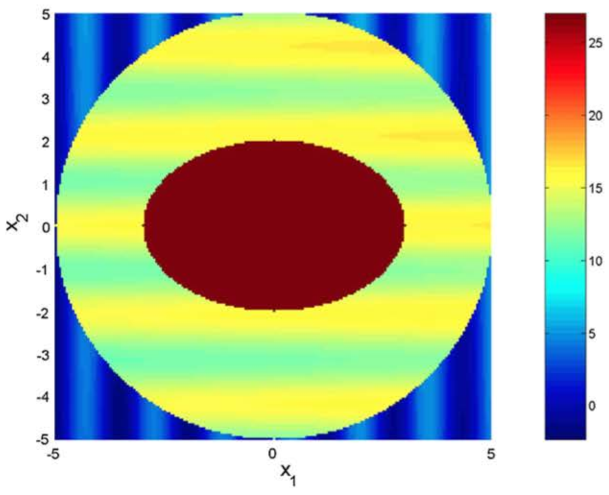

Fig. 12 Function $f_4(x_1, x_2, 0)$

The technique of finding edges in different 2D sections of a 3D image is fast and efficient. The results for EDAS-3 are reported in Table 10; see Figs. 10(c) and 10(d). It can be used for medical diagnosis, but if we are interested in a description of the 3D structure we need to use additional reconstruction algorithms.

4.5 General Piecewise Continuous Function

In the case of general piecewise continuous functions which are not piecewise constant, Theorem 1 is not directly applicable. In spite of this fact, EDAS-3 can be applied to solve some of these problems as we see below. Define

$$f_4(x_1, x_2, x_3) \equiv \begin{cases} 2 + 0.1(x_1 + x_2 + x_3) + 3\sin(4x_1), & \text{if } (x_1, x_2, x_3) \in R - B_4, \\ 14 + 0.1(x_1 + x_2) + 2\cos(3x_2), & \text{if } (x_1, x_2, x_3) \in B_4 - E, \\ 27, & \text{if } (x_1, x_2, x_3) \in E, \end{cases}$$

where $R = [-10, 10]^3$, $B_4 \equiv \{(x_1, x_2, x_3) : x_1^2 + x_2^2 + x_3^2 \leq 25\}$, and $E \equiv \{(x_1, x_2, x_3) : x_1^2/9 + x_2^2/4 + x_3^2/4 \leq 1\}$. Figure 12 shows the intersection of the graph of f_4 with the plane $x_3 = 0$.

The results obtained are reported in Table 11. Figure 13 shows a comparison between the slices of the cloud of points generated by EDAS-3 and the intersections of the interface with planes perpendicular to the x_3-axis.

Table 11 Performance of EDAS-3 on f_4

E_1	E_2	E_3	E_4	E_5	TNT	$TNJT$	CPU (s)
0.1	0.1	5.0	1.0	2.0	9474706	1073228	340.8

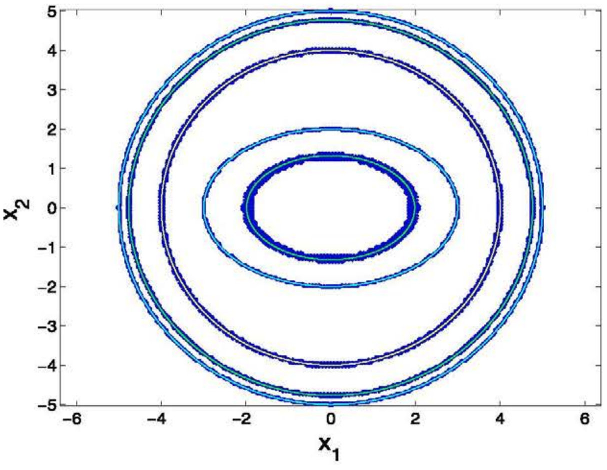

Fig. 13 (Color online) Performance of EDAS-3 on f_4: Slice $\langle -0.05, 0.05\rangle$, exact solution for $x_3 = 0$ (*light blue line*); slice $\langle 1.45, 1.55\rangle$, exact solution for $x_3 = 1.5$ (*green line*); slice $\langle 2.95, 3.05\rangle$, exact solution for $x_3 = 3$ (*yellow line*)

5 Concluding Remarks

In this paper we study the case $d = 3$ of the algorithm EDAS-d. EDAS-3 can approximate the jump discontinuity set of functions defined almost everywhere on convex subsets of $\mathbb{R}^3$. The method is based on adaptive splitting of the domain of the function guided by the value of an average integral. The numerical computation of these integrals introduces an "implicit smoothing" that allows a fast convergence away from the jump points. The algorithm needs to specify five parameters which have an intuitive meaning. They can be easily fixed in most problems. While difference filters and deformable models approximate the jump discontinuity set by a set of points and a set of surfaces respectively, EDAS-3 provides a triangulated 3D set that contains the jump discontinuity set. The accuracy of this approximation is fixed beforehand. This is a basic difference with other methods. The output of EDAS-3 does not require reconstruction algorithms and can be used directly as a 3D model.

