UNIVERSITY OF GRANADA

Representation and Visualization of Volumetric Data

PHD Thesis of

Dr. Francisco Velasco-Anguita

English Extended Abstract

Octubre de 2002

ADVISOR

Dr. Juan Carlos Torres-Cantero

Software Engineering Department

This Phd Thesis has been supported by the *Ministerio de Ciencia y Tecnología* and by *FEDER* through grant TIC2001-2099-C03-02.

Contents

List of Figures

List of Tables

Chapter 1

A formalization of volumetric models

1.1 Chapter abstract

A volume can be represented by a function $\alpha : V \subset \mathbb{R}^3 \rightarrow \Gamma$ where Γ is a property domain. In this work we present a formalization of such functions. We formalize the property domain Γ classifing into discrete and continuous domains. We then formalize the volumetric model making again a classification in discrete and continuous models. Several results are shown, for example, a representation schema of discrete volumes on the basis of the CSG schema, or operations to transform continuous models in discrete ones.

1.2 Introduction

Volumetric data are usually represented by a set of property values γ_i measured in a set of points $(x_i, y_i, z_i) : i = 1, 2, \cdots, N$. These values are samples from an unknown function $f(x, y, z)$.

In order to operate with the representation of the volume[Brodlie, 01], a function is defined to estimate the property values at the points between the known sam-

ples. We call α to this function. In this chapter we develop a formalization of this function α on the basis of a previous formalization of the domain of property Γ.

There are related works like those of Mallgren[Mallgren, 82], Fiume[Fiume, 89] and Torres[Torres, 93] where several formalizations of graphic objets are developed. However, the Mallgren's proposal is 2D oriented and the concepts he uses are not easily extended to 3D.

Fiume formalizes the visualization process defining a static object as (Z_0, I_0) where $Z_0 \subseteq \mathbb{R}^3$ and $I_0 : Z_0 \rightarrow C$ being C a space of color. His proposal has a handicap, the operators he defines are not boolean operators. His proposal can be used as an abstraction of rendering but not of modeling.

Torres develops a generalization of the concept of object that is proposed by Fiume. He defines a graphic object on the basis of the concepts of presence and aspect.

Presence describes the ocupation of space of a graphic object. It is not only a value of 0 or 1, but a countable value that allow to build objets by superposition (value > 1) or substraction (negative value) of objets.

The aspect of an object defines its appearance and visual properties. It can represent information such as color, opacity, etc.

Our proposal is centered on the formalization of volumetric objects. We consider that the presence just has two allowed values, 0 and 1, and the aspect is split in two domains: the domain of property values that are represented in the volume and the domain of visual attributes that allow us to render an interpretation of the volume. Property domain values can be discrete or coninuous. The study of the volumetric models has also been developed depending on the kind of property domain on which the volume is defined.

Next section presents our formalization of property domains. It is presented by means of a classification of property domains in continuous and discrete. Next section presents the formalization of volumetric models. Operators and a classification of volumetric models in discrete and continuous are also shown. Respect to the discrete volumetric models, a boolean algebra is defined and a representation schema, called constructive discrete-volume geometry, is presented. Respect to the continuous volumetric models, several algebras are shown. A way to define continuous volumetric models from a finite set of samples and an operator to transform continuous volumetric models into discrete ones are also presented.

1.3 Mathematical model of property domain

A property domain is a set that contains all the values that can be represented in a volume. However, a definition like the previous one is poor because if you want to operate with volumes you need to operate with their property values, so it is necessary to define several operations on the property domain that allow us to operate with volumes. Moreover, you can need to represent properties whose values have a discrete variation, like the material, or properties whose values show a continuous variation, like the temperature.

So we have studied the property domains making a classification of them in continuous domains and discrete ones.

1.3.1 Continuous property domain

A continuous property domain can have infinite property values between every two property values.

Definition 1 (Continuous property domain) *We say that* Γ *is a continuous property domain iff it is homeomorfic to* $\mathbb{R}^n$ *for some* $n > 0$ *and it has defined the following operations:*

- $+$*: internal sum, such that* $(\Gamma, +)$ *is an additive group.*
- $*$*: external product on a body K, being* $(\Gamma, +, *)$ *a vectorial K-space.*
- $\times$*: internal product that satisfies the following properties: associative, commutative, and existence of identity element.*

When in a volume you want to represent more than one property, the first step will be to compose the corresponding property domains to obtain just one property domain that, although compound, represents all the properties to be studied in the volume.

The composition operator is called $\uplus$ and it is defined using the cartesian product of the domains to be composed.

Definition 2 (Compound continuous property domain) *Let Γ_1 and Γ_2 be continuous property domains; let $\gamma_{1_1}, \gamma_{1_2} \in \Gamma_1$ and $\gamma_{2_1}, \gamma_{2_2} \in \Gamma_2$ be property values; and let K be a body with $k \in K$.*
The compound continuous property domain $\Gamma = \Gamma_1 \uplus \Gamma_2$ is defined as follows:

The set of values is defined as $\Gamma = \Gamma_1 \times \Gamma_2$

The operators are defined as:
$+ : \Gamma \times \Gamma \to \Gamma$
$(\gamma_{1_1}, \gamma_{1_2}) + (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} + \gamma_{2_1}, \gamma_{1_2} + \gamma_{2_2})$

$\times : \Gamma \times \Gamma \to \Gamma$
$(\gamma_{1_1}, \gamma_{1_2}) \times (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} \times \gamma_{2_1}, \gamma_{1_2} \times \gamma_{2_2})$

$* : K \times \Gamma \to \Gamma$
$k * (\gamma_{1_1}, \gamma_{1_2}) = (k * \gamma_{1_1}, k * \gamma_{1_2})$

Theorem 1 *For any two conitnuous property domains, Γ_1 and Γ_2, the composition $\Gamma_1 \uplus \Gamma_2 = \Gamma$ with the previously defined operations is a continuous property domain.*
Proof:
If Γ_1 and Γ_2 are continuous property domains, then for some m and n, Γ_1 is homeomorfic to $\mathbb{R}^m$ and Γ_2 is homeomorfic to $\mathbb{R}^n$. Then, some continuous functions $f_1 : \Gamma_1 \to \mathbb{R}^m$ and $f_2 : \Gamma_2 \to \mathbb{R}^n$ exist.
If $f : \Gamma_1 \times \Gamma_2 \to \mathbb{R}^{m+n}$ is defined as $f(x,y) = (f_1(x), f_2(y))$, then f is continuous and $\Gamma = \Gamma_1 \times \Gamma_2$ is homeomorfic to $\mathbb{R}^{m+n}$.
It is easy to see that the operators $+, \times, *$ satisfy the conditions required in the definition 1.

1.3.2 Discrete property domain

A discrete property domain has a finite cardinality and is a boolean algebra. Discrete domains are simplified representation of continuous domains.

Definition 3 (Discrete property domain) *Let $\overline{\Gamma}$ a continuous property domain. We say that $\Gamma \subset \overline{\Gamma}$ is a discrete property domain extracted from $\overline{\Gamma}$ iff Γ has finite cardinality and it is a boolean algebra.*

The neutral element is called as 0 *and the universal one as* ω.

Example 1 *From a property domain,* color*, we can extract the discrete property domain* $\{black = 0,\ red, green, blue, cyan, magenta, yellow, white = \omega\}$ *where the operator* $\cup$ *represents the additive mixture of colors and the operator* $\cap$ *represents the substractive mixture of colors.*
The neutral element is the black color and the universal element is the white color. The operators are defined as it is shown in the following tables.
$K = black,\ R = red,\ G = green,\ B = blue,$
$C = cyan,\ M = magenta,\ Y = yellow,\ W = white$

$\cup$	K	R	G	B	C	M	Y	W
K	K	R	G	B	C	M	Y	W
R	R	R	Y	M	W	M	Y	W
G	G	Y	G	C	C	W	Y	W
B	B	M	C	B	C	M	W	W
C	C	W	C	C	C	W	W	W
M	M	M	W	M	W	M	W	W
Y	Y	Y	Y	W	W	W	Y	W
W	W	W	W	W	W	W	W	W

$\cap$	K	R	G	B	C	M	Y	W
K	K	K	K	K	K	K	K	K
R	K	R	K	K	K	R	R	R
G	K	K	G	K	G	K	G	G
B	K	K	K	B	B	B	K	B
C	K	K	G	B	C	B	G	C
M	K	R	K	B	B	M	R	M
Y	K	R	G	K	G	R	Y	Y
W	K	R	G	B	C	M	Y	W

A discrete property domain can be characterized by a subset of values called boolean base of properties.

Definition 4 (Boolean base of properties) *Let* Γ *be a discrete property domain. The set of property values* $\Gamma_b = \{\gamma_0, \gamma_1, \ldots, \gamma_n\} \subseteq \Gamma$ *will be a boolean base of* Γ *iff:*

1. $\forall \gamma_k \in \Gamma \quad \exists \gamma_i, \ldots, \gamma_j \in \Gamma_b$ such that $\gamma_k = \bigcup_{l=i}^{j} \gamma_l$
2. $\bigcup_{\gamma_i \in \Gamma_b} \gamma_i = \omega$
3. $\forall \gamma_i, \gamma_j \in \Gamma_b \ : \ \gamma_i \neq \gamma_j \Rightarrow \ \gamma_i \cap \gamma_j = 0$

For all discrete property domain with two or more elements, a boolean base exists and it is unique.

Theorem 2 *For all discrete property domain* Γ *with two or more values of property, a boolean base of properties exists.*
Proof:
On the basis of [Permingeat, 88]:

1. *A relationship* $a \leq b$ *sii* $a \cap b = a$ *can be defined such that* $(\Gamma, \leq, \sim, \omega, 0)$ *is a boolean reticulum.*
2. *An atom can be defined as an element* $a \neq 0$ *such that* $\forall x \in \Gamma$, $x \cap a = a$ *or* $x \cap a = 0$.
3. *At least an atom exists.*
4. *If* a_1 *and* a_2 *are atoms, then* $a_1 \cap a_2 \neq 0 \Rightarrow a_1 = a_2$.
5. *If* $\{a_i, \ldots, a_j\}$ *is the set of atoms such that* $a_i \leq x \in \Gamma - \{0\}$, *then* $x = a_i \cup \ldots \cup a_j$.

So, the set of atoms of Γ *is a boolean base of* Γ.

Theorem 3 *For all discrete property domain* Γ *with two or more values of property, its boolean base of properties is unique.*
Proof:
Let supose that two boolean bases Γ_{b1} *and* Γ_{b2} *exist. Their definitions are*
$\Gamma_{b1} = \{\gamma_{1_1}, \ldots, \gamma_{1_n}\}$
$\Gamma_{b2} = \{\gamma_{2_1}, \ldots, \gamma_{2_m}\}$
Every γ_{1_i} *can be defined by an union of elements of* Γ_{b2}, *but simultaneously, every* γ_{2_k} *can be defined by an union of elements of* Γ_{b1}. *Then, every* γ_{1_i} *could be defined by an union of elements of* Γ_{b1}, *but this is not possible because* Γ_{b1} *is a boolean base. So, the two bases are the same.*

Once we have proven that every discrete property domain (with more than one element) has just one boolean base, we can use a set of property values as generator set of a discrete property domain.

Theorem 4 *Let $\overline{\Gamma}$ be a continuous property domain, and let $\Gamma_b \subset \overline{\Gamma}$ be a finite set of values that has defined the following operations:*
$\cup : \Gamma_b \times \Gamma_b \to \overline{\Gamma}$
$\cap : \Gamma_b \times \Gamma_b \to \overline{\Gamma}$
$\sim : \Gamma_b \to \overline{\Gamma}$
If for all $x, y, z \in \Gamma_b$, the following are hold:
- $x \neq y \Rightarrow x \cap y = 0$
- $x \cup (y \cup z) = (x \cup y) \cup z$
- $x \cap (y \cap z) = (x \cap y) \cap z$
- $x \cup y = y \cup x$
- $x \cap y = y \cap x$
- $x \cup x = x$
- $x \cap x = x$
- $x \cup (x \cap y) = x$
- $x \cap (x \cup y) = x$
- $x \cup \sim x = \omega$
- $x \cap \sim x = 0$
- $x \cup (y \cap z) = (x \cup y) \cap (x \cup z)$
- $x \cap (y \cup z) = (x \cap y) \cup (x \cap z)$

(where $\omega = \bigcup_{x \in \Gamma_b} x$).
Then, the set $\Gamma = \{\gamma \in \overline{\Gamma} : \gamma = \bigcup_{\gamma_i \in P} \gamma_i \ \ \forall P \in \wp(\Gamma_b)\}$ is a discrete property domain generated by Γ_b. This is represented as $\Gamma = \mathcal{G}(\Gamma_b)$.
Proof:
Trivial.

We have also defined a composition operator for discrete property domain.

Theorem 5 *Let Γ_1 and Γ_2 be two discrete property domains. A new discrete property domain Γ can be defined as composition of them.*
Proof:
A compound boolean base can be built, and then, a discrete property domain can be generated.

Case 1: *Γ_1 and Γ_2 are subset of the same continuous property domain $\overline{\Gamma}$*
Let Γ_{b1} and Γ_{b2} be the boolean bases of Γ_1 and Γ_2. The boolean base of Γ is built as:

1. $\Gamma'_b = \Gamma_{b1} \cup \Gamma_{b2}$
2. *Every couple of values $\gamma_i, \gamma_j \in \Gamma'_b$ such that $\gamma_i \cap \gamma_j = \gamma_k \neq 0$ is replaced by $\gamma_i - \gamma_k$, $\gamma_j - \gamma_k$, and γ_k producing a new set Γ_b. The intersection among every two different values of Γ_b is 0 and so, Γ_b is a boolean base.*

Case 2: *Γ_1 and Γ_2 are subset of different continuous property domains, $\overline{\Gamma_1}$ and $\overline{\Gamma_2}$ respectively.*
The boolean base of Γ is built by the following process:

1. *The continuous domain $\overline{\Gamma}$ is built by composition of $\overline{\Gamma_1}$ and $\overline{\Gamma_2}$.*
2. *The boolean base Γ_b of Γ is built as $\{(x,0) : x \in \Gamma_{b1}\} \cup \{(0,y) : y \in \Gamma_{b2}\}$*
3. *The operations $\cup$, $\cap$ and $\sim$ are defined as:*
   $\cup : \Gamma_b \times \Gamma_b \rightarrow \overline{\Gamma}$
   $(\gamma_{1_1}, \gamma_{1_2}) \cup (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} \cup \gamma_{2_1}, \gamma_{1_2} \cup \gamma_{2_2})$
   $\cap : \Gamma_b \times \Gamma_b \rightarrow \overline{\Gamma}$
   $(\gamma_{1_1}, \gamma_{1_2}) \cap (\gamma_{2_1}, \gamma_{2_2}) = (\gamma_{1_1} \cap \gamma_{2_1}, \gamma_{1_2} \cap \gamma_{2_2})$
   $\sim : \Gamma_b \rightarrow \overline{\Gamma}$
   $\sim (\gamma_{1_1}, \gamma_{1_2}) = (\sim \gamma_{1_1}, \sim \gamma_{1_2})$
   So Γ_b is a boolean base.

Using the set Γ_b, the compound discrete domain $\Gamma = \Gamma_1 \uplus \Gamma_2$ is built as $\Gamma = \mathcal{G}(\Gamma_b)$.

1.3.3 Some operators more

Finally, we present some operators on property domains (continuous or discrete) oriented to:

- Project a continuous domain in a discrete one.
- Define property domains depending on other domains.
- Interpret the property values by means of visual attributes.

The projection operation from a continuous property domain $\overline{\Gamma}$ in a discrete one Γ needs that $\Gamma \subset \overline{\Gamma}$. Then, the operation is defined by an aplication $P : \overline{\Gamma} \rightarrow \Gamma$. This operation allows us to represent a continuous domain by a set with finite cardinality and with boolean operations, however some information will be lost due to the approximation.

The dependence among property domains is implemeted defining functions β from the independent property domain to the dependent one. The dependent domains do not need to be explicitly represented, they are implicitly represented by the dependence functions.

Finally, in order to visualize a volumetric model that represents several properties, it is necessary to do a mapping from the compound domain of property Γ to a domain of visual attributes Λ, like the color or the opacity. This mapping is represented using a function $\nu : \Gamma \longrightarrow \Lambda$, that is called interpretation function. This function help us to understand the meaning of the properties that are being represented in a volumetric model. For instance, the clasification funtions used in volume rendering [Levoy, 88, Westover, 90] use this kind of interpretation functions.

1.4 Mathematical model of volume

In this section we are going to describe the formalization of a volumetric model on the basis of the property domains previously presented.

We will define an equivalence relationship among volumetric models and generic operations among them. Next, we will introduce a classification of volumetric models in discrete models and continuous ones.

Definition 5 (Volumetric model) *Let Γ be a property domain that represents the properties that will be modeled of a volume. We define a volumetric model O_α as:*

$$\alpha : V \subset \mathbb{R}^n \longrightarrow \Gamma$$

where V is closed, bounded and regular.

