Noname manuscript No.
(will be inserted by the editor)

A doubly adaptive algorithm for edge detection in 3D images

Sagrario Lantarón · M. Dolores López · Javier Rodrigo

Abstract This paper proposes a new algorithm (DA3DED) for edge detection in 3D images. DA3DED is doubly adaptive because it is based on the adaptive algorithm EDAS-1 for detecting edges in functions of one variable and a second adaptive procedure based on the concept of projective complexity of a 3D image. DA3DED has been tested on 3D images that modelize real problems (composites and fractures). It has been much faster than the 1D edge detection algorithm for 3D images derived from EDAS-1.

Keywords edge detection · 3D images · Projective complexity · Doubly adaptive algorithm

1 Introduction

Three dimensional imaging is useful in many fields such as medicine and materials science. 3D medical imaging models organs and internal structures of the human body. These models are important in image guided surgery, assessment of the quality of bones, diagnostics, etc; see, for example, [1–3]. New materials (composites, foams, etc.) have a complex geometric structure. 3D images of materials provide data such as the 3D connectivity of a structure, distribution of particles, fibre orientation, etc. These data can be used in simulations to compute macroscopic material properties. They constitute the basis of virtual material design [4].

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.

S. Lantarón and M.D. López
Departamento de Matemática e Informática aplicadas a las Ingenierías Civil y Naval. ETSI de Caminos. Universidad Politécnica de Madrid. Avda. Profesor Aranguren s/n. 28040 Madrid. Spain.
Tel.: +34-91-3366669
E-mail: sagrario.lantaron@upm.es; marilo.lopez@upm.es
J. Rodrigo
Departamento de Matemática Aplicada. Escuela Técnica Superior de Ingeniería (ICAI). Universidad Pontificia Comillas. C/ Alberto Aguilera 25. 28015 Madrid, Spain.
E-mail: jrodrigo@comillas.edu

  - 1D edge detection methods (1D3DED). 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 [5,6].
  - Spatial difference filters [7]. 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. 3D difference filters were first reported in [8]. In some cases the computation cost of these methods may become excessive. A subvoxel edge detector based on the Canny algorithm is described in [9].
  - Polynomial fitting method [10]. 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 is minimized, the internal energy constraints the shape of the active contour and the external energy locates the contour in desired image features [11–17]. Deformable models present some inconveniences such as sensibility to the initial contours, stopping at local minima of the energy, slowness and computational cost.
  - Adaptive splitting methods. An algorithm (EDAS-3) to approximate the jump discontinuity set of functions defined on subsets of $\mathbb{R}^3$ was proposed in [18]. The procedure is based on the adaptive splitting of the domain of the function guided by the value of an average integral. This algorithm overcomes most inconvenients presented by deformable models. It does not depend on the initial surface choice. It exhibits effective edge detection in 3D models with different interface topologies. Moreover, EDAS-3 can obtain the edges with a prefixed accuracy.
- 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 [19,20] and registration based segmentation [21].

In this paper we present an accurate algorithm (DA3DED) that can obtain a point based representation of the jump discontinuity set of a 3D image. The algorithm is a type of 1D edge detection method (1D3DED). It begins by considering a region in the plane x_1x_2 containing the projection of the image and a set of points in this region. Then it considers straight lines perpendicular to the plane x_1x_2 and computes the edge points lying on these straight lines. This task is accomplished using the 1D edge detector EDAS-1 [22]. If the number of edges detected in the region is high or the distance between them is small, the algorithm concludes that the region under consideration is complex and performs a subdivision process, otherwise the region is not divided and the algorithm studies a contiguous zone. The process is repeated for the x_1x_3 and x_2x_3 planes. The edge points obtained are stored in a file. In this way complex regions are crossed by many lines and simple regions are crossed by few lines. The resulting algorithm is doubly adaptive because the "straight line step" is performed in an adaptive way by EDAS-1 and the second step performs an adaptive process based on the complexity of the image. The algorithm uses less CPU time than previous algorithm [18]. We will give a C-like pseudocode description of the constituent algorithms of DA3DED.

Experimental results show a good behaviour of DA3DED on practical 3D problems that modelize different types of composite materials and fractures.

The paper is organized as follows. Section 2 gives some mathematical preliminary results and details the conceptual and algorithmic constituents of DA3DED: EDAS-1 algorithm, discrete and continuous representation of 3D images, projective complexity, subdivision procedures of discrete rectangles and split criterion. It also describes the algorithm DA3DED. Section 3 shows computational experiments on models of complex materials. Section 4 provides some concluding remarks.

2 Materials and Methods

2.1 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

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

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

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

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 definitions, 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 $\Gamma \equiv \cup_{i=1}^n \Gamma_i$. Let $\mathbf{x} \in \Gamma$, then for some $m \leq n, \mathbf{x} \in \Gamma_{i_j}, j = 1,...,m$, where $i_j \in \{1,2,\ldots,n\}$.

Define

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

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 following notation: Let $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$ be a set of $d+1$ vectors in $\mathbb{R}^d$. This set is called affinely independent if the vectors $\{\mathbf{v}_1 - \mathbf{v}_0, \mathbf{v}_2 - \mathbf{v}_0, \ldots, \mathbf{v}_d - \mathbf{v}_0\}$ are linearly independent. Suppose now that $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$ is an affinely independent subset of $\mathbb{R}^d$. The *d-simplex* T generated by $\{\mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d\}$, 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.

Proposition 1*([24])* Let $T = \langle \mathbf{v}_0, \mathbf{v}_1, \ldots, \mathbf{v}_d \rangle$ be a *d-simplex* and let $w_0, w_1, ..., w_d$ be *arbitrary real numbers. Then there is a unique affine map L_T from T to $\mathbb{R}$ such that $L_T(\mathbf{v}_j) = w_j, j = 0, 1, \ldots, d$.*

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

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

In this subsection we study the average integral

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

where $v(T)$ 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 described in the next section. The comportment of $AI_T(f)$ when f is continuous is given by the following results.

Proposition 2*([24]) Let $S \subset \mathbb{R}^d$ be compact and let $f : S \to \mathbb{R}$ be continuous. Then given $\varepsilon > 0$, there exists $\delta > 0$ such that*

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

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 $\varepsilon > 0$, we have that $AI_T(f) < \varepsilon$ for any small enough d-simplex $T \subset S$.*

From the above results, 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 $d = 1$.

Consider the Heaviside function

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

Define the funcion $h(x) \equiv JH(x - \alpha)$ where $J > 0$ and α are real numbers. We have the following result

Proposition 3*([22]) Let T be a compact interval in $\mathbb{R}$ and $\alpha \in \overset{\circ}{T}$. Then*

$$\frac{J}{4} \le \frac{\int_T |h(x) - L_T h(x)|\, dx}{v(T)} < J \tag{1}$$

where $v(T)$ is the length of the interval T.

2.2 Constituents of the algorithm DA3DED

DA3DED is based on several concepts and algorithms that are described in this subsection.

2.2.1 Algorithm EDAS-1

EDAS-1 is an algorithm, based on inequality (1), to approximate the jump discontinuity set of functions of one variable. It is a particular case of the algorithm EDAS-*d* for functions defined on subsets of $\mathbb{R}^d$. In this section we give an outline of EDAS-*d*. Its validity for $d = 1,2$ has been established in [22]. The case $d = 3$ has been studied in [18]. Consider the *d*-interval $R = [\mathbf{a},\mathbf{b}]$, where $\mathbf{a},\mathbf{b} \in \mathbb{R}^d$. We can find n (closed) simplices T_i such that

$$\overset{\circ}{T}_i \cap \overset{\circ}{T}_j = \emptyset,\ i \neq j,$$
$$\bigcup_{i=1}^{n} T_i = R,$$

(for example, see [25,26]). We call $P \equiv \{T_i\}_{i=1}^n$ a partition of R. The set of partitions of R is denoted by $\mathscr{P}(R)$. Given a *d*-simplex T we denote by $\mathscr{V}(T)$ the set of its vertices.