EDAS-3 allows to handle automatically interfaces with complex topological structure. Initial conditions are not necessary. Since it is not a variational approach, it does not present the problem of stopping at local minima of an energy. In the case of difficult problems we can partition the domain and apply EDAS-3 independently to each subdomain (parallel processing).

We have studied from a theoretical and computational point of view the case $d = 3$. From the experiments, we can draw the following conclusions about EDAS-3:

It provides a precise determination of the jump points.

The resulting piecewise affine function L does not show oscillatory behavior near the jump points. This makes possible to obtain accurate values of the magnitude of the jumps. As a consequence, we can obtain stratified edges (Γ_l^J, Γ_{lu}^J) in a straightforward manner. In the case of images, this fact is important because stratified edges correspond to "individual objects" in a scene.

It is robust against continuous and discontinuous random perturbations.

It gives good results for general piecewise continuous functions (not necessarily piecewise constant).

The CPU-time of EDAS-3 depends on the complexity of the 3D image and on the precision required. In the experiments performed on real images the order of magnitude of the CPU time has been similar to that reported by active volume algorithms.

Some caution is necessary when we apply EDAS-3:

The approximate method of evaluation of integrals can lead to the oversight of some jump points. This can be avoided by decreasing the exploration parameter E_4 or by modifying the adaptive procedure to compute integrals.

If the interface presents acute dihedral angles, Theorem 1 may not be applicable to tetrahedra intersecting the edges of such angles. This part of Γ^J might be erroneously approximated.

To sum up, we can state that EDAS-3 is a general procedure to approximate the jump discontinuity set of functions defined on $\mathbb{R}^3$. EDAS-3 can be applied to 3D medical images (MRI, CT, etc). This method is specially suitable for dealing with high-resolution 3D images.

Acknowledgement We would like to thank the anonymous reviewers for their careful reading of our manuscript and the insightful comments they provided.

Appendix

In this section we give a proof of Theorem 1. Let $\alpha = (\alpha_1, \alpha_2, \dots \alpha_d)$, where $\alpha_i \in \mathbb{Z}^+ \cup \{0\},\ i = 1, \dots, d$. Define

$$I_d f \equiv \int_0^1 dx_1 \int_0^{1-x_1} dx_2 \dots \int_0^{1-x_1 \dots -x_{d-1}} f(x_1, x_2, \dots, x_d) dx_d,$$
$$\mathbf{x}^\alpha \equiv \mathbf{x} = (x_1, \dots, x_d) \to x_1^{\alpha_1} \dots x_d^{\alpha_d}.$$

Lemma 1 ([17])

$$I_d \mathbf{x}^\alpha = \alpha_1! \dots \alpha_d! / (d + |\alpha|)!,$$

where $|\alpha| = \alpha_1 + \cdots + \alpha_d$.

We need some results about range computation of polynomials using the Bernstein form. Define

$$\mathbf{i} \equiv (i_1, \dots, i_d), \qquad \mathbf{l} \equiv (l_1, \dots, l_d), \qquad \overline{\mathbf{x}} \equiv (\overline{x}_1, \dots, \overline{x}_d), \qquad \underline{\mathbf{x}} \equiv (\underline{x}_1, \dots, \underline{x}_d),$$
$$\mathbf{x}^{\mathbf{i}} \equiv \prod_{\mu=1}^{d} x_\mu^{i_\mu}, \qquad \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{l}} \equiv \sum_{i_1=0}^{l_1} \dots \sum_{i_d=0}^{l_d}, \qquad \binom{\mathbf{l}}{\mathbf{i}} \equiv \prod_{\mu=1}^{d} \binom{l_\mu}{i_\mu}.$$

A d-variate polynomial of degree $\mathbf{l} = (l_1, l_2, \dots, l_d)$ can be written as $p(\mathbf{x}) = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{l}} a_{\mathbf{i}} \mathbf{x}^{\mathbf{i}}$, where $\mathbf{x} = (x_1, x_2, \dots, x_d)$.