Example 2 *A solid[Mantyla, 88] is a particular case of volumetric model. Let O_α be a volumetric model with:*

- $n = 3$

- $\Gamma = \{empty, full\}$
- *V rigid set*
- $\alpha(p) = full \ \ \forall p \in V$

Definition 6 (Relationship operator $\circ$) *Let O_{α_1} and O_{α_2} be two volumetric models, with $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ and $\alpha_2 : V_2 \subset \mathbb{R}^n \to \Gamma$*
Let $V_1' = \{p \in V_1 : \alpha_1(p) \neq 0\}$ and $V_2' = \{p \in V_2 : \alpha_2(p) \neq 0\}$

O_{α_1} is related with O_{α_2} ($O_{\alpha_1} \circ O_{\alpha_2}$) iff $V_1' = V_2'$ and $\alpha_1(p) = \alpha_2(p) \ \ \forall p \in V_1' = V_2'$.

Theorem 6 *The relationship operator $\circ$ is an equivalence relationship.*
Proof:
Trivial because the operator $\circ$ is defined on the basis of the operator $=$.

The set of equivalence classes that can be defined from $\mathbb{R}^n$ to Γ is called O_Γ^n. When we refer to a volumetric model O_α, we will be referring to the equivalence class of O_α.

The operators among volumetric models will be defined by operations on Γ and boolean operations on $\mathbb{R}^n$.

Definition 7 (Internal unary operation) *Let O_{α_1} be a volumetric model defined as $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma_1$ and let $op1i$ be an internal unary operation defined on Γ_1.*
We define $O_\alpha = op1i(O_{\alpha_1})$ as:

$$\alpha : V_1 \subset \mathbb{R}^n \to \Gamma_1$$
$$\alpha(p \in V_1) = op1i(\alpha_1(p))$$

Definition 8 (Binary operation on an external body) *Let O_{α_1} be a volumetric model defined as $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma_1$, let K be a body and let $op2e$ be a binary operation defined on Γ_1 and K.*
We define $O_\alpha = K \ op2e \ O_{\alpha_1}$ as:

$$\alpha : V_1 \subset \mathbb{R}^n \to \Gamma_1$$
$$\alpha(p \in V_1) = K \ op2e \ \alpha_1(p)$$

Definition 9 (Internal binary operation) *Let O_{α_1} and O_{α_2} be two volumetric models defined as $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma_1$ and $\alpha_2 : V_2 \subset \mathbb{R}^n \to \Gamma_2$, let $op2i$ be an internal*

binary operation defined on $\Gamma = \Gamma_1 \uplus \Gamma_2$*, and let opb be a regularized boolean operation on* $\mathbb{R}^n$.
We define $O_\alpha = O_{\alpha_1}\ op2i\ O_{\alpha_2}$ *as:*

$\alpha : V \in R^n \rightarrow \Gamma$ *donde*

$V = V_1\ opb\ V_2$

$\Gamma = \Gamma_1 \uplus \Gamma_2$

α_1 *and* α_2 *are extended or restricted in order to have* α'_1 *and* α'_2 *defined from V to* Γ

$$\alpha(p \in V) = \alpha'_1(p)\ op2i\ \alpha'_2(p)$$

Example 3 *Let us define the operation* $+_\cup$*, that is to say, two volumetric models will be operated using the boolean* $\cup$ *in* $\mathbb{R}^n$ *and the internal* $+$ *in* Γ.

Let $\alpha_1 : V_1 \subset \mathbb{R}^n \rightarrow \Gamma_1$ *and* $\alpha_2 : V_2 \subset \mathbb{R}^n \rightarrow \Gamma_2$ *be the two volumetric models.*

Γ *is defined as* $\Gamma_1 \uplus \Gamma_2$.

V *is defined as* $V_1 \cup^* V_2$.

α_i *is redefined as* $\alpha'_i(p \in V) = \begin{cases} \alpha_i(p) & \text{if } p \in V_i \\ 0 & \text{if } p \notin V_i \end{cases}$

Then, the result α *is defined as:*

$\alpha : V \rightarrow \Gamma$

$$\alpha(p \in V) = \alpha'_1(p) + \alpha'_2(p)$$

To continue formalizing the volumetric models they have been classified in discrete and continuous on the basis of the property domain where they are defined.

1.4.1 Discrete volumetric model

Discrete volumetric models will be characterized for being defined on a discrete property domain and for being composed of constant volumetric models. A constant volumetric model will be a model that presents the same property value in all the volume.

In this section we will use boolean operations among discrete volumetric models. In order to define these operations, it is necessary to be able to extend the α

functions to construct the new composed volume V, ant to operate the two functions α_1 and α_2.

The extension of α_i is done as:

$$\alpha_i'(p) = \begin{cases} \alpha_i(p) & \text{if } p \in V_i \\ 0 & \text{if } p \notin V_i \end{cases}$$

The operation $\cup$ is defined as:

$V = V_1 \cup V_2$

$\alpha(p) = \alpha_1'(p) \cup \alpha_2'(p)$

The operation $\cap$ is defined as:

$V = V_1 \cap V_2$

$\alpha(p) = \alpha_1'(p) \cap \alpha_2'(p)$

The operation $-$ is defined as:

$V = V_1$

$$\alpha(p) = \begin{cases} \alpha_1'(p) & \text{si} \quad p \notin i(V_2) \\ \alpha_1'(p) - \alpha_2'(p) & \text{si} \quad p \in i(V_2) \end{cases}$$

where $i(V)$ computes the interior of V.

Before defining the discrete volumetric models we are going to define the constant volumetric models.

Definition 10 (γ-constant volumetric model) *Let Γ be a discrete property domain, and let $\gamma \in \Gamma$ be a property value.*
We define a γ-constant volumetric model as:

$\alpha : V \subset \mathbb{R}^n \to \Gamma$ *where* $\forall p \in V \Rightarrow \alpha(p) = \gamma$

The function α will be called as α_γ, and a γ-constant volumetric model will be called as O_γ. When the property belongs to a boolean base Γ_b the model will be called Γ_b-constant volumetric model.

It can be easily proved that the previously defined boolean operations, when they are applied among constant volumetric models, satify the asociative, conmutative, idempowerful, absorbent, existence of complement element and distributive properties.

On the basis of this constant volumetric models, the discrete volumetric models are defined.

Definition 11 (Discrete volumetric model) *Let Γ be a discrete property domain and let Γ_b its boolean base of properties.*
A volumetric model O_α will be a discrete volumetric model iff there are constant volumetric models $O_{\gamma_i}, \cdots, O_{\gamma_j}$ such that:

1. $\{\gamma_i, \ldots, \gamma_j\} \subseteq \Gamma_b$.
2. $O_\alpha = \bigcup_{l=i}^{j} O_{\gamma_l}$

The set of all discrete volumetric models that can be defined from $\mathbb{R}^n$ to Γ is called $O^n_{\Gamma_d}$.

This set of discrete volumetric models together with the boolean operators previously defined is a boolean algebra.

1.4.1.1 Boolean algebra of discrete volumetric models

The following lemmas prove that the boolean operations among discrete volumetric models produce as result a discrete volumetric model.

Lemma 1 *The union of discrete volumetric models produces as result a discrete volumetric model.*
Proof:
Let O_{α_1} and O_{α_2} be two discrete volumetric models, the union of both is

$$O_\alpha = O_{\alpha_1} \cup O_{\alpha_2}$$

As O_{α_1} is a discrete volumetric model, there are

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

constant volumetric models such that

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

where Γ_b is a boolean base of Γ and

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

With O_{α_2} defined in a similar way. Then, we can write

$$O_{\alpha} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}} \cup O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}}$$

So, the result is composed for a union of Γ_b-constant volumetric models.

Lemma 2 *The intersection of discrete volumetric models produces as result a discrete volumetric model.*
Proof:
Let O_{α_1} and O_{α_2} be two discrete volumetric models, the intersection of both is

$$O_{\alpha} = O_{\alpha_1} \cap O_{\alpha_2}$$

If O_{α_1} is a discrete volumetric model, there are

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

constant volumetric models such that

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

where Γ_b is a boolean base of Γ and

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

With O_{α_2} defined in a similar way. Then, we can write

$$O_{\alpha} = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap (O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}})$$

Using the distributive property

$$O_{\alpha} = \bigcup (O_{\gamma_{1_k}} \cap O_{\gamma_{2_l}})$$

$\forall O_{\gamma_{1_k}}$ belonging to the components of O_{α_1} and $\forall O_{\gamma_{2_l}}$ belonging to the components of O_{α_2}.
As the intersection among Γ_b-constant volumetric models is a Γ_b-constant volumetric model or an empty model, we can say that the result is composed for a union of Γ_b-constant volumetric models.

Before proving the difference among discrete volumetric models, we are going to prove that the difference among constant volumetric models produces a discrete volumetric model, and the difference among a discrete volumetric model and a constant volumetric model produces a discrete volumetric model.

Lemma 3 *The difference among constant volumetric models produces a discrete volumetric model.*
Proof:
Let O_{γ_1} *and* O_{γ_2} *be two constant volumetric models defined as* $\alpha_1 : V_1 \to \Gamma$ *and* $\alpha_2 : V_2 \to \Gamma$

The difference among both as it is defined in the page 12 is $O_\alpha = O_{\gamma_1} - O_{\gamma_2}$ *where*
$\alpha(p) = \begin{cases} \gamma_1 & si \quad p \notin i(V_2) \\ \gamma_1 - \gamma_2 & si \quad p \in i(V_2) \end{cases}$
being $i(V)$ *the interior of* V.

O_α *can be written as* $O_\alpha = O_{\alpha_a} \cup O_{\alpha_b}$ *where* $\alpha_a : V_1 - V_2 \to \Gamma$ *with* $\alpha_a(p) = \gamma_1$ *and* $\alpha_b : V_1 \cap V_2 \to \Gamma$ *with* $\alpha_a(p) = \gamma_1 - \gamma_2$ *and so, it is a discrete volumetric model.*

Lemma 4 *The difference among a discrete volumetric model and a constant volumetric model produces a discrete volumetric model.*
Proof:
Let O_{α_1} *be a discrete volumetric model and let* O_γ *be a constant volumetric model, the difference among both is*

$$O_\alpha = O_{\alpha_1} - O_\gamma$$

If O_{α_1} *is a discrete volumetric model, there are*

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

constant volumetric models such that

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

being Γ_b *a boolean base of* Γ *and*

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

Then, O_α can be written as:

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) - O_\gamma$$

Sustituting the difference by the intersection with the complementary we have

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap \sim O_\gamma$$

Using the distributive property we have

$$O_\alpha = \bigcup (O_{\gamma_{1_k}} \cap \sim O_\gamma)$$

Sustituting the intersection with the complementary by the difference we have

$$O_\alpha = \bigcup (O_{\gamma_{1_k}} - O_\gamma)$$

O_α is defined by the union of discrete volumetric models, so O_α is a discrete volumetric model.

Lemma 5 *The difference among discrete volumetric models is a discrete volumetric model.*
Proof:
Let O_{α_1} and O_{α_2} be two discrete volumetric models, the difference among both is

$$O_\alpha = O_{\alpha_1} - O_{\alpha_2}$$

If O_{α_1} is a discrete volumetric model, there are

$$O_{\gamma_{1_i}}, \ldots, O_{\gamma_{1_j}}$$

constant volumetric models such that

$$\{\gamma_{1_i}, \ldots, \gamma_{1_j}\} \subseteq \Gamma_b$$

being Γ_b a boolean base of Γ and

$$O_{\alpha_1} = O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}$$

With respect to O_{α_2} is similar. So O_α can be written as:

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) - (O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}})$$

Sustituting the difference by the intersection with the complementary we have

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap \sim (O_{\gamma_{2_i}} \cup \ldots \cup O_{\gamma_{2_j}})$$

The complementary of an union of objets is equal to the intersection of the objets' complementary

$$O_\alpha = (O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap (\sim O_{\gamma_{2_i}} \cap \ldots \cap \sim O_{\gamma_{2_j}})$$

Using the asociative property we have

$$O_\alpha = \underline{((O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) \cap \sim O_{\gamma_{2_i}})} \cap (\sim O_{\gamma_{2_k}} \cap \ldots \cap \sim O_{\gamma_{2_j}})$$

Sustituting inside the underlining parenthesis the intersection with a complementary by the difference we have

$$O_\alpha = \underline{((O_{\gamma_{1_i}} \cup \ldots \cup O_{\gamma_{1_j}}) - O_{\gamma_{2_i}})} \cap (\sim O_{\gamma_{2_k}} \cap \ldots \cap \sim O_{\gamma_{2_j}})$$

That is to say, inside the underlining parenthesis we have the difference among a discrete volumetric model and a constant volumetric model, which produces a discrete volumetric model. Repeating the process by means of the associative property with the other constant volumetric models belonging to the descomposition of O_{α_2}, the differences are done one by one getting as result a discrete volumetric model.

On the basis of the previous lemmas, we can define a boolean algebra of discrete volumetric models.

Theorem 7 (Boolean algebra of discrete volumetric models) *The quadruple $(O^n_{\Gamma_d}, \cup, \cap, \sim)$ is a boolean algebra.*
Proof:
We have proven that the operations are closed. We should prove that the operations satisfy the properties that characterize the boolean algebras. This properties are satisfied because the operations among discrete volumetric models are based on operations among constant volumetric models that satisfy such properties.

1.4.1.2 Constructive Discrete-Volume Geometry

On the basis of the boolean algebra of discrete volumetric models, we are going to present a representation schema of discrete volumes. It is called *Constructive Discrete-Volume Geometry* (CDVG). It is based on the Constructive Solid Geometry schema [Requicha, 82], which represents a solid representing the building process of the solid from geometric primitives using regularized boolean operations.

It is rather used to model solids [Requicha, 80] because of its expressive power, validity, non ambiguity, use easiness by means of languages. Moreover, it is concise and closed by the boolean operations [Mantyla, 88].

It considers that solids just have one property value, that is to say, it just represents homogeneous solids. Using the formalization presented about discrete volumetric models, solids with more than one property value can be represented. Our proposal allows us to represent heterogeneous solids (discrete volumes).

By a CDVG, every primitive represents a geometry and a property value. Primitives are constant volumetric models. Operanting with primitives by the previously defined boolean operations a representation of a discrete volume is got.

A CDVG tree is defined as:

<CDVG tree> ::=
<primitive with property value> |
<CDVG tree><boolean operation><CDVG tree> |
<transformation><CDVG tree>

Figure 1.1 represents a 2D discrete volume using a CDVG.

1.4.2 Continuous volumetric model

Continuous volumetric models will be characterized for being defined on a continuous property domain.

Definition 12 (Continuous volumetric model) *A volumetric model O_α with $\alpha : V \subset \mathbb{R}^n \rightarrow \Gamma$ will be a continuous volumetric model iff Γ is a continuous property domain.*

The set of all continuous volumetric models that can be defined from $\mathbb{R}^n$ to Γ is called $O^n_{\Gamma_c}$.

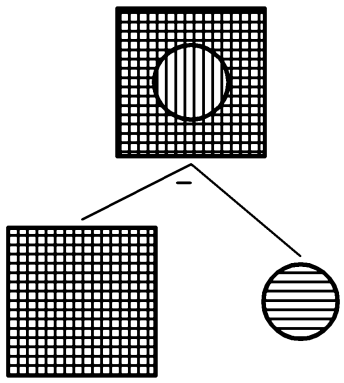


Figure 1.1: CDVG of a 2D discrete volume

Defining several operations on $O^n_{\Gamma_c}$, some algebras of continuous volumetric models can be defined.

1.4.2.1 Algebras of continuous volumetric models

The algebras will be defined on the basis of the internal operators $+_\cup$ and $\times_\cup$; and on the basis of the external operator $*$ on a body K. The operators are defined as it is previously shown in this section. Moreover, some particular objects will be needed.

Definition 13 (Neutral element for $+_\cup$) *The neutral class of equivalence for the operation $+_\cup$ is the class that contains the object O_α with $\alpha : \mathbb{R}^n \to \Gamma$ defined as $\alpha(p) = 0 \ \forall p$ where 0 is the neutral element for the operator $+$ on the domain of property. It is called as O_0.*

Definition 14 (Opposed element for $+_\cup$) *Given a volumetric object O_α, the opposed element for $+_\cup$ is defined as O_{α^-} where $\alpha^-(p) = -\alpha(p) \ \ \forall p$. The element $-\gamma$ represents the opposed element of γ for the operator $+$ on the property domain.*

Theorem 8 *The pair $(O^n_{\Gamma_c}, +_\cup)$ is an additive group.*
Proof:
The asociative and conmutative properties are satisfied because $+_\cup$ is defined on the

basis of the operators $\cup$ on $\mathbb{R}^n$ and $+$ on Γ, which satisfy such properties. Let's prove the existence of the neutral and opposed element.