The proposed algorithm builds a partition $P \in \mathscr{P}(R)$ 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

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

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. E_1 also provides a detection threshold, because the algorithm will detect the jumps with magnitude greater than $4E_1$. 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 $\mathscr{G}_1$ and the bad simplices are put into the set $\mathscr{B}_1$.
Step 2. At each step j we have a set of good simplices $\mathscr{G}_j$ and a set of bad simplices $\mathscr{B}_j$. Divide each bad simplex into two simplices, by splitting its largest edge. Test whether these children are good or bad to obtain the sets $\mathscr{G}_{j+1}$ and $\mathscr{B}_{j+1}$
Step 3. The algorithm stops if $\mathscr{B}_j = \emptyset$. Then $G = \mathscr{G}_j$ is the searched partition. If the stopping criterion is not satisfied, go to Step 2.
Step 4. Obtain the following subset of G

$$A\Gamma^J_{E_3} \equiv \left\{ T \in G : AI_T(f) > E_1 \textit{ and } j^a \equiv \underset{\mathbf{v}_i \in \mathscr{V}(T)}{max} \{f(\mathbf{v}_i)\} - \underset{\mathbf{v}_i \in \mathscr{V}(T)}{min} \{f(\mathbf{v}_i)\} > E_3 \right\},$$

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. $A\Gamma_{E_3}^J$ is a set of simplices containing points of $\Gamma_{E_3}^J$. This constitutes an approximation of $\Gamma_{E_3}^J$. In the cases considered by Proposition 4, we have that $A\Gamma_{E_3}^J \supset \Gamma_{E_3}^J$. In the images studied in Section 3, a voxel is considered as an edge voxel if it contains some barycenter of the simplices in $A\Gamma_{E_3}^J$.

In practice we have implemented the above algorithm using a binary tree whose leaves correspond to good simplices. In the case $d = 1$, the integrals $\int_T |f(x) - L_T f(x)|\,dx$ have been computed using the Gauss-Legendre integration formula [27]. More details about the implementation and performance of EDAS-1 can be found in [22].

2.2.2 Continuous and discrete representations of a 3D Image

The algorithm DA3DED uses two different mathematical representations of a 3D image. It considers that the image is a function defined on a box in $\mathbb{R}^3$ when it computes edge points using EDAS-1. This continuous representation is also useful to define the concept of "projective complexity"; we will see this in the next subsection. In the remaining steps DA3DED uses the usual discrete representation. Below we define these concepts.

Let n and m be positive integers with $n < m$, define

$$[[n,m]] \equiv \{n, n+1, \dots, m\}$$

A 3D image is given by an $n_1 \times n_2 \times n_3$ array $I = \{I_{ijk}\}$ with indices $(i,j,k) \in [[1,n_1]] \times [[1,n_2]] \times [[1,n_3]]$. From a 3D image I, one can build a piecewise constant function $\tilde{I}$, defined on $[0.5, n_1 + 0.5] \times [0.5, n_2 + 0.5] \times [0.5, n_3 + 0.5]$ in the following way

$$\tilde{I}(x_1,x_2,x_3) \equiv \begin{cases} I_{111}, \text{ if } (x_1,x_2,x_3) \in [0.5,1.5] \times [0.5,1.5] \times [0.5,1.5], \\ I_{i11}, \text{ if } (x_1,x_2,x_3) \in [i-0.5,i+0.5] \times [0.5,1.5] \times [0.5,1.5] \ (i > 1), \\ I_{1j1}, \text{ if } (x_1,x_2,x_3) \in [0.5,1.5] \times [j-0.5,j+0.5] \times [0.5,1.5] \ (j > 1), \\ I_{ij1}, \text{ if } (x_1,x_2,x_3) \in [i-0.5,i+0.5] \times [j-0.5,j+0.5] \times [0.5,1.5] \ (i,j > 1), \\ I_{11k}, \text{ if } (x_1,x_2,x_3) \in [0.5,1.5] \times [0.5,1.5] \times [k-0.5,k+0.5] \ (k > 1), \\ I_{i1k}, \text{ if } (x_1,x_2,x_3) \in [i-0.5,i+0.5] \times [0.5,1.5] \times [k-0.5,k+0.5] \ (i,k > 1), \\ I_{1jk}, \text{ if } (x_1,x_2,x_3) \in [0.5,1.5] \times [j-0.5,j+0.5] \times [k-0.5,k+0.5] \ (j,k > 1), \\ I_{ijk}, \text{ if } (x_1,x_2,x_3) \in [i-0.5,i+0.5] \times [j-0.5,j+0.5] \times [k-0.5,k+0.5] \\ \quad (i,j,k > 1), \end{cases}$$

Fig. 1 shows the difference between both representations in the case of a 2D image.

2.2.3 Projective complexity of a 3D Image

Several complexity measures for 2D images have been proposed [28,29]. Some definitions are based on statistical gray level features (global versus local histograms). Other definitions are based on spatial distributions of gray levels (edges, symmetry, texture). Edges highlight image zones in which there are abrupt changes in luminosity level usually associated with surface discontinuities. The rationale is straightforward: images with more borders contain more objects (or objects with more facets), thus resulting in a greater perceived complexity. This has lead to the following edge based measure of complexity. Complexity is measured by the number of edges per unit area in the image [30]. In the case of 3D images we can extend this definition, for example, complexity can be measured by the number of edge voxels per unit volume. This definition is not suitable for adaptive algorithms based on 1D edge detection methods. Therefore we propose a new definition consistent with the above one.

Consider a continuously defined function $f : [a,b]^3 \to \mathbb{R}$. Let $[\alpha,\beta] \subset [a,b]$ and $S = [\alpha,\beta]^2$. Let $(x_1,x_2) \in S$ be a fixed point in S. Call $e(x1,x2)$ the number of edge points of the function of one variable $f(x_1,x_2,x)$ when $x \in [a,b]$. The projective complexity of f on the set $K = S \times [a,b]$ is defined as

$$PC(f,K) \equiv \frac{\int_S e(x_1,x_2)dx_1 dx_2}{v(S)}. \tag{2}$$

Let d be a positive integer. To approximate the above quantity we subdivide S into d^2 identical squares. Denote the center of the squares by $(x_{1i},x_{2j}),\ 1 \le i,j \le d$. Then (2) can be approximated by

$$\begin{aligned} PC^a(f,K) &= (\sum_{i,j=1}^{d} e(x_{1i},x_{2j})((\beta-\alpha)/d)^2)/(\beta-\alpha)^2 = \\ &= \sum_{i,j=1}^{d} e(x_{1i},x_{2j})/d^2. \end{aligned} \tag{3}$$

Observe that $PC^a(f,K)$ does not depend on the size of the square S. We can consider e as a vector with d^2 components and write (3) as

$$PC^a(f,K) = \|e\|_1 / d^2.$$

In each experiment, the value of d has been almost constant. Then, using the norm equivalence, we can define the following measure of complexity

$$PC^*(f,K) = \|e\|_\infty .$$

Although we have considered sets in the plane x_1x_2, similar definitions can be stated for sets in the planes x_1x_3 or x_2x_3. The above definitions are applicable to 3D images because they can be extended to continuously defined functions using the procedure detailed in Subsection 2.2.2.

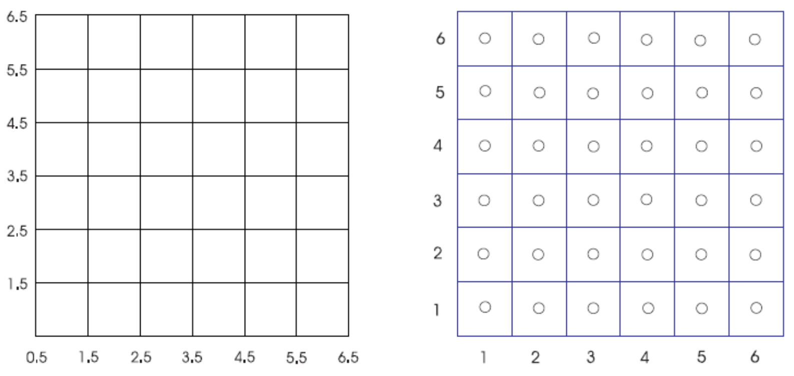


Fig. 1: Continuous representation (left) and discrete representation (right) of a 2D image.