Consider the d-box $X = [\underline{x}_1, \overline{x}_1] \times \cdots \times [\underline{x}_d, \overline{x}_d]$. A d-variate polynomial p of degree $\mathbf{l}$ can be represented over X as $p(\mathbf{x}) = \sum_{\mathbf{i}=\mathbf{0}}^{\mathbf{l}} b_{\mathbf{i}} B_{\mathbf{i}}(\mathbf{x})$, where $B_{\mathbf{i}}(\mathbf{x}) = \binom{\mathbf{l}}{\mathbf{i}} \mathbf{x}^{\mathbf{i}} (\mathbf{1} - \mathbf{x})^{\mathbf{l}-\mathbf{i}}$. The so-called Bernstein coefficients are given by the following expression (see [34])

$$b_{\mathbf{i}} = \sum_{\mathbf{j}=\mathbf{0}}^{\mathbf{i}} \frac{\binom{\mathbf{i}}{\mathbf{j}}}{\binom{\mathbf{l}}{\mathbf{j}}} (\overline{\mathbf{x}} - \underline{\mathbf{x}})^{\mathbf{j}} \sum_{\mathbf{k}=\mathbf{j}}^{\mathbf{l}} \binom{\mathbf{k}}{\mathbf{j}} \underline{\mathbf{x}}^{\mathbf{k}-\mathbf{j}} a_{\mathbf{k}}, \quad \mathbf{0} \le \mathbf{i} \le \mathbf{l}.$$

Lemma 2 (Range enclosing property, [34]) *The range of p over X is contained within the interval spanned by the minimum and maximum Bernstein coefficients, that is*

$$\min_{\mathbf{i}}\{b_{\mathbf{i}}\} \le p(\mathbf{x}) \le \max_{\mathbf{i}}\{b_{\mathbf{i}}\}, \quad \mathbf{x} \in X.$$

Theorem 1 *Let $h : \mathbb{R}^3 \to \mathbb{R}$ be the function defined by*

$$h(\mathbf{x}) \equiv JH(ax_1 + bx_2 + cx_3 + d),$$

where a, b, c, d, and J are real numbers such that either a or b or c are distinct from zero and $J > 0$. Define $r \equiv \{\mathbf{x} = (x_1, x_2, x_3) \in \mathbb{R}^3 : ax_1 + bx_2 + cx_3 + d = 0\}$ and let T be an arbitrary tetrahedron in $\mathbb{R}^3$ such that $r \cap \overset{\circ}{T} \neq \emptyset$. Then

$$\frac{7J}{32} \le \frac{\int_T |h(\mathbf{x}) - L_T h(\mathbf{x})| d\mathbf{x}}{\mathrm{v}(T)} \le \frac{3J}{4}, \tag{1}$$

where $\mathrm{v}(T)$ is the volume of T.

Proof Let $A(a_1, a_2, a_3)$, $B(b_1, b_2, b_3)$, $C(c_1, c_2, c_3)$, and $D(d_1, d_2, d_3)$ be the vertices of the tetrahedron T. Consider the change of variables $\mathbf{x} = \tau(\mathbf{y})$ defined by

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 - a_1 & c_1 - a_1 & d_1 - a_1 \\ b_2 - a_2 & c_2 - a_2 & d_2 - a_2 \\ b_3 - a_3 & c_3 - a_3 & d_3 - a_3 \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} a_1 \\ a_2 \\ a_3 \end{pmatrix}.$$

Let

$$M_\tau = \begin{pmatrix} b_1 - a_1 & c_1 - a_1 & d_1 - a_1 \\ b_2 - a_2 & c_2 - a_2 & d_2 - a_2 \\ b_3 - a_3 & c_3 - a_3 & d_3 - a_3 \end{pmatrix}.$$

We have $T = \tau(\Delta)$, where Δ is the tetrahedron in the space (y_1, y_2, y_3) with vertices $P_1(0, 0, 0)$, $P_2(1, 0, 0)$, $P_3(0, 1, 0)$ and $P_4(0, 0, 1)$.

If we use the change of variables formula for multiple integrals and consider that $\mathrm{v}(T) = |det(M_\tau)|/6$

$$AI_T(h) = \frac{\int_T |h(\mathbf{x}) - L_T h(\mathbf{x})| d\mathbf{x}}{\mathrm{v}(T)} = \frac{\int_\Delta |h(\tau(\mathbf{y})) - L_T h(\tau(\mathbf{y}))||det(M_\tau)| d\mathbf{y}}{\mathrm{v}(T)}$$

$$= 6 \int_\Delta |h(\tau(\mathbf{y})) - L_T h(\tau(\mathbf{y}))| d\mathbf{y}. \tag{2}$$

There are three possible cases:

(i) h is equal to J at two vertices of T and is equal to zero at the other two vertices.
(ii) h is equal to J at one vertex of T and is equal to zero at the other three vertices.
(iii) h is equal to J at three vertices of T and is equal to zero at the other vertex.