Given O_{α_1} a volumetric model where $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ the operation

$$O_{\alpha_1} +_\cup O_0$$

is equal to an O_α where

$$\alpha(p) = \begin{cases} 0 & if \quad p \notin V_1 \\ \alpha_1(p) + 0 & if \quad p \in V_1 \end{cases}$$

Then, $O_\alpha \circ O_{\alpha_1}$.

With respect on the opposed element, the operation

$$O_{\alpha_1} +_\cup O_{\alpha_1^-}$$

is equal to an O_α where

$$\alpha(p) = \alpha(p) - \alpha(p) = 0$$

then, $O_\alpha \circ O_0$.

Theorem 9 *Given a body K and the external operator $*$ on K, the trio $(O^n_{\Gamma_c}, +_\cup, *)$ is a vectorial K-space.*
Proof:
Trivial, $(O^n_{\Gamma_c}, +_\cup)$ is an additive group, and the operator $*$ on O_{Γ_c} and K is defined on the basis of $*$ on Γ and K. So, the properties to be a vectorial K-space are satisfied.

Definition 15 (Identity element for $\times_\cup$) *The identity equivalence class for the operator $\times_\cup$ is the class that contains the object O_α such that $\alpha : \mathbb{R}^n \to \Gamma$ with $\alpha(p) = 1 \ \ \forall p$ where 1 is the identity element for $\times$ on Γ. It is called as O_1.*

Theorem 10 *The internal product $\times_\cup$ on $O^n_{\Gamma_c}$ satisfies the associative, conmutative and existence of identity element properties.*
Proof:
The associative and commutative properties are satisfied because $\times_\cup$ on $O^n_{\Gamma_c}$ is defined on the basis of $\cup$ on $\mathbb{R}^n$ and $\times$ on Γ, which satisfy such properties. Let's prove the existence of identity element.

Given O_{α_1} a volumetric model where $\alpha_1 : V_1 \subset \mathbb{R}^n \to \Gamma$ the operation

$$O_{\alpha_1} \times_{\cup} O_1$$

is equal to an O_α where

$$\alpha(p) = \begin{cases} 0 \times 1 = 0 & si \quad p \notin V_1 \\ \alpha_1(p) \times 1 = \alpha_1(p) & si \quad p \in V_1 \end{cases}$$

Then, $O_\alpha \circ O_{\alpha_1}$.

1.4.2.2 Some operations more

In this section we are going to show how to define a continuous volumetric model from a finite set of samples, how to simplify it and how to transform continuous volumetric models in discrete ones.

Usually, the function α is not known, having just a finite set of samples V as volume representation, for example the data that are got from a computed tomography. In these cases, the function

$$\alpha : V \subset \mathbb{R}^n \to \Gamma$$

is not a volumetric model because V is not regular. However, a volumetric model can be built from V by the function

$$\overline{\alpha} : \overline{V} \subset \mathbb{R}^n \to \Gamma$$

where:

- $\overline{V}$ is the convex hull of V.
- $\overline{\alpha}$ is interpolation function that allow us to estimate property values between the known points. It must satisfy the condition $\overline{\alpha}(p) = \alpha(p)\ \forall p \in V$. Usually it is a n-linear interpolation.

Some times, in order to reduce the storage requeriments, only a subset of samples $W \subset V$ is represented. We have studied how this subset must be chosen to minimize the error that is produced with the reduction.

From W, a volumetric model can be defined as

$$\overline{\alpha}' : \overline{W} \subset \mathbb{R}^n \rightarrow \Gamma$$

but the following conditions must be satisfied to reduce the error of aproximation:

- W must have the same convex hull than V, that is to say, $\overline{W} = \overline{V}$
- All the points that have a local minimun or a local maximun in V must be in W.

This error can be computed as:

$$error = \frac{1}{Card(V)} \sum_{p \in V} |\overline{\alpha}'(p) - \alpha(p)|$$

On the basis of this idea of simplification of volumetric models a representation schema was presented in [Velasco, 01a], where it is shown as example the representation of a human head with a space reduction of 20% and an error of 0.14 units. In this particular example, the error is measured visualizing the volume by isosurface extraction and estimating the average deviation between isosurfaces. One unit is the distance between two consecutive samples.

Finally, we are going to present an operation to transform a continuous volumetric model in a discrete one, using the projection functions that can be defined from continuous property domains to discrete ones.

Let Γ_c be a continuous property domain, let $\Gamma_d \subset \Gamma_c$ be a discrete property domain, let Γ_{d_b} be its boolean base of properties and let $P : \Gamma_c \rightarrow \Gamma_d$ be a projection from Γ_c to Γ_d.

Let $\alpha_c : V \subset \mathbb{R}^n \rightarrow \Gamma_c$ be a continuous volumetric model. A function

$$\alpha_d : V \subset \mathbb{R}^n \rightarrow \Gamma_d$$

could be defined as

$$\alpha_d(p) = P(\alpha_c(p)) \;\; \forall p \in V$$

but we could not assure that α_d is a discrete volumetric model because we do not know whether it can be built by an union of constant volumetric models or not. But we can use it to define a discrete volumetric model from it:

1. Several sets $V_i = \{p \in V : \alpha_d(p) \cap \gamma_i = \gamma_i\} \ \ \forall \gamma_i \in \Gamma_{d_b}$ are defined.
2. The sets V_i of the previous step are regularized to get regular sets V_i'.
3. Several constant volumetric models
$$\alpha_i' : V_i' \subset \mathbb{R}^n \rightarrow \Gamma_d$$
are defined as
$$\alpha_i'(p) = \gamma_i \ \ \forall p \in V_i'$$
4. Then, a discrete volumetric model can be defined as
$$O_\alpha' = \bigcup_i O_{\alpha_i'}$$

1.5 Chapter conclusions

In this chapter, we have presented a formalization of volumetric models. This formalization has been developed using the function $\alpha : V \subset \mathbb{R}^n \rightarrow \Gamma$ as representation of a volume, where Γ is the domain of property values that are represented by the volume.

The formalization has been developed in two steps, first we have formalized the domain of properties defining several operations and defining several algebras. A classification of domains of properties has been developed: discrete domains and continuous ones.

We have formalized the volumetric models defining several operations and defining several algebras. A classification of volumetric models has been developed: discrete volumetric models and continuous ones.

A boolean algebra of discrete volumetric models has been defined, and a representation schema that is based on it has been presented: Constructive Discrete-Volume Geometry (CDVG).

Respect to the continuous volumetric models, we have formalized how to build a continuous volumetric model from a finite set of samples, and how to transform a continuous volumetric model in a discrete one.

As future work, we plan to formalize mixed domains of properties, continuous and discrete, as well as to formalize mixed volumetric models.

Chapter 2

Cell Octrees

2.1 Chapter abstract

Today it is usual to have volumes represented by several millions of data due to the improvements of scaners. So it is neccesary to have a representation scheme which allows us to manipulate them quickly. In this chapter we propose a multiresolution representation scheme, called cell octree, which represents the volume using the highest resolution where it is necessary and a low resolution in the areas of the volume where the represented information verifies some user-defined criterion. So the volume is only represented at the highest resolution where it is really usefull.

The method is oriented to render the model using marching cubes. The representation, which is independent on the threshold, is built in such a way that it guarantees that the isosurfaces are crack-free.

2.2 Introduction

A volume, in a general way, can be mathematically represented as a function (with several restrictions) from the Euclidean space $\mathbb{R}^3$ towards the property set that are being represented [Velasco, 03].

When the volume is obtained from a 3D scanner, for example a TAC (tomog-

raphy axial computerised) from a patient, just a set of samples is got as representation of it.

In order to have a good representation, a large data set is necessary, so it is necessary to have a data structure which allows us to handle the information quickly.

Wilhelms et al. propose a data structure called bono (Branch On Need Octree) that indexes the data set, being this data set arranged as an isotropic, regular, rectilinear and structured grid [Wilhelms, 92]. A bono represents the data set as a cubic cells grid together with a tree that indexes the grid. When the volume is to be visualized, a isosurface is extracted from the volume acording to a threshold property value. Grey nodes store the maximum and minimum property values of its cells.

The bono structure allows to access quickly to the cells that are crossed by the isosurface for a given threshold. The isosurface is built by marching cubes [Lorensen, 87].

However, data sets are larger and larger, so it is necessary to have a data structure that needs less storage space. In this chapter we propose a data structure called *Cell Octree* which reduces the storage space taking into account the regularity of the information.

> **Voxel vs. Cell.** We call *voxel* to a cell which represents just one constant property value for all the points inside it. The clasical octrees [Meagher, 80] are based on voxels whereas the structure we are proposing is based on *cells*. A cell stores eight property values at its eight vertices, so it represents a continuous property values field inside it.

Next section shows the cell octree scheme defining several user-dependent criteria to build it. Section four shows the results we have got on real and synthetic volumes and the scheme is compared with other approaches. The final section shows the conclusions of this chapter and outlines some works for the future.

2.3 Cell octree

The idea is to study the cells and find out which cells can be grouped whitout changing the generate isosurface. Where they verify some conditions, eight cells are grouped into just one, as it is shown in the figure 2.1.

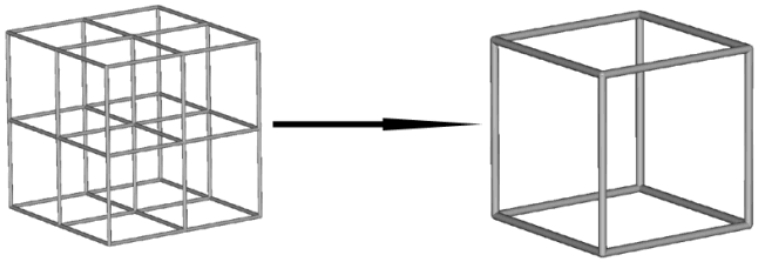


Figure 2.1: Unique allowed way of grouping cells

We use a bono as start point on which, grouping is implemented prunning the tree.

Before presenting the cell octree, let us see the main characteristics of the bono structure (the figure 2.2 shows a 2D bono):

- The internal nodes of the tree can have 8 sons (4, 2 or 1 for the singular cases).
- The tree indexes the grid independently on its size.
- The number of internal nodes is minimal.
- Its leaf nodes represent 8 cells (4, 2, or 1 for the singular cases).
- It has all its leaf nodes in the same level.

The singular cases appear when the size of at least one dimension of the bounding box is not a power of two or when the size of the three dimensions of the bounding box is not the same. These sigular nodes represent up to three faces that are sharing a corner of the bounding box. In the figure 2.2, the upper and right edges produce singular nodes.

A cell octree conserves all the characteristics of a bono less the last one, a cell octree can have leaf nodes at every level. As the size of the cell represented by a leaf node depends on its level, a cell octree can represent cells of different size.

Another difference is that a cell octree can have leaf nodes that represent just one cell (not only for singular cases). We will call it an OneCellLeafNode when a

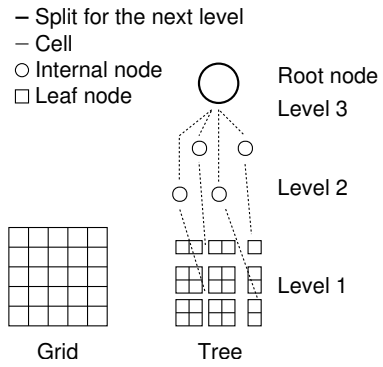


Figure 2.2: Grid and tree for a 2D bono

non-singular leaf node represents just one cell, and about MoreCellsLeafNode for a leaf node that represents more than one cell.

The process to build a cell octree consists on traversing a bono in a bottom-up way and every time that the eight cells represented by a leaf node (MoreCellsLeafNode) verify the prunne criterion the leaf node is transformed into an OneCellLeafNode, and every time that eight brother OneCellLeafNode exist, theirs father internal node is transformed into a MoreCellsLeafNode. See the figure 2.3.

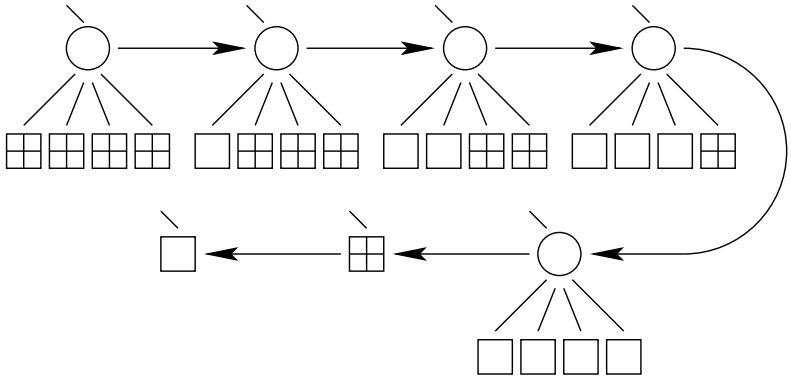


Figure 2.3: Prunning process for a 2D cell octree

The figure 2.3 shows a part of a 2D grid (16 cells) which is represented by an internal node and four MoreCellsLeafNodes. In the first step, the cells represented

by one of these leaf nodes verify the prune criterion and it is transformed into OneCellLeafNode, the following three steps show the transformation of the other three MoreCellsLeafNodes into three OneCellLeafNode. Then, there is an internal node that is father of four OneCellLeafNode, so these leaf nodes are prunned and the internal node is transformed into a MoreCellsLeafNode. Again the four cells represented by this leaf node are tested and if they verify the prunne criterion (as in the figure) the MoreCellsLeafNode is transformed into an OneCellLeafNode. After these six steps this area of the 2D grid is now represented by one larger cell.

The figure 2.4 shows a 2D grid (a), the 2D bono for the grid (b), a possible 2D cell octree (cell quadtree) built from the previous bono (c), and the non regular grid that is represented by the cell quadtree.

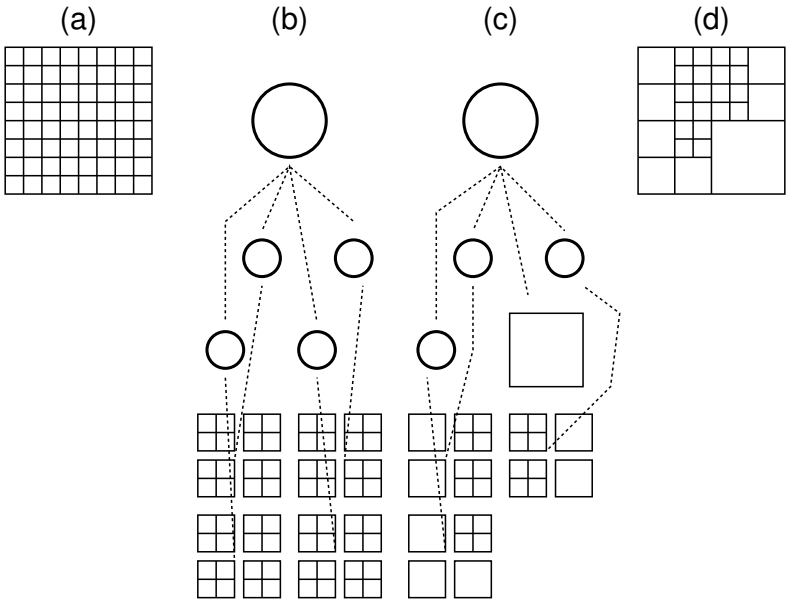


Figure 2.4: Cell quadtree from a 2D grid

By a cell octree, a volume can be represented using less amount of information and by an user-defined prunning criterion. In the following sections we will present several prunning criteria which have been defined thinking on visualizing the volume by isosurface extraction methods. However, the prunning criteria are threshold independent, so the cell octree does not need to be re-built every time the

threshold changes.

In the rest of the thesis, we will refer to the grouping operation and to the prunning operation indistinctly.

2.3.1 Definitions

Let us define some geometric elements. See the figure 2.5.

c-group: *Candidate group*. A group of eight cells which are tested in order to group them into one larger cell.

c-face: *Candidate face*. A group of four faces on the same plane which belong to the boundary of a c-group. If a c-group is grouped, its six c-faces will be the six faces of the larger cell.

c-edge: *Candidate edge*. A group of two edges on the same line which belong to the boundary of a c-face. If a c-group is grouped, its twelve c-edges will be the twelve edges of the larger cell.

Together with these elements there will be six kind of vertices.

In a c-edge we will distinguish two kind of vertices: one *c-edge central vertex* and two *c-edge final vertices*.

In a c-face we will distinguish two kind of vertices: one *c-face central vertex* and four *c-face corner vertices*.

In a c-group we will distinguish two kind of vertices: one *c-group central vertex* and eight *c-group corner vertices*.

Finaly, a vertex v is *positive* for a particular threshold value γ if $F(v) \geq \gamma$ and *negative* if $F(v) < \gamma$ [Lorensen, 87]. $F(v)$ is the function that returns the property value at the vertex v.