2.2.4 Subdivision and selection procedures

Consider a 3D image given by an $n_1 \times n_2 \times n_3$ array $I = \{I_{ijk}\}$ with indices $(i,j,k) \in [[1,n_1]] \times [[1,n_2]] \times [[1,n_3]]$.

The doubly adaptive algorithm performs split and point selection operations. These procedures are defined on the set of indices, that is, they consider discrete rectangles. First, we describe how a discrete rectangle is divided into four almost equal discrete subrectangles.

Consider the discrete rectangle $\mathscr{R} = [[n,m]] \times [[p,q]]$ with $n < m$ and $p < q$. The subdivision is performed computing $r = n + \lfloor (m-n)/2 \rfloor$ and $s = p + \lfloor (q-p)/2 \rfloor$ ($\lfloor x \rfloor$ denotes the floor function that returns the greatest integer less than or equal to x). The resulting discrete rectangles are

$$\begin{aligned}
\mathscr{R}_1 &= [[n,r]] \times [[p,s]], \\
\mathscr{R}_2 &= [[n,r]] \times [[s+1,q]], \\
\mathscr{R}_3 &= [[r+1,m]] \times [[s+1,q]], \\
\mathscr{R}_4 &= [[r+1,m]] \times [[p,s]].
\end{aligned}$$

Suppose now that we want to select $(d+1)^2$ points from $\mathscr{R}$, distributed regularly. The selected points are obtained as the set

$$\mathscr{N} = \{h[0], h[1], \ldots, h[d]\} \times \{v[0], v[1], \ldots, v[d]\},$$

where the vectors h and v are given by the following algorithm

```
$left = n - 0.5; right = m + 0.5; up = q + 0.5; down = p - 0.5;$
$h[0] = n; h[d] = m; v[0] = p; v[d] = q;$
$for(i = 1; i < d; i++)$
{
    $ht = left + i(right - left)/d;$
    $vt = down + i(up - down)/d;$
    $h[i] = \lfloor ht + 0.5 \rfloor$; if$(ht - \lfloor ht \rfloor == 0.5) h[i] = h[i] - 1;$
    $v[i] = \lfloor vt + 0.5 \rfloor$; if$(vt - \lfloor vt \rfloor == 0.5) v[i] = v[i] - 1;$
}
```

Observe that $\lfloor x + 0.5 \rfloor$ returns the integer nearest to x. The above algorithm obtains a subset of $[[n,m]]([[p,q]])$ whose points are distanced approximately by $(n-m)/d((q-p)/d)$. If $d > n-m$ or $d > q-p$ the subdivision is not possible.

2.2.5 Split criterion

The set of indices of a 3D image is given by

$$\mathscr{M} = [[1,n_1]] \times [[1,n_2]] \times [[1,n_3]].$$

The projections of this set on the coordinate planes are

$$\begin{aligned}
\mathscr{R}_a &= [[1,n_1]] \times [[1,n_2]], \\
\mathscr{R}_b &= [[1,n_1]] \times [[1,n_3]], \\
\mathscr{R}_c &= [[1,n_2]] \times [[1,n_3]].
\end{aligned}$$

Consider a set $\mathscr{R}_\alpha$, $(\alpha = a,b,c)$. Suppose that $\mathscr{R} = [[n,m]] \times [[p,q]]$ is a subset of $\mathscr{R}_\alpha$. The split criterion for $\mathscr{R}$ is defined by the following procedure

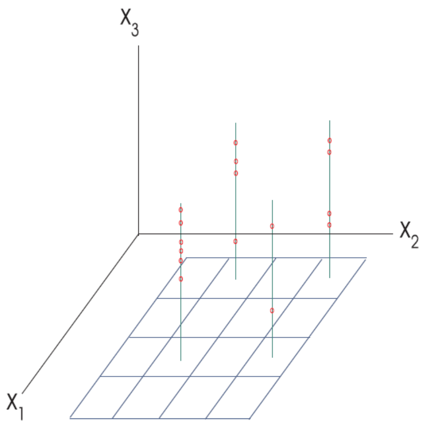


Fig. 2: EDAS-1 obtains edge points (red) lying on the straight lines perpendicular to the considered plane.

Split criterion

```
Step 1. if $(d > n-m || d > q-p)$ return 0;
Step 2. Select $(d+1)^2$ points from $\mathscr{R}$, distributed regularly (Subsection 2.2.4). The result is the set $\mathscr{N}$.
Step 3. Consider the straight lines perpendicular to the plane containing $\mathscr{R}_\alpha$, passing through each point in $\mathscr{N}$.
Step 4. Apply EDAS-1 to each one of these lines to obtain the number of edge points corresponding to each point $\mathbf{k} \in \mathscr{N} (NEP(\mathbf{k}))$; see Fig. 2. Compute the minimum distance between these edge points $(MIND(\mathbf{k}))$. In this step all the edge points found are stored in a file.
Step 5. Compute
        $MAXNEP = \max_{\mathbf{k}\in\mathscr{N}} NEP(\mathbf{k}),$
        $MIND = \min_{\mathbf{k}\in\mathscr{N}} MIND(\mathbf{k}).$
Step 6. If we call $TNEP$ the limit of the maximum number of edge points and $TMIND$ the limit of the minimum distance between edge points, then
    if $((MAXNEP \geq TNEP) || (MIND \leq TMIND)$return 1;
    else return 0;
```

A discrete rectangle is subdivided when the measure of projective complexity corresponding to the discrete rectangle (PC^*) is greater than a fixed beforehand limit or when the minimum distance between edge points is less than a fixed threshold. In this last case, it is possible that the discontinuity surface be non smooth and this makes necessary to consider small discrete rectangles to obtain a detailed description of the jump discontinuity set. Given a discrete rectangle $\mathscr{R}$, the split criterion is implemented with the function SPC. $SPC(\mathscr{R})$ takes the value 1 if the split criterion is satisfied and 0 in the contrary case.

2.3 Doubly adaptive algorithm

Consider the discrete rectangle $\mathscr{R}_\alpha = [[1,n_i]] \times [[1,n_j]]$. Let $\mathscr{R}$ be a discrete rectangle contained in $\mathscr{R}_\alpha$. If $SPC(\mathscr{R}) = 0$ we call $\mathscr{R}$ a good discrete rectangle otherwise $\mathscr{R}$ is called bad.

Doubly Adaptive algorithm for 3D Edge Detection (DA3DED)

Step 1. Consider $\mathscr{R}_\alpha = [[1,n_1]] \times [[1,n_2]]$. Divide $\mathscr{R}_\alpha$ into four discrete rectangles using the procedure described in Subsection 2.2.4. Apply the split criterion to the resulting rectangles. The good rectangles are put into the set $\mathscr{G}_1$ and the bad rectangles put into the set $\mathscr{B}_1$.
Step 2. At each step j we have a set of good discrete rectangles $\mathscr{G}_j$ and a set of bad discrete rectangles $\mathscr{B}_j$. Divide each bad discrete rectangle into four discrete rectangles using the procedure described in Subsection 2.2.4. Test whether these children are good or bad to obtain the sets $\mathscr{G}_{j+1}$ and $\mathscr{B}_{j+1}$.
Step 3. The algorithm stops if $\mathscr{B}_j = \emptyset$. If the stopping criterion is not satisfied, go to Step 2.
Step 4. Repeat the above steps for $\mathscr{R}_b = [[1,n_1]] \times [[1,n_3]]$ and $\mathscr{R}_c = [[1,n_2]] \times [[1,n_3]]$.