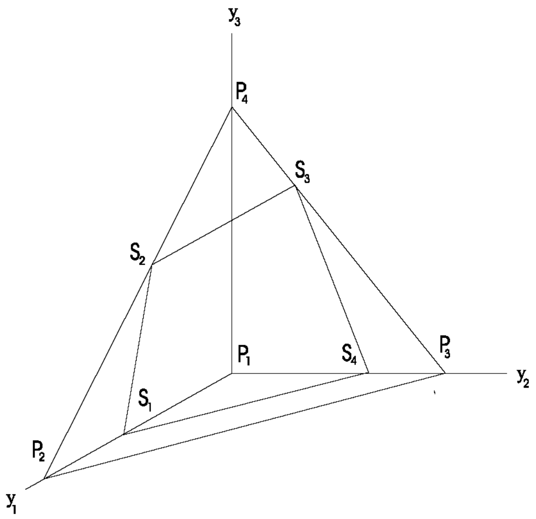

Fig. 14 Illustration for the proof of Theorem 1

(i) Assume that h is equal to J at B and C and is equal to zero at A and D. To obtain $L_T h(\tau(\mathbf{y}))$ it suffices to find the affine interpolant L_Δ such that $L_\Delta(P_2) = L_\Delta(P_3) = J$ and $L_\Delta(P_1) = L_\Delta(P_4) = 0$. It is clear that $L_\Delta(\mathbf{y}) = J(y_1 + y_2)$. Therefore

$$L_T h(\tau(\mathbf{y})) = L_\Delta(\mathbf{y}) = J(y_1 + y_2),$$

and (2) can be written as

$$AI_T(h) = 6\int_\Delta |h(\tau(\mathbf{y})) - J(y_1 + y_2)|d\mathbf{y}.$$

Consider a plane intersecting Δ such as that shown in Fig. 14. The intersection is a quadrilateral with vertices $S_1 = (\alpha, 0, 0)$, $S_2 = (\beta, 0, 1-\beta)$, $S_3 = (0, \gamma, 1-\gamma)$, and $S_4 = (0, \delta, 0)$. Define

$$\begin{aligned}
T_1 &\equiv \langle P_1, S_2, P_4, S_3\rangle,\\
T_2 &\equiv \langle P_1, S_1, S_2, S_3\rangle,\\
T_3 &\equiv \langle P_1, S_1, S_3, S_4\rangle,\\
Q_1 &\equiv T_1 \cup T_2 \cup T_3,\\
Q_2 &\equiv \Delta - Q_1.
\end{aligned}$$

Then

$$\begin{aligned}
AI_T(h) &= 6\int_\Delta |h(\tau(\mathbf{y})) - L_T h(\tau(\mathbf{y}))|d\mathbf{y}\\
&= 6\left(\int_{Q_1} |0 - J(y_1 + y_2)|d\mathbf{y} + \int_{Q_2} |J - J(y_1 + y_2)|d\mathbf{y}\right)\\
&= 6J\left(\int_{Q_1} (2y_1 + 2y_2 - 1)d\mathbf{y} + \int_\Delta (1 - y_1 - y_2)d\mathbf{y}\right).
\end{aligned}$$

By Lemma 1,

$$\int_{\Delta}(1-y_1-y_2)d\mathbf{y}=1/12.$$

Then we have

$$AI_T(h)=6J\Big(\int_{Q_1}(2y_1+2y_2-1)d\mathbf{y}+1/12\Big)=6J(I_1+I_2+I_3+1/12),$$

where $I_i\equiv\int_{T_i}(2x_1+2x_2-1)d\mathbf{x},\ i=1,2,3$. The evaluation of the integrals I_i can be made by means of suitable changes of variables.

The application τ_1 defined by

$$\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}\beta & 0 & 0\\ 0 & \gamma & 0\\ 1-\beta & 1-\gamma & 1\end{pmatrix}\begin{pmatrix}y_1\\y_2\\y_3\end{pmatrix}$$

is such that $T_1=\tau_1(\Delta)$. Using τ_1, we have

$$I_1=\int_{T_1}(2x_1+2x_2-1)d\mathbf{x}=\int_{\Delta}(2\beta y_1+2\gamma y_2-1)\beta\gamma d\mathbf{y}.$$