2.3.2 Basic prunning criterion

The main idea is to group eight cells into just one when this operation does not lead a non-desired error. That is to say, the error between the volume represented by a

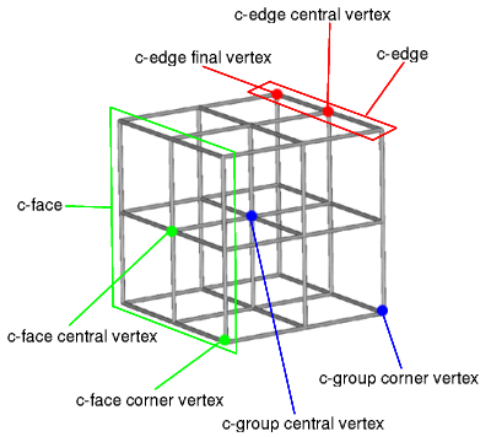


Figure 2.5: C-group and other definitions

grouped cell and the volume represented by the eight cells before grouping must be small enough as to be allowed by the user.

As the visualization method we use is by isosurface extraction, let us analize which errors on the isosurface can be arisen when eight cells are grouped. Mainly the errors are due to the values that are not accessed yet in a grouped cell and that continue being accessed in the smaller joint cells (blue points in the figure 2.6).

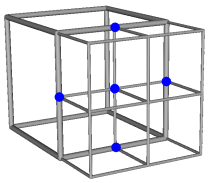


Figure 2.6: Vertices that are an error source

Let us see the non-desired error (see figure 2.7, where red points are positive points). Let E be a c-edge, if its c-edge central vertex value is greater than the values at its both c-edge final vertices or if it is lesser than them, then the situation that is shown in the figure 2.7(a) can arise. In this figure we can see for a particular threshold value there are isosurface inside the small cells but there is not any isosurface inside

the larger cell, so the isosurface has a hole whose border is rendered in the figure in red. This error is due because the c-edge central vertex is positive where the c-edge final vertices are negative.

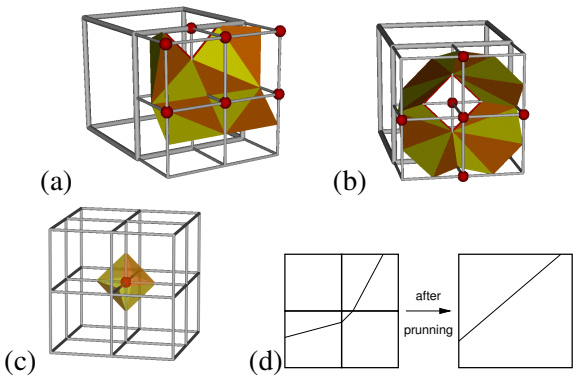


Figure 2.7: Non-desired errors

In order to avoid holes as the one shown in the figure 2.7(a) the equation 2.1 must be true for every c-edge of the c-group. Obviously, the c-group of the large cell of the figure 2.7(a) did not make true the equation and it should not have been grouped.

$$\begin{array}{lcl} [(F(c-edge\ final\ vertex_1) & \leq & F(c-edge\ central\ vertex)) \wedge \\ (F(c-edge\ central\ vertex) & \leq & F(c-edge\ final\ vertex_2))] \\ & \vee & \\ [(F(c-edge\ final\ vertex_1) & \geq & F(c-edge\ central\ vertex)) \wedge \\ (F(c-edge\ central\ vertex) & \geq & F(c-edge\ final\ vertex_2))] \end{array} \quad (2.1)$$

Another non-desired error is the one shown in the figure 2.7(b).

In this case, for a particular threshold the c-face central vertex is positive while the four c-face corner vertices are negative, so there is isosurface outside the large cell but there is not inside it.

To avoid this kind of holes the equation 2.2 must be true for every c-face of the c-group. Let us supose that the condition 2.1 is true for every c-edge of the c-face.

Then, the condition is,

$$\begin{array}{rcl} F(c-face\ central\ v.) & \leq & F(\max\{c-face\ corner\ v.\}) \\ & \wedge & \\ F(c-face\ central\ v.) & \geq & F(\min\{c-face\ corner\ v.\}) \end{array} \quad (2.2)$$

Another non-desired error is the losing of parts of the isosurface. Let us supose a c-group as the one shown in the figure 2.7(c). For a particular threshold there is a part of isosurface around the c-group central vertex.

If this c-group is grouped, this part of the isosurface will be lost. So another condition is set to avoid these situations. The c-group central vertex can not have the minimum or the maximum property value among the 27 property values of the 27 vertices of the c-group. Let us supose that the conditions 2.1 and 2.2 are true for every c-edge and c-face of the c-group. Then, the condition is,

$$\begin{array}{rcl} F(c-group\ central\ v.) & \leq & F(\max\{c-group\ corner\ v.\}) \\ & \wedge & \\ F(c-group\ central\ v.) & \geq & F(\min\{c-group\ corner\ v.\}) \end{array} \quad (2.3)$$

Finnaly, when a prune is done, the isosurface built inside a grouped cell for a particular threshold can be very different of the isosurface which would be built inside the c-group before grouping it. This is so because the isosurface inside a grouped cell is built using 8 vertices instead of the 27 vertices which would be used in the eight cells of the c-group. See a 2D example in the figure 2.7(d).

So there is one condition more.

Let $\Delta\gamma$ be the maximum deviation allowed for a vertex' property value. It is defined by the user.
Let $F(v)$ be the property value known for the vertex v.
Let $F'(v)$ be the property value estimated for the vertex v in a grouped cell.
Then, the condition is

$$\begin{array}{l} \forall c - edge\ central\ vertex\ \ v \in c - group \\ \quad |F(v) - F'(v)| \leq \Delta\gamma \\ \forall c - face\ central\ vertex\ \ v \in c - group \\ \quad |F(v) - F'(v)| \leq \Delta\gamma \\ \forall c - group\ central\ vertex\ \ v \in c - group \\ \quad |F(v) - F'(v)| \leq \Delta\gamma \end{array} \tag{2.4}$$

So, the criterion is defined as follows:

Basic prunning criterion

A c-group will be grouped iff:

1. The condition 2.1 is true for every c-edge of the c-group.
2. The condition 2.2 is true for every c-face of the c-group.
3. The condition 2.3 is true for the c-group central vertex of the c-group.
4. Once the user has defined the allowed error $\Delta\gamma$, the condition 2.4 is true for every c-edge central vertex, every c-face central vertex and the c-group central vertex of the c-group.

By the use of this criterion, prunnes are done avoiding some non-desired errors, however grouped cells can be ambiguous. A cell is ambiguous when for a particular threshold there is one configuration of positive and negative vertices but different topologies of isosurface are possible. To determine which concrete topology must be built, several computations must be done [Chernyaev, 95, Lopes, 99, Cignoni, 00].

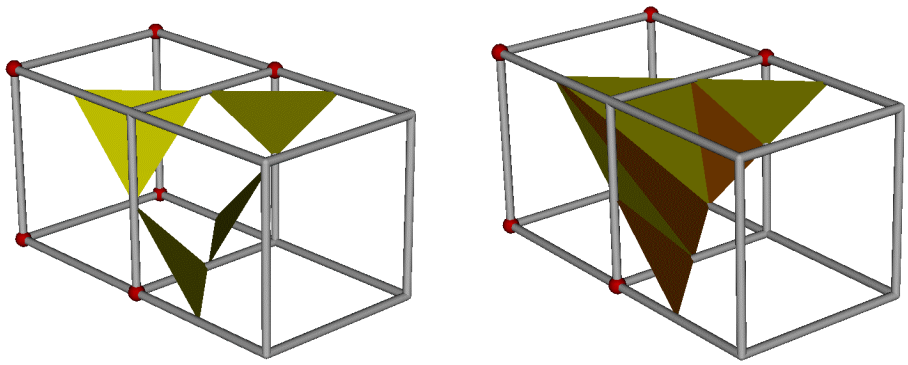
In order to decrease this kind of computations, a new condition can be added to the basic prunning criterion with the goal of avoiding the grouped cells to be ambiguous for all thresholds.

Next section shows a criterion which is based on this one and avoids ambiguous grouped cells.

2.3.3 Monotonous prunning criterion

The ambiguous cells are those which have an ambiguous face, that is to say, a face which have 2 positive vertices in a diagonal and 2 negative vertices in the other diagonal, see figure 2.8. A cell will also be ambiguous, even when its faces are not

ambiguous, if it has 2 positive vertices in a main diagonal and the other vertices are negatives, see figure 2.9. In both figures (2.8 and 2.9) the cases (a) and (b) show different possible triangulations.



(a) (b)

Figure 2.8: Two triangulations for a cell within an ambiguous face

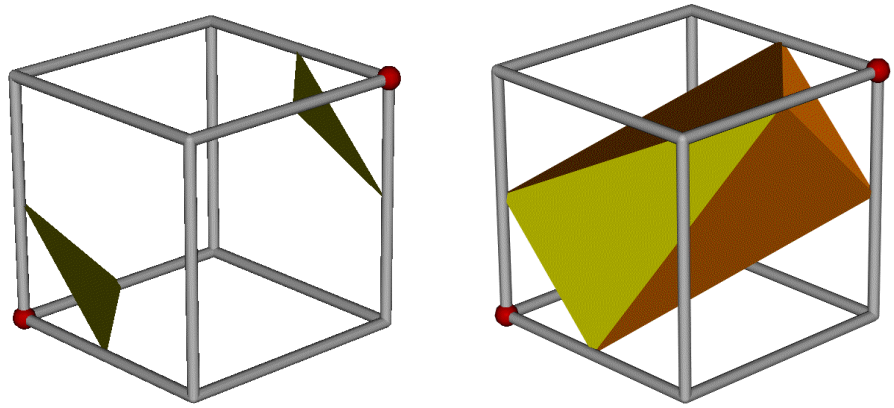
In order to increase the prunne criterion to avoid ambiguous grouped cells we will define the concept of *monotonous cell*, a cell will be monotonous if it is not ambiguous for any thershold. Before defining a characterization of this concept we must define some previous concepts.

Definition 16 (Monotonous face)
Let c be a face and $\mathcal{F}(c)$ its boundary.

c is monotonous iff $F_{|\mathcal{F}(c)}(x,y,z)$ only has one minimum and one maximum.

Theorem 11
Let c be a face.
If c is monotonous then c is unambiguous for all threshold.
Proof:
Let us supose that there is a threshold γ so that c is ambiguous, then c has two positive vertices in a diagonal and two negative vertices in the other diagonal, so $F_{|\mathcal{F}(c)}(x,y,z)$ would have 2 maximums and 2 minimums and c would not be monotonous.

Definition 17 (Potentialy cut vertex)



(a) (b)

Figure 2.9: Two triangulations for a ambiguous cell without ambiguous faces

A vertex v_0 of a cell c will be called potentialy cut *if one of the following conditions is true (see figure 2.10*(a)*):*

1. $F(v_0) < \min\{F(v_1), F(v_2), F(v_3)\}$
2. $F(v_0) > \max\{F(v_1), F(v_2), F(v_3)\}$

where v_1, v_2 y v_3 are vertices which share some edge with v_0.

A potentialy cut vertex has an associated active interval, which is

$$[F(v_0), \min\{F(v_1), F(v_2), F(v_3)\}] \tag{2.5}$$

for the condition 1, or

$$[\max\{F(v_1), F(v_2), F(v_3)\}, F(v_0)] \tag{2.6}$$

for the condition 2.

Theorem 12
A cell within a potentialy cut vertex will be crossed for every isosurface whose threshold belongs to its associated active interval. The potentialy cut vertex will be on a

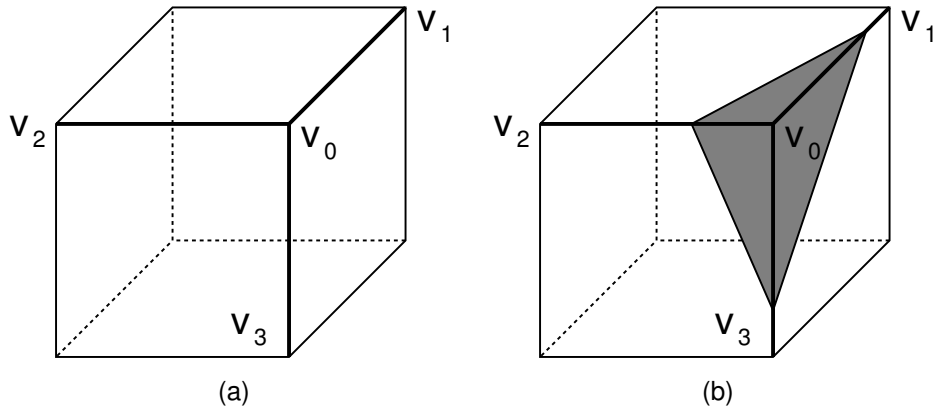


Figure 2.10: Potentialy cut vertex

side of the isosurface and the other vertices will be on the oposite side. See figure 2.10(b).
Proof:
Trivial. For all threshold that belongs to the associated interval, the sign of v_0 *will be different to the sign of* v_1, v_2 *and* v_3.

Definition 18 (Non monotonous diagonal)
Let c be a cell, let d be a main diagonal of c.
d is non monotonous if its ending vertices are potentialy cut vertices and the intersection of their associated intervals is non empty. This intersection set is called N_d.

A cell whose faces are monotonous but within a non monotonous diagonal d will have a configuration as the one shown in the figure 2.9, in particular for every threshold that belongs to N_d.

Definition 19 (Monotonous diagonal)
Every main diagonal that can not be labelled as non monotonous *will be defined as monotonous diagonal.*

Now, we can define the concept *monotonous cell* on the basis of the previous definitions.

Definition 20 (Monotonous cell)
A cell will be monotonous if its 6 faces are monotonous and its 4 main diagonals also are monotonous.

Theorem 13
Let c be a cell. If c is monotonous then there is no threshold so that c is ambiguous.
Proof:
Trivial.

So, if the user wants to avoid the grouped cells to be ambiguous for all thresholds, a new condition has to be added to the basic prunning criterion getting a new criterion defined as follows:

Monotonous prunning criterion

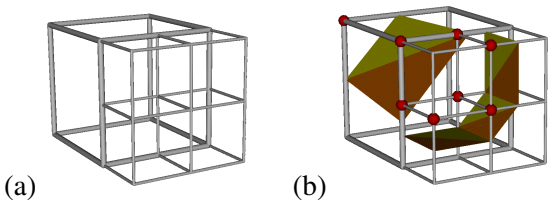
A c-group will be grouped iff:

1. The c-group observes the basic prunning criterion conditions.
2. The cell after grouping is monotonous.

Next section shows another criterion, in this case the goal is to avoid cracks on the isosurface when there are joined cells of different size.

2.3.4 Non cracks prunning criterion

In a cell octree is usual to have joined cells of different size (see figure 2.11(a)).



(a) (b)

Figure 2.11: Joined cells of different size and a possible crack on the isosurface inside them

In these cases some cracks as the one shown in the figure 2.11(b) can be arisen in the boundary between cells of different size.

This happens because the isosurface on a side of that boundary is computed using 9 vertices whereas on the other side is computed using only 4 vertices.

There are different methods to cover these holes. It can be done drawing a polygon [Shu, 95], moving the vertices of the *small triangles* towards the *larger one* [Shekhar, 96], or replacing some triangle in the *large cell* by a fan of triangles which are put to fill the hole [Westermann, 99].

In these methods the process is done after a particular threshold is set and after the triangles have been built. We propose a method which directly produces the isosurface without cracks and without post-process.

The basic idea is to modify the property value of some vertices, so that when the isosurface is built inside a large cell and inside the joined smaller cells, both parts of isosurface coincide on the boundary between those cells. This modification would be done just once, so new computations will not be needed every time the threshold changes.

Let us see what is happen on the boundary between one large cell and four smaller cells.

Usually, the function $F(x,y,z)$ used to interpolate the interior of a cell is the trilinear interpolation, then when $F(x,y,z)$ is restricted to a face, the function $G(s,t)$ which interpolates the interior of a face is a bilinear interpolation which depends on the four corners of the face. So, the boundary between one large cell and four small cells is modelled by five bilinear interpolations. One from the large cell (v_1,v_3,v_7,v_9) and four from the small cells: (v_1,v_2,v_4,v_5), (v_2,v_3,v_5,v_6), (v_4,v_5,v_7,v_8), and (v_5,v_6,v_8,v_9). See figure 2.12.

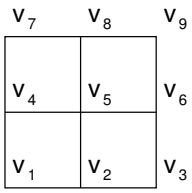


Figure 2.12: Boundary between cells of different size

The first step would be to make the four small bilinear interpolations coincide

with the large bilinear interpolation by the modification of the property value at some vertices, as it is shown in the following equations:

$$\begin{aligned} F(v_2) &\leftarrow \frac{F(v_1)+F(v_3)}{2} \\ F(v_4) &\leftarrow \frac{F(v_1)+F(v_7)}{2} \\ F(v_6) &\leftarrow \frac{F(v_3)+F(v_9)}{2} \\ F(v_8) &\leftarrow \frac{F(v_7)+F(v_9)}{2} \\ F(v_5) &\leftarrow \frac{F(v_1)+F(v_3)+F(v_7)+F(v_9)}{4} \end{aligned} \tag{2.7}$$

However this does not avoid the cracks because the isosurface is approximated as triangles with different resolution for the large and the small cells. The isosurface restricted to a face is an isocurve which is approximated by a polyline as it is shown in the figure 2.13, where it is approximated by one segment in the large cell and until three segments in the small cells, so the crack is still arisen.