DA3DED has been implemented using a quadtree. Below we detail this algorithm.

2.3.1 Algorithm to generate the quadtree

In this subsection we provide a pseudocode of the algorithm used by DA3DED to generate the quadtree. We start by defining a structure of type node (each node is associated with a discrete rectangle $\mathscr{R}$)

```
struct node
{
 int n; int m; int p; int q;
 int spc;
 int pt;
 int ch[4];
 int d;
}Q[ ],
```

where, n,m,p,q are the coordinates of the corners of the discrete rectangle.
spc: indicator variable that takes the value 1 if the algorithm has applied the *split criterion* to the rectangle and 0 otherwise.
pt: number of the node parent of the current one.
$ch[j]$: *jth* child of the current node (-1 in case of leaf nodes). $ch[0]$ is the left child and $ch[3]$ is the right child.
d: determines the subset of $\mathscr{R}$ used by the split criterion.

The algorithm uses alternately two distinct values of d: d_1 and d_2, in order to avoid the repetition of computations. In practice these values are almost equal. We denote by R_c the discrete rectangle associated with the current node c.

Algorithm to generate the quadtree

```
– Input of data:$n_k, n_l, 1 \leq k < l \leq 3, E_1, E_2, TNEP, TDMIN, d_1, d_2, po = 0$.
– Initialize the node 0:
  $Q[0].n = 1, Q[0].m = n_k,; Q[0].p = 1; Q[0].q = n_l; Q[0].spc = 1; Q[0].pt = -1;$
– Split the node 0 into four discrete rectangles and initialize the nodes 1, 2, 3 and 4.
  $Q[j].n, Q[j].m, Q[j].p, Q[j].q, j = 1,\dots,4$, are obtained according to the method described
  in Subsection 2.2.4.
  for$(j = 1; j \leq 4; j++)\{Q[j].spc = 0; Q[j].pt = 0; Q[j].d = d_1;\}$
– Complete the members of the node 0,
  for$(j = 0; j \leq 3; j++)Q[0].ch[j] = j + 1;$
– Initialize the current node and the counter of nodes $c = 1; nn = 4;$
– while$(po < 4)$
  {
    $rch = Q[c].ch[3]$; (right child of the current node)
    if$(Q[c].spc == 0)$ (the split criterion has not been applied to the current node)
    {
        if$(SPC(\mathscr{R}_c) == 1)$ (the current node satisfies the split criterion)
        {
            Split $\mathscr{R}_c$ into four discrete rectangles,
            compute $Q[nn + j].n, Q[nn + j].m, Q[nn + j].p, Q[nn + j].q$,
            $j = 1,\dots,4$, by the method described in Subsection 2.2.4.
            if$(Q[c].d == d_1)DV = d_2;$
            else $DV = d_1;$
            for$(j = 1; j \leq 4; j++)$
            $\{Q[nn + j].spc = 0; Q[nn + j].pt = c;$
            $Q[nn + j].d = DV; Q[c].ch[j - 1] = nn + j;\}$
            $Q[c].spc = 1;$
            $c = nn + 1$; (new current node)
            $nn = nn + 4;$
        }
        else (the current node does not satisfy the split criterion)
        {
            $Q[c].spc = 1;$
            for$(j = 0; j \leq 3; j++)Q[c].ch[j] = -1;$
            $c = Q[c].pt;$
            if$(c == 0)po = po + 1;$
        }
    }
    else (the split criterion has been applied to the current node)
    {
        if$(Q[rch].spc == 1)$(all the childs have been visited. Go to the parent)
        {
            $c = Q[c].pt;$
            if$(c == 0)po = po + 1;$
        }
        else (there exist childs not visited)
        {
            for($j = 1; j \leq 3; j++$) if ($Q[Q[c].ch[j]].spc == 0$)$break$;
            $c = Q[c].ch[j]$;
            $if(c == 0)po = po + 1$;
        }
    }
}
```

3 Results and Discussion

In this section we have tested DA3DED with several synthetic material models. The results have been compared with those obtained with other algorithms: 3D Sobel, 3D Prewitt and 3D EDAS-1. The test models have been the following:

– Internal inclusions within a material.
– Short fiber composites.
– Fracture models.

Materials with internal inclusions are also called particle composites (filled materials). For example, internal pores in aluminium alloys can be obtained by introducing hydrogen in the melt metal. Since the solubility of the hydrogen decreases with the temperature, when the metal solidifies, the gas is rejected and forms gas pores which are easily visualized using X-ray tomography [31]. Icosahedral inclusions have been described in [32]. Short fiber composites are a kind of composite materials with short fibers of materials included in a matrix material. For example, glass or carbon fibers are used to reinforce polymers. The habitual fibers present a circular cross-section. Nowadays composite research is being conducted with more exotic fibers: hollow glass fibers and triangular glass fibers. In the case of fibers with triangular cross sections is difficult to compute the fibre orientations from 2D images. This makes necessary 3D imaging [33]. Finally, we have considered a model of fractured material. The interface of a fracture model is rather difficult to approximate because it presents acute dihedral angles. In fact, many procedures to detect 3D edges assume that the jump discontinuity set is smooth.

To test the algorithms, we have designed examples having different complexity in different zones. The box containing the composite has been divided into eight equal cubes. The inclusions and fibers have been randomly placed in each cube with the following distribution:

– Spherical inclusions: 60 spheres (with radius 8) in one of the cubes, each one of the remaining parts contains 4 spheres.
– Icosahedral inclusions: 60 icosahedra in one of the cubes, each one of the remaining parts contains 4 icosahedra.
– Cylindrical fibers: 30 cylinders in one of the cubes, each one of the remaining parts contains 2 cylinders.
– Prismatic fibers: 30 triangular prisms in one of the cubes, each one of the remaining parts contains 2 triangular prisms.

The 3D images used in the experiments have been generated in two steps. First, we have considered a continuous function with value 1 inside and 0 outside the bodies. In the case of icosahedral inclusions, prismatic fibers and fracture model we have used the algorithm

Table 1: Performance of 3D EDAS-1 on the test models.

3D Image	E_1	E_2	E_3	E_4	E_5	TNI	TNJI	CPU(s)
Spherical inclusions	10^{-1}	10^{-2}	10^{-2}	10	10	4770465	93447	109.8
Icosahedral inclusions	10^{-1}	10^{-2}	10^{-2}	10	10	6123908	229703	156.7
Cylindrical fibers	10^{-1}	10^{-2}	10^{-2}	10	10	4854444	102482	111.8
Prismatic fibers	10^{-1}	10^{-2}	10^{-2}	10	10	4940281	112624	115.1
Fracture model	10^{-1}	10^{-2}	10^{-2}	10	10	8297400	452000	234.4

proposed in [34] to find the distance from a point to a polyhedron. Then, the above continuous function has been discretized using a $200 \times 200 \times 200$ mesh and taking its value at the center of each cell.

The experimental environment has been the following:

– CPU Intel(R)Core(TM) i7-4500U CPU 1.8 GHz.
– Running Software Microsoft Visual C++ 2012.

The test 3D images exhibit a complex structure containing several three dimensional objects. In complex problems, it may be difficult to use deformable model algorithms. Therefore, we have compared DA3DED 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

$$\begin{aligned}
MX(:,:,1) &= [-2\ 0\ 2; -3\ 0\ 3; -2\ 0\ 2],\\
MX(:,:,2) &= [-3\ 0\ 3; -6\ 0\ 6; -3\ 0\ 3],\\
MX(:,:,3) &= [-2\ 0\ 2; -3\ 0\ 3; -2\ 0\ 2].
\end{aligned}$$

The corresponding mask used for the 3D Prewitt method is

$$\begin{aligned}
MX(:,:,1) &= [-1\ 0\ 1; -1\ 0\ 1; -1\ 0\ 1],\\
MX(:,:,2) &= [-1\ 0\ 1; -1\ 0\ 1; -1\ 0\ 1],\\
MX(:,:,3) &= [-1\ 0\ 1; -1\ 0\ 1; -1\ 0\ 1].
\end{aligned}$$

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 [7,35].

