UNIVERSIDAD CENTRAL DE VENEZUELA
FACULTAD DE CIENCIAS
ESCUELA DE MATEMÁTICA

Homología persistente para la evaluación de reconstrucciones 3D

Trabajo Especial de Grado presentado ante la ilustre Universidad Central de Venezuela por el **Br. Carlos Valladares** para optar al título de Licenciado en Matemática.

Tutor: Dr. Mauricio Angel.

Caracas, Venezuela
Febrero, 2014

Nosotros, los abajo firmantes, designados por la Universidad Central de Venezuela como integrantes del Jurado Examinador del Trabajo Especial de Grado titulado "**Homología Persistente para la evaluación de reconstrucciones 3D**", presentado por el **Br. Carlos Valladares**, titular de la Cédula de Identidad **19378914**, certificamos que este trabajo cumple con los requisitos exigidos por nuestra Magna Casa de Estudios para optar al título de **Licenciado en Matemática**.

Dr. Mauricio Angel
Tutor

Dr. Wuilian Torres
Jurado

Msc. Jonathan Otero
Jurado

Dedicatoria

A Dios y mi familia.

Agradecimiento

Agradezco a mi familia por su incondicional apoyo durante todos mis estudios; al tutor, quien gracias a sus consejos, revisiones y experiencia hicieron que este trabajo se desarrollara de la mejor manera posible; a la Dra. Rocio González, Universidad de Sevilla, quien me suministró información y material necesario para la ejecución de este trabajo.

CONTENIDO

Introducción

El objetivo principal de este Trabajo Especial de Grado consiste en la evaluación topológica de reconstrucciones 3D obtenidas a partir de la técnica de voxel carving; entendiéndose, durante el desarrollo de este trabajo, el término evaluación topológica o sólo evaluación, como la determinación de ciertos invariantes topológicos como lo son las componentes conexas y los agujeros n-dimensionales, en el objeto reconstruído para luego comparar los mismos con dichos invariantes en el objeto real. Todo esto se realiza con la finalidad de verificar si el modelo preserva estos invariantes en comparación con el objeto inicial. Por ejemplo, si un objeto posee un par de componentes conexas y un agujero, se desea evaluar si los mismos también se encuentran presentes en una reconstrucción del objeto a través de la técnica de voxel carving.

Para poder determinar los invariantes topológicos mencionados anteriormente se deben comprender ciertos términos como lo son grupos de homología, números de Betti, entre otros; los cuales pueden ser obtenidos a través del estudio de un modelo algebraico topológico, conocido como modelo AT. Puesto que se desea calcular dicho modelo computacionalmente, se deben introducir un par de algoritmos con esta finalidad, los cuales son el algoritmo de incremento y decrecimiento, éstos son estudiados por Gónzalez en [**5**] y [**8**].

Por otra parte, en modelación geométrica, se tiene que el Voxel Carving es un método conocido para la construcción de modelos tridimensionales de objetos a partir de un conjunto de imágenes asociadas al mismo. El proceso basicamente consiste en capturar, desde diversas perspectivas, imágenes del objeto. Luego, se extrae en cada imagen la silueta del objeto y se procede a esculpir las mismas de manera progresiva en un paralelepípedo para obtener de esta manera como resultado un modelo tridimensional aproximado al objeto real, este método es desarrollado detalladamente por Balan en [**1**]. No obstante, dicho método aun cuando genera una representación tridimensional del objeto sujeto a estudio, la misma no nos garantiza que el modelo sea topológicamente correcto, es decir, que se preserven las propiedades topológicas del objeto original.

Por tanto, la motivación para realizar este trabajo radica principalmente en el hecho de que aun cuando la técnica de voxel carving es ampliamente utilizada, en la mayoría de los casos no se determina de manera exacta y fiable si la reconstrucción preserva las propiedades topológicas del objeto inicial; en tal sentido se desea aplicar el método de voxel carving en ciertos objetos, con propiedades topológicas conocidas, para luego evaluar la reconstrucción de los mismos mediante el estudio de su modelo AT y en última instancia verificar bajo cúales condiciones se obtiene un modelo topológicamente correcto.

El Trabajo Especial de Grado se encuentra estructurado de la siguiente manera, en el primer capítulo se presentan los preliminares necesarios para el desarrollo y referencia del mismo, específicamente se exponen algunas definiciones, proposiciones y observaciones de álgebra homológica, adicionalmente se hace énfasis en aspectos básicos de conjuntos cúbicos. En el segundo capítulo se expondrán las nociones de topología computacional, entre las que cabe destacar una técnica para la reconstrucción computacionalmente de objetos 3D, conocida como voxel carving, de igual manera se explicará el modelo AT y la forma de obtener el mismo a partir de algoritmo de incremento y decrecimiento para el cálculo de dicho modelo, dicho cómputo será realizado computacionalmente gracias al software EditCup, el cual también será expuesto en el mismo. Por útlimo, en el tercer capítulo se mostrarán algunos demostraciones en donde se reconstruye un modelo 3D a través de la técnica de voxel carving y luego se procede al cálculo de su modelo AT correspondiente mediante el software EditCup.

CAPITULO 1

Preliminares

Este capítulo tiene como finalidad ofrecer los preliminares necesarios para el desarrollo de este trabajo, así como también proporcionar una descripción teórica de aspectos relacionados con el álgebra homológica, como lo son los conceptos de categoría abeliana, cadena de complejos, grupos de homología, homología y homología persistente. Además se hace énfasis en las bases teóricas de conjuntos cúbicos [**12**], pues los mismos representan la unidad con la que se trabajará en el desarrollo de la parte computacional de este trabajo. Unas referencias estándar en álgebra homológica y topología algebraica son el libro de Weibel [**13**], el libro de teoría de categorías de Mitchell [**10**] y el libro de Munkres [**11**].

1. Categoría abeliana

El concepto de *categoría abeliana* fue introducido por Buchsbaum en [**4**], con la finalidad de unificar varias teorías de cohomología. La categoría abeliana es fundamental para la construcción de otras definiciones que permitirán el desarrollo del concepto de homología en términos categóricos. Por esta razón, antes de presentar los contenidos de mayor relevancia en la ejecución de este trabajo rse recordarán algunas definiciones de la teoría de categoría que serán utilizadas en el mismo.

DEFINICIÓN 1.1. Una *categoría* es una terna $C = (Ob(C), Hom_{_C}, \circ_{_C})$ donde:

(i) $Ob(C)$ es una clase de elementos (de algún espacio universal), denominados objetos de la categoría.

(ii) Para cada par $x, y \in Ob(C)$, $Hom_{_C}(x, y)$ representa un conjunto, denominado morfismos o flechas de x a y.

(iii) $\circ_{_C}$ es una ley de composición sobre $Hom_{_C}$ asociado, es decir, sean $w, x, y, z \in Ob(C)$

- Para todo $f \in Hom_{_C}(x, y)$ y $\forall\, g \in Hom_{_C}(y, z)$, se cumple que $g \circ_{_C} f \in Hom_{_C}(x, z)$.
- Para todo $f \in Hom_{_C}(w, x), \forall\, g \in Hom_{_C}(x, y), \forall\, h \in Hom_{_C}(y, z)$ se cumple que $(h \circ_{_C} g) \circ_{_C} f = h \circ_{_C} (g \circ_{_C} f)$.

Estos datos se encuentran sujetos a los siguientes axiomas:

(iv) Para todo $x \in Ob(C)$ existe un morfismo o flecha $id_x \in Hom_C(x,x)$ (la identidad sobre x), tal que si $f \in Hom_C(x,y)$, se satisface que $f \circ_C id_x = f$ e $id_y \circ_C f = f$.

(v) $\circ_C$ es cerrada en Hom_C, es decir, si $f \in Hom_C$ y $g \in Hom_C(y,z)$ entonces existe $h \in Hom_C(x,z)$ tal que $h = g \circ_C f$.

Nota. Si $Ob(C)$ es un conjunto decimos que C es una *categoría pequeña*.

Ejemplos.

1. Las estructuras de grupo forman un categoría denotada por Grp, en donde, $Ob(Grp)$ se encuentra constituido por todos los grupos, Hom_{Grp} son los homomorfismos de grupo y por último, la composición es la composición usual de funciones, representada por $\circ_{Grp}$.
2. La categoria Set es aquella en la cual la clase de elementos se encuentra conformada por todos los conjuntos. Si A y B son conjuntos, entonces tenemos que $Hom_{Set}(A,B)$ es el conjunto de las funciones con dominio en A y codominio en B, y la composición es la composición usual de funciones. Set es una de las categorías más utilizadas.
3. Todo espacio topológico $(X, \mathcal{T}_X)$ es una categoría. La clase de elementos son los abiertos de la topología $\mathcal{T}(X)$, si $U, V \in \mathcal{T}(X)$ y $U \subseteq V$ entonces $Hom_X(U,V) = \{i_U^V\}$ donde i_U^V es la función inclusión de U a V. En caso contrario, $Hom_X(U,V) = \emptyset$

Definición 1.2. Sea $C = (Ob(C), Hom_C, \circ_C)$ una categoría.

1. $I \in Ob(C)$ es un *objeto inicial* en C si $\forall\, A \in C$ existe una única flecha $I \to A$
2. $F \in Ob(C)$ es un *objeto final* en C si $\forall\, A \in Ob(C)$ existe una única flecha $A \to F$.
3. Si $K \in C$ es tanto un objeto inicial como final, decimos que K es un *objeto cero* para la categoría C.

Proposición 1.3. *Sea $C = (Ob(C), Hom_C, \circ_C)$ una categoría. Si C tiene un objeto inicial entonces es único, salvo isomorfismos.*

Demostración. Sean $I, I' \in Ob(C)$ objetos iniciales en C. Como I es objeto inicial existe una única flecha $I \xrightarrow{f} I'$. Como I' también satisface que es un objeto inicial existe una única flecha $I' \xrightarrow{g} I$. Componiendo ambas flechas obtenemos $I \xrightarrow{g \circ f} I$ y $I' \xrightarrow{f \circ g} I'$. Pero

por ser I inicial existe una sola flecha $I \to I$ además dicha flecha sólo puede ser id_I pues estamos en una categoría, por tanto, se tiene que $g \circ f = id_I$. De manera ánaloga se obtiene que $f \circ g = id_{I'}$. Por tanto, se concluye que f y g son isomorfismos e $I \cong I'$ □

Existen proposiciones análogas a la anterior, en el caso de que la categoría tenga objeto inicial u objeto cero.

Observación. Para cualquier categoría C los objetos finales o iniciales no necesariamente existen.

En la Figura 1.1 se muestra un ejemplo de categoría que no tiene objeto inicial ni final. Se asume que todos los elementos presentan la flecha id.

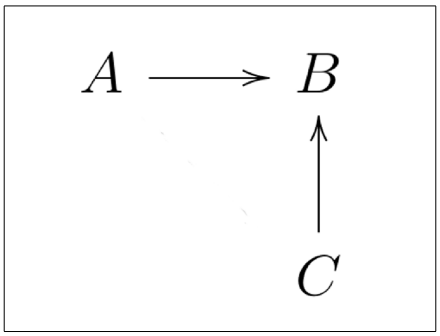


FIGURA 1.1

En la Figura 1.2 se presenta un ejemplo de una categoría con objeto cero representado por 0. Se asume que todos los elementos presentan la flecha id.

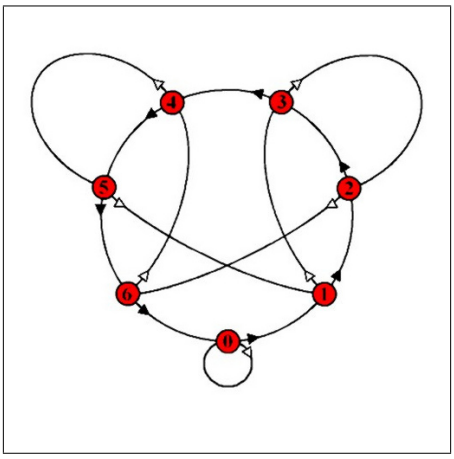


FIGURA 1.2. (0) Objeto cero

Notación. Se denotará al objeto cero de una categoría C por $0_{_C}$ y a las flechas que se tienen del mismo, por ser objeto inicial y final, como:

$$\forall\, A \in Ob(C)$$
$$i_{_0}^{A} : 0_{_C} \longrightarrow A$$
$$P_{_0}^{A} : A \longrightarrow 0_{_C}\ .$$

Definición 1.4. Sea $C = (Ob(C), Hom_{_C}, \circ_{_C})$ una categoría. Sean $A, B \in Ob(C)$, si C tiene objeto cero y $f : A \longrightarrow B$, se define el *kernel* o *núcleo* de f como un objeto $ker(f) \in Ob(C)$ tal que existe $k : ker(f) \longrightarrow A$ que satisface $f \circ_{_C} k = 0_{_C}$

$$\begin{array}{ccccc} ker(f) & \xrightarrow{k} & A & \xrightarrow{P_0^A} & 0_C \\ & & & \searrow^{f} & \downarrow{\scriptstyle i_0^B} \\ & & & & B \end{array}$$

Definición 1.5. Sea $C = (Ob(C), Hom_{_C}, \circ_{_C})$ una categoría. Sean $A, B \in Ob(C)$, si C tiene un objeto cero y $f : A \longrightarrow B$, se define el *cokernel* de f como un objeto $coker(f) \in Ob(C)$ tal que existe $j : B \longrightarrow coker(f)$ que satisface $j \circ_{_C} f = 0_{_C}$

$$\begin{array}{ccccc} A & \xrightarrow{f} & B & \xrightarrow{j} & coker(f) \\ & \searrow^{f\circ j} & & \swarrow_{P_0^{coker(f)}} & \\ & & 0 & & \end{array}$$

Definición 1.6. Sea $C = (Ob(C), Hom_{_C}, \circ_{_C})$ una categoría. Decimos que C es una *categoría abeliana* si se satisfacen las siguientes condiciones:

1. C posee elemento cero.
2. Toda flecha o morfimso admite kernel y cokernel.
3. $\forall x, y \in Ob(C)$, $Hom_{_C}(x, y)$ es un grupo abeliano.
4. C es cerrada por sumas directas finitas y productos directos finitos.

Ejemplos.