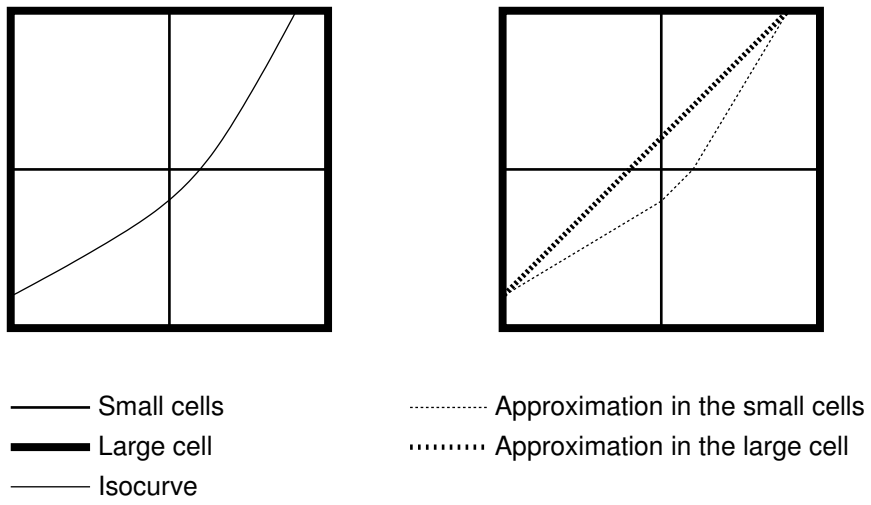


Figure 2.13: Approximation of an isocurve

So, the property value $F(v_5)$ must be modified on the basis of the approximation of the isocurves. There are three different cases of isocurves, let us see them.

For these cases, when we refer to *the bilinear surface* we are refering to the bilinear of the large cell, which determines how $F(v_5)$ must be modified to make coincide the approximation of the isocurve on the small faces with the approximation of the isocurve on the large face.

Case 1: Flat bilinear

In this case, the representation of the bilinear surface 1is a plane (see figure 2.14). All the isocurves are straight lines, so $F(v_5)$ can be modified as it was shown in the equation 2.7. No cracks will be arisen on this face for all threshold.

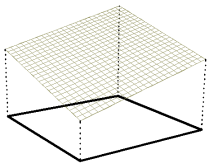


Figure 2.14: Plot of a bilinear for the case 1

Case 2: Isocurve between two contiguous edges

Let us supose that for all threshold the isocurve is plot between two continguous edges, so the bilinear (figure 2.15(a)) would be approximated by two triangles (figure 2.15(b)).

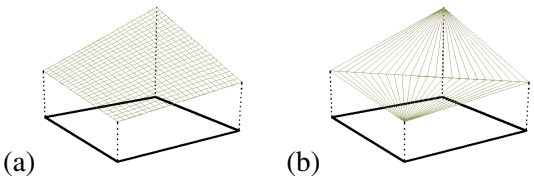


Figure 2.15: Plot of a bilinear for the case 2 and its approximation

In this case $F(v_5)$ must be modified with the average value of the ending values of the diagonal where the two triangles are joined.

But the figure shows the ideal case where the two ending vertices of such a diagonal have the same value. When these vertices have different values (a and b), for all threshold in the opened interval $]a,b[$ its isocurve will not be between two contiguous edges.

In general, if a and b are the values of the ending vertices of a diagonal of the face and c and d are the values of the ending vertices of the other diagonal of the face, $\forall \gamma \in]a,b[\cap]c,d[$ the isocurve will not be in this case. The isocurve will be between two opposed edges.

Case 3: Isocurve between two opposed edges

In this case (see figure 2.16(a)), to make both isocurves coincide, $F(V_5)$ must be modified according to the equation 2.8

$$F(v_5) \leftarrow \frac{\gamma_0 - F(v_4)}{x_a} + F(v_4) \tag{2.8}$$

where

$$x_a = \frac{\frac{\gamma_0 - F(v_1)}{F(v_2) - F(v_1)} + \frac{\gamma_0 - F(v_7)}{F(v_8) - F(v_7)}}{2}$$

So the new value for $F(V_5)$ depends on the threshold γ_0. Moreover, when the isocurve crosses three small cells (see figure 2.16(b)) $F(V_5)$ must be modified using two different values simultaneously, one of them must be used to compute the correct position of V_a and the other one must be used to compute the correct position of V_b.

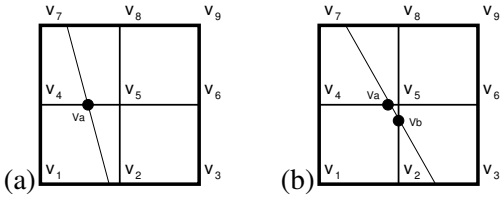


Figure 2.16: Approximation of an isocurve between two opposed edges. (a) crossing 2 small cells. (b) crossing 3 small cells

This case is not independent on the threshold and a preprocess would be necessary every time the threshold changes. As our goal is to allow changes on the threshold without preprocessing, we are going to define a prunning criterion which avoids this *case 3*.

For that, the user must define a set of important threshold values and then the cell octree is built acording to these values and the user can visualize the volume for these values whitout any preprocess due to a threshold change and, more important, without cracks on the isosurface.

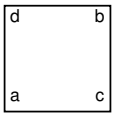


Figure 2.17: Face corner values

Definition 21 (Valid Face)
Let C_Γ be the set of threshold values which are important for the user.

Let F be a face, let a,b,c,d be the property values on its four corners as it is shown in the figure 2.17.

F will be valid for C_Γ if its bilinear is flat (case 1 previously shown) or belongs to the case 2 previously shown for all threshold in C_Γ.

In particular, the following condition must hold:

$$(c-a) = (b-d) \quad \text{or} \tag{2.9}$$
$$C_\Gamma \cap]a,b[= \emptyset \quad \text{or} \tag{2.10}$$
$$C_\Gamma \cap]c,d[= \emptyset \tag{2.11}$$

Non cracks prunning criterion

A c-group will be grouped if:

1. The c-group observes the monotonous prunning criterion conditions.
2. The cell after grouping has all its faces valid for C_Γ.

Once the cell octree is built it is traversed by first-breadth mode and every time there is a boundary between cells of different size (see figure 2.12) the values $F(v_2), F(v_4), F(v_6), F(v_8)$ are modified as it is shown in the equation 2.7 and $F(v_5)$ is modified as:

- $F(v_5) \leftarrow \frac{F(v_1)+F(v_3)+F(v_7)+F(v_9}{4}$ if the equation 2.9 is true.
- $F(v_5) \leftarrow \frac{F(v_1)+F(v_9)}{2}$ if the equation 2.10 is true.
- $F(v_5) \leftarrow \frac{F(v_3)+F(v_7)}{2}$ if the equation 2.11 is true.

So, the isosurfaces for every value in C_Γ will be built without cracks.

2.4 Results

The proposals which have been shown in this chapter have been implemented and tested using real and simulated volumes. As real volumes we have used volumetric information from a tomography axial computerized (TAC), in particular, data from the *Visible Human Project* without filtering and normalized between 0 and 255. The threshold visualized is 60 (see figure 2.18(left)). The resolutions are: $100 \times 80 \times 80$ for the neck, $150 \times 120 \times 120$ for the knee and $200 \times 240 \times 240$ for the head.

We have also tested the proposed method using models whitin a well know definition function. In particular we have used three models whose definition is:

$$\begin{gathered} \alpha(x,y,z) : V \subset \mathbb{R}^3 \rightarrow [0,255] \\ \alpha(x,y,z) = \begin{cases} 255 \cdot (1-r_i) & \text{if} \quad r_i \leq 1 \\ 0 & \text{default} \end{cases} \end{gathered} \tag{2.12}$$

where V is an aligned cube with edge equal to 2 units centered at the origin.

r_i is defined depending on the particular model as follows:

- **Model 1**

$$r_1 = \sqrt{x^2+y^2+z^2} \tag{2.13}$$

- **Model 2**

$$\begin{gathered} r_2 = \sqrt{x^2+y^2+z^2}+ \\ +0.05 \cdot (\sin(50 \cdot \arctan\frac{z}{x}) + \cos(40 \cdot \arctan\frac{y}{x})) \end{gathered} \tag{2.14}$$

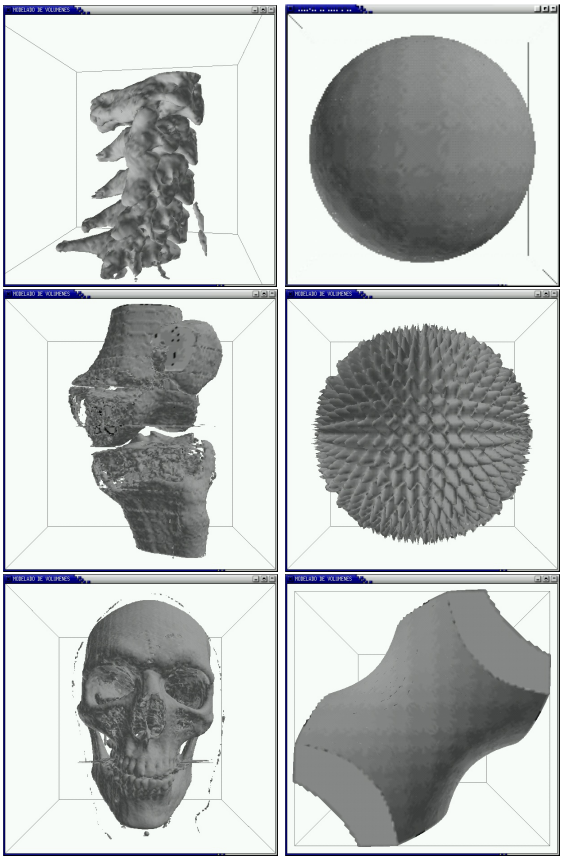


Figure 2.18: Real volume and mathematical model images

- **Model 3**

$$r_3 = \sqrt{x^2 + 2 \cdot y \cdot z} \tag{2.15}$$

A cell octree has been built from these models within a resolution of $100 \times 100 \times 100$. Using 60 as threshold the images in the figure 2.18(right) are got.

All the models have been represented by a bono and by a cell octree using the *non cracks prunning criterion*. The tree size has been measured and the time spent in building it. The time spent in building the triangle mesh has also been measured and the number of triangles for the mesh. Results are shown in the figure 2.19.

It can be seen that the time spent in building the cell octree is longer than the spent in building the bono, but the tree is built just once and the threshold can be changed without building the tree again. The time spent in building the triangle mesh and its number of triangles is similar. It can be seen as a proof of its quality. The main advantage is found in the cell octree size which is smaller than bono tree.

We have also measured the error which is due to the prunnes, this error has been measured as the average distance between the isosurface built from a cell octree and a reference isosurface. For the models 1, 2 and 3 it is easy to have the reference isosurface because the mathematical definition of the models is known. For the human models, the reference isosurface is built from the data at the highest resolution and the test is done using a cell octree at one level less of resolution.

The test has been done representing the models by a cell octree and by a bono. The results are shown in the table 2.1.

Model	Representation	Error
Neck	Bono	0.38
	Cell octree	0.38
Knee	Bono	0.37
	Cell octree	0.38
Head	Bono	0.31
	Cell octree	0.31
Model 1	Bono	0.00
	Cell octree	0.04
Model 2	Bono	0.16
	Cell octree	0.17
Model 3	Bono	0.30
	Cell octree	0.35

Table 2.1: Error measurement

As you can see the error using a cell octree is similar to the error using a bono, so we have a good representation with less space requirements.

Finally we have rendered the models using different colors depending on the error as it is shown in the figure 2.20. The error is positive or negative depending on where the isosurface is respect on the reference isosurface, inside or outside. Of course, the average distance is computed without using the sign.

The colored images are shown in the figure 2.21. It can be seen that, in general the images have little error. Take into account that the human models are represented using a level less of resolution.

The wide yellow areas in the image of the knee and in the model 3 are due to that they are cut by the bounding box.

2.5 Chapter conclusions

A data structure for volume representation, called *Cell Octree* has been showed. It is based on a tree, so it serves as index and allows to access to the different parts of the represented volume in a quick way. The tree is allowed to have its leaf nodes on different level, so it can be used to represent a volume by an adaptive way.

A cell octree can be built using different prunning criteria, in this chapter three different prunning criteria have been presented. These prunning criteria have been defined to use as visualization method the marching cubes [Lorensen, 87] method. So they have been defined depending on the cualities of this method. In particular the goal is to avoid cracks on an isosurface which is built in a multiresolution grid.

The data structure has been tested using real volumes from medical data and also using mathematic volumes. It has been compared whith the structure called bono [Wilhelms, 92] respect on the time spent to build the tree, its size, the time spent to build the isosurface, its number of triangles and its error respect on the real isosurface.

As future work, we plan to study and define new prunning criteria and also to use the data structure for progressive transmission of the volume. Another line on which we are working is the adaptive visualization of the volume using the scheme presented in this chapter.

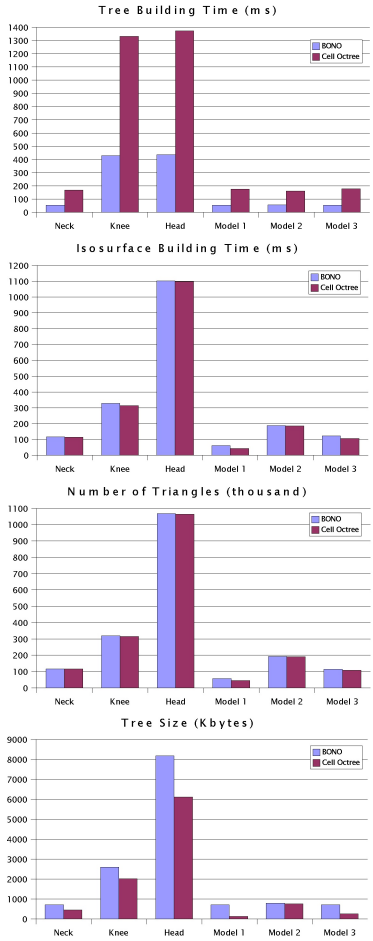


Figure 2.19: Bono vs. Cell Octree

Figure 2.20: Colors to render the error

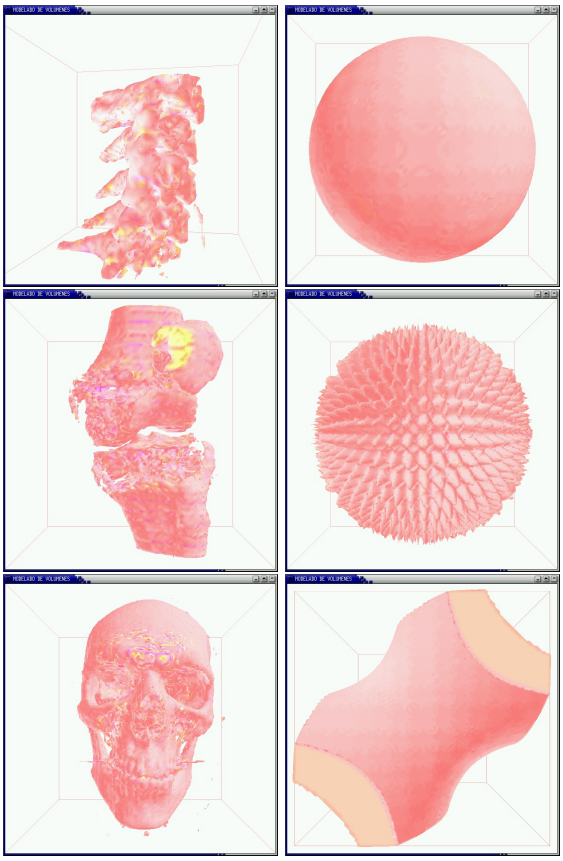


Figure 2.21: Colored images respect on the error

Chapter 3

Progressive Transmission of Cell Octrees

3.1 Chapter abstract

Due to the improvements of 3D scanners, volumetric model resolutions get higher and higher. So data structures as well as algorithms are needed to handle them properly. In this chapter, we present an algorithm to transmit a volume through a network allowing the user to start working on it without waiting for the whole volume to be received. The algorithm uses the cell octree scheme as volume representation and carries out the transmission in a progressive way. User receives the model level by level from a poor resolution level upto the highest resolution available. Some test has been done to show how the user can carry out some kind of work on the model at a medium resolution level (with less than 5% of the whole volume transmited) without waiting for the whole transmission.

3.2 Introduction

Today, systems at whatever computer science scope are frequently interconnected by network, this allows us for resource sharing, information centralization, information

sending/receiving to/from other systems, etc. So, volume modeling and visualization systems also enjoy this kind of advantages; however, the volumetric models size is higher and higher, so the transmission trhough network of volumetric models becomes slower and slower.