The results obtained with the Prewitt method are similar to those obtained with the Sobel method, we have not included them to save space.

3D EDAS-1 consists of applying EDAS-1 to the straight lines, perpendicular to the plane $x_i x_j$, passing through each one of the points of $[[1,n_i]] \times [[1,n_j]]$. This process is performed for each coordinate plane.

The numerical results for 3D EDAS-1 and DA3DED are reported in Tables 1 and 2. We call TNI the total number of intervals and $TNJI$ the total number of intervals containing jumps, generated in the applications of EDAS-1. $E5$ is a parameter of adaptive cubature; see [22]. The results in Table 2 have been obtained with the same values of E_i, $i = 1, \ldots, 5$, than those in Table 1.

The synthetic models are shown in Figs. 3, 6, 9, 12 and 15. A graphic comparison of the results obtained with 3D Sobel, 3D EDAS-1 and DA3DED is shown in Figs. 4, 5, 7, 8, 10, 11, 13, 14, 16, 17 and 18.

Table 2: Performance of DA3DED on the test models.

3D Image	TMIND	TNEP	d_1	d_2	TNI	TNJI	CPU(s)
Spherical inclusions	3	25	10	9	2137515	71586	51.3
Icosahedral inclusions	3	25	10	9	4339122	192821	110.4
Cylindrical fibers	3	25	10	8	2774380	89438	65.8
Prismatic fibers	4	25	6	5	3369707	108608	79.5
Fracture model	4	17	8	7	3983960	338064	124.9

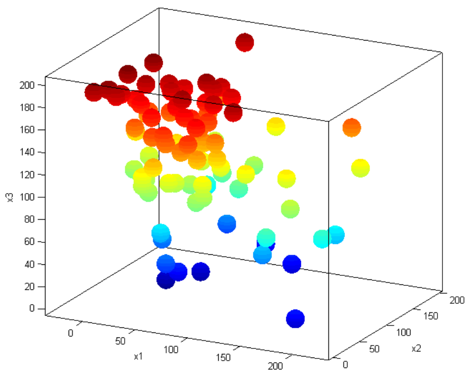
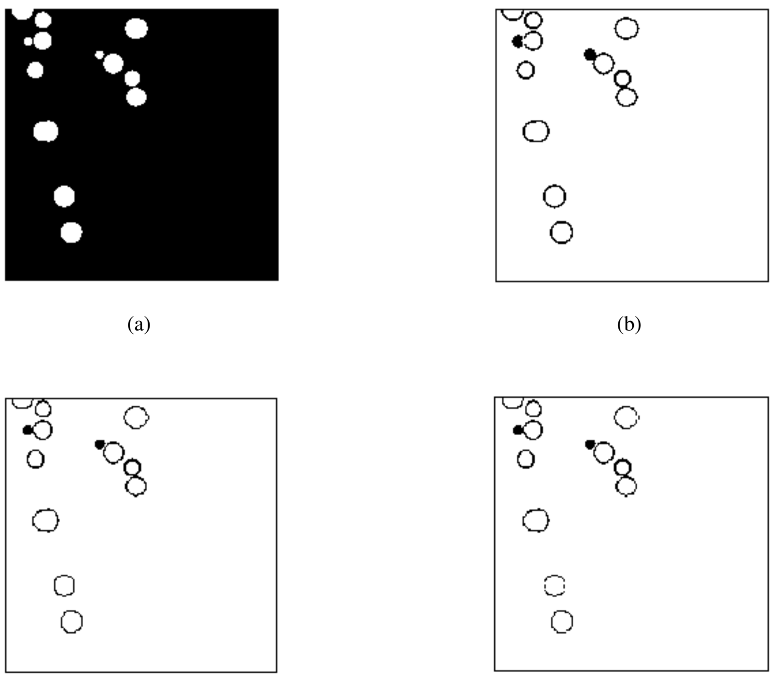


Fig. 3: Spherical inclusions.



(a) (b) (c) (d)

Fig. 4: Spherical inclusions (intersection with the plane $x1$=50). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

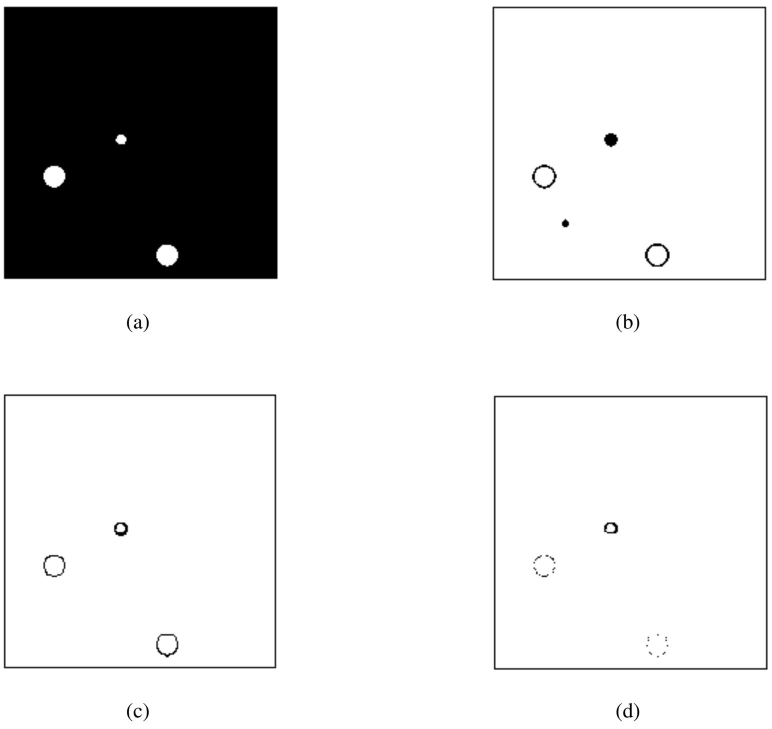


Fig. 5: Spherical inclusions (intersection with the plane x_1=150). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

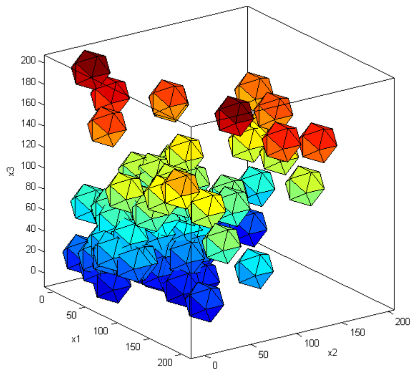


Fig. 6: Icosahedral inclusions.