By Lemma 1,

$$I_1=\beta\gamma(\beta+\gamma-2)/12.$$

The application τ_2 defined by

$$\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}\alpha & 0 & \beta\\ 0 & \gamma & 0\\ 0 & 1-\gamma & 1-\beta\end{pmatrix}\begin{pmatrix}y_1\\y_2\\y_3\end{pmatrix}$$

is such that $T_2=\tau_2(\Delta)$. Using τ_2, we have

$$I_2=\int_{T_2}(2x_1+2x_2-1)d\mathbf{x}=\int_{\Delta}(2\alpha y_1+2\gamma y_2+2\beta y_3-1)\alpha\gamma(1-\beta)d\mathbf{y}.$$

By Lemma 1,

$$I_2=\alpha(1-\beta)\gamma(\alpha+\beta+\gamma-2)/12.$$

The application τ_3 defined by

$$\begin{pmatrix}x_1\\x_2\\x_3\end{pmatrix}=\begin{pmatrix}\alpha & 0 & 0\\ 0 & \delta & \gamma\\ 0 & 0 & 1-\gamma\end{pmatrix}\begin{pmatrix}y_1\\y_2\\y_3\end{pmatrix}$$

is such that $T_3=\tau_3(\Delta)$. Using τ_3, we have

$$I_3=\int_{T_3}(2x_1+2x_2-1)d\mathbf{x}=\int_{\Delta}(2\alpha y_1+2\delta y_2+2\gamma y_3-1)\alpha(1-\gamma)\delta d\mathbf{y}.$$

By Lemma 1,

$$I_3=\alpha(1-\gamma)\delta(\alpha+\gamma+\delta-2)/12.$$

Define

$$\begin{aligned} K &\equiv 12(I_1 + I_2 + I_3) \\ &= \beta\gamma(\beta+\gamma-2) + \alpha(1-\beta)\gamma(\alpha+\beta+\gamma-2) + \alpha(1-\gamma)\delta(\alpha+\gamma+\delta-2). \end{aligned}$$

Then $AI_T(h) = J(K+1)/2$.

Let $\mathbf{z} = (\alpha, \beta, \gamma, \delta)$. It is necessary to determine lower and upper bounds of $K(\mathbf{z})$ on $R = [0, 1]^4$. In order to achieve tight bounds we can partition R into n 4-intervals, that is $R = \bigcup_{i=1}^{n} R_i$ with $\overset{\circ}{R}_i \cap \overset{\circ}{R}_j = \emptyset,\ i \neq j$, and consider that

$$\underset{\mathbf{z}\in R}{LBK}(\mathbf{z}) = \min_{1\le i\le n}(\underset{\mathbf{z}\in R_i}{LBK}(\mathbf{z})), \tag{3}$$

$$\underset{\mathbf{z}\in R}{UBK}(\mathbf{z}) = \max_{1\le i\le n}(\underset{\mathbf{z}\in R_i}{UBK}(\mathbf{z})), \tag{4}$$

where LBK denotes a lower bound of K and UBK denotes an upper bound of K. We have divided each side of R into 16 equal parts obtaining a partition of R into 65536 identical 4-intervals. By applying Lemma 2 to each R_i, (3), and (4), we have

$$-0.375488 \le K(\mathbf{z}) \le 0, \quad \mathbf{z} \in R.$$

To appreciate the tightness of these bounds observe that K attains a local minimum at $\mathbf{z}^* = (1/2, 1/2, 1/2, 1/2)$ with $K(\mathbf{z}^*) = -3/8 = -0.375$. On the other hand, $K(1, 1, 1, 1) = 0$.

Consequently, we have

$$0.312256J \le AI_T(h) \le J/2. \tag{5}$$

(ii) Assume that h is equal to J at B and is equal to zero at A, C and D. To obtain $L_T h(\tau(\mathbf{y}))$ it suffices to find the affine interpolant L_Δ such that $L_\Delta(P_2) = J$ and $L_\Delta(P_1) = L_\Delta(P_3) = L_\Delta(P_4) = 0$. It is clear that $L_\Delta(\mathbf{y}) = Jy_1$. Therefore

$$L_T h(\tau(\mathbf{y})) = L_\Delta(\mathbf{y}) = Jy_1,$$

and (2) can be written as

$$AI_T(h) = 6\int_\Delta |h(\tau(\mathbf{y})) - Jy_1|d\mathbf{y}.$$

Consider a plane intersecting Δ such as that shown in Fig. 15. The intersection is a triangle with vertices $S_1 = (\alpha, 0, 0)$, $S_2 = (\beta, 0, 1-\beta)$, and $S_3 = (\gamma, 1-\gamma, 0)$. Define

$$\begin{aligned} T_1 &\equiv \langle P_2, S_1, S_2, S_3\rangle, \\ Q_1 &\equiv \Delta - T_1. \end{aligned}$$