Previous works tray to make faster the communication by transmitting only the information which is useful to visualize the volume but the model is always stored on the server [Engel, 99]. So, frequently a new visualization requires a new transmission, depending on which information is stored on the server and which one is stored on the client. For example, some times a camera change can imply a new transmission.

We propose to carry out the transmission of the whole volume, but at a progressive way. Volumetric model is transmitted using different resolutions of accuracy, in such a way that the process begins by transmitting the model using a low resolution (few data obviously means little transmission time), the receiver has a version of the model rather soon to work on it; the user can select a point of view as well as do other visualization adjustements. Meanwhile, the system continues transmitting more data in order to improve the resolution of the model which is being received by the *client*. The system receives the whole volume but the user has not had to wait for the whole transmission in order to have an operative volumetric model.

The progressive transmission is implemented on the basis of the cell octree representation scheme [Velasco, 01b]. This scheme is useful for this purpose because it is a multiresolution scheme that allows the user to handle the volume at different resolution levels depending on the particular task to do.

In the following section, we describe the progressive transmission algortithm. next section shows results and images of a volume transmitted at a progressive way. The final section contains the conclusions drawn and outlines some possibilities for future work.

3.3 Progressive transmission algorithm

As a cell octree is composed by a grid and a tree, in order to do a cell octree transmission we have to transmit grid values (to which we will simply note as *values*) and tree nodes (simply noted as *nodes*). Sender and receiver must be synchronized so that the information for which the receiver is waiting, be the same that the sender is sending

to. Our proposal allows us to do the transmission without sending any extra information for syncrhonization, even thougth the transmission is carried out at a progressive way.

In this section, we are going to show the value transmission algorithm, the node transmission algorithm and how both algorithms are mixed to get the cell octree progressive transmission algorithm.

3.3.1 Value transmission

Values must be transmitted in such a way that the *client* can have a complete version of the model from the begining; a model whose resolution will be improved step by step, but it will always be a model of the whole volume. So the values that are sent on the first step are the bounding box corners, and at every new step, the values that are sent are the intermediate values between the previous step values. This proccess is explained in detail in this section.

The transmission begins sending the bounding box size, noted as `MAXX`, `MAXY` and `MAXZ`. From these data, sender and receiver can compute a new datum, noted as `distance`, which will be the value transmission director. It is computed using the equation 3.1.

$$distance = \min\{x = 2^n : n \in \mathbb{N} \wedge x \geq \max\{MAXX, MAXY, MAXZ\}\} \tag{3.1}$$

`distance` defines how far two consecutive values are for a particular resolution level. Equation 3.1 computes the `distance` datum for the worst resolution level (`distance` is equal to a high number, so not all grid values are used). At every new resolution level it is computed again by dividing itself by 2. `distance` will be equal to 1 for the best resolution level (every grid value is used).

Once the *client* has received the model size and it has computed the `distance` datum it knows how much space needs to store the grid, and it knows in which order it will receive the different values.

The algorithm, written in pseudo-C language, for the worst resolution transmission (resp. reception) is shown in the figure 3.1.

For the other resolution levels, the algorithm has to be careful with not to

```
FUNCTION sendRootValues ()
{
   for (i = 0; i <= MAXX; i += distance)
      for (j = 0; j <= MAXY; j += distance)
         for (k = 0; k <= MAXZ; k += distance)
            send (grid[i][j][k]);
}
```

Figure 3.1: Value transmission for the worst level (root level)

send values which have been already sent. For example, see in the figure 3.2 where the worst resolution level for this 2D grid are the values represented by squares, the medium resolution level are the values represented by circles together with the ones represented by squares and the best resolution are all the values. When the system is sending the medium resolution level, it only has to send the circle values, because the square values have been already sent, and when it is sending the best resolution level, it only has to send the diamond values because the others have been already sent.

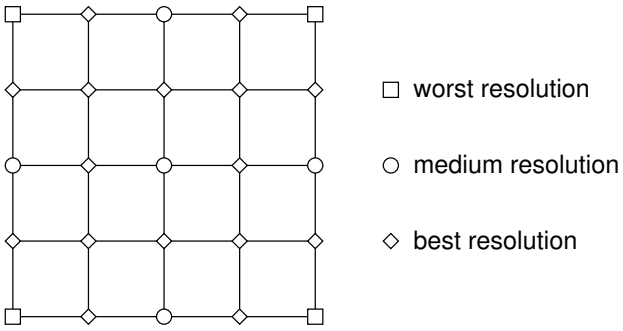


Figure 3.2: Progressive value transmission

The algorithm for all the resolution level transmission but the worst one is shown in the figure 3.3. Let us explain it. A new resolution level uses the values previously sent plus the values which are located in the middle position with respect to the previous resolution level, for example, the medium resolution level in the figure 3.2 uses the previously sent values (squares) and the circle values (located on a middle position between the square values). So, in 2D there are already sent values at the

rows which are multiple of the `distance` datum for the previous resolution level. In 3D there are already sent values at the layers which are multiple of the `distance` datum for the previous resolution level and for every layer within values, these are located at the rows which are multiple of the `distance` datum for the previous resolution level.

```
FUNCTION sendLevelValues ()
{
  oldDistance = distance;
  distance /= 2;
  for (i = 0, auxi = TRUE; i <= MAXX; i += distance, auxi = NOT auxi)
    for (j = 0; auxj = TRUE; j <= MAXY; j += distance, auxj = NOT auxj)
      if (auxi && auxj)
        for (k = distance; k <= MAXZ; k += oldDistance)
          send (grid[i][j][k]);
      else
        for (k = 0; k <= MAXZ; k += distance)
          send (grid[i][j][k]);
}
```

Figure 3.3: Value transmission for other levels

To avoid sending values more than once, it is necessary to handle both distance data, the one that belongs to the current resolution level and the one that belongs to the previous level. In the algorithm shown in the figure 3.3 both distances are handled (`distance` and `oldDistance`), and using the variables `auxi` and `auxj`, it is possible to know whether the sender is located on a layer and on a row within values already sent or not. If so, only the intermediate values (which are located every `oldDistance` positions) are sent, else, all the values for this resolution level are sent (located every `distance` positions).

3.3.2 Node transmission

The tree information has to be also transmited in such a way that the *client* always have a model of the whole volume, from a poor resolution towards the highest one. So the tree has to be transmited level by level. This section explains in detail the process.

The tree transmission is carried out by traversing it by the first-breadth way,

which allows us to traverse it level by level. So, a data structure *queue* will be necessary to carry out the traversing, and a mark, `beginLevel`, will be necessary to know when a new level begins.

Figure 3.4 shows the algorithm to transmit the tree, before explaining it let us show the functions to handle the queue:

- `emptyQueue()`: Returns `TRUE` if the queue is empty, `FALSE` if not.
- `lastQueue(element)`: Adds the `element` to the queue ending.
- `firstQueue()`: Returns the element located at the first position of the queue, it is also erased from the queue.

The queue can store two kinds of elements: nodes and the mark `beginLevel`.

```
send (root);
lastQueue (beginLevel);
if (isInternalNode (root))
  lastQueue (root);
exit = FALSE;
do {
  element = firstQueue ();
  if (element == beginLevel)
     if (not emptyQueue())
        lastQueue (beginLevel);
     else
        exit = TRUE;
  else {   /* element is an internal node */
     for each element's son {
        send (son);
        if (isInternalNode (son))
           lastQueue (son);
     }
  }
} while (not exit)
```

Figure 3.4: Tree transmission

The main loop of the algorithm uses the queue to store internal nodes, and for every internal node that is at the first position of the queue, it is erased from the queue and all its sons are visited. If they are internal nodes, they are added at the

queue ending; the loop stops when the queue is empty. The algorithm begins visiting and adding the root node to the queue and continues as we explained above. It is easy to see how the tree is traversed level by level.

This basic algorithm is modified with sentences to send nodes and to know when a new level begins. For example:

- Before adding the root node to the queue, which is added only if it is an internal node, this is sent and the mark `beginLevel` is added to the queue; the root node is the last node of its level.

- Every time the mark `beginLevel` is at the first position of the queue, a level is finished. If the queue is empty, the tree traversal is finished; if not, a new level begins and the mark `beginLevel` is again added to the queue ending.

- Every time a node is at the first position of the queue, it is erased from the queue, all its sons are sent, and those which are internal nodes are added to the queue. As all the internal nodes are visited level by level, theirs sons are also visited (sent) level by level.

3.3.3 Cell octree progressive transmission

To send a cell octree, which is composed by a grid and a tree, we have to combine the previously shown algorithms into a new algorithm which is shown in figure 3.5. It can be seen that this algorithm basically is the tree transmission algorithm but it has been modified by inserting sentences to send the values for a level before sending the nodes for that level (lines 4 and 14 in figure 3.5). In this way, all the values that can be accessed through a received node will be stored in the receiver grid.

The reception algorithm, similar to the transmission one, is shown in the figure 3.6. This algorithm includes sentences to build the tree (lines 6 and 23 in figure 3.6) and to visualize the cell octree during the reception (lines 7 and 24); in other applications this visualization could be unnecessary or other kind of process can be carried out. Every node (leaf of internal) is visualized, when the sons nodes of an internal node are received, the triangles built for the internal node are replaced by the triangulation built for its sons. By this way, the visualization is refined meanwhile the nodes are comming.

```
 1:   send (MAXX, MAXY, MAXZ);
 2:   distance = computeDistance (MAXX,MAXY,MAXZ);
 3:      /* computeDistance uses the equation (1) */
 4:   call sendRootValues ()
 5:   send (root);
 6:   lastQueue (beginLevel);
 7:   if (isInternalNode (root))
 8:      lastQueue (root);
 9:   exit = FALSE;
10:   do {
11:      element = firstQueue ();
12:      if (element == beginLevel)
13:         if (not emptyQueue()) {
14:            call sendLevelValues();
15:            lastQueue (beginLevel);
16:         } else
17:            exit = TRUE;
18:      else {   /* element is an internal node */
19:         for each element's son {
20:            send (son);
21:            if (isInternalNode (son))
22:               lastQueue (son);
23:         }
24:      }
25:   } while (not exit)
```

Figure 3.5: Cell octree transmission

3.4 Results

The proposal shown in this chapter has been implemented and tested using real volumes from the *Visible Human Project* without filtering and using a resolution of $220 \times 230 \times 220$. The size of the data that are sent at every transmission level was measured, and images for every transmission level were rendered. Data are shown in table 3.1 and images in figures 3.7, 3.8, y 3.9.

In order to evaluate the image quality for levels 5, 6 and 7, these images

```
 1:   receive (MAXX, MAXY, MAXZ);
 2:   distance = computeDistance (MAXX,MAXY,MAXZ);
 3:      /* computeDistance uses the equation (1) */
 4:   call receiveRootValues ()
 5:   receive (root);
 6:   newTree (root);
 7:   visualize (root);
 8:   lastQueue (beginLevel);
 9:   if (isInternalNode (root))
10:      lastQueue (root);
11:   exit = FALSE;
12:   do {
13:      element = firstQueue ();
14:      if (element == beginLevel)
15:         if (not emptyQueue()) {
16:            call receiveLevelValues();
17:            lastQueue (beginLevel);
18:         } else
19:            exit = TRUE;
20:      else   /* element is an internal node */
21:         for each possible element's son {
22:            receive (son);
23:            set element as son's father;
24:            visualize (son);
25:            if (isInternalNode (son))
26:               lastQueue (son);
27:         }
28:   } while (not exit)
```

Figure 3.6: Cell octree reception

were coloured according to the distance (error) between theirs isosurfaces and the isosurface for level 8 (used as pattern). The error is positive or negative depending on where the isosurface is respect on the pattern isosurface (i.e. inside or outside). The average error was also measured, obviously without using the sign. Coloured images and colour range are shown in figue 3.10.

It can be seen how at the level 6, with just the 5% of the total size transmited, a good approximation is got allowing the user to carry out some kind of work, like adjusting the camera, choicing the isovalue, etc. Meanwhile, new information is coming and at the level 7 (only the 30% of the total size) a very good approximation is got allowing the user to work with the model perfectly.

Level	Size for Level (bytes)	Total size (bytes)
0	10	10
1	78	88
2	624	712
3	4.752	5.464
4	28.776	34.240
5	186.856	221.096
6	1.141.012	1.362.108
7	7.339.684	8.701.792
8	21.896.104	30.597.896

Table 3.1: Transmission data

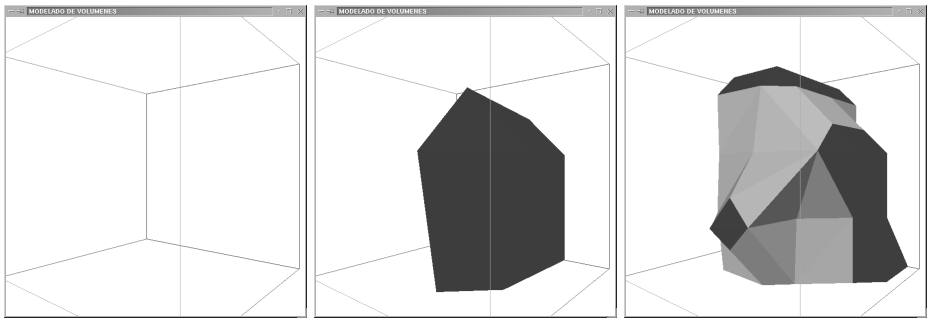


Figure 3.7: Images for levels 0, 1 and 2

Figure 3.11 shows a logarithmic diagram where it can be seen the relation among time and image got. A constant bit rate transmission is assumed, so the relation can be established among size and image.

3.5 Chapter conclusions

An algorithm for progressive transmission of volumes is described. The algorithm uses a particular representation of volumes, the cell octree scheme, which is very useful for this operation because it is a multiresolution scheme, so volumes are transmited using the different resolution levels stored in the model. The user begins re-

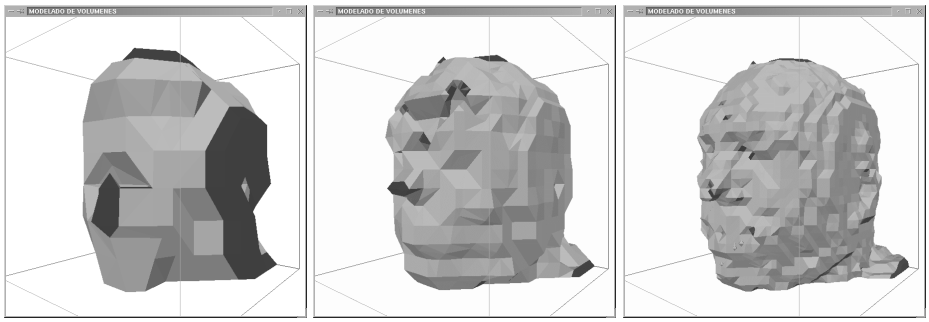


Figure 3.8: Images for levels 3, 4 and 5

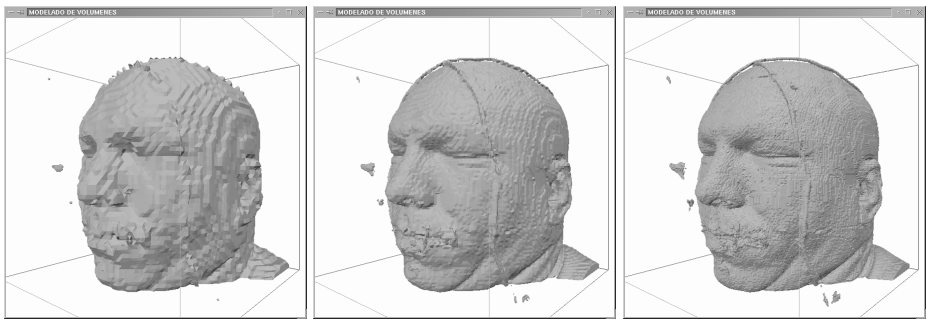
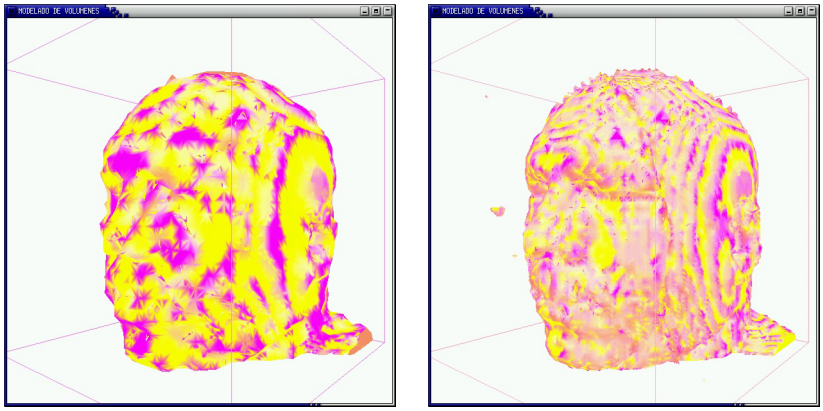


Figure 3.9: Images for levels 6, 7 and 8

ceiving a low resolution (but fast) model that allows him or her to carry out some kind of work, meanwhile the resolution model is improved by the new transmissions.