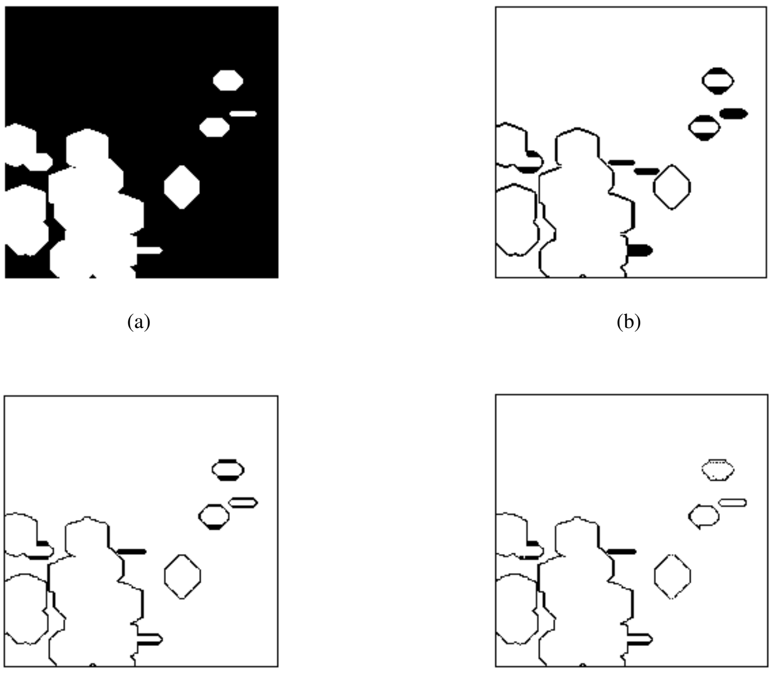
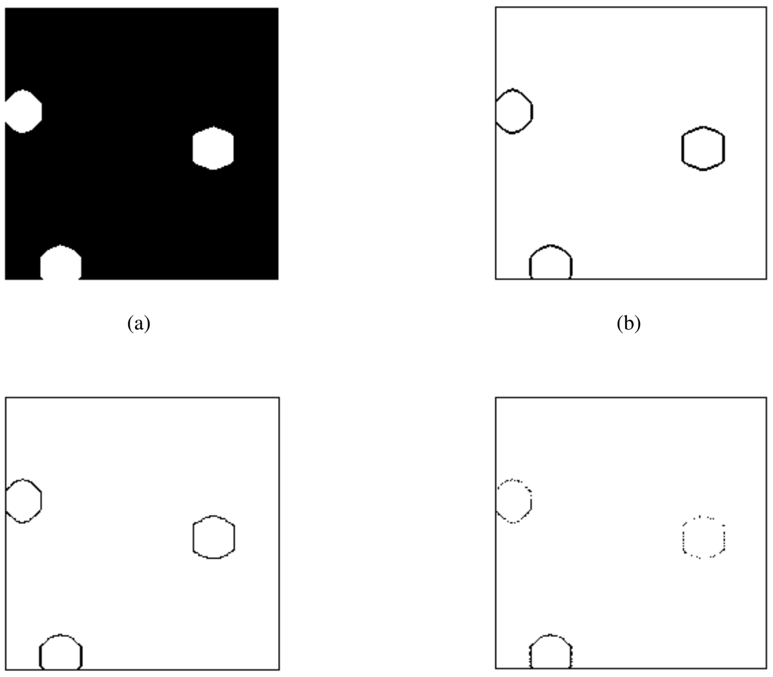


Fig. 7: Icosahedral inclusions (intersection with the plane $x1$=75). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.


(a)

(b)

(c)

(d)

Fig. 8: Icosahedral inclusions (intersection with the plane x_1=150). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

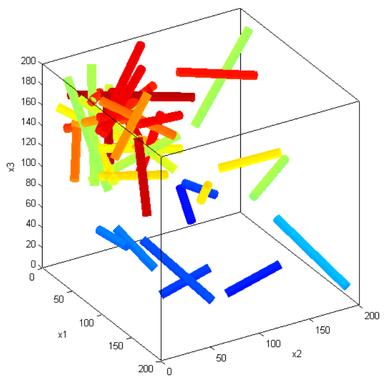


Fig. 9: Cylindrical fibers (circular cross-section).

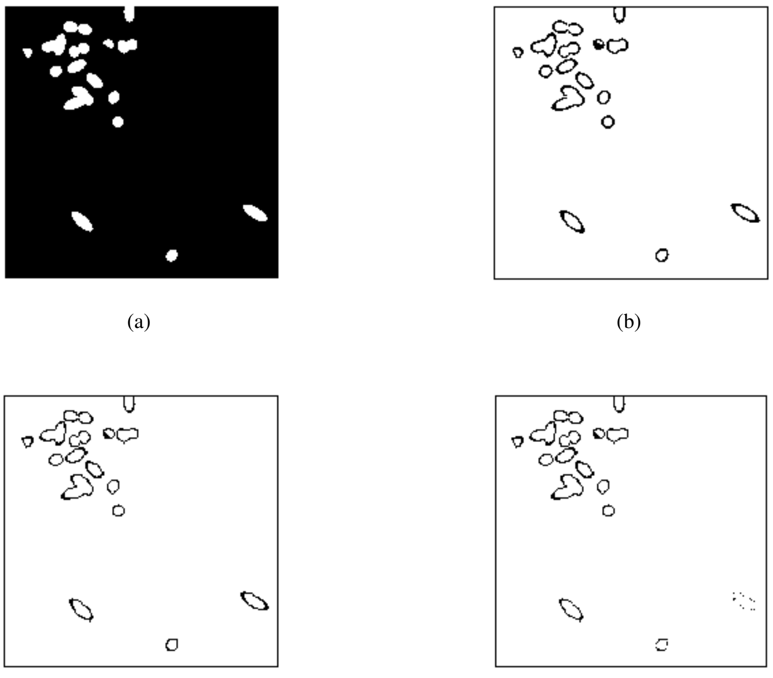


Fig. 10: Cylindrical fibers (intersection with the plane x_1=50). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

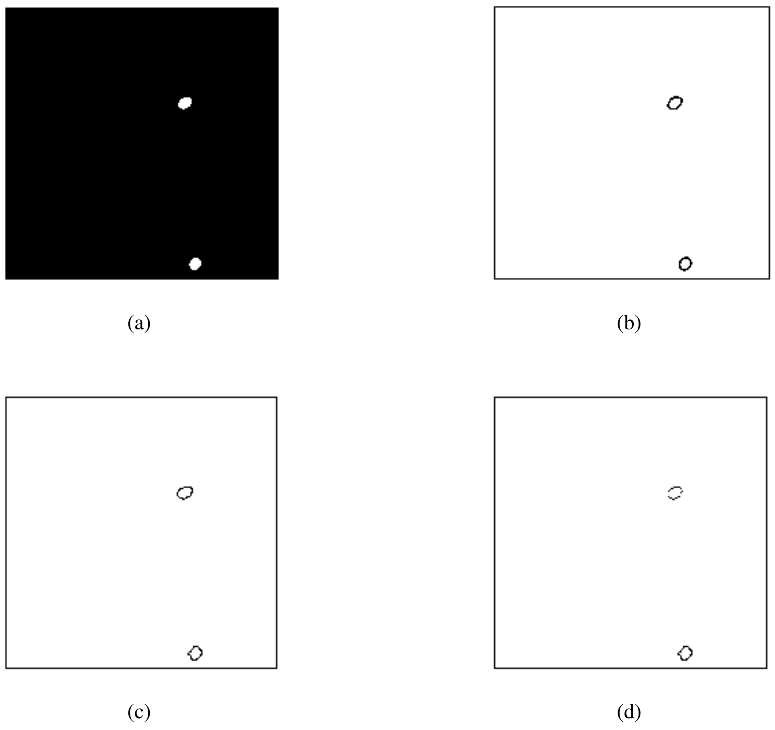

(a) (b) (c) (d)

Fig. 11: Cylindrical fibers (intersection with the plane x_1=110). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

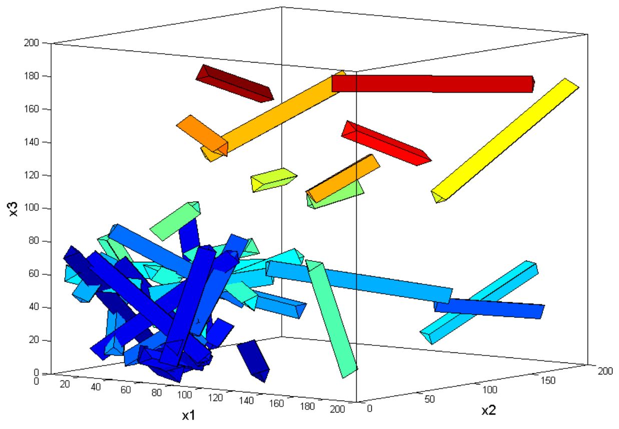


Fig. 12: Prismatic fibers (triangular cross-section).