Then we have

$$\begin{aligned} AI_T(h) &= 6\int_\Delta |h(\tau(\mathbf{y})) - L_T h(\tau(\mathbf{y}))|d\mathbf{y} \\ &= 6\left(\int_{T_1} |J - Jy_1|d\mathbf{y} + \int_{Q_1} |0 - Jy_1|d\mathbf{y}\right) \\ &= 6J\left(\int_{T_1} (1-2y_1)d\mathbf{y} + \int_\Delta y_1 d\mathbf{y}\right). \end{aligned}$$

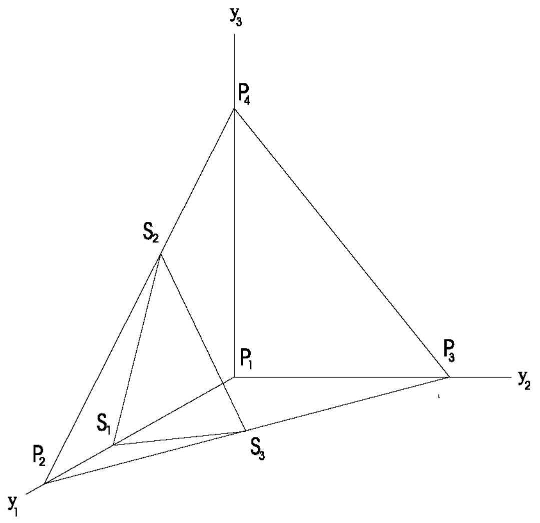

Fig. 15 Illustration for the proof of Theorem 1

By Lemma 1,

$$\int_{\Delta} y_1 d\mathbf{y} = 1/24.$$

Then we have

$$AI_T(h) = 6J(I_1 + 1/24),$$

where $I_1 \equiv \int_{T_1}(1-2x_1)d\mathbf{x}$.

The application τ_4 defined by

$$\begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} 1-\alpha & \gamma-\alpha & \beta-\alpha \\ 0 & 1-\gamma & 0 \\ 0 & 0 & 1-\beta \end{pmatrix} \begin{pmatrix} y_1 \\ y_2 \\ y_3 \end{pmatrix} + \begin{pmatrix} \alpha \\ 0 \\ 0 \end{pmatrix}$$

is such that $T_1 = \tau_4(\Delta)$. Using τ_4, we have

$$I_1 = \int_{T_1}(1-2x_1)d\mathbf{x} = \int_{\Delta}(1-2((1-\alpha)y_1 + (\gamma-\alpha)y_2 + (\beta-\alpha)y_3 + \alpha))d\mathbf{y}.$$

Let $u = (1-\alpha)(1-\beta)(1-\gamma)$, by Lemma 1, $I_1 = u(1-\alpha-\beta-\gamma)/12$. Let $K = 12I_1 = u(1-\alpha-\beta-\gamma)$, then we have $AI_T(h) = J(K+1/2)/2$.

Let $\mathbf{z} = (\alpha, \beta, \gamma)$ and $R = [0,1]^3$. Using standard analytical techniques we find that $K(\mathbf{z})$ has only one local minimum in $\overset{\circ}{R}$ that is attained at $\mathbf{z}^* = (1/2, 1/2, 1/2)$. It is easy to check (studying the values of $K(\mathbf{z})$ at the boundary of R) that the global minimum of K over R is $K(\mathbf{z}^*) = -1/16 = -0.0625$. Moreover, applying the Bernstein procedure with a partition of R into 2097152 identical cubes, we have

$$-0.062504 \le K(\mathbf{z}) \le 1, \quad \mathbf{z} \in R.$$

The upper bound is sharp because $K(0,0,0) = 1$.

Consequently, we have

$$7J/32 \le AI_T(h) \le 3J/4. \tag{6}$$

(iii) Assume that h is equal to zero at B and is equal to J at A, C and D. To obtain $L_T h(\tau(\mathbf{y}))$ it suffices to find the affine interpolant L_Δ such that $L_\Delta(P_2) = 0$ and $L_\Delta(P_1) = L_\Delta(P_3) = L_\Delta(P_4) = J$. It is clear that $L_\Delta(\mathbf{y}) = J(1 - y_1)$. Therefore,

$$L_T h(\tau(\mathbf{y})) = L_\Delta(\mathbf{y}) = J(1 - y_1),$$

and (2) can be written as

$$AI_T(h) = 6\int_\Delta |h(\tau(\mathbf{y})) - (1 - Jy_1)|d\mathbf{y}.$$

Consider a plane intersecting Δ such as that shown in Fig. 15. The intersection is a triangle whose vertices are those described in section (ii).