1. Recordar que Grp es la categoría conformada por todos los grupos. Si se restringen estos elementos sólo a los grupos abelianos y se consideran nuevamente los homomorfismos de grupo como la clase de morfismos, y la composición como la usual, se tendrá como resultado una categoría abeliana.

2. Si R es un anillo entonces la categoría de todos los R-módulos izquierdos o derechos es una categoría abeliana.

DEFINICIÓN 1.7. Dadas dos categorías $C = (Ob(C), Hom_C, \circ_C)$ y $D = (Ob(D), Hom_D, \circ_D)$. Un *funtor* de C a D consiste en un par $F = (F_O, F_H)$ tal que

$$F_O : Ob(C) \longrightarrow Ob(D).$$

$$F_H : Hom(C) \longrightarrow Hom(D).$$

Se satisface que

$$F_H(id_A) = id_{F_O(A)}, \text{ preserva la identidad.}$$

$$F_H(\alpha \circ_C \beta) = F_H(\alpha) \circ_D F_H(\beta), \text{ preserva la composición.}$$

Ejemplo. Sea $C = (Ob(C), Hom_C, \circ_C)$ una categoría. Se define el funtor $Id : C \longrightarrow C$ (*funtor identidad*) como $Id = (Id_O, Id_H)$ tal que $Id_O(x) = x$ e $Id_H(\ x \xrightarrow{f} y\ ) = \ x \xrightarrow{f} y$

DEFINICIÓN 1.8. Sean $C = (Ob(C), Hom_C, \circ_C)$, $D = (Ob(D), Hom_D, \circ_D)$ dos categorías y $F = (F_O, F_H)$ un funtor de C a D. Decimos que:

1. F es *fiel* si F_H es inyectiva.
2. F es *pleno* si F_H es sobreyectiva.

2. Homología

DEFINICIÓN 1.9. Una *cadena de complejos* es un par $(\{C_i\}, \{d_i\})_{i \in I}, I \subseteq \mathbb{Z}$, donde

- $\{C_i\}_{i \in I}$ es una colección de objetos de una categoría abeliana C.
- d_i con $i \in I$ son morfismos tales que

$$d_i : C_i \longrightarrow C_{i-1}$$
$$d_{i-1} \circ_C d_i = 0_C \ .$$

DEFINICIÓN 1.10. Una *cocadena de complejos* es un par $(\{C_i\}, \{d_i\})_{i \in I}, I \subseteq \mathbb{Z}$, donde

- $\{C_i\}_{i \in I}$ es una colección de objetos de una categoría abeliana C.
- d_i con $i \in I$ son morfismos tales que

$$d_i : C_i \longrightarrow C_{i+1}$$
$$d_{i+1} \circ_C d_i = 0_C \ .$$

DEFINICIÓN 1.11. Sea $(\{C_i\},\{d_i\})_{i\in I}, I \subseteq \mathbb{Z}$, una cadena de complejos que satisface $im(d_{i+1}) = ker(d_i), \forall\, i \in I$, en este caso se dice que la cadena de complejos forma una *sucesión exacta*.

DEFINICIÓN 1.12. Un *funtor exacto* es un funtor de una categoría abeliana a otra categoría abeliana que preserva sucesiones exactas, pueden ser sucesiones exactas cortas.

TEOREMA 1.13. ***(Teorema Freyd-Mitchell)*** *Sea C una categoría abeliana pequeña. Entonces existe un anillo R y un funtor fiel, pleno y exacto $F : C \longrightarrow R - mod$.*

DEMOSTRACIÓN. Ver [**10**, pág. 151]. □

El teorema anterior establece que cualquier categoría abeliana pequeña es equivalente a una categoría de módulos, por tal razón bastará estudiar categorías de módulos sobre algún anillo.

Las ideas que sirven como base para la definición de la homología tienen su origen en la fórmula de Euler para poliedros (Descartes, 1639) y en el concepto de conectividad introducido por Riemann en el caso de superficies y luego, generalizado por Betti a dimensiones superiores. No obstante, la fórmula de Euler y la idea de conectividad permanecen aisladas hasta el año de 1895 cuando Poincaré las asocia para dar inicio a lo que se conoce como teoría de la homología.

DEFINICIÓN 1.14. Dada una cadena de complejos $C = (\{C_i\},\{d_i\})_{i\in I}, I \subseteq \mathbb{Z}$. Definimos el *k-ésimo grupo de homología* de la cadena de complejos C como

$$H_k(C) = ker(d_k)/im(d_{k+1}).$$

DEFINICIÓN 1.15. Dada una cocadena de complejos $C = (\{C_i\},\{d_i\})_{i\in I}, I \subseteq \mathbb{Z}$. Definimos el *k-ésimo grupo de cohomología* de la cocadena C como

$$H^k(C) = ker(d_{k+1})/im(d_k).$$

DEFINICIÓN 1.16. La *homología* de una cadena de complejos $C = (\{C_i\},\{d_i\})_{i\in I}, I \subseteq \mathbb{Z}$ es

$$H(C) = \bigoplus_{i\in I} H_i C.$$

2.1. Homología simplicial. Sean $p_0, \ldots, p_r, \in \mathbb{R}^m$ con $m \geq r$ puntos geométricamente independientes, es decir no existe plano $(r-1)$ dimensional que contenga los $r+1$ puntos. El r-símplice generado por los $r+1$ puntos geométricamente independientes, denotado por $\sigma_r =< p_0, \ldots, p_r >$, es el conjunto

$$\sigma_r = \{x \in \mathbb{R}^m \mid x = \sum_{i=0}^{r} \lambda_i p_i, \lambda_i \geq 0 \text{ para } i = 0, \ldots, r \text{ y } \sum_{i=0}^{r} \lambda_i = 1\}.$$

Nótese que un 0-símplice es un punto $\sigma_0 =< p_0 >$, un 1-símplice $\sigma_1 =< p_0, p_1 >$ es el segmento de recta que une p_0 con p_1, un 2-símplice $\sigma_2 =< p_0, p_1, p_2 >$ es el triángulo de vértices p_0, p_1, p_2 con su interior incluido.

En la Figura 1.3 se muestra un 0-símplice $< a >$, un 1-símplice $< a, b >$, un 2-símplice $< a, b, c >$ y un 3-símplice $< a, b, c, d >$.

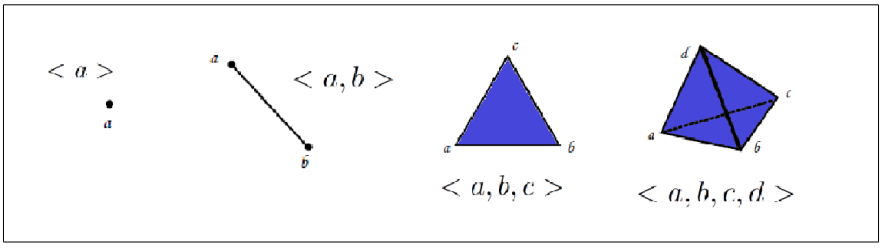


FIGURA 1.3. Símplices

Sea $q \in \mathbb{Z}$ tal que $0 \leq q \leq r$, si se escogen $q+1$ puntos $p_{i_0}, \ldots, p_{i_q}$ de $\{p_0, \ldots, p_r\}$, esos $q+1$ puntos generan un símplice σ_q, que se llamará q-cara de σ_r, para denotar este hecho se utilizará $\sigma_q \leq \sigma_r$. Si $\sigma_q \neq \sigma_r$, se dice que σ_q es una cara propia de σ_r y se denotará por $\sigma_q < \sigma_r$, esto induce un orden parcial.

DEFINICIÓN 1.17. Sea K un conjunto finito de simplices en $\mathbb{R}^d$, decimos que K es un *complejo simplicial* si los simplices de K satisfacen:

i. Una cara arbitraria de un símplice en K pertenece a K.

ii. Si σ, σ' son dos símplices en K, entonces $\sigma \cap \sigma' = \emptyset$ o $\sigma \cap \sigma' \leq \sigma$ y $\sigma \cap \sigma' \leq \sigma'$

Se utilizará la notación $(\ldots)$ para representar un símplice orientado. Un 1-símplice orientado $\sigma_1 = (p_0, p_1)$ es el segmento de recta que une p_0 y p_1, recorrido desde p_0 hasta p_1, pero si se considera el otro 1-símplice que contiene a los puntos anteriores e igual representa

al segmento de recta que une a los mismos, pero recorrido desde p_1 hasta p_0, se refiere a (p_1, p_0). Dada la orientación de $< p_0, p_1 >$ por (p_0, p_1) se denotará al otro símplice orientado $< p_1, p_0 >$, por $(p_1, p_0) = -(p_0, p_1)$.

Un r-símplice orientado, $r \geq 1$ es construido de la siguiente manera. Sean: $p_0, \ldots, p_r \in \mathbb{R}^m$ $(m \geq r)$, $r+1$ puntos linealmente independientes y sea $\{p_{i_0}, \ldots, p_{i_r}\}$ una permutación de ellos. Se dice que $\{p_0, \ldots, p_r\}$ es equivalente a $\{p_{i_0}, \ldots, p_{i_r}\}$ si

$$P = \begin{pmatrix} 0 & 1 & 2 & \ldots & r \\ i_0 & i_1 & i_2 & \ldots & i_r \end{pmatrix}$$

es una permutación par. Esto es una relación de equivalencia y cada clase de equivalencia es llamado un r-símplice orientado. Existen dos clases, una conformada por las permutaciones pares de $p_0, \ldots, p_r$ y otra por las permutaciones impares. La clase de equivalecia que contiene a $p_0, \ldots, p_r$ es denotada por $\sigma_r = (p_0, \ldots, p_r)$ y la otra es denotada por $-\sigma_r = -(p_0, \ldots, p_r)$, en resumen $(p_{i_0}, \ldots, p_{i_r}) = signo(P)(p_0, \ldots, p_r)$.

Sea $K = \{\sigma_\alpha\}_{\alpha \in I}, I \subset \mathbb{Z}$ un complejo simplicial n-dimensional. Consideremos los símplices σ_β de K como símplices orientados.

El grupo de r-cadenas $C_r(K)$ de un complejo simplicial K es el grupo abeliano generado por los r-símplices orientados de K. Si $r > dim(K)$, se define $C_r(K)$ como el grupo $\{0\}$. Un elemento perteneciente a $C_r(K)$ es llamado r-cadena.

Sea I_r el número de r-símplices de K. Se denotarán dichos símplices por $\sigma_{r,i}, 1 \leq i \leq I_r$. Un elemento $c \in C_r(K)$ se expresa como

$$c = \sum_{i=1}^{I_r} c_i \sigma_{r,i},$$

donde $c_i \in \mathbb{Z}$ y sólo un número finito de ellos es distinto de cero. Los enteros c_i son llamados coeficientes de c. La estructura de grupo asociada a $C_r(K)$ se encuentra definida como sigue:

- **Suma.** Sean $c = \sum_i c_i \sigma_{r,i}$ y $c' = \sum_i c'_i \sigma_{r,i}$ dos r-cadenas, se define la suma de $c + c'$ como
$$c + c' = \sum_i (c_i + c'_i)\sigma_{r,i}.$$
- **Elemento neutro.** El elemento neutro se define como $0 = \sum_i 0\sigma_{r,i}$.
- **Elemento inverso.** Si $c = \sum_i c_i \sigma_{r,i}$, su inverso es $-c = \sum_i - c_i \sigma_{r,i}$.

Por tanto, se tiene que $C_r(K)$ es un grupo abeliano.

Se define el operador *frontera* o *borde* ∂_r para un r-símplice de la siguiente manera. Sea $\sigma_r = (p_0, \ldots, p_r)$ un r-símplice orientado. El *borde* $\partial_r\sigma_r$ de σ_r es la (r-1)-cadena definida por

$$\partial_r\sigma_r = \sum_{i=1}^{r}(-1)^i(p_0, \ldots, \widehat{p_i}, \ldots, p_r),$$

donde el símbolo $\widehat{p_i}$ representa que dicho punto es omitido. Se entiende a ∂_r como un operador actuando sobre σ_r para producir su borde.

PROPOSICIÓN 1.18. *$(C_r(K), \partial_r)_{1\leq r\leq I_r}$ forma una cadena de complejos simpliciales.*

2.2. Homología persistente.

DEFINICIÓN 1.19. Sea $C = (\{C_i\}, \{d_i\})_{i\in I}, I \subseteq \mathbb{Z}$ una cadena de complejos. Se define el *k-ésimo número de Betti* β_K como la dimensión del k-ésimo grupo de homología. Es decir,

$$\beta_k(C) = dim(H_k(C)) = dim\left(\frac{Ker(d_k)}{Im(d_{k+1})}\right).$$

Intuitivamente, el k-ésimo número de Betti hace referencia al número k-dimensional de superficies no conectadas. Así, los siguientes números de Betti tienen las siguientes interpretaciones:

- β_0 es el número de componentes conexas.
- β_1 es el número de agujeros bidimensionales.
- β_2 es el número de agujeros tridimensionales (cavidades).

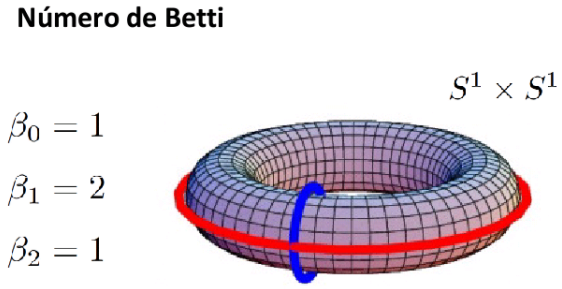


FIGURA 1.4. Toro

En la Figura 1.4 se muestra un toro junto con sus números de Betti. Un toro tiene una componente conexa, dos agujeros, y una cavidad tridimensional en el interior del mismo.

DEFINICIÓN 1.20. Un *intervalo k-dimensional de Betti* con extremos $[t_{start}, t_{end})$ corresponde a un agujero k-dimensional que aparece en la filtración en un tiempo t_{start}, permanece abierto para $t_{start} \leq t < t_{end}$ y se cierra en un tiempo t_{end}.

Recordar que un filtración es una secuencia anidada de complejos.