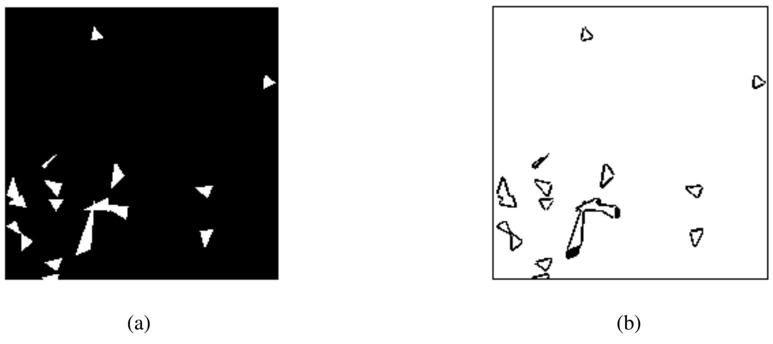
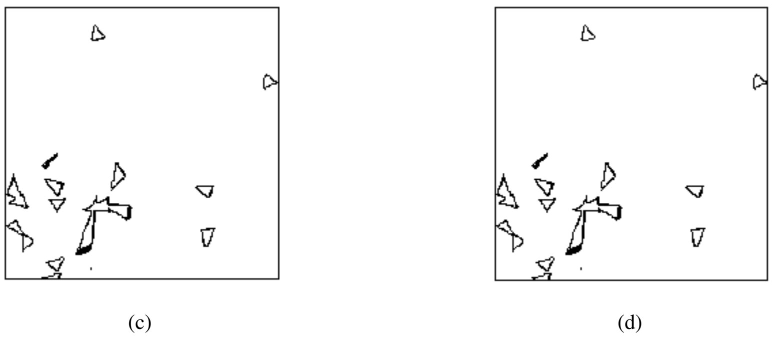



Fig. 13: Prismatic fibers (intersection with the plane x_1=50). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

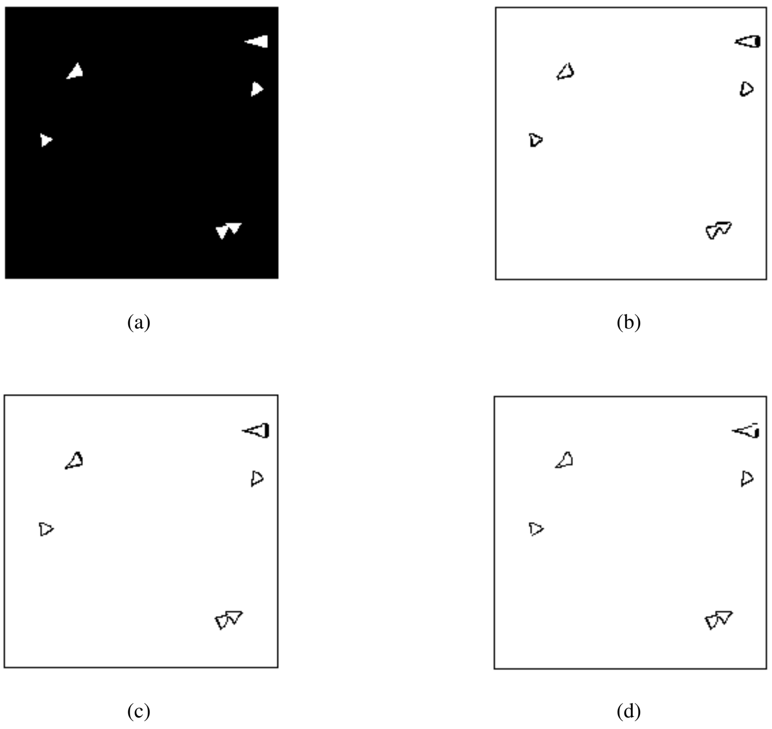


Fig. 14: Prismatic fibers (intersection with the plane x_1=150). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

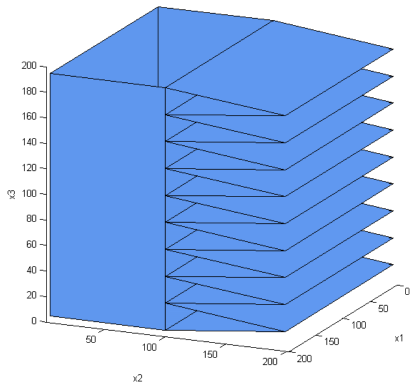


Fig. 15: Fracture model.

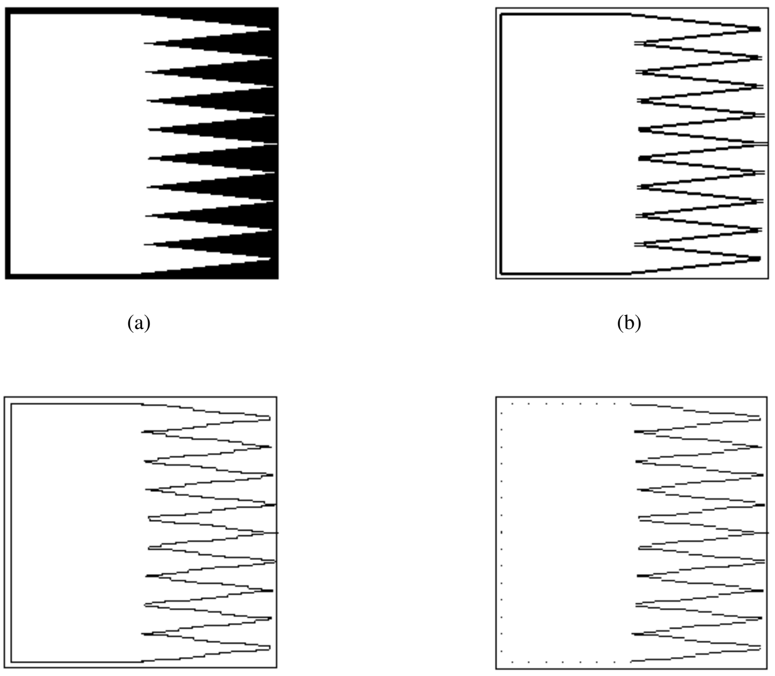


Fig. 16: Fracture model (section perpendicular to the x_1-axis). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

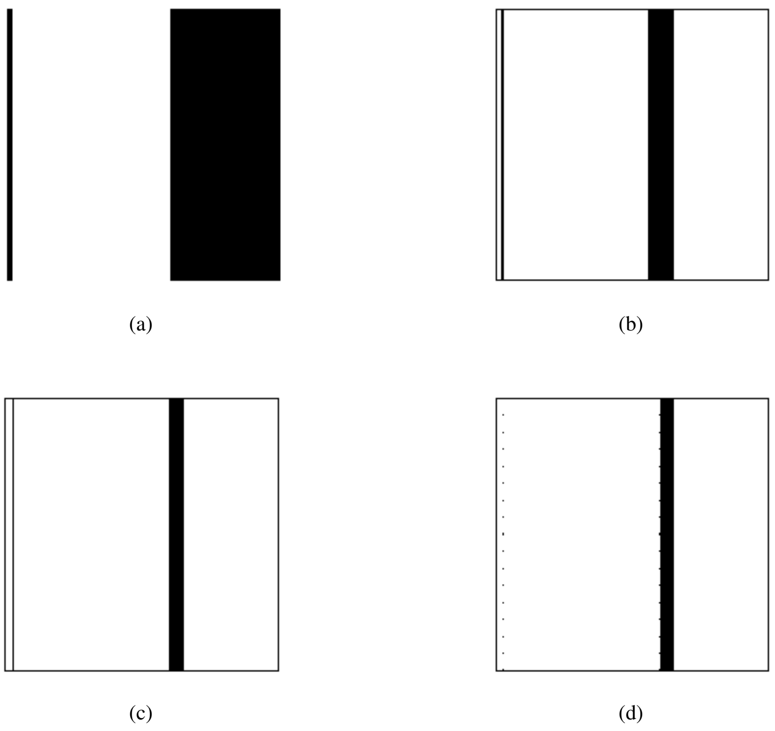


Fig. 17: Fracture model (intersection with the plane x_3=175). (a) Exact intersection, (b) 3D Sobel, (c) 3D EDAS-1, (d) DA3DED.