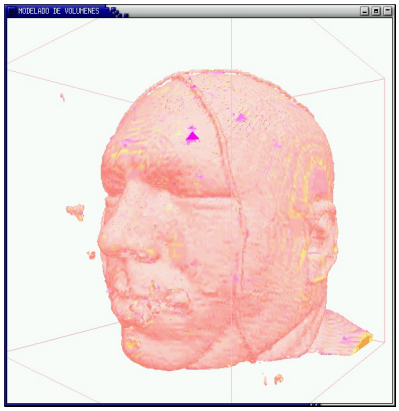
The main advantage of the proposal is that the system does not require to send or receive extra information even although the transmission is carried out at a progressive way. Moreover the model is stored in the receiver, so a change of point of view or isovalue does not need new transmissions.

The proposal can be improved by carrying out an *adaptive* transmission: the user select an interest area and then the sender continues by only transmiting this area of the model, so the user receives the interest area at a high resolution and the rest of the model at a medium or poor resolution.

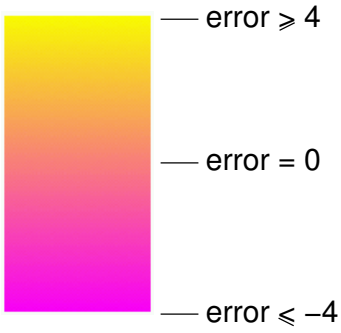


Level 5 - average error 3.20

Level 6 - average error 1.28



Level 7 - average error 0.34



Colour range

Figure 3.10: Coloured images according to error

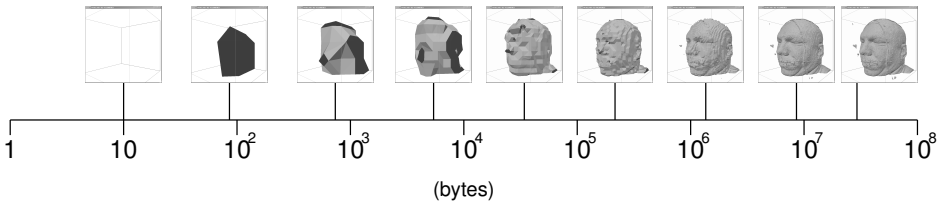


Figure 3.11: Relation among *time* and image got

Chapter 4

A Method for Isosurface Extraction

4.1 Chapter abstract

Volumetric data can be represented as a rectilinear structured grid and can be displayed by the rendering of an isosurface that is built from the grid and defining a threshold value. The know marching cubes method builds the isosurface cell by cell. However, from a set of cells that comply with some conditions, a bigger cell can be built. In these cases, the volume is represented by a set of cells of different size. Then, the isosurface built by marching cubes will have holes on the borders between cells of different size. In this chapter, we show a new method, called marching edges, to build the isosurface edge by edge, that generates a hole-free isosurface.

4.2 Introduction

Volumetric data are usually represented as a set of property values v_i at a set of points $(x_i, y_i, z_i) \ : \ i = 1, 2, \ldots, N$; these values are samples from some unknown continuous function $f(x, y, z)$. In order to obtain pictures from that set of samples, a process of three step can be done [Haber, 90]:

1. *Data enrichment:* An estimation, $F(x,y,z)$, of the unknown function $f(x,y,z)$ is made with which new couples $(point, property\ value)$ can be estimated.

2. *Mapping:* Some geometric interpretation of the function $F(x,y,z)$ is chosen in order to understand its behaviour. This geometric interpretation will typically be a three dimensional object that can be rendered in the next step.

3. *Rendering:* The geometry obtained in the previous step is rendered using standard computer graphics techniques.

An usual way to render volumetric data is the marching cubes method [Lorensen, 87]. A rectilinear structured grid of the samples is made from the original volume data; an isosurface $F(x,y,z) = v_k$ is built and rendered for some threshold value v_k. Only a subset of the volume is rendered.

In order to build the surface, every cubic cell in the grid is processed: each cell is classified like one of 15 distinct cases by comparing v_k with the property values of the 8 vertices of the cell; every case has a triangulation that represents the isosurface inside the cell; the union of all piece of isosurfaces of all cells forms the isosurface that is rendered.

Since 1987, several papers have been published about the improvement of marching cubes method [Brodlie, 01].

Concretely, this method needs a long time, most of which is spent processing cells that do not have isosurface inside. In order to reduce this time, several methods have been proposed to search for cells intersecting the surface. These methods can be classified according to the search criteria used:

- *Range-based*, each cell is labelled with the interval that it spans in the range of the property field. This allows to search for intervals that contain a given threshold value. Cignoni et al. propose to use an interval tree structure [Edelsbrunner, 80, Preparata, 85] in order to retrieve these intervals, from which active cells are found [Cignoni, 97]. In particular, they use a method that combines range-based and surface-based search.

- *Surface-based*, only the cells that are adjacent to cells for which the surface has been previously rendered are processed [Shekhar, 96]. This solution does not display the complete surface when it has more than one component. It is

necessary to have a seed set of cells from which all the components can be rendered. Bajaj et al. propose a method to generate seed sets that allow to get all components for all possible threshold values [Bajaj, 96].

- *Space-based*, domain spanned by the data set is partitioned in order to search active cells. These partitions can be hierarchical [Wilhelms, 92].

A tipical space-based searching is the bono structure. Wilhelms et al. proposed to build an octree [Meagher, 80] that indexes the grid, and which stores, in every octree node, the maximum and minimum property value of the subvolume covered by the node [Wilhelms, 92]. This information is used when traversing the volume and allows that those nodes for which the property interval does not contain the threshold to be skipped. This structure, called BONO (Branch On Need Octree), makes the rendering faster but increases the storage requirement as it must store both the grid and the octree.

Other advantage of the tree is the multiresolution capability. Volume can be rendered at different levels of detail by pruning the tree at a specific level when it is rendered. Furthermore, an adaptive isosurface can be built if different branches of the tree are pruned at different levels. Works like [Ohlberger, 97, Westermann, 99] use these issues, however, the prune condition is dependent on the threshold and, in the case of [Westermann, 99], also on the point of view.

It is also possible to use hierarchical structures and prunning of branches oriented to the visualization process. Livnat et al. use a bono in order to do a hierarchical front-to-back traversal of the data set with dynamic pruning of sections that are hidden from the point of view by previously extracted sections of the isosurface [Livnat, 98]. Only the isosurface that is visible from the point of view is built. When the viewpoint changes, the isosurface must be re-built.

However, when different branches of the tree are pruned at different levels, there will be joined cells of different size. Then, holes (or cracks) on the isosurface can arise on the border between those cells of different size. This is because the isosurface inside the large cell is built from less information than the isosurface inside the small ones (see figure 4.1).

Several solutions have been proposed for this problem. Shu et al. propose covering the crack with a polygon [Shu, 95]; Shekhar et al. propose removing it by moving the vertices of the triangles in the small cells so that they coincide with the edges of the triangle in the big one [Shekhar, 96]; Westermann et al. propose to

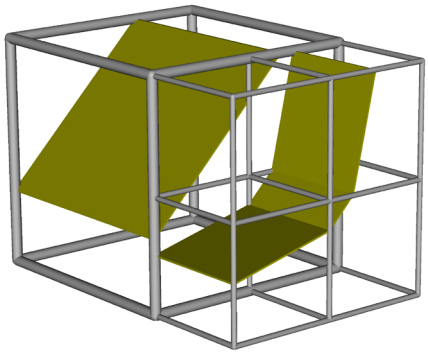


Figure 4.1: Example of crack

replace a triangle in the large cell with a fan of triangles, in order to adapt the isosurface inside the large cell to the isosurface inside the small ones [Westermann, 99]; this method with several improvementes is used by Engel et al. to implement a web-based volume visualization system [Engel, 99]. In these cases, the process can be performed after the triangle mesh has been built or when it is being built.

In a previous chapter we proposed a modification of bono, called *cell octree*, in which the branches that represent a set of cells whose property values verify some uniform gradient conditions are forever pruned [Velasco, 01a]. The main difference between a cell octree and other approaches is that the prune condition is independent on the threshold. Note that this does not forbit to use an aditional view dependent prune condition.

In a cell octree, cells of different size can be adjacents. In order to avoid crashes on surfaces that cross over two adjacent cells of different size, the prune condition is very restrictive. That is, it allows to prune a branch only when it can be ensured that no crack will appear for any threshold value.

In this chapter, we present a method to build the triangles of the isosurface, edge by edge instead of cell by cell. Isosurfaces built with this method can cross the border between cells of different size without any discontinuity. This is so because there are no triangle's vertices on the border between cells, then it is not necessary to make coincide any geometric data on both sides of that border. The cells will have up to one triangle's vertex that will be inside the cell. So, triangles will cross the border.

The quality of no discontinuity allows us to use a less restrictive prune condition obtaining a cell octree with less storage requirements.

In section three our proposal, called *marching edges*, is presented, where the concept of *isopoint* is explained. In section four we explain how to compute the isopoints, how to label the edges to avoid processing an edge more than once, and the rendering algorithm. Finally, section five shows data and pictures from real volumes.

4.3 Marching edges

Let c be a cell intersected by the isosurface. This cell, called *active cell*, has some edges that are crossed by the isosurface. These edges are called *active edges*. Let us consider the active edge (A,B) on the figure 4.2; the active edge is shared by four active cells. For every cell one point inside of it, called *isopoint*, is calculated and two triangles are built by the four isopoints.

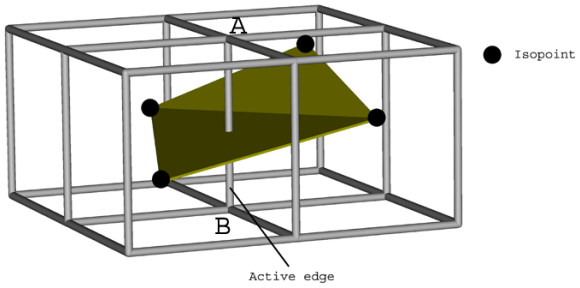


Figure 4.2: Triangulation for an active edge

Let us consider now cells of different size. Only two distinct cases are possible: first, the active edge (A,B) in the smallest cell is on a face of a larger one (figure 4.3); second, the active edge (A,B) in the smallest cell is on an edge (C,B) of a larger one and it is not on a face of another cell (figure 4.4).

Every cell that shares the active edge contributes one isopoint to the triangulation, so the triangulation of the first case is made using one triangle, whereas the triangulation of the second case is made using two triangles.

In both cases, if the small edge is active, then the large cell is active too, as

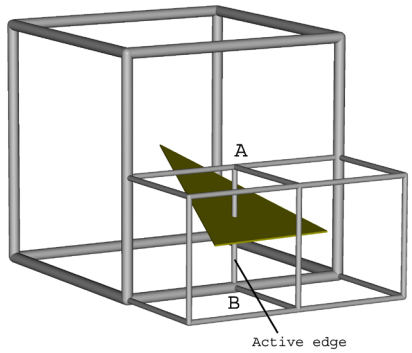


Figure 4.3: Triangulation by 1 triangle

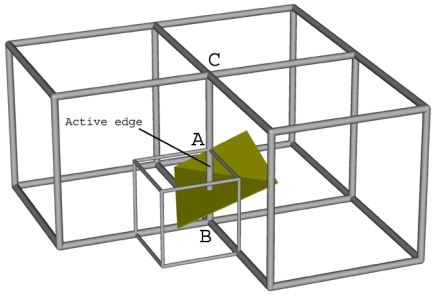


Figure 4.4: Triangulation by 2 triangles

the large cell is monotonous.

So, the number of distinct cases by marching edges is three, when the edge is not active, no one triangle is generated; when the edge is shared by three cells (figure 4.3), one triangle is generated; and when the edge is shared by four cells (figure 4.4), two triangles are generated.

This method has a drawback: some cases of active cells (the cases 1, 3, 4, 6, 7, and 13 of [Lorensen, 87]) generate more triangles by marching edges than by marching cubes. However, other cases (the cases 5, 9, 11 and 14 of [Lorensen, 87]) generate less triagles by marching edges than by marching cubes.

For example, in the figure 4.5 we can see the case number 1 of [Lorensen, 87] that generates one triangle using the marching cubes method. As shown in the figure 4.6 the same configuration has three active edges. For each active edge, up to two triangles are generated using marching edges. These triangles are shared by up to four cells; then, the active cell generates 1.5 triangles.

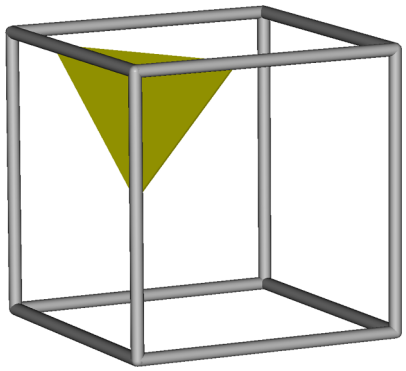


Figure 4.5: Case 1 by marching cubes

The case number 8 of [Lorensen, 87] (figure 4.7) is one of the most common in data sets, figure 4.8 shows how it is triangulated by marching edges, it can be seen that it generates the same number of triangles than using marching cubes. The isopoints have been marked by black circles.

The number of triangles built by our method can be between a 25 % less and a 50 % greater than when the marching cubes method is used.

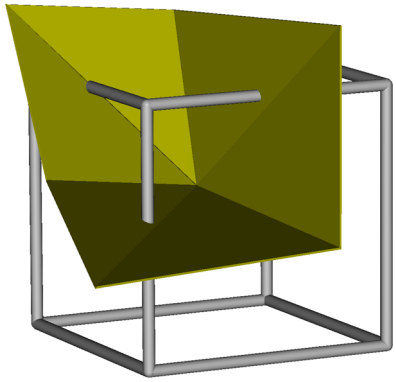


Figure 4.6: Case 1 by marching edges

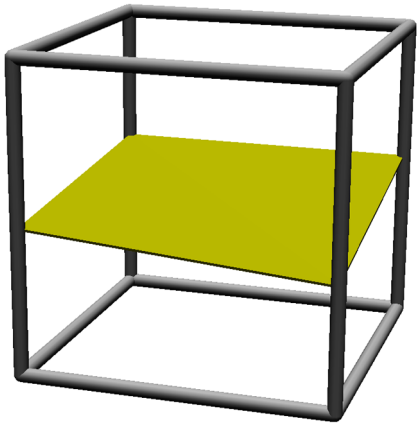


Figure 4.7: Case 8 by marching cubes

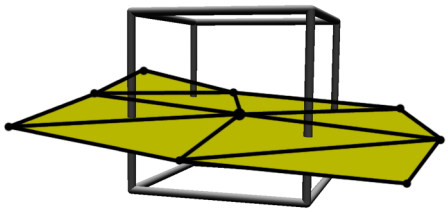


Figure 4.8: Case 8 by marching edges

However, the advantages of our proposal are:

- The border between cells of different size will not have any crack, because triangle's vertices will not be on the faces, and so, they will not depend on the different number of samples for that border in the large cell with respect to the small one. Triangles will cross those borders (see figure 4.9).
- Thanks to the previous advantage, the prune critera used in [Velasco, 01a] can be less restrictive; only the monotony conditions are used, the condition about the center of the faces are not needed. So more prunes are done. Moreover, any property value needs to be changed at any vertex.
- The more prunes are done, the less storage space is needed.

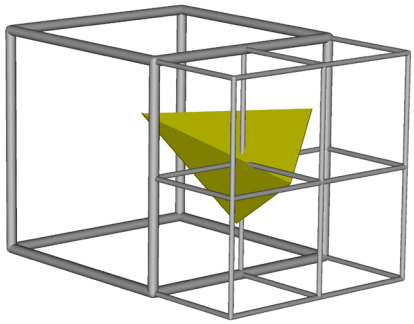


Figure 4.9: The advantage of no cracks

The edges on the limit of the volume do not generate triangles even if they are actives, because there are no cells on the other side. If it is needed, volume must be resized or repositioned.

4.4 Implementation

Every active edge generates one or two triangles. The vertices of these triangles are into the cells sharing the edge. For every active cell the location of the triangle vertex, that is unique, is computed using only information local to the cell. This ensure that all triangles crossing the cell are joint at the same vertex avoiding crash. We will show now how the location of the vertex, that we call *isopoint*, is computed.

4.4.1 Computation of Isopoints

We present the method we use to calculate the isopoint that uses information about the cell and the threshold. Other methods like using a fix relative position can be faster but also they can generate worse isosurfaces.

Let $c = (c.x, c.y, c.z, c.s)$ be an active cell where $(c.x, c.y, c.z)$ is the cell's vertex nearest to the origin, and $c.s$ is the size of the cell. And let v_k be the threshold value. The isopoint of the cell c will be a point $P \in c$ that represents the cell in order to build triangles.