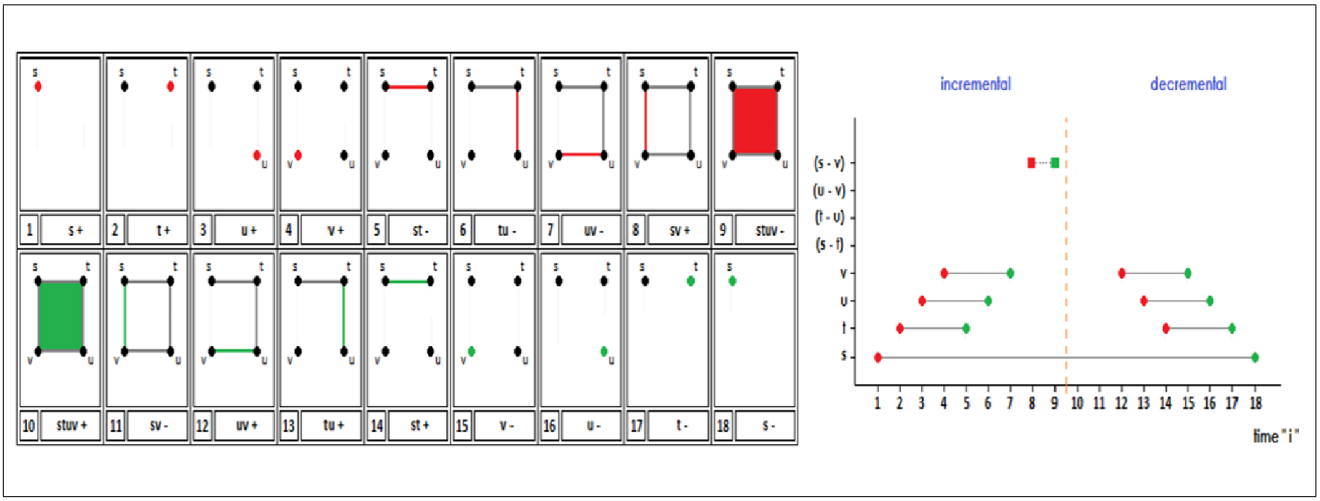


FIGURA 1.5

En la Figura 1.5 se puede observar un ejemplo de los intervalos de Betti. En la parte izquierda de la figura se encuentra la evolución de un complejo a lo largo del "tiempo" (filtración). Desde el cuadro número 1 al número 9 se agregan elementos al complejo, esto se representa con color rojo; luego del cuadro número 10 al 18 se eliminan algunos elementos, en este caso se identifican con color verde. Por otra parte a la derecha de la Figura 1.5, se observa una gráfica que representa el tiempo de vida de los distintos elementos. Se puede apreciar que el único agujero que se forma inicia en 8 y finaliza en 9. Nótese además que cuando se unen 2 puntos basta representarlo en la gráfica como un sólo punto, por ejemplo en 5 los puntos s y t se unen y se identifican los mismos simplemente como s, análogamente en 9 cuando se cierra el agujero basta representarlo como el segmento (s-v).

DEFINICIÓN 1.21. Dada una categoría $C = (Ob(C), Hom_C, \circ_C)$. Sea $A \in Ob(C)$. Una *C-filtración* de A es una sucesión $\{C_j, f_j\}_{j \in J} \subset C, J \subset \mathbb{Z}$ tal que

- $\forall j \in J$, se tiene que $f_j : C_j \longrightarrow C_{j-1}$ es inyectiva.
- $\prod_{j \in J} C_j = A$, donde $\prod$ representa el producto cartesiano.

DEFINICIÓN 1.22. Dada una categoría $C = (Ob(C), Hom_C, \circ_C)$. Sea $A \in Ob(C)$ y $\{C_j, f_j\}_{j\in J} \subset C, J \subset \mathbb{Z}$ una C-filtración de complejos de A. Se define la *homología persistente de* A, $PH(A)$ como:

$$PH(A) = \prod_{j\in J} H(C_j).$$

3. Conjuntos cúbicos

El principal objetivo de esta sección y la siguiente es proporcionar una descripción de los aspectos algebraicos de los conjuntos cúbicos, así como también de la homología cúbica.

DEFINICIÓN 1.23. Un *cubo elemental* de $\mathbb{R}^d$, $d \in \mathbb{N}$ es un conjunto de la forma

$$Q = I_1 \times I_2 \times \cdots \times I_d \ ,$$

donde cada I_i, con $i \in \{1, 2, \ldots, d\}$ es un intervalo elemental de la forma $[l, l] = [l]$ ó $[l, l + 1]$ con $l \in \mathbb{Z}$. Los intervalos elementales que contienen un único punto se denominan *degenerados*, mientras que los que tienen longitud 1 se denominan *no degenerados*.

DEFINICIÓN 1.24. Cada cubo elemental $Q = I_1 \times I_2 \times \cdots \times I_d$ de $\mathbb{R}^d$ define una *celda elemental*

$$\mathring{Q} = \mathring{I}_1 \times \mathring{I}_2 \times \cdots \times \mathring{I}_d \ ,$$

donde

$$\mathring{I} = \begin{cases} (l, l+1) & \text{si } I = [l, l+1] \\ [l] & \text{si } I = [l] \end{cases} .$$

DEFINICIÓN 1.25. El *conjunto de todos los cubos elementales* en $\mathbb{R}^d$ se denotará mediante K^d. Por otra parte, el conjunto de todos los cubos elementales se denotará mediante K, es decir

$$K := \bigcup_{d=1}^{\infty} K^d \ .$$

DEFINICIÓN 1.26. Dado un cubo elemental $Q = I_1 \times I_2 \times \cdots \times I_d \subset \mathbb{R}^d$, el número de intervalos elementales que foman Q se denota $emb(Q) = d$. Por otra parte, se llamará $dim(Q)$ al número de elementos no degenerados en Q.

Nótese que si $emb(Q) = d$, entonces $Q \in K^d$, es decir, Q es uno de los cubos elementales de $\mathbb{R}^d$. Haciendo uso de *emb* y *dim* se pueden definir las siguientes clases de cubos elementales contenidas en K:

$$K_k = \{Q \in K | dim(Q) = k\}.$$

$$K_k^d = K_k \cap K^d.$$

Producto. Sean $Q_1 \in K_{k_1}^{d_1}$ y $Q_2 \in K_{k_2}^{d_2}$, es posible contruir un nuevo cubo elemental haciendo uso del producto cartesiano de conjuntos:

$$Q_1 \times Q_2 \in K_{k_1+k_2}^{d_1+d_2} \ .$$

DEFINICIÓN 1.27. $T \in K$ es una *cara* de $P \in K$ si $T \subset P$ y se denotará mediante $T \preceq P$. Si $T \preceq P$ y $T \neq P$ se dirá que T es *cara propia* de P, y será denotado por $T \prec P$. Por último, T es *cara primitiva* de P, si T es una cara de P y $dim(Q) = dim(P) - 1$.

DEFINICIÓN 1.28. Diremos que un conjunto $X \subset \mathbb{R}^d$ es un *conjunto cúbico* o simplemente *cúbico* si X puede ser escrito como unión finita de cubos elementales.

Notación. Dado un conjunto cúbico $X \subset \mathbb{R}^d$ se utilizará la siguiente notación en esta sección, pues en la sección 2 del capítulo 2 representará una cadena de complejos cualquiera.

$$K(X) = \{Q \in K | Q \subset X\} = K^d(X).$$

$$K_k(X) = \{Q \in K(X) | dim(Q) = k\}.$$

$$K_k^d(X) = K^d(X) \cap K_k(X).$$

El conjunto $K_0(X)$ contiene todos los vértices de X, $K_1(X)$ todas sus aristas y, en general, $K_k(X)$ todos los k-cubos de X. En la Figura 1.6 se muestra un ejemplo de 4 k-cubos.

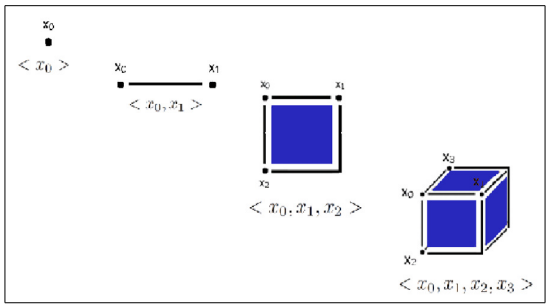


FIGURA 1.6

4. Homología cúbica

Históricamente, el primer método que se utilizó para definir la homología fue desarrollado por Poincaré en 1893 y esto trae como consecuencia lo que hoy se conoce como homología simplicial. Dicho método se encontraba limitado al uso de conjuntos que podían ser triangularizables, es decir descompuestos en celdas elementales obtenidas a partir de símplices. La homología cúbica no presenta ninguna diferencia conceptual con relación a la simplicial, pues la idea de la misma radica basicamente en cambiar las piezas triangulares, en las cuales se está descomponiendo el conjunto, por celdas cúbicas. Cabe destacar, que la homología simplicial abarca la cúbica pues todo cubo elemental es triangularizable. No obstante, la homología cúbica supone ventajas computacionales al "disminuir" la cantidad de información a procesar y hacer uso de una estructura que ya es "conocida" por el ordenador.

4.1. Cadenas cúbicas. Puesto que la homología es una herramienta algebraica, para poder definir la homología de los conjuntos cúbicos, se necesita asociar a cada cubo elemental $Q \in K_k^d$ un objeto algebraico, que se denotará $\widehat{Q}$ y al que se llamará *k-cadena* de $\mathbb{R}^d$. Es posible dar una interpretación útil para las k-cadenas $\widehat{Q} \in \widehat{K}_k^d$ como funciones en K. Específicamente, dado $Q \in K_k^d$, se puede pensar en $\widehat{Q}$ como la función $\widehat{Q} : K_k^d \to \mathbb{Z}$ definida mediante

$$\widehat{Q}(P) := \begin{cases} 1, & \text{si } P = Q \\ 0, & \text{si } P \neq Q \end{cases}, \quad P \in K_k^d.$$

Utilizando esta asociación, junto con la notación que se introdujo en la sección anterior para los conjuntos cúbicos, se obtienen sus análogos para los conjuntos de k-cadenas:

- Conjunto de todas las k-cadenas elementales de $\mathbb{R}^d$:
$$\widehat{K}_k^d := \{\widehat{Q} | Q \in K_k^d\}.$$
- Conjunto de todas las cadenas elementales de $\mathbb{R}^d$:
$$\widehat{K}^d := \bigcup_{k=0}^{\infty} \widehat{K}_k^d \ .$$

La estructura algebraica que se utilizará para las cadenas serán obtenidas de la siguiente manera: dada una colección finita $\{\widehat{Q}_1, \dots \widehat{Q}_m\} \subset \widehat{K}_k^d$ se considerarán las combinaciones lineales de la forma:

$$c = \alpha_1 \widehat{Q}_1 + \cdots + \alpha_m \widehat{Q}_m, \text{ con } \alpha_i \in \mathbb{Z}, i \in \{1, \dots, m\}.$$

El conjunto de todas estas combinaciones lineales constituye el conjunto C_d^k de k-cadenas de d-dimensiones. Por contrucción, C_d^k es un grupo abeliano libre generado por la base de $\widehat{K}_k^d$. Los cubos elementales se usan para generar elementos de la base.

Sea $c = \alpha_1\widehat{Q}_1 + \cdots + \alpha_m\widehat{Q}_m \in C_k^d$, haciendo uso de la interpretación que se ofreció anteriormente, es directo pensar en c como la función $c : K_k^d \to \mathbb{Z}$ definida mediante

$$c(P) = \alpha_1\widehat{Q}_1(P) + \cdots + \alpha_m\widehat{Q}_m(P), \text{ con } P \in K_k^d \ .$$

Por último, es posible definir un análogo algebraico del producto cartesiano de cubos elementales. Se representará dicha operación a través del símbolo operación $\diamond$ (*operación diamante*) definida mediante las siguientes reglas:

- Si $Q_1 \in K_{k_1}^{d_1}$ y $Q_2 \in K_{k_2}^{d_2}$ definimos $\widehat{Q}_1 \diamond \widehat{Q}_2 := \widehat{Q_1 \times Q_2} \in \widehat{K}_{k_1+k_2}^{d_1+d_2}$.

4.2. Operador frontera. Para cada $k \in \mathbb{Z}$, el *operador frontera*

$$\partial_k : C_k^d \to C_{k-1}^d$$

es un homomorfismo que cumple

$$\partial_{k-1}\partial_k = 0_{C}.$$

Recordar que un cubo elemental de $\mathbb{R}^d$, $d \in \mathbb{N}$ es un conjunto de la forma

$$Q = I_{_1} \times I_{_2} \times \cdots \times I_{_d} \ ,$$

donde cada $I_{_i}$, con $i \in \{1, 2, \ldots, d\}$ es un intervalo elemental de la forma $[l, l] = [l]$ ó $[l, l+1]$ con $l \in \mathbb{Z}$. Por lo tanto, se tiene que

$$\widehat{Q} = \widehat{I}_1 \diamond \widehat{I}_2 \diamond \cdots \diamond \widehat{I}_d \ .$$

A partir de esto, se puede definir el operador frontera a través de su acción sobre las cadenas elementales $\widehat{Q} \in \widehat{K}_k^d$, mediante inducción sobre la dimensión d, de la siguiente manera:

- **Caso** $d = 1$**:** Si Q es un intervalo elemental, es decir $Q = I_1$, se define

$$\partial_k\widehat{Q} := \begin{cases} 0, & \text{si } Q = [l] \\ \widehat{[l+1]} - \widehat{[l]}, & \text{si } Q = [l, l+1] \end{cases}, \quad l \in \mathbb{Z}.$$

- **Caso** $d > 1$**:** En este caso $\widehat{Q} = \widehat{I}_1 \diamond \cdots \diamond \widehat{I}_d$. Reescribiéndolo se tiene que $\widehat{Q} = I_1 \diamond \widehat{P}_1$, donde $\widehat{P}_1 = \widehat{I}_2 \diamond \cdots \diamond \widehat{I}_d$. Luego, se define

$$\partial_k\widehat{Q} = (\partial\widehat{I}_1 \diamond \widehat{P}_1) - (\widehat{I}_1 \diamond \partial_{k-1}\widehat{P}_1).$$

En el caso de que $emb(P_1) > 1$, se efectúa el proceso de manera análoga, es decir, se reescribe

$$\widehat{P}_1 = \widehat{I}_2 \diamond \widehat{P}_2, \text{ donde } \widehat{P}_2 = \widehat{I}_3 \diamond \cdots \diamond \widehat{I}_d,$$

luego se le aplica el operador frontera. El proceso continua hasta llegar a $\widehat{P}_{d-1} = \widehat{I}_d$.