4 Conclusions

Fast and accurate edge detection in 3D images is necessary in many applications. Some typical examples include, image guided surgery, 3D assisted face recognition, safety and security applications [36], seismic imaging, etc.

The problem is challenging because 3D images involve an enormous amount of data (voxels) that must be processed. Algorithms that provide a simple point based representation of the jump discontinuity set (3D filters) process all the voxels in an image.

However, in most images the complexity varies from a region to another. The proposed algorithm (DA3DED) is based on this fact. The depth of the treatment is proportional to the complexity of the region into consideration.

DA3DED has been tested on 3D images that modelize real problems (composites, fractures). It has been much faster than the 1D edge detection algorithm for 3D images derived from EDAS-1.

The doubly adaptive algorithm provides a thorough description of the edges in complex zones. Regions with low complexity are described in less detail, but in this case the discontinuity surface can be effectively reconstructed using techniques such as interpolation.

References

1. A. Bosnjak, G. Montilla, R. Villegas, I. Jara, 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, Lyon, France, pp. 5263-5266 (2007)
2. Ch. Bauer, T. Pock, E. Sorantin, H. Bischof, R. Beichel, Segmentation of interwoven 3d tubular tree structures utilizing shape priors and graph cuts, Med. Image Anal. 14, 172-184 (2010)
3. L. Qin, H.K. Genant, J.F. Griffith, K.-S. Leung (Eds.), Advanced Bioimaging Technologies in Assessment of the Quality of Bone and Scaffold Materials, Springer (2007)
4. J. Ohser, K. Schladitz, 3D Images of Materials Structures, WILEY-VCH, (2009)
5. R. Archibald, K. Chen, A. Gelb, R. Renaut, Improving tissue segmentation of human brain MRI through preprocessing by the Gegenbauer reconstruction method, Neuroimage 20, 489-502 (2003)
6. R. Archibald, J. Hu, A. Gelb, G. Farin, Improving the accuracy of volumetric segmentation using pre-processing boundary detection and image reconstruction, IEEE Trans. Image Process. 13, 459-466 (2004)
7. J. Toriwaki, H. Yoshida, Fundamentals of Three-Dimensional Digital Image Processing, Springer, Dordrecht, (2009)
8. H.K. Liu, Two- and three-dimensional boundary detection, Computer Graphics and Image Processing 6, 123-134 (1977)
9. Ch. Bänisch, P. Stelldinger, U. Kothe, Fast and accurate 3D edge detection for surface reconstruction, in: J. Denzler, G. Notni, H. Sübe (Eds.), Proc. DAGM 2009, LNCS 5748, Springer, pp. 111-120 (2009)
10. R. Archibald, A. Gelb, J. Yoon, Polynomial fitting for edge detection in irregularly sampled signals and images, SIAM J. Numer. Anal. 43, 259-279 (2005)
11. M. Kass, A. Witkin, D. Terzopoulos, Snakes: Active contour models, Int. J. Comput. Vis. 1, 321-331 (1988)
12. D. Terzopoulos, A. Witkin, M. Kass, Constraints on deformable models: recovering 3D shape and non-rigid motion, Artificial Intelligence 36, 91-123 (1988)
13. T. McInerney, D. Terzopoulos, T-snakes: topology adaptive snakes, Med. Image Anal. 4, 73-91 (2000).
14. N. Barreira, M.G. Penedo, L. Cohen, M. Ortega, Topological active volumes: A topology-adaptive deformable model for volume segmentation, Pattern Recognit. 43, 255-266 (2010)
15. T. Shen, H. Li, Z. Qian, X. Huang, Active volume models for 3D medical image segmentation. In: IEEE Conference on Computer Vision and Pattern Recognition, pp. 707-714 (2009)
16. J.A. Sethian, Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, Cambridge, UK (1999)
17. E. Meinhardt, E. Zacur, A.F. Frangi, V. Caselles, 3D edge detection by selection of level surface patches, J. Math. Imaging Vis. 34, 1-16 (2009)
18. B. Llanas, S. Lantarón, Edge detection by adaptive splitting II. The three dimensional case, J. Sci. Comput. 55, 474-503 (2012)
19. T. Kitasaka, K. Mori, J. Hasegawa, J. Toriwaki, K. Katada, Recognition of aorta and pulmonary artery in the mediastinum using medial-line models from 3D CT images without contrast material, Medical Imaging Technology 20, 572-583 (2002)
20. K. Yokoyama, T. Kitasaka, K. Mori, Y. Mekada, J. Hasegawa, J. Toriwaki, Liver region extraction from 3D abdominal X-ray CT images using distribution features of abdominal organs, Journal of Computer Aided Diagnosis of Medical Images 7, 1-11 (2003)
21. J.S. Suri, D.L. Wilson, S. Laxminarayan (Eds.), Handbook of Biomedical Image Analysis, Volume III: Registration Models, Kluwer Academic / Plenum Publishers, New York (2005)
22. B. Llanas, S. Lantarón, Edge detection by adaptive splitting, J. Sci. Comput. 46, 485-518 (2011).
23. G. Wang, Q.M.J. Wu, Guide to Three Dimensional Structure and Motion Factorization, Springer, Dordrecht (2011)
24. Th. W. Gamelin, R. E. Greene, Introduction to Topology, Dover, Mineola, New York (1999)
25. A. Bliss, F. E. Su, Lower bounds for simplicial covers and triangulations of cubes, Discrete Comput. Geom. 33, 669-686 (2005)
26. D. Orden, F. Santos, Asymptotically efficient triangulations of the d-cube. Discrete Comput. Geom. 30, 509-528 (2003)
27. V.I. Krylov, Approximate Calculation of Integrals, Dover, Mineola, New York (2005)
28. R.A. Peters II, R. N. Strickland, Image complexity metrics for automatic target recognizers, in: 1990 Automatic Target Recognizer Systems and Technology Conference, Naval Surface Warfare Center, Silver Spring, MD, pp. 1-17 (1990)
29. M. Cardaci, V. di Gesú, M. Petrou, M.E. Tabacchi, A fuzzy approach to the evaluation of image complexity, Fuzzy Sets and Systems 160, 1474-1484 (2009)

30. B. Bhanu, Automatic target recognition: state of the art survey, IEEE Trans. Aero. and Elec. Sys. 22, 364-379 (1986)
31. J.Y. Buffiére, S. Savelli, E. Maire, Characterisation of MMCp and cast aluminium alloys, in: J. Baruchel, J.-Y. Buffiere, E. Maine, P. Merle, G. Peix (Eds.), X-Ray Tomography in Material Science, Hermes Science Publications, Paris, pp. 103-113 (2000)
32. Z.Y. Kutikhina, L.N. Larikov, O.A. Shnatko, Influence of icosahedral inclusions on the thermal stability of Al-Fe-Mg system alloys, Metal Science and Heat Treatment 32, 208-210 (1990)
33. A.R. Clarke, C. N. Eberhardt, Microscopy Techniques for Materials Science, Woodhead Publishing and CRC Press (2002)
34. B. Llanas, M. Fernández de Sevilla, V. Feliú, An iterative algorithm for finding a nearest pair of points in two convex subsets of Rn, Comp. Math. Appl. 40, 971-983(2000)
35. D. Wang, D. M. Doddrell, G. Cowin, A novel phantom and method for comprehensive 3-dimensional measurement and correction of geometric distortion in magnetic resonance imaging, Magn. Reson. Imaging 22, 529-542 (2004)
36. A. Koschan, M. Pollefeys, M. A. Abidi (Eds.), 3D Imaging for Safety and Security, Springer (2007)