Define

$$T_1 \equiv \langle P_2, S_1, S_2, S_3\rangle,$$
$$Q_1 \equiv \Delta - T_1.$$

Then we have

$$\begin{aligned} AI_T(h) &= 6\int_\Delta |h(\tau(\mathbf{y})) - L_T h(\tau(\mathbf{y}))|d\mathbf{y} \\ &= 6\left(\int_{T_1} |0 - J(1 - y_1)|d\mathbf{y} + \int_{Q_1} |J - J(1 - y_1)|d\mathbf{y}\right) \\ &= 6J\left(\int_{T_1} (1 - 2y_1)d\mathbf{y} + \int_\Delta y_1 d\mathbf{y}\right). \end{aligned}$$

Therefore, this case transforms into case (ii). The bounds of $AI_T(h)$ are given by (6). Since $7/32 = 0.218750$, (1) follows from inequalities (5) and (6). □

References

1. Archibald, R., Chen, K., Gelb, A., Renaut, R.: Improving tissue segmentation of human brain MRI through preprocessing by the Gegenbauer reconstruction method. NeuroImage **20**, 489–502 (2003)
2. Archibald, R., Gelb, A., Gottlieb, S., Ryan, J.: One-sided post-processing for the discontinuous Galerkin method using ENO type stencil choosing and the local edge detection method. J. Sci. Comput. **28**, 167–190 (2006)
3. Archibald, R., Gelb, A., Saxena, R., Xiu, D.: Discontinuity detection in multivariate space for stochastic simulations. J. Comput. Phys. **228**, 2676–2689 (2009)
4. Archibald, R., Gelb, A., Yoon, J.: Polynomial fitting for edge detection in irregularly sampled signals and images. SIAM J. Numer. Anal. **43**, 259–279 (2005)
5. Archibald, R., Hu, J., Gelb, A., Farin, G.: Improving the accuracy of volumetric segmentation using pre-processing boundary detection and image reconstruction. IEEE Trans. Image Process. **13**, 459–466 (2004)
6. Bänsch, E., Mikula, K.: Adaptivity in 3D image processing. Comput. Vis. Sci. **4**, 21–30 (2001)
7. Barreira, N., Penedo, M.G., Cohen, L., Ortega, M.: Topological active volumes: A topology-adaptive deformable model for volume segmentation. Pattern Recognit. **43**, 255–266 (2010)
8. Bauer, Ch., Pock, T., Sorantin, E., Bischof, H., Beichel, R.: Segmentation of interwoven 3d tubular tree structures utilizing shape priors and graph cuts. Med. Image Anal. **14**, 172–184 (2010)
9. Bliss, A., Su, F.E.: Lower bounds for simplicial covers and triangulations of cubes. Discrete Comput. Geom. **33**, 669–686 (2005)