Es inmediato comprobar que, dado un conjunto cúbico cualquiera $X \subset \mathbb{R}^d$, la acción del operador frontera transforma cadenas de X en cadenas de X. Es decir, se satisface que

$$\partial_k(C_k(X)) \subset C_{k-1}(X).$$

Notación. La restricción de ∂_k al conjunto $C_k(X)$ se denota mediante ∂_k^X y satisface que

$$\partial_k^X : C_k(X) \to C_{k-1}(X).$$

Definición 1.29. El *complejo de cadenas cúbico* $C(X)$ de un conjunto cúbico $X \subset \mathbb{R}^d$ se define como

$$C(X) := \{C_k(X), \partial_k^X\},$$

donde $C_k(X)$ son los grupos de k-cadenas cúbicas generadas por $K_k(X)$ con la operación de composición y ∂_k^X es el operador frontera cúbico restringido al conjunto X.

Proposición 1.30. *Sea* $C(X) := \{C_k(X), \partial_k^X\}$ *un complejo de cadenas cúbico. Entonces se cumple que*

$$\partial_{k-1}^X \partial_k^X = 0_{C}.$$

A partir de los operadores frontera se pueden introducir los siguientes conceptos algebraicos para el conjunto cúbico X.

Definición 1.31. Sea $C(X) := \{C_k(X), \partial_k^X\}$ un complejo de cadenas cúbico.

- El conjunto de los *k-ciclos* de X se define como $Z_k(X) = ker(\partial_k^X)$.
- El conjunto de los *k-bordes* de X se definen como $B_k(X) = im(\partial_{k+1}^X)$.

Puesto que $\partial_{k-1}^X \partial_k^X = 0$ se tiene que $B_k(X)$ es siempre un subgrupo de $Z_k(X)$. Nuestro interés radica en caracterizar aquellos ciclos que no sean bordes. Estos objetos constituyen el *k-ésimo grupo de homología cúbica*.

DEFINICIÓN 1.32. Sea $C(X) := \{C_k(X), \partial_k^X\}$ un complejo de cadenas cúbico. El *k-ésimo grupo de homología cúbica* del conjunto X es:

$$H_k(X) := Z_k(X)/B_k(X).$$

DEFINICIÓN 1.33. La colección de todos los grupos de homología de un conjunto X se denomina homología cúbica de X, y se expresa como:

$$H(X) := \{H_k(X)\}_{k\in\mathbb{Z}} \ .$$

CAPITULO 2

Nociones de topología computacional

Este capítulo tiene como objetivo suministrar aspectos teóricos y prácticos acerca de las nociones de topología computacional. Específicamente, se detallará una técnica utilizada para producir objetos 3D digitalmente, dicha técnica es el voxel carving [**1**]. Por otra parte, se expondrá la teoría relacionada con el modelo algebraico topológico, conocido como modelo AT [**5**], el cual permitirá realizar el estudio homológico a los objetos reconstruídos en 3D, en particular aquellos objetos obtenidos a través del voxel carving. Se explicarán algunos algoritmos para obtener el modelo AT computacionalmente, el algoritmo de incremento y decrecimiento [**5**] y por último se presentará el software EditCup, el cual permite realizar el cálculo del modelo AT [**2**],[**3**],[**6**].

1. Voxel carving

El voxel carving es una técnica empleada para producir un objeto tridimensional digitalmente a partir de imágenes capturadas desde un conjunto de cámaras colocadas alrededor del objeto y en posiciones conocidas.

El proceso de reconstrucción involucra capturar un conjunto de imágenes de un objeto, y mediante el análisis de dichas imágenes obtener una descripción de la forma del mismo. En cada imagen se realiza como primer paso la segmentación de la región de interés (silueta del objeto) con respecto al fondo de la misma, luego se crea una "caja virtual" alrededor de la posición del objeto en el espacio 3D, a continuación, se hace uso de las siluetas extraídas de cada imagen para de esta manera eliminar los voxeles inconsistentes (voxeles que representan fondo y no silueta) del volumen definido, esta iteración se repite con todas las siluetas obtenidas de las distintas imágenes.

1.1. Ejemplo. Se explicará de forma más detallada el proceso del voxel carving haciendo uso del siguiente ejemplo, el cual representa una de las formas más básicas de la técnica de voxel o space carving, en la cual cada imagen es utilizada como una máscara. Seguidamente,

se coloca una caja en el centro de la escena y a partir de cada imagen nos limitamos a observar lo que se encuentra fuera de la silueta del objeto, cualquier cosa fuera del mismo es eliminada de la caja. Evidentemente, este proceso requiere que se conozca la posición de las cámaras con respecto al objeto en el momento de la captura de las imágenes, este hecho representa un problema independiente a la técnica de voxel carving.

Esta técnica se ha perfeccionado a lo largo de la última década y puede ser optimizada para ser computacionalmente eficiente. No obstante, en este ejemplo prevalece la simplicidad sobre la eficiencia, puesto que nuestro principal objetivo es explicar la técnica y mostrarla en MATLAB (Ver Apéndice 1.)

Las imágenes utilizadas en este ejemplo fueron proporcionadas por Wolfgang Niem de la Universidad de Hannover, los datos de las cámaras suministradas por el Dr. A. W. Fitzgibbon y el Prof. A. Zisserman de el Grupo de Investigación en Robótica de la Universidad de Oxford. Todas la funciones para este ejemplo se encuentran en el paquete "spacecarving" y los datos en el archivo "DinosaurData", los cuales pueden ser obtenido en http://www.robots.ox.ac.uk/∼ vgg/data/data-mview.html, la cual fue consultada el 03 de noviembre de 2013. Esta demostracón fue desarrolada por Loren Shure.

Cargar los datos de las imágenes y las cámaras. El primer paso a realizar consiste en la lectura del archivo en donde se encuentren los datos, en este caso "DinosaurData", para luego poder hacer uso de la información de las cámaras, así como también la imagen asociada a cada una de ellas. En este ejemplo, y particularmente en este trabajo, no existe interés en determinar el proceso para la captura de las imágenes. La Figura 2.1 está conformada por las imágenes sujetas a estudio en este ejemplo.

FIGURA 2.1

Conversión de las imágenes a siluetas. La imagen obtenida de cada cámara se convierte en una imagen binaria, utilizando en este caso el hecho de que el fondo de las mismas es azul y también se emplean operadores morfológicos para limpiar los bordes. El resultado se convertirá en la máscara de la silueta. Los comandos (MATLAB) "bwareaopen" e "imopen" son sumamente útiles en el proceso de creación de las siluetas. La Figura 2.2 muestra una comparación entre la imagen de entrada y la silueta.

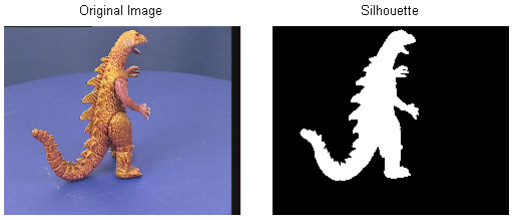


FIGURA 2.2

Es importante destacar el hecho de que los agujeros en las máscaras deben ser obtenidos de manera precisa, ya que pueden ser eliminados ciertos voxels que forman parte del objeto y no del fondo, además se generarían en el modelo final agujeros que el objeto no presenta.

Matriz voxel. En este paso, se genera una matriz voxel 3D (caja) de elementos listos para ser tallados (removidos) en el proceso de voxel carving (Figura 2.3).

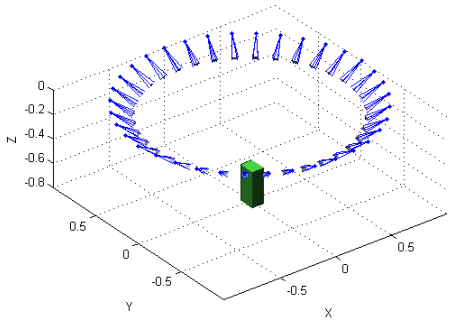


FIGURA 2.3. Matriz voxel y ubicación de cámaras.

Nótese que mientras mayor sea la cantidad de elementos de la matriz se obtendrá un modelo más detallado del objeto, pero de igual manera se necesitará mayor cantidad de espacio disponible en el disco duro del computador, y el tiempo de cómputo se verá incrementado. Por otra parte, si se utilizan pocos elementos se puede perder información y obtenerse un

modelo que no representa fielmente al objeto de estudio. En este ejemplo empleamos 6000000 de voxels.

Tallar voxels con la primera imagen. Se proyecta desde la camara, con la cual fue adquirida la primera imagen, la silueta sobre la matriz voxel. Cualquier voxel que se encuentre fuera de la silueta será removido, dejando de esta manera sólo los que se encuentran dentro de la misma (Figura 2.4).

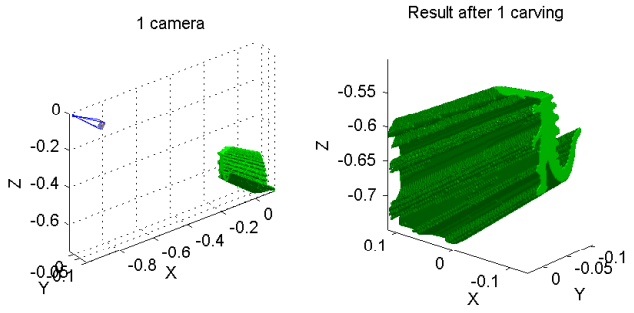


FIGURA 2.4

Haciendo uso de una única imagen los datos que se obtienen no proporcionan información relevante para el análisis topológico del objeto, es por esta razón que en esta técnica deben ser empleadas una mayor cantidad de imágenes obtenidas desde diversos puntos de vista.

Agregando mayor cantidad de puntos de vista. A medida que se agregan más puntos de vistas del objeto se refina la forma de la reconstrucción. Por ejemplo en este caso, si se incluyen dos más, obtenemos una figura que es más "parecida" al objeto. En la Figura 2.5 se observa la aplicación del método de voxel carving con 3 imágenes.

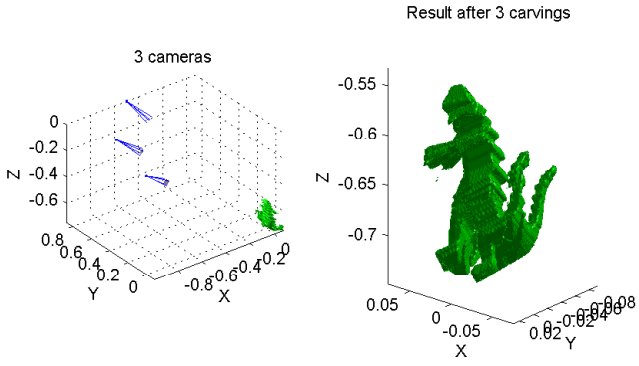


FIGURA 2.5

Todas las vistas. El último paso consiste en aplicar el procedimiento a todas las imágenes. En este ejemplo se consideran los 36 puntos de vista (36 imágenes) del objeto. Figura 2.6 representa el modelado final con las 36 imágenes.

FIGURA 2.6

2. Modelo AT

En la sección anterior se ofrece una manera para recontruir un objeto en 3D a partir de distintas imágenes del mismo, no obstante el objetivo de este trabajo no se radica exclusivamente con el conocimiento de dicha reconstrucción, sino que además se quieren calcular los grupos de homología para de esta manera determinar la cantidad de agujeros n-dimensionales que presente el objeto, con esta finalidad se debe conocer la definición del modelo AT, así como también algunos algoritmos [**7**] para el cómputo del mismo. En este sección la operación que se utilizará entre funciones sera la composición por tal motivo no se colocará el símbolo de dicha operación.

DEFINICIÓN 2.1. Una *contracción de cadena* de un complejo de cadena (C, ∂_C) a otro complejo de cadena $(C', \partial_{C'})$ es un conjunto de tres homomorfismos (f, g, ϕ) tal que $f : C \longrightarrow C'$ y $g : C' \longrightarrow C$ son aplicaciones de cadena; fg es la aplicación identidad de C'; $\phi : C \longrightarrow C$ es una homotopía de cadena de la aplicación identidad de C (id_C) a gf, es decir, $\phi\partial_C + \partial_{C'}\phi = id_C - gf$. Además, se satisfacen las propiedades de aniquilación $f\phi = 0_{C'}$, $\phi g = 0_C$ y $\phi\phi = 0_C$.

Algunas propiedades importantes de las contracciones de cadena son: C' tiene menor o igual cantidad de generadores que C. C y C' tienen grupos de homología isomórficos.

DEFINICIÓN 2.2. Un *modelo AT* para un complejo de cadena (C, ∂) es el conjunto (K, h, f, g, ϕ), donde h es un conjunto de generadores de una cadena de complejos H isomórfica a la homología de K y (f, g, ϕ) es una contracción de cadena de K a H. Esto implica que $f\partial = 0_{H}$ y $\partial g = 0_{C}$

Observación. En este trabajo se considerará el modelo AT como el conjunto (K, H, f, g, ϕ), donde H es la cadena de complejo que se describe en la definición anterior, puesto que los algoritmos para el cálculo del modelo AT permiten obtener directamente dicho conjunto.

A continuación se describirán dos algoritmos para el cálculo de modelos AT, el primero de ellos es el algoritmo de incremento y el segundo corresponde al de decrecimiento.

2.1. Algoritmo de incremento. Sea K una cadena de complejos con un orden parcial en sus elementos $\{\sigma_1, \sigma_2, \ldots, \sigma_n\}$, es decir, satisface que si σ_i es una cara de σ_j entonces $i < j$. A continuación, considérese una filtración de K, es decir una secuencia anidada de subcomplejos, $\emptyset = K_0 \subseteq K_1 \subseteq \cdots \subseteq K_n$, tal que $K_i = \{\sigma_1, \ldots, \sigma_i\}$, nótese que todas las caras propias de σ_i se encuentran contenidas en K_{i-1}.