In order to calculate the isopoint P, (see the figure 4.10) the central point p_c and its property value is computed by the function F. For every cell's vertex p_i we can know if the isosurface crosses the virtual line l_i between p_i and p_c. If l_i is crossed, an isopoint P_i can be computed by linear interpolation. The isopoint $P = (P.x, P.y, P.z)$ will be the average point of all the P_i computed in the cell.

The gradient vector $G = (G.x, G.y, G.z)$ at P is computed as:

$$G.x(P) = (G.x^0 \cdot (1 - \Delta y) + G.x^1 \cdot \Delta y) \cdot (1 - \Delta z) +$$

$$(G.x^2 \cdot (1 - \Delta y) + G.x^3 \cdot \Delta y) \cdot \Delta z$$

being

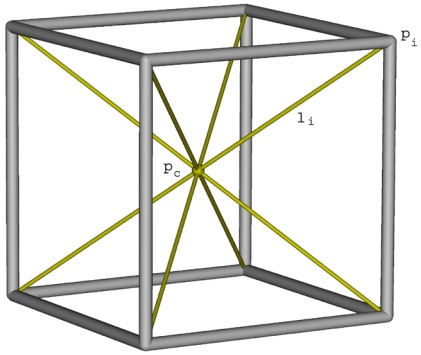


Figure 4.10: Computation of isopoints

$$\Delta x = \frac{P.x - c.x}{c.s}$$

$$\Delta y = \frac{P.y - c.y}{c.s}$$

$$\Delta z = \frac{P.z - c.z}{c.s}$$

and

$G.x^0 = F(c.x + c.s, c.y, c.z) - F(c.x, c.y, c.z)$

$G.x^1 = F(c.x + c.s, c.y + c.s, c.z) - F(c.x, c.y + c.s, c.z)$

$G.x^2 = F(c.x + c.s, c.y, c.z + c.s) - F(c.x, c.y, c.z + c.s)$

$G.x^3 = F(c.x + c.s, c.y + c.s, c.z + c.s) -$
$F(c.x, c.y + c.s, c.z + c.s)$

$G.y$ and $G.z$ are computed in a similar way.

The gradient vector is used to allow a shadowed rendering, like the one displayed by the phong method [Bui-Tuong, 75].

Every isopoint is computed just once, they are stored in a set of isopoints to use them when it is necessary.

4.4.2 Edge Labeling

Every edge can be shared by up to four cells. If the volume is processed cell by cell, and for each cell, all its edges are processed, every edge can be processed up to four times. So every cell has a label to indicate what edges must be processed. Every edge in the volume is processed just once.

A cell octree can have cells of different size, the edges that are shared by cells of different size (figures 4.3 and 4.4) are labeled and processed in the smallest cell. This is so in order to make easy the searching of neighbourg cells to find the needed isopoints to build triangles. When a labeled active edge is being processed, it is necessary to find the isopoints (to build triangles) inside the neighbour cells. Note that our method can process only edges that are shared by up to 4 cells. When an edge is shared by cells of different size, we process the edge on the smallest cell.

In every cell, only the labeled edges are processed, then the minimun and maximun property values stored at each cell are only computed in function of the labeled edge's vertices, so the intervals $[min, max]$ stored in the tree's nodes can be smaller and some tree's branches will not be processed with a higher probability and the visualization time will be shorter.

Note that the prune criteria and the edges labeling is independent on the threshold value.

4.4.3 Visualization

The pseudocode of the procedure to render the volume is shown in the figure 4.11, where `V1` and `V2` are the vertices of the edge `E` and `F(vertex)` is the estimated volume function.

4.5 Results

The proposed method has been implemented and compared with bono using marching cubes, and cell octree using marching cubes. Tests have been performed with the volumes showed in figure 4.14. The volumes have been modelled from TAC and the data have been normalised between 0 and 255 (the threshold for the tests is 60).

Table 4.1 shows the storage requirements for the octree. Table 4.2 shows the

```
render_volume (node N,threshold T)
{
   if (N has sons)
      for each son S of N
         if (T is between min and max of S)
            render_volume (S,T)
   else /* it is a leaf node */
      for each labeled edge E=(V1,V2) of N
         if (T is between F(V1) and F(V2))
         {
            classify E
            compute the isopoints
            show the triangles
         }
}
```

Figure 4.11: Pseudocode of render_volume

time used to build the tree. Table 4.3 shows the number of triangles of the isosurface. The time used to build the isosurface is shown in table 4.4. Figures 4.12 and 4.13 shows a plot of these data against the size of the volume.

Volume	Finger	Eye	Head
Size	$17 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
Bono	24.144	299.600	2.396.752
Cell Octree (m. cubes)	23.248	190.032	1.587.088
Cell Octree (m. edges)	18.775	144.535	1.330.857

Table 4.1: Storage required by tree (bytes)

In the diagrams about the isosurface the lines of *bono* and *cell octree (m. cubes)* are in the same position as the results of both methods are very similar (see tables 4.3 and 4.4).

We can see that there is a reduction in the storage requeriment and in the time to build the isosurface whereas the quality of the images is good, figure 4.15 shows

Volume	Finger	Eye	Head
Size	$17 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
Bono	1	8	58
Cell Octree (m. cubes)	4	29	231
Cell Octree (m. edges)	8	64	590

Table 4.2: Time to build the tree (ms)

Volume	Finger	Eye	Head
Size	$17 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
Bono	3.586	34.232	376.998
Cell Octree (m. cubes)	3.586	34.226	376.804
Cell Octree (m. edges)	4.828	40.328	463.682

Table 4.3: Number of triangles of the isosurface

Volume	Finger	Eye	Head
Size	$17 \times 32 \times 32$	$64 \times 64 \times 64$	$128 \times 128 \times 128$
Bono	5	45	513
Cell Octree (m. cubes)	4	45	512
Cell Octree (m. edges)	2	18	236

Table 4.4: Time to build the isosurface (ms)

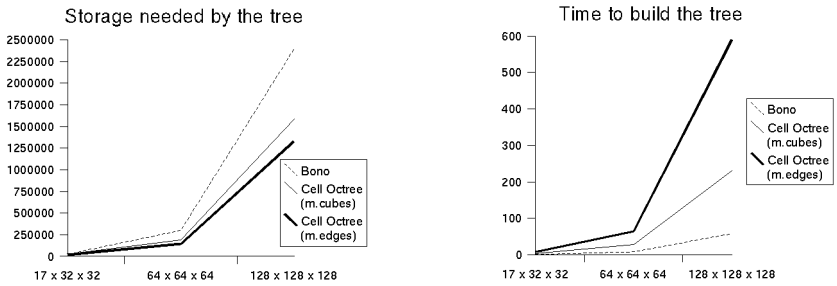


Figure 4.12: Diagrams about the tree.

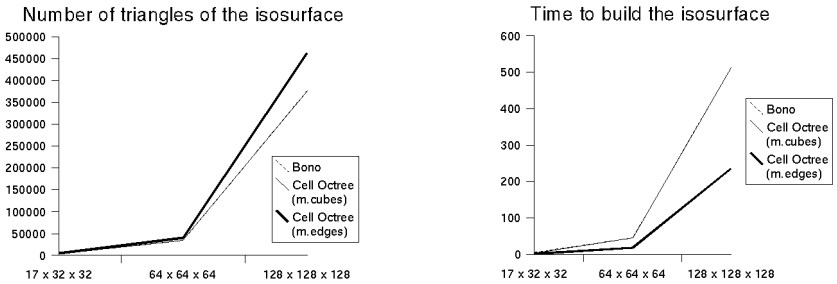


Figure 4.13: Diagrams about the isosurface.

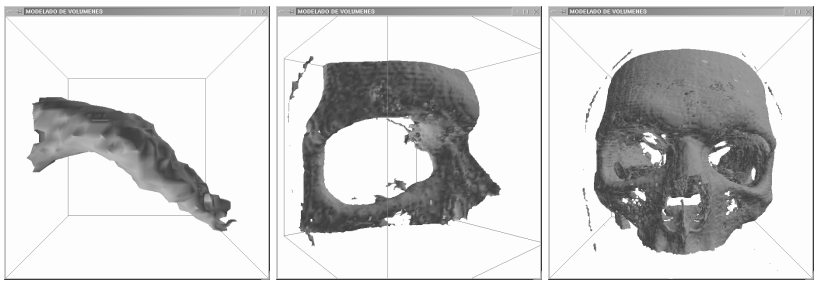


Figure 4.14: Images from volumes used

on the left side a zoomed picture by marching cubes and on the right side the same subvolume rendered by marching edges. The time to build the tree is much longer, but this process is done just once. The number of triangles of the isosurface is greater than using marching cubes.

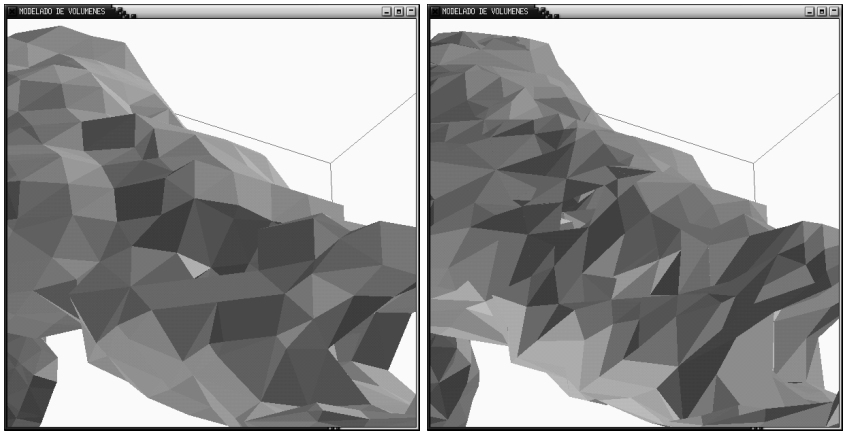


Figure 4.15: Marching cubes vs. marching edges

The reduction in the storage requeriment shows how the prunning criteria is less restrictive.

The number of triangles can be reduced by decimation. Schroeder et al. propose deleting points of the triangle mesh and making a retriangulation [Schroeder, 92]. Montani et al. propose to build the isosurface so that triangles can easily be in the same plane, and then retriangulate them [Montani, 94]. In our proposal, if the isopoints that are chosen to represents the cells are in the center of the cells, many triangles will be in the same plane too.

4.6 Chapter conclusions

A new method for volume visualization by surface extraction has been proposed. We have presented a method to build the triangles edge by edge instead of cell by cell. This method reduces the number of distinct cases from 15 (marching cubes [Lorensen, 87]) to 3. Also, it avoids that cracks are arisen when the surface crosses the border between cells of different sizes, neither pre-process nor post-process are

needful. Thanks to it, the prunning criteria used in [Velasco, 01a] can be less restrictive. So, more prunes can be done and at a higher level of the tree. As a consequence of this, the storage space requirement of the tree has been reduced.

We are currently working on an improvement to store the properties in the tree avoiding the use of the grid, to obtain further reduction in space used. Also, we are studying particular cases of cells in order to reduce the number of triangles built.

Bibliography

[Bajaj, 96] Bajaj, C., Pascucci, V., and Schikore, D. (1996). Fast isocontouring for improved interactivity. In *ACM Symposium on Volume Visualization*, pages 39–46,99, San Francisco, USA.

[Brodlie, 01] Brodlie, K. and Wood, J. (2001). Recent advances in volume visualization. *Computers Graphics forum*, 20(2):125–148.

[Bui-Tuong, 75] Bui-Tuong, P. (1975). Ilumination for computer generated pictures. *CADM*, 18(6):311–317.

[Chernyaev, 95] Chernyaev, E. (1995). Marching cubes 33: construccion of topologically correct isosurfaces. Technical Report CN/95-17. Avaiable as `http://wwwinfo.cern.ch/asdoc/psdir/mc.ps.gz`, CERN.

[Cignoni, 00] Cignoni, P., Ganovelli, F., Montani, C., and Scopigno, R. (2000). Reconstruction of topologically correct and adaptive trilineal surfaces. *Computer and Graphics*, 24(3):399–418.

[Cignoni, 97] Cignoni, P., Marino, P., Montani, C., Puppo, E., and Scopigno, R. (1997). Speeding up isosurface extraction using interval trees. *IEEE Transactions on Visualization and Computer Graphics*, 3(2):158–170.

[Edelsbrunner, 80] Edelsbrunner, H. (1980). Dynamic data structures for orthogonal intersection queries. Technical Report F59, Inst. Informations-verarb., Tech. Univ. Graz, Graz, Austria.

[Engel, 99] Engel, K., Westermann, R., and Ertl, T. (1999). Isosurface extraction techniques for web-based volume visualization. In *IEEE Visualization*, pages 139–146.

[Fiume, 89] Fiume, E. (1989). *The mathematical structure of raster graphics*. Academic Press, Boston.

[Haber, 90] Haber, R. and McNabb, D. (1990). *Visualization idioms: a conceptual model for scientific visualization systems*, chapter of Visualization in Scientific Computing, pages 74–93. B. Shriver, G.M. Nielson and L.J. Rosenblum (eds).

[Levoy, 88] Levoy, M. (1988). Display of surfaces from volume data. *IEEE Computer Graphics & Applications*, pages 29–37.

[Livnat, 98] Livnat, Y. and Hansen, C. (1998). View dependent isosurface extraction. In *IEEE Visualization*, pages 175–181.

[Lopes, 99] Lopes, A. (1999). *Accuracy in Scientific Visualization*. PhD thesis, Univiersity of Leeds, Leeds, UK.

[Lorensen, 87] Lorensen, W. and Cline, H. (1987). Marching cubes: A high resolution 3d surface construction algorithm. *ACM Computer Graphics*, 21(4):163–169.

[Mallgren, 82] Mallgren, W. (1982). *Formal specification of interactive graphics programming languages*. MIT Press, Cambridge.

[Mantyla, 88] Mantyla, M. (1988). *An Introduction to Solid Modelling*. Computer Science Press.

[Meagher, 80] Meagher, D. (1980). Octree encoding: a new tecnique for the representation, manipulation and display of arbitrary three dimensional objects by computer. Technical Report IPL-TR-80-111, Polytechnic Inst., Revisselaer.

[Montani, 94] Montani, C., Scateni, R., and Scopigno, R. (1994). Discretized marching cubes. In *Proceedings of Visualization*, pages 281–287, Los Alamitos, CA, USA.

[Ohlberger, 97] Ohlberger, M. and Rumpf, M. (1997). Hierarchical and adaptive visualization on nested grids. *Computing*, 59(4):365–385.

[Permingeat, 88] Permingeat, N. and Glaude, D. (1988). *Álgebra de Boole. Teoría, métodos de cálculo, aplicaciones.* Vicens Vives.

[Preparata, 85] Preparata, F. and Shamos, M. (1985). *Computational Geometry: An Introduction*. Springer-Verlag.

[Requicha, 80] Requicha, A. (1980). Representations for rigid solids: theory, methods and systems. *Computer surveys*, 12(4).

[Requicha, 82] Requicha, A. and Voelcker, H. (1982). Solid modelling: A historical summary and contemporary assessment. *IEEE Computer Graphics and Applications*, 2(2):9–24.

[Schroeder, 92] Schroeder, W., Zarge, J., and Lorensen, W. (1992). Decimation of triangle meshes. *ACM Computer Graphics*, 25(2):65–70.

[Shekhar, 96] Shekhar, R., Fayyad, E., Yagel, R., and Cornhill, F. (1996). Octree-based decimation of marching cubes surfaces. In *Visualization'96*, pages 335–342,499, San Francisco, USA. IEEE.

[Shu, 95] Shu, R., Zhou, C., and Kankanhalli, M. (1995). Adaptive marching cubes. *The Visual Computer*, 11:202–217.

[Torres, 93] Torres, J. and Clares, B. (1993). Graphics objects: A mathematical abstract model for computer graphics. *Computer Graphics Forum*, 12(5):311–327.

[Velasco, 01a] Velasco, F. and Torres, J. C. (2001a). Cell octree: A new data structure for volume modeling and visualization. In *VI Fall Workshop on Vision, Modeling and Visualization*, pages 151–158, Stuttgart, Germany.

[Velasco, 01b] Velasco, F. and Torres, J. C. (2001b). Uso de octrees de celdas para visualización de volúmenes. In *XI Congreso Español de Informática Gráfica*, pages 99–111, Girona.

[Velasco, 03] Velasco, F., Torres, J. C., and Cano, P. (2003). A formalization of volumetric models. In *XIII Spanish Conference on Computer Graphics (CEIG'03)*, pages 101–114, A Coruña (Spain).

[Westermann, 99] Westermann, R., Kobbelt, L., and Ertl, T. (1999). Real-time exploration of regular volume data by adaptive reconstruction of isosurfaces. *The Visual Computer*, 15:100–111.

[Westover, 90] Westover, L. (1990). Footprint evaluation for volume rendering. *Computer Graphics*, 24(4):367–376.

[Wilhelms, 92] Wilhelms, J. and Gelder, A. V. (1992). Octrees for faster isosurface generation. *ACM Transactions on Graphics*, 11(3):201–227.