10. Bosnjak, A., Montilla, G., Villegas, R., Jara, I.: 3D segmentation with an application of level set-method using MRI volumes for image guided surgery. In: Proceedings of the 29th Annual International Conference of the IEEE EMBS, 23–26 August, Lyon, France, pp. 5263–5266 (2007)
11. Catté, F., Lions, P.L., Morel, J.M., Coll, T.: Image selective smoothing and edge detection by nonlinear diffusion. SIAM J. Numer. Anal. **29**, 182–193 (1992)
12. Di, Y., Li, R.: Computation of dendritic growth with level set model using a multi-mesh adaptive finite element method. J. Sci. Comput. **39**, 441–453 (2009)
13. Gamelin, Th.W., Greene, R.E.: Introduction to Topology. Dover, New York (1999)
14. Gelb, A., Tadmor, E.: Detection of edges in spectral data. Appl. Comput. Harmon. Anal. **7**, 101–135 (1999)
15. Gottlieb, D., Shu, Ch.-W.: On the Gibbs phenomenon and its resolution. SIAM Rev. **39**, 644–668 (1997)
16. Grevera, G.J., Udupa, J.K., Miki, Y.: A task-specific evaluation of three-dimensional image interpolation techniques. IEEE Trans. Med. Imaging **18**, 137–143 (1999)
17. Grundmann, A., Möller, H.M.: Invariant integration formulas for the n-simplex by combinatorial methods. SIAM J. Numer. Anal. **15**, 282–290 (1978)
18. Horowitz, S.L., Pavlidis, T.: Picture segmentation by a tree traversal algorithm. J. ACM **23**, 368–388 (1975)
19. http://en.wikipedia.org/wiki/File:Computed_tomography_of_human_brain_-_large.png#filehistory
20. Kass, M., Witkin, A., Terzopoulos, D.: Snakes: Active contour models. Int. J. Comput. Vis. **1**, 321–331 (1988)
21. Kitasaka, T., Mori, K., Hasegawa, J., Toriwaki, J., Katada, K.: Recognition of aorta and pulmonary artery in the mediastinum using medial-line models from 3D CT images without contrast material. Med. Imaging Technol. **20**, 572–583 (2002)
22. Liu, H.K.: Two- and three-dimensional boundary detection. Comput. Graph. Image Process. **6**, 123–134 (1977)
23. Lizier, M.A.S., Martins, D.C. Jr., Cuadros-Vargas, A.J., Cesar, R.M. Jr., Nonato, L.G.: Generated segmented meshes from textured color images. J. Vis. Commun. Image Represent. **20**, 190–203 (2009)
24. Llanas, B., Lantarón, S.: Edge detection by adaptive splitting. J. Sci. Comput. **46**, 485–518 (2011)
25. Llanas, B., Sáinz, F.J.: Fast training of neural trees by adaptive splitting based on cubature. Neurocomputing **71**, 3387–3408 (2008)
26. McInerney, T., Terzopoulos, D.: T-snakes: topology adaptive snakes. Med. Image Anal. **4**, 73–91 (2000)
27. Meinhardt, E., Zacur, E., Frangi, A.F., Caselles, V.: 3D edge detection by selection of level surface patches. J. Math. Imaging Vis. **34**, 1–16 (2009)
28. Orden, D., Santos, F.: Asymptotically efficient triangulations of the d-cube. Discrete Comput. Geom. **30**, 509–528 (2003)
29. Perona, P., Malik, J.: Scale space and edge detection using anisotropic diffusion. In: Proceedings of the IEEE Workshop on Computer Vision (Miami), pp. 16–22 (1987)
30. Rumpf, M., Voigt, A., Berkels, B., Rätz, A.: Extracting grain boundaries and macroscopic deformations from images on atomic scale. J. Sci. Comput. **35**, 1–23 (2008)
31. Sethian, J.A.: Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science. Cambridge University Press, Cambridge (1999)
32. Shen, T., Li, H., Qian, Z., Huang, X.: Active volume models for 3D medical image segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition 2009, pp. 707–714 (2009)
33. Shilling, R.Z., Robbie, T.Q., Bailloeul, T., Mewes, K., Mersereau, R.M., Brummer, M.E.: A super-resolution framework for 3-D high-resolution and high-contrast imaging using 2-D multislice MRI. IEEE Trans. Med. Imaging **28**, 633–644 (2009)
34. Smith, A.P.: Fast construction of constant bound functions for sparse polynomials. J. Glob. Optim. **43**, 445–458 (2009)
35. Suri, J.S., Wilson, D.L., Laxminarayan, S. (eds.): Handbook of Biomedical Image Analysis Volume III: Registration Models. Kluwer Academic/Plenum Publishers, New York (2005)
36. Terzopoulos, D., Witkin, A., Kass, M.: Constraints on deformable models: recovering 3D shape and nonrigid motion. Artif. Intell. **36**, 91–123 (1988)
37. Toriwaki, J., Yoshida, H.: Fundamentals of Three-Dimensional Digital Image Processing. Springer, Dordrecht (2009)
38. Vukadinovic, D., van Walsum, Th., Manniesing, R., Rozie, S., Hameeteman, R., de Weert, T.T., van der Lugt, A., Niessen, W.J.: Segmentation of the outer vessel wall of the common carotid artery in CTA. IEEE Trans. Med. Imaging **29**, 65–76 (2010)
39. Wang, D., Doddrell, D.M., Cowin, G.: A novel phantom and method for comprehensive 3-dimensional measurement and correction of geometric distortion in magnetic resonance imaging. J. Magn. Reson. Imaging **22**, 529–542 (2004)

40. Wang, G., Wu, Q.M.J.: Guide to Three Dimensional Structure and Motion Factorization. Springer, Dordrecht (2011)
41. Wu, X.: Adaptive split-and-merge segmentation based on piecewise least-square approximation. IEEE Trans. Pattern Anal. Mach. Intell. **15**, 808–815 (1993)
42. Yokoyama, K., Kitasaka, T., Mori, K., Mekada, Y., Hasegawa, J., Toriwaki, J.: Liver region extraction from 3D abdominal X-ray CT images using distribution features of abdominal organs. J. Comput. Aided Diag. Medical Images **7**, 1–11 (2003)