Primero, se define un modelo AT $(K_1, H_1, f_1, g_1, \phi_1)$ para K_1 de la siguiente manera: $H_1 := \{\sigma_1\}$, $f_1(\sigma_1) = \sigma_1$, $g_1(\sigma_1) = \sigma_1$ y $\phi_1(\sigma_1) = 0$. Luego, se procede a agregar un complejo nuevo σ_i a la vez, con $i = 2, \ldots, n$ y se calcula un modelo AT en cada adición $(K_i, H_i, f_i, g_i, \phi_i)$, donde $K_i = K_{i-1} \cup \{\sigma_i\}$, como sigue: inicialmente, $H_i := H_{i-1}$; $f_i(\mu) := f_{i-1}(\mu)$ y $\phi_i(\mu) := \phi_{i-1}(\mu)$ para cualquier $\mu \in K_i \setminus \{\sigma_i\}$; $g_i(h) := g_{i-1}(h)$ para cualquier $h \in H_i$. Luego, tómese en consideración $f_{i-1}\partial(\sigma_i)$ para detectar si σ_i crea o destruye una clase de homología:

(1) Si $f_{i-1}\partial(\sigma_i) = 0$ se crea una nueva clase de homología, por tanto, $H_i := H_{i-1} \cup \{\sigma_i\}$, $f_i(\sigma_i) := \sigma_i$, $g_i(\sigma_i) := \sigma_i + \phi_{i-1}\partial(\sigma_i)$ y $\phi_i(\sigma_i) := 0$.
(2) Si $f_{i-1}\partial(\sigma_i) \neq 0$ se destruye una clase de homología. Sea j el índice más grande tal que $\sigma_j \in f_{i-1}\partial(\sigma_i)$, entonces notemos que $j < i$ y $dim(\sigma_j) = dim(\sigma_i) - 1$. Luego, $H_i := H_{i-1} \setminus \{\sigma_j\}$, $f_i(\sigma_i) := 0$, $\phi_i(\sigma_i) := 0$.

Además dos operaciones son aplicadas a todos los complejos $x \in K_i$ de tal manera que $\sigma_j \in f_{i-1}(x)$:

- Actualización de f: $f_i(x) := f_{i-1}(x) + f_{i-1}\partial(\sigma_i)$.
- Actualización de ϕ: $\phi_i(x) := \phi_{i-1}(x) + \sigma_i + \phi_{i-1}\partial(\sigma_i)$.

LEMA 2.3. *σ_i pertenece a un $k-$ciclo en $C(K_i)$ si y sólo si $f_{i-1}\partial(\sigma_i) = 0$.*

DEMOSTRACIÓN. Si $f_{i-1}\partial(\sigma_i) = 0$, entonces $\partial\sigma_i = \partial\phi_{i-1}(\partial\sigma_i)$. Por lo tanto, $\sigma_i + \phi_{i-1}\partial(\sigma_i)$ es un $k-$ciclo y σ_i pertenece a este. Por otra parte, si σ_i pertenece a un $k-$ciclo a en $C(K_i)$, entonces $a = \sigma_i + b$ donde b e una $k-$cadena en $C(K_i)$. Puesto que $\partial a = 0$ entonces $f_{i-1}\partial(\sigma_i) = f_{i-1}\partial(b)$. Además como $f_{i-1}\partial(b) = \partial f_{i-1}(b) = 0$ entonces $f_{i-1}\partial(\sigma_i) = 0$. □

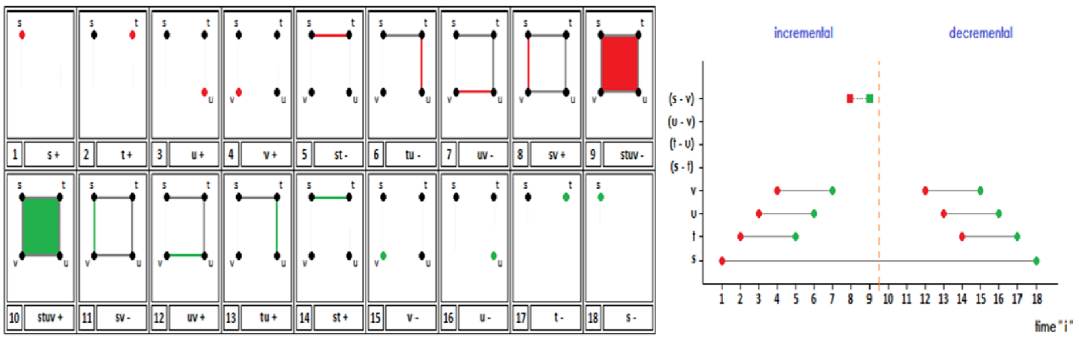
2.2. Algoritmo de decrecimiento. Sea (K, H, f, g, ϕ) un modelo AT para un complejo K. Sea σ un complejo maximal de K. Entonces un modelo AT para $K' = K \setminus \{\sigma\}$ puede ser construido de la siguiente manera:

Inicialmente $H' := H$, $g'(h) := g(h)$ para todo $h \in H'$, $f'(x) := f(x)$ y $\phi'(x) := \phi(x)$ para todo $x \in K'$.

(1) Si existe $\beta \in H$ tal que $\sigma = g(\beta)$ entonces σ destruye la clase de homología $[g(\beta)]$ creada antes por β. Por lo tanto, $H' := H \setminus \{\beta\}$; $f'(x) := f(x) + \beta$ si $\beta \in f(x)$ y $x \in K$:'; $g'(h) := g(h) + g(\beta)$ si $\sigma \in g(h)$ y $h \in H'$; $\phi'(y) := \phi(y) + g(b)$ si $\sigma \in \phi(x)$ y $x \in K'$.

(2) De lo contrario, puesto que $\sigma \in \phi\partial\sigma$, existe $\mu \in \partial\sigma$, $\mu \notin H$, tal que $\sigma \in \phi(\mu)$. Entonces μ crea una nueva clase de homología $[g'(\mu)]$. Por lo tanto $H' := H \cup \{\mu\}$; $g'(\mu) := gf(\mu) - \partial\phi(\mu)$. $f'(x) := f(x) + \mu + f(\mu)$ y $\phi'(x) := \phi(x) + \phi(\mu)$ si $\sigma \in \phi(x)$ y $x \in K'$.

En la Figura 2.17 se puede apreciar un ejemplo en donde se puede aplicar el algoritmo de incremento y decrecimiento, desde el paso 1 al 9 de la filtración se agrega un símplice a la vez y luego de la 10 a la 18 se elimina un símplice maximal, por tanto en la primera parte podría ser aplicado el algoritmo de incremento y en la segunda el de decrecimiento.



3. EditCup

El software EditCup, [**2**],[**3**],[**6**], desarrollado por J.M. Berrio, M.M. Maraver y F.Leal, permite visualizar conceptos abstractos en objetos tridimensionales, representado a estos a partir de complejos simpliciales. Algunos de estos conceptos son invariantes topológicos como los grupos de homología y cohomología, así como también el anillo de cohomología. El software a grandes rasgos opera de la siguiente forma, primero reduce el complejo simplicial inicial (objeto 3D) a otro complejo simplicial, de tal manera que dicho complejo presenta una equivalencia de homotopía con el primero (contracción de cadena), luego calcula los invariantes del esqueleto resultante.

3.1. Software EditCup. EditCup es un software en fase beta que ha sido diseñado con la finalidad de ser una herramienta de visualización de objetos 3D, que mejore la comprensión de conceptos altamente abstractos de topología algebraica, como lo son la (co)homología y el anillo de (co)homología, mediante el cálculo del modelo AT. La Figura 2.7 representa la interfaz de usuario del software. En éste, un proyecto (mundo virtual) consiste de una representación simplicial particular. Haciendo uso de los cursores podemos movernos en cualquier dirección y sentido dentro del mismo. El software cuenta con la siguiente leyenda para facilitar la comprensión de los objetos, el color rojo representa a los tetraedros, verde para los triángulos, azul para los lados y negro para los vértices, se añade un pequeño voxel rojo dentro de cada tetraedro para saber si el mismo se encuentra hueco o no.

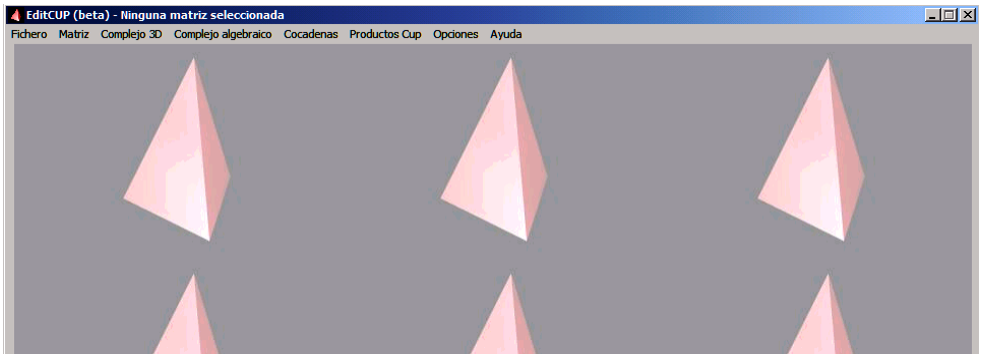


FIGURA 2.7. Interfaz de usuario de EditCup

Para crear un nuevo proyecto, el usuario debe seleccionar la opción fichero en la interfaz de usuario y luego seleccionar nuevo, seguidamente se debe establecer el número de filas, columnas y de capas del proyecto que se va a generar.

Por ejemplo, en la Figura 2.8 se muestra la creación de un fichero llamado Prueba con 5 filas, 3 columnas y 2 capas.

FIGURA 2.8. Prueba

Luego, se puede crear una representación de un complejo simplicial seleccionando el comando Editar Matriz. Seguidamente se desplegará una ventana (Figura 2.9), en la parte izquierda de la misma se pueden seleccionar los símplices necesarios (vértices, segmentos, triángulos, tetraedros), en la parte superior derecha de la ventana se irá generando la representación simplicial en construcción y por último en la parte inferior derecha se encuentran las opciones de fila, columna y capa las cuales permiten el desplazamiento por los distintos voxeles que conforman el proyecto, visualmente se apreciará un voxel de color amarillo que indica el lugar en donde se encuentra el mismo, adicionalmente en la parte inferior derecha se encuentran las opciones para poder desplazarse en la representación.

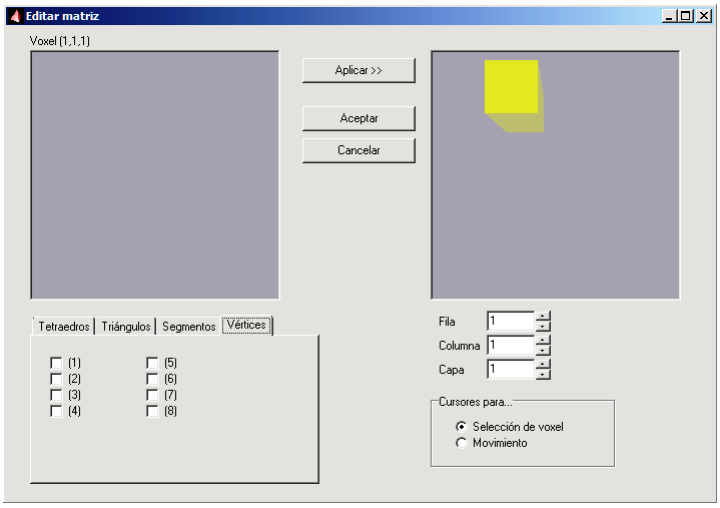


FIGURA 2.9. Interfaz

A continuación en la Figura 2.10 se muestra un ejemplo de una representación simplicial realizada con el programa que contiene 2 voxeles, 1 tetaraedro, 1 triángulo y 1 segmento.

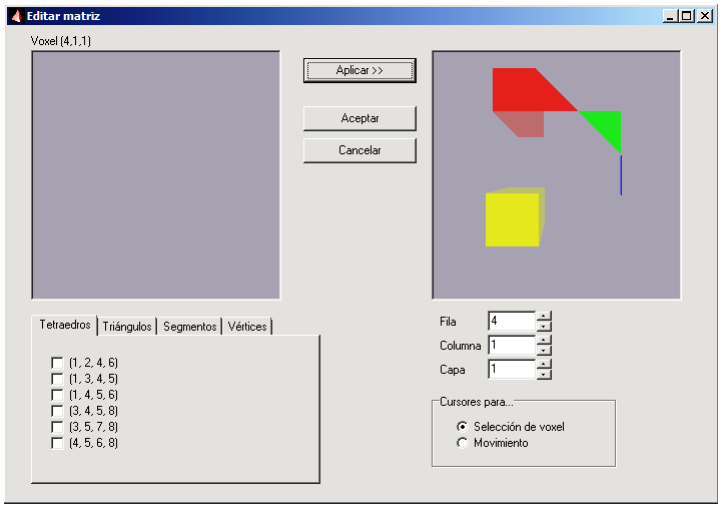


FIGURA 2.10

Observación. El software EditCup permite hacer las representaciones de complejos simpliciales, no obstante estamos interesados en analizar las reconstrucciones por voxel carving las cuales en un principio son complejos cúbicos, por tanto para obtener dicha representación en EditCup basta escoger los 6 tetraedros asociados al voxel que se desea generar.

EditCup permite calcular las clases de (co)homología de la representación, las cuales ofrecerán información acerca de la cantidad de componentes conexas, túneles y cavidades que presenta la misma. Para ello el software emplea los algoritmos que fueron descritos en la sección anterior, en primer lugar el programa realiza un adelgazamiento topológico (una contracción de cadena) al objeto y luego procede al cómputo del adelgazamiento algebraico (cadenas de (co)homología).

Para obtener estos datos se debe, una vez creado el fichero, seleccionar la opción Complejo 3D y luego Visualizar, a continuación se debe presionar el botón Calcular Clases de (Co)homología, una vez que concluya los cómputos correspondientes se desplegará una ventana notificando que las clases han sido correctamente calculadas

En el ejemplo anterior, procediendo a realizar dichos cálculos se obtendrá que sólo existe 1 componente conexa y no existen ni túneles ni cavidades (Figura 2.11). Resultado este que coincide con lo esperado pues la figura no posee ningún agujero y tampoco posee ninguna separación.

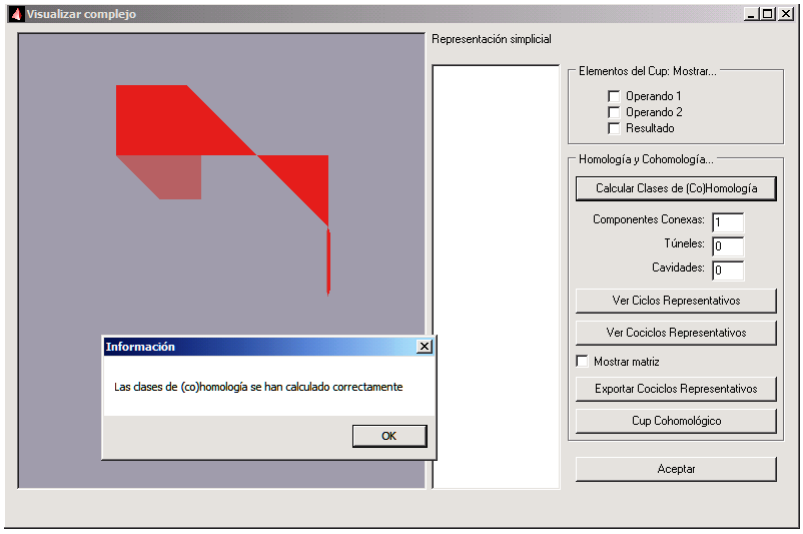


FIGURA 2.11

EditCup cuenta con una aplicación de gran utilidad para visualizar los ciclos representativos asociados al modelo en estudio. Para ello se debe presionar el botón Ver Ciclos Representativos que se encuentra en la misma pantalla en la cual se calculan las clases, en este caso la leyenda asociada a la figura que se obtiene es la siguiente: un voxel de color azul representa una componente conexa, una curva cerrada de color amarillo hace referencia a un agujero y un objeto de color verde representa una cavidad. En el ejemplo que se ha utilizado sólo hay un ciclo y este se encuentra representado por un voxel de color azul, pues este objeto sólo tiene una componente conexa, este resultado se aprecia en Figura 2.12.

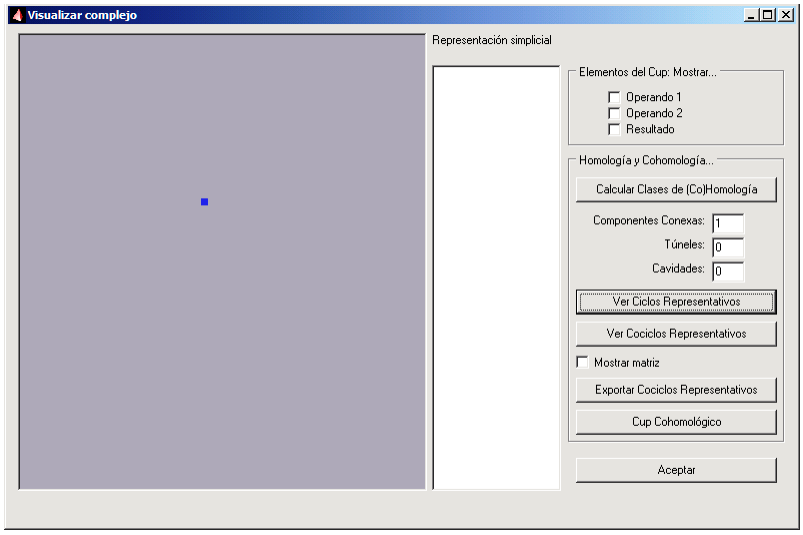


FIGURA 2.12

Ahora, considérese el caso de un toro, si se realizan los cálculos que fueron aplicados en el ejemplo anterior, se obtendrá que el número de componentes conexas es 1, el número de túneles es 2 y el número de cavidades es 1. La Figura 2.13 representa este ejemplo.

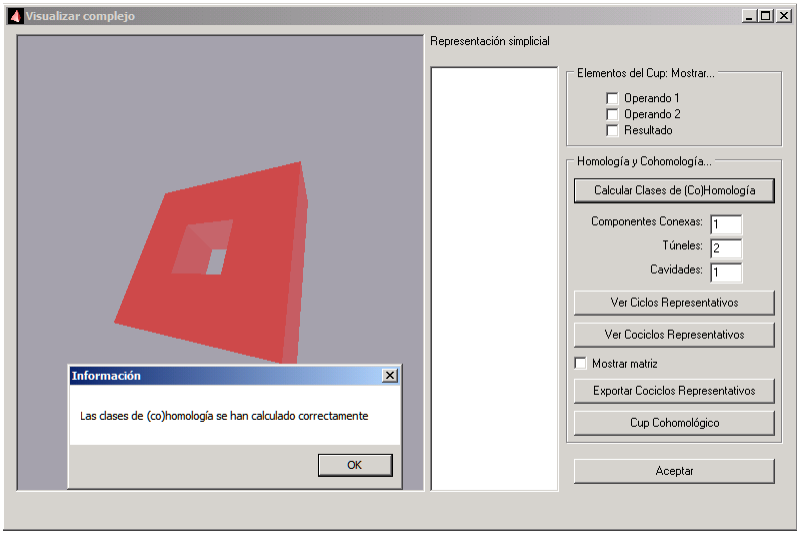


FIGURA 2.13. Toro

Sus ciclos representativos se encuentran en la Figura 2.14.

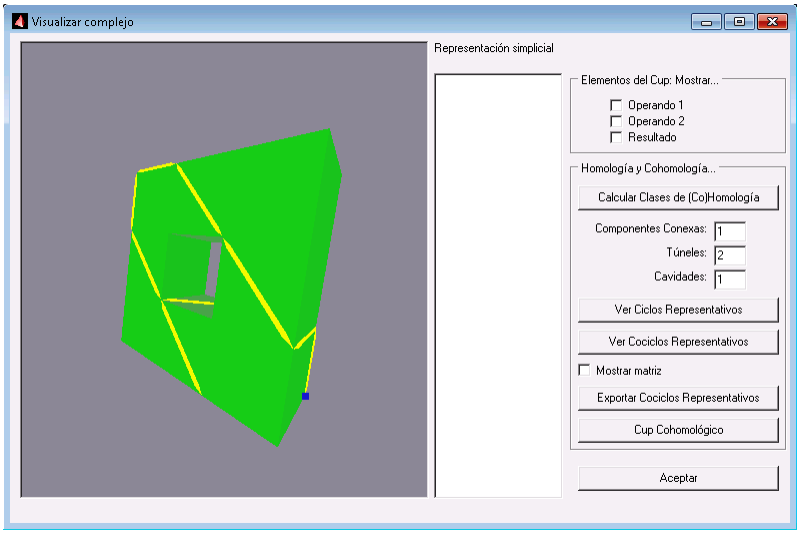


FIGURA 2.14. Toro

Nótese que los resultados que se obtienen corresponden con los resultados teóricos esperados que se fueron comentados en los ejemplos de la Definición 1.19 relacionada a los números de Betti. La Figura 2.15 establece los números de Betti para el toro.

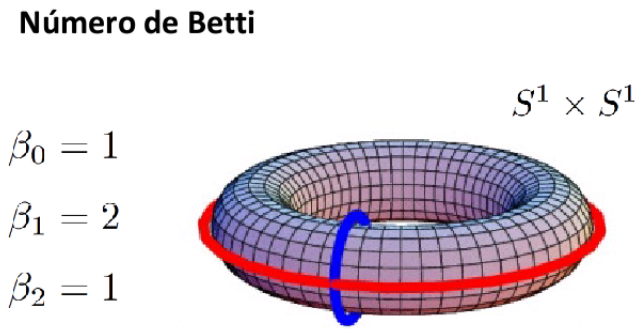


FIGURA 2.15. Toro

Observación. (Tiempo de cómputo) El ordenador en el cual se ejecutaron todas las aplicaciones de esta sección presenta las siguientes características: procesador AMD PHENOM II X4 955 a 3.20 GHz, 8 GB memoria RAM DDR3 a 1600mhz, tarjeta de video NVIDIA GeForce GT430 1GB DDR3. En todos los ejemplos de esta sección no se utilizaron aplicaciones distintas a los softwares necesarios.

El tiempo de cálculo del ejemplo del método de voxel carving que se realizó tiene una duración estimada de 75 segundos. Por otra parte, el tiempo de cómputo para los ejemplos que se ejecutaron en el software EditCup fueron variables, es decir en el caso del toro en algunas oportunidades el tiempo fue de 15 segundos, en otras alrededor de 2 minutos, no obstante en una cantidad considerable de casos (40%) el software mostraba la notificación "Out of memory" (sin memoria), cabe destacar que las condiciones bajo las cuales se ejecutó el software para la realización de todos los ejemplos fueron las mismas. Es probable, que esta gran diferencia en los tiempos de cálculo se debe al hecho de que EditCup se encuenta en fase beta y quizás aun no ha sido optimizado en la utilización de los recuersos del ordenador.

CAPITULO 3

Modelo AT y homología

Este capítulo tiene como finalidad presentar algunos ejemplos en los cuales se evidencie la reconstrucción por el método de voxel carving de ciertos objetos, para luego proceder al cálculo del modelo AT a los mismos mediante el software EditCup y de esta manera evaluar si las reconstrucciones son topológicamente correctas en relación al objeto inicial.

1. Reconstrucción de taza

En este ejemplo se trabajará con una taza, de la cual se obtienen un conjunto de imágenes a partir de cámaras ubicadas en distintas y conocidas posiciones.

En primer lugar se procederá a realizar la construcción del objeto en 3D haciendo uso de la técnica de voxel-carving que ya fue estudiada y por tal razón no se volverá a enfatizar en los detalles de cada paso de este proceso sino por el contrario solamente serán nombrados los mismos y detallados los pasos que sean de mayor importancia para este caso particular.

El primer paso consiste en la carga de los datos, es decir, las imágenes, posición de las cámaras y la relación imagen-cámara. Las imágenes sujetas a este estudio sse encuentran representadas en la Figura 3.1.

FIGURA 3.1. Imágenes taza

El siguiente procedimiento es convertir las imágenes en siluetas (Figura 3.2).

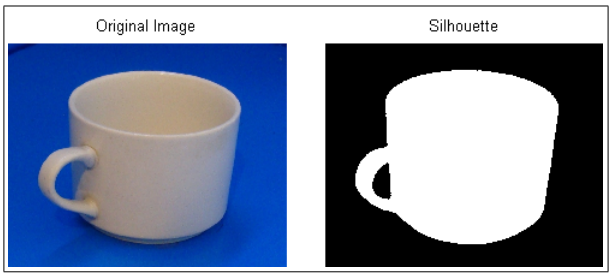


FIGURA 3.2. Siluetas taza

Luego se crea la matriz de voxel (Figura 3.3) de la cual serán tallados todos los voxeles que no pertenezcan a las siluetas.

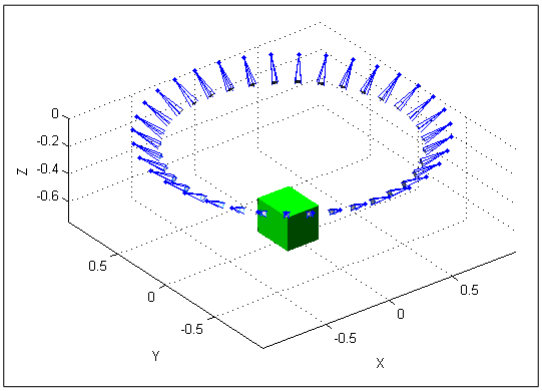


FIGURA 3.3. Matriz voxel

Se tallan los voxeles haciendo uso de la primera imagen. El resultado de este paso se observa en la Figura 3.4.

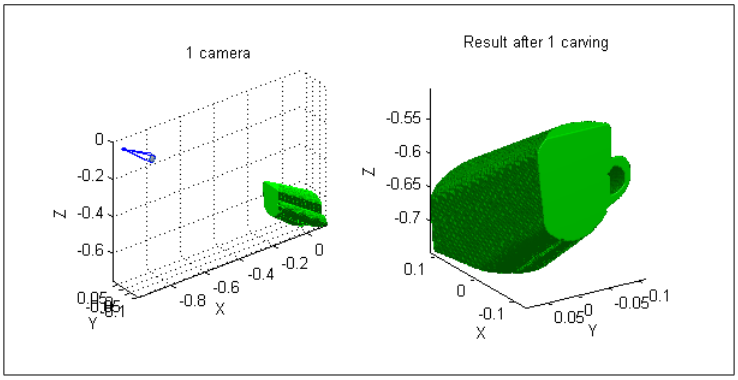


FIGURA 3.4

Luego, se agregan un par de vistas adicionales. Resultado Figura 3.5.

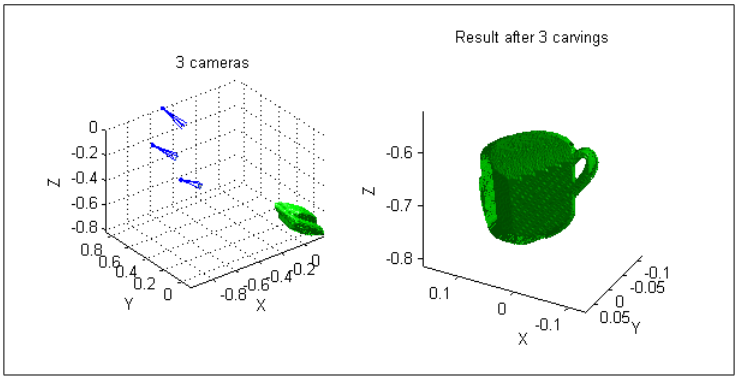


FIGURA 3.5

Por útlimo se debe terminar de realizar la reconstrucción haciendo uso todas las imágenes, no obstante en este ejemplo sólo serán agregadas las primeras 20 y las imágenes de la 28 a la 35, el resultado de esta reconstrucción puede ser visto en la Figura 3.6. La selección de este conjunto específico de imágenes se debe al hecho de que cuando se utilizan todas, el asa de la taza pierde la conexidad pues la misma es muy delgada en el objeto real y trae como consecuencia una gran cantidad de dificultades para establecer la correcta posición de las cámaras, dichas dificultades fueron solucionados en su mayoría y para evitar este problema basta utilizar las imágenes que son especificadas anteriormente, además el modelo que se obtiene con estas es homotópico al objeto real.

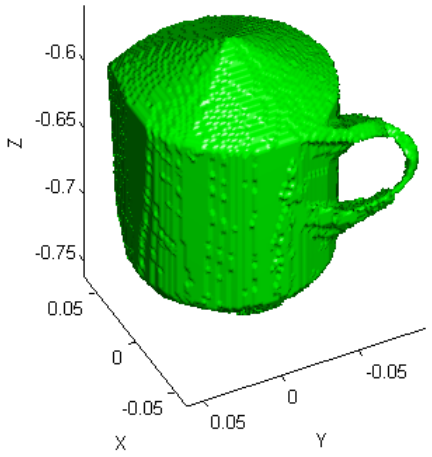


FIGURA 3.6

Nótese que la taza no presenta la cavidad que tienen todas las tazas, esto se debe a uno de los problemas de la técnica del voxel-carving, el cual es que al obtener las siluetas a partir de las imágenes, las mismas no presentan la información acerca de las depresiones del objeto. No obstante, esto no representa un problema pues el resultado sigue siendo topológicamente correcto.

Luego, a partir de la reconstrucción 3D que se obtuvo de la taza y haciendo uso de estos datos en el software EditCup se pueden obtener los resultados deseados, como lo son los grupos de homología y por tanto poder determinar el número de componentes conexas, túneles y cavidades. La Figura 3.7 representa el resultado del análisis de estos datos en el software luego de haber calculado las clases de homología y además se muestran los ciclos representativos.

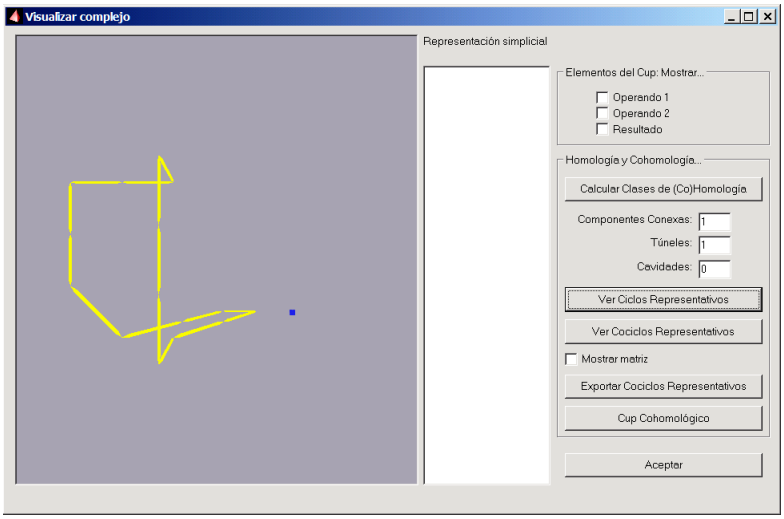


FIGURA 3.7

Se puede observar que los resultados son los esperados para esta reconstrucción, es decir, el objeto presenta una componente conexa, un túnel y no tiene cavidades. En los ciclos representativos se puede apreciar un voxel azul que representa la componente conexa, y de igual forma una curva cerrada de color amarillo, ésta identifica el agujero correspondiente al asa de la taza.

Ahora mostremos otro ejemplo, para ello se empleará la reconstrucción del objeto que se obtiene al utilizar las 36 imágenes que fueron tomadas al mismo, en este caso la taza del asa se encuentra "rota" y por tanto el objeto 3D no presenta el agujero en el mismo. Este hecho deberá ser verificado haciendo uso de EditCup.

En la Figura 3.8 se presenta el modelo obtenido a partir de las 36 imágenes.

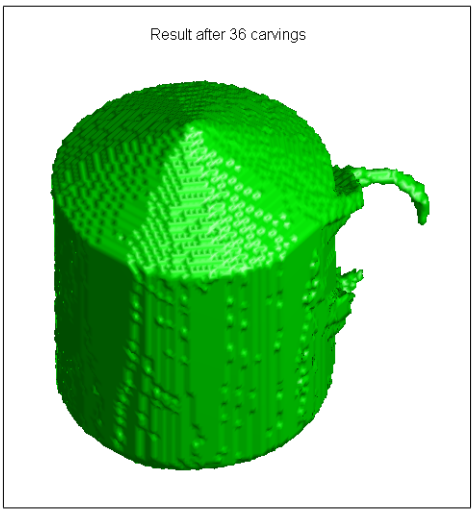


FIGURA 3.8

Luego, el resultado que se obtiene en EditCup se encuentra representado en la Figura 3.9, y establece la existencia solamente de una componente conexa y de igual manera, establece la ausencia tanto de túneles como de cavidades, siendo este resultado lo esperado. En la Figura 3.9 sólo se aprecia un voxel azul que representa el único ciclo representativo de este modelo.

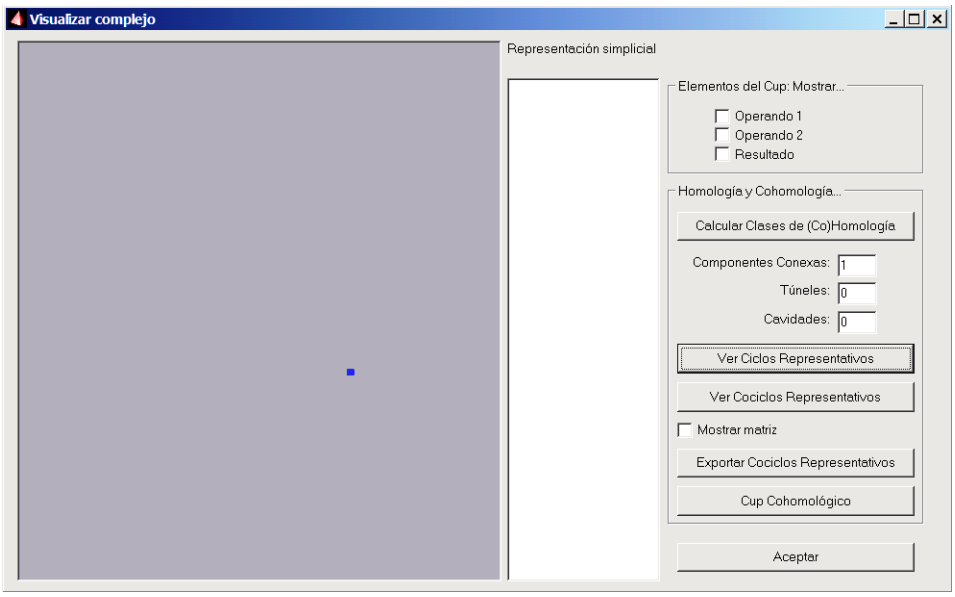


FIGURA 3.9

Con este par de ejemplos expuestos se puede comprender la utilidad de poder conocer si ciertos invariantes topológicos se preservan en el modelo que se obtiene del objeto real, pues la mayor preocupación es obtener un modelo que sea topológicamente correcto con el real.

2. Reconstrucción de un tenista

En la mayoría de los casos de modelado por el método de voxel-carving al utilizar una mayor cantidad de imágenes el objeto será más similar en todos los aspectos al objeto real, en el caso previo de la taza no sucedió esto debido a dificultades que ya fueron planteadas. No obstante, a continuación se presenta un ejemplo desarrollado en [**9**] en el cual se ilustrará este hecho.

A continuación se presenta la Figura 3.10, la cual está conformada por un par de reconstrucciones por la técnica de voxel carving de un tenista, en la primera a) se utiliza la información obtenida a partir de 4 cámaras, en la segunda b) se emplean 10 cámaras, además en ambos casos se superponen los ciclos representativos. Por otra parte, en la misma figura se muestra un diagrama en donde se compara la cantidad de componentes conexas (α_1) y túneles ($\gamma_1, \gamma_2, \gamma_3$) en función de la cantidad de cámaras empleadas para la reconstrucción (desde 1 a 50).

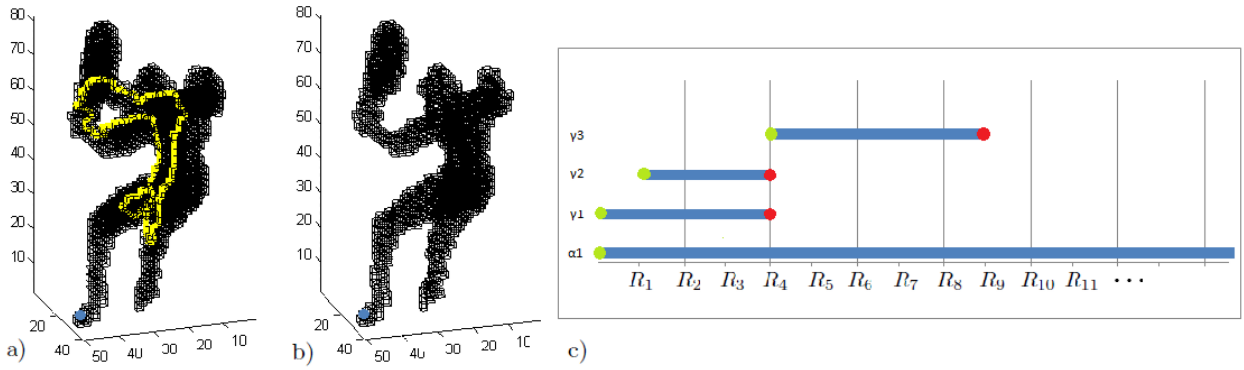


FIGURA 3.10

Nótese que cuando se realiza la reconstrucción haciendo uso de 1, 2 o 3 cámaras se obtiene 1 componente conexa (α_1) y 2 túneles (γ_1, γ_2); cuando se emplean de 4 a 9 cámaras se obtiene 1 componente conexa (α_1) y un túnel (γ_3); no obstante a partir de la utilización de 10 cámaras o más es cuando se obtiene un modelo que es topológicamente correcto con

el objeto sujeto a estudio, pues en estos casos se obtiene 1 componente conexa (resultado esperado).

Ahora considere el siguiente ejemplo que también fue desarrollado en [**9**].

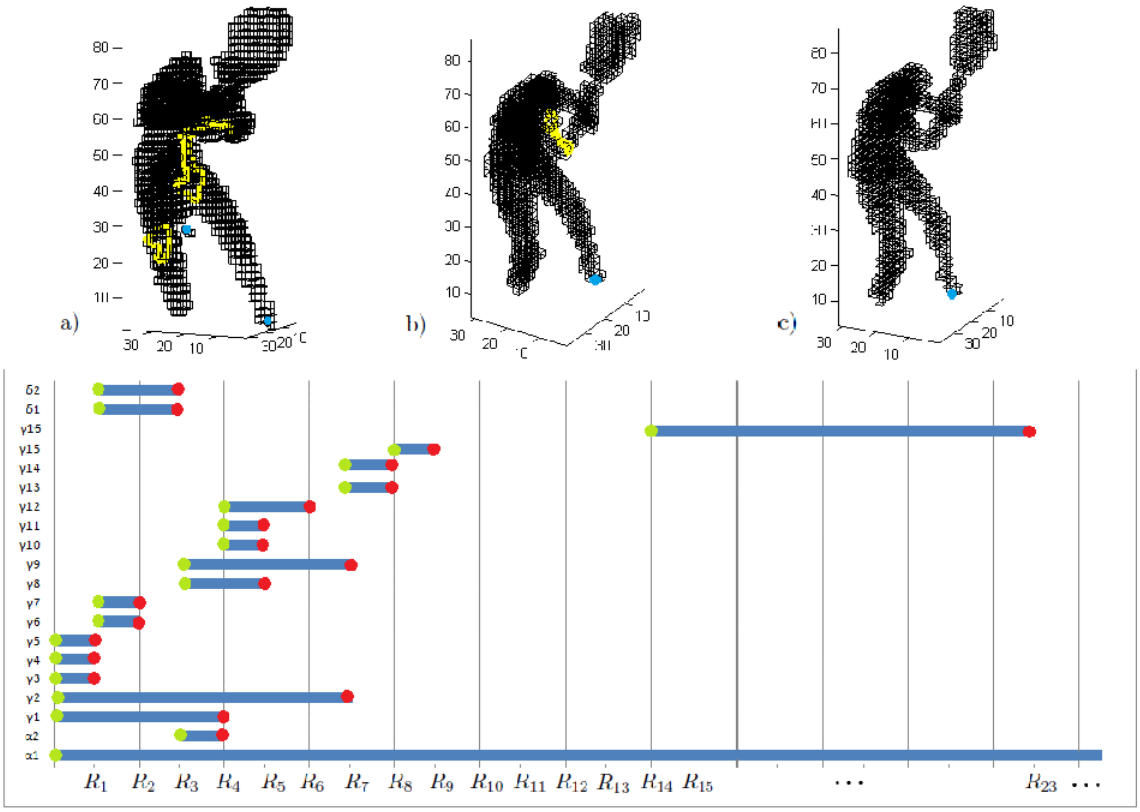


FIGURA 3.11

En la Figura 3.11 se pueden observar tres reconstrucciones diferentes de un tenista que sujeta la raqueta con ambas manos, por tal razón el objeto real presenta una componente conexa y un túnel, en la primera reconstrucción a) se emplean 4 cámaras, y de acuerdo al diagrama que también se incluye en esta figura, puede ser determinado que dicha reconstrucción tiene 2 componentes conexas (α_1, α_2) y 6 túneles $(\gamma_2, \gamma_8, \gamma_9, \gamma_{10}, \gamma_{11}, \gamma_{12})$; en la segunda reconstrucción b) se utiliza la información de 15 cámaras y la misma contiene 1 componente conexa (α_1) y un túnel (γ_{16}), nótse que ya estos datos concuerdan con los que presenta el objeto real; y por último, la reconstrucción c) obtenida de 24 cámaras tiene los mismos resultados que en el caso anterior. En este ejemplo, nuevamente se superponen los ciclos representativos y α representa componentes conexas, γ los túneles, y δ las cavidades.

Esto ejemplos traen como consecuencia que se genere la siguiente interrogante ¿cuál sería la menor cantidad de imágenes que necesitamos de un objeto para poder generar un modelo topológicamente correcto?, la respuesta a esta interrogante no es trivial, pues la misma depende de distintos factores como lo son: la forma del objeto, las posibles ubicaciones para las cámaras, la identificación de las siluetas, entre otros.

CAPITULO 4

Conclusiones y trabajo a futuro

La reconstrucción por voxel carving ha demostrado a través de los ejemplos desarrollados en este trabajo ser una técnica efectiva en la reconstrucción de objetos 3D, sin embargo la misma presenta algunos inconvenientes como lo son la incapacidad de determinar algunas cavidades en los objetos reconstruídos, por ejemplo el caso de la reconstrucción de la taza; no obstante por lo general este hecho no representa un problema para el cálculo de los invariantes topológicos, puesto que en su mayoría dichas cavidades no representan ningún tipo de agujero.

El mayor problema que presenta este método es que dependiendo de la cantidad de imágenes que se consideren del objeto, la posición de las mismas y la forma del objeto, puede suceder que la reconstrucción que se obtiene presente propiedades topológicas que el objeto sujeto a estudio no presenta. Un ejemplo de esto fue el que se trabajó acerca de las reconstrucciones de los tenistas.

En este trabajo se presentó una manera de evaluar las propiedades topológicas (componentes conexas, agujeros n-dimensionales) en la reconstrucción con la finalidad de determinar si la misma es topológicamente correcta con relación al objeto inicial; dicha manera se ve realizada en el estudio homológico de la reconstrucción haciendo uso del modelo AT, él cual en este trabajo se obtuvo gracias al software EditCup, demostrando éste, aun cuando se encuentra en fase beta, ser una herramienta útil para este objetivo pues en todos los casos en los cuales fue aplicado durante la realización de este trabajo obtuvo los resultados teóricos, permitiendo así el análisis comparativo entre las propiedades topológicas en función a la cantidad de cámaras utilizadas.

Por tanto se puede concluir a partir de los resultados obtenidos de este trabajo que se pueden estudiar las propiedades topológicas de los objetos reconstruídos por el método de voxel carving con la finalidad de verificar que en los mismos se preserven dichas propiedades en relación con el objeto sujeto a estudio.

Tomando en consideración este trabajo, particularmente las dificultades presentadas en él, algunos de los posibles trabajos que se podrían generar con el objetivo de mejorarlo pueden ser:

- Optimizar el algoritmo para la reconstrucción de objetos 3D por la técnica de voxel-carving.
- Desarrollar un nuevo software que permita realizar las funciones del software EditCup o en su defecto mejorar éste (se desconoce si los autores están trabajando en este aspecto).
- Automatizar el proceso de reconstrucción y cálculo de las clases de (co)homología, esto se podría desarrollar generando una función en MatLab que aplique el algoritmo de incremento-decrecimiento para el cálculo del modelo AT y así realizar todos los procesos en el mismo.

Apéndice

A1.

```
%% Carving a Dinosaur
% This is a demo of reconstructing a 3D shape from multiple images using
a
% simple space-carving approach. This technique is usually used when you
% need a 3D model of a small artefact which can be placed on a turntable,
% allowing dozens, even hundreds of images to be captured from around the
% object. It has been used pretty successfully by museums and the like to
% create online virtual galleries.
%
% _Note: This demo requires the
% <http://www.mathworks.com/access/helpdesk/help/toolbox/images |Image
Processing Toolbox|>._
%
% Author: Ben Tordoff
%
% Copyright 2005-2009 The MathWorks, Inc.

%% Introduction
% A little while ago (is it really four years?!) I was asked to prepare a
% demonstration for a customer visit. The customer had some samples that
% they wanted to photograph in order to estimate the volume occupied
before
% and after a chemical process. These samples were smooth but irregularly
% shaped such that a simple "volume of revolution" calculation was
% inaccurate. They wanted to know if accurate volume estimation from
images
% was possible, and if so how you might do it.
%
% The demo I produced is enumerated below and is the most basic form of a
% technique called "space carving" or "shape from silhouettes", where
each
% image is just used as a mask. A lump of voxel "clay" is placed in the
% middle of the scene and from each image we simply look and see what is
% outside the object silhouette. Anything outside is carved away.
% Obviously, this requires us to know where the camera was relative to
the
% object when the picture was taken, which is a whole separate problem.
%
% This technique has been refined over the last decade and can be
% done in some computationally and memory efficient ways. My approach is
% neither of these - I went for simplicity over efficiency since my only
% aim was to explain the technique and show it in MATLAB.
```

```
%
% *Acknowledgements*
%
% The dinosaur images used here were provided by Wolfgang Niem at
% the University of Hannover.
%
% The camera data used in this example was provided by
% <http://research.microsoft.com/en-us/um/people/awf |Dr A. W.
Fitzgibbon|>
% and <http://www.robots.ox.ac.uk/~az |Prof A. Zisserman|>
% from the <http://www.robots.ox.ac.uk |University of Oxford Robotics
Research Group|>.
%
% The images and camera data can both be downloaded from the
<http://www.robots.ox.ac.uk/~vgg/data/data-mview.html
% |Visual Geometry Group web-pages|> at the <http://www.robots.ox.ac.uk
% |University of Oxford Robotics Research Group|>.

%% Setup
% All functions for this demo are in the "spacecarving" package and the
% data in the "DinosaurData" folder.
import spacecarving.*;
datadir = fullfile( fileparts( mfilename( 'fullpath' ) ), 'DinosaurData'
);
close all;


%% Load the Camera and Image Data
% This reads the "Dinosaur" directory, loading the camera definitions
% (internal and external calibration) and image file for each camera.
These
% calibrations have previously been determined from the
% images using an automatic process that we won't worry about here.
cameras = loadcameradata( datadir )

montage( cat( 4, cameras.Image ) );
set( gcf(), 'Position', [100 100 600 600] )
axis off;

%% Convert the Images into Silhouettes
% The image in each camera is converted to a binary image using the
% blue-screen background and some morphological operators to clean up the
% edges. This becomes the "mask" referred to above. Holes in this mask
are
% particularly dangerous as they will cause voxels to be carved away that
% shouldn't be - we can end up drilling a hole through the object! The
% Image Processing Toolbox functions
%
<http://www.mathworks.com/access/helpdesk/help/toolbox/images/bwareaopen.
html |bwareaopen|>
% and
<http://www.mathworks.com/access/helpdesk/help/toolbox/images/imclose.htm
l |imclose|>
% are your friends for this job!
for c=1:numel(cameras)
    cameras(c).Silhouette = getsilhouette( cameras(c).Image );
```

```
end

figure('Position',[100 100 600 300]);

subplot(1,2,1)
imshow( cameras(c).Image );
title( 'Original Image' )
axis off

subplot(1,2,2)
imshow( cameras(c).Silhouette );
title( 'Silhouette' )
axis off

makeFullAxes( gcf );

%% Work out the space occupied by the scene
% Initially we have no idea where to look for the model. We will assume
% that the model lies in the space spanned by the cameras and their
% principal view directions. We then perform a very low-res space-carve
% using all the cameras to narrow down exactly where the object is. This
% isn't foolproof, but good enough for this demo.
[xlim,ylim,zlim] = findmodel( cameras );


%% Create a Voxel Array
% This creates a regular 3D grid of elements ready for carving away. The
% input arguments set the bounding box and the approximate number of
voxels
% to create. Since the voxels must be cubes, the actual number generated
% may be a little more or less. We'll start with about six million voxels
% (you may need to reduce this if you don't have enough memory).
%
% For "real world" implementations of space carving you certainly
wouldn't
% create a uniform 3D matrix like this. OctTrees and other refinement
% representations give much better efficiency, both in memory and
% computational time.
voxels = makevoxels( xlim, ylim, zlim, 6000000 );
starting_volume = numel( voxels.XData );

% Show the whole scene
figure('Position',[100 100 600 400]);
showscene( cameras, voxels );

%% Carve the Voxels Using the First Camera Image
% The silhouette is projected onto the voxel array.
% Any voxels that lie outside the silhouette are carved away, leaving
only
% points inside the model. Using just one camera, we end up with a
% dinosaur-prism - a single camera provides no information on depth.
voxels = carve( voxels, cameras(1) );

% Show Result
figure('Position',[100 100 600 300]);
```

```
subplot(1,2,1)
showscene( cameras(1), voxels );
title( '1 camera' )
subplot(1,2,2)
showsurface( voxels )
title( 'Result after 1 carving' )

%% Add More Views
% Adding more views refines the shape. If we include two more, we already
% have something recognisable, albeit a bit "boxy".
voxels = carve( voxels, cameras(4) );
voxels = carve( voxels, cameras(7) );

% Show Result
figure('Position',[100 100 600 300]);
subplot(1,2,1)
title( '3 cameras' )
showscene( cameras([1 4 7]), voxels );
subplot(1,2,2)
showsurface(voxels)
title( 'Result after 3 carvings' )


%% Now Include All the Views
% In this case we have 36 views (roughly every 10 degrees). For a very
% detailed model and if you have an automatic capture rig you would use
far
% more - the only limit being time and disk-space. When using a computer
% controlled turn-table (as is done in museums) storage is the only real
% limitation.
for ii=1:numel(cameras)
    voxels = carve( voxels, cameras(ii) );
end
figure('Position',[100 100 600 700]);
showsurface(voxels)
set(gca,'Position',[-0.2 0 1.4 0.95])
axis off
title( 'Result after 36 carvings' )

final_volume = numel( voxels.XData );
fprintf( 'Final volume is %d (%1.2f%%)\n', ...
    final_volume, 100 * final_volume / starting_volume )

%% Get real values
% We ideally want much higher resolution, but would run out of memory.
% Instead we can use a trick and assign real values to each voxel instead
% of a binary value. We do this by moving all voxels a third of a square
in
% each direction then seeing if they get carved off. The ratio of carved
to
% non-carved for each voxel gives its score (which is roughly equivalent
to
% estimating how much of the voxel is inside).
offset_vec = 1/3 * voxels.Resolution * [-1 0 1];
[off_x, off_y, off_z] = meshgrid( offset_vec, offset_vec, offset_vec );
```

```
num_voxels = numel( voxels.Value );
num_offsets = numel( off_x );
scores = zeros( num_voxels, 1 );
for jj=1:num_offsets
    keep = true( num_voxels, 1 );
    myvoxels = voxels;
    myvoxels.XData = voxels.XData + off_x(jj);
    myvoxels.YData = voxels.YData + off_y(jj);
    myvoxels.ZData = voxels.ZData + off_z(jj);
    for ii=1:numel( cameras )
        [~,mykeep] = carve( myvoxels, cameras(ii) );
        keep(setdiff( 1:num_voxels, mykeep )) = false;
    end
    scores(keep) = scores(keep) + 1;
end
voxels.Value = scores / num_offsets;
figure('Position',[100 100 600 700]);
showsurface( voxels );
set(gca,'Position',[-0.2 0 1.4 0.95])
axis off
title( 'Result after 36 carvings with refinement' )


%% Final Result
% For online galleries and the like we would colour each voxel from the
% image with the best view (i.e. nearest normal vector), leading to a
% colour 3D model. This makes zero difference to the volume estimate
(which
% was the main purpose of the demo), but does look pretty!
figure('Position',[100 100 600 700]);
ptch = showsurface( voxels );
colorsurface( ptch, cameras );
set(gca,'Position',[-0.2 0 1.4 0.95])
axis off
title( 'Result after 36 carvings with refinement and colour' )


%% Conclusion
% Hopefully this demo has given you a taste for what is possible by
simple
% image masking and space-carving. If this has whetted your appetite,
have
% a look at the references below. Converting each image to a binary mask
% throws away a lot of information. Instead of using these silhouettes,
we
% could use the image values (either greyscale or colour) and a
% photo-consistency constraint. This is *much* harder to get right, but
% copes much better with concavities and holes in the model.
%
% Have you ever been asked about volume estimation from images? Do you
% fancy trying this at home? Perhaps you've implemented a better way to
do
% this? I'd love to hear from you
% <http://blogs.mathworks.com/loren/?p=210#respond here>.
%
%
```

```
%% File Availability
% The functions created for this demo will ultimately appear on the File
Exchange.
% Once they are available, we'll update this post.  As a reminder,
there's a link
% above for getting the data.
%
%% References
% Some good references for this (including the original paper that used
these images) are:
%
% * *Automatic 3D model construction for turn-table sequences*, _A. W.
Fitzgibbon, G. Cross, and A. Zisserman,
% In 3D Structure from Multiple Images of Large-Scale Environments,
Springer LNCS 1506, pages 155--170, 1998_
% * *A Theory of Shape by Space Carving*, _K. N. Kutulakos & S. M. Seitz,
% International Journal of Computer Vision 38(3), 199-218, 2000_
% * *Foundations of Image Understanding*, Chapter 16, _edited by L. S.
Davis,
% Kluwer, 2001_
```

Bibliografía

[1] Balan A. O, Voxel Carving and Coloring - Constructing a 3D Model of an Object from 2D Images. 2003.

[2] Berrio R., González-Díaz R.,Leal F., Real P, Visualizing Cohomology Aspects of 3D Objects. Proc. of the 6th Asian Technology Conf. in Math. pp 459-468. 2001.

[3] Berrio R., González-Díaz R.,Leal F., Real P, A graphical Tool for Understanding Cohomology. Actas del 8Â° Encuentro de Álgebra Computacional y Aplicaciones, EACA-2002. pp 103-106. 2002.

[4] Buchsbaum. D. A Exact Categories and Duality. *Transactions of the American Mathematical Society*, Volume 80, Issue 1, pp 1-34. 1955.

[5] González-Díaz R., Ion A., Jiménez M., Poyatos R, Incremental-Decremental Algorithm for Computing AT-models and Persistent Homology. CAIP 2011. LNCS vol. 6854, pp 286293. Springer, Heidelberg. 2011.

[6] González-Díaz R., Leal F., Real P., Programas informáticos en matemáticas. La Gaceta de la RSME, Vol 7.2, pp 553-566. 2004.

[7] González-Díaz R., Medrano B., Real P., Sanchez-Pelaez A, Simplicial perturbation techniques and effective homology. CASC 2006, LNCS 4194, 166-177. 2006.

[8] González-Díaz R., Real P, On the cohomology of 3D digital images. Discrete Applied Math 147 (2-3), 245-263. 2005.

[9] Gutierrez A., Monaghan D,. Jiménez M,. O'Connor N, Persistent Homology for 3D Reconstruction Evaluation. CTIC 2012 4th International Workshop on Computational Topology in Image Context, Bertinoro, Italy. May 2012.

[10] Mitchell B, Theory of Categories. 1965.

[11] Munkres J, Elements of Algebraic Topology. Addison-Wesley Co. 1984.

[12] Quintero B. E, Homología cúbica: Algoritmos para el cálculo de la aplicación inducida por una función continua. Universidad Carlos III de Madrid. Proyecto fin de carrera. 2009.

[13] Weibel C, An introduction to Homological Algebra. Cambridge University Press. 1995.