VNIVERSIDAD NACIONAL
AVTNOMA DE
MEXICO

UNIVERSIDAD NACIONAL AUTÓNOMA DE MÉXICO

POSGRADO EN CIENCIA E INGENIERÍA DE LA COMPUTACIÓN

"VISUALIZACIÓN DE VOLÚMENES USANDO SUPERFICIES RASTREADAS CON NORMALES SUAVES OBTENIDAS POR ART "

T E S I S

QUE PARA OBTENER EL GRADO DE:

MAESTRA EN CIENCIAS (COMPUTACIÓN)

P R E S E N T A:

FÁTIMA VICENTA QUINTAL FLORES

DIRECTOR DE TESIS:

DR. EDGAR GARGUÑO ÁNGELES

México, D.F. **2011.**

Para mi familia y mis queridos GoMaPe, por todos los años compartidos

Agradecimientos

El presente trabajo representa el esfuerzo de varias personas y organizaciones que directa o indirectamente colaboraron en su desarrollo. Agradezco al Dr. Edgar Garduño por haberme confiado este trabajo y por la dirección del mismo, por su paciencia al leer, opinar y corregir este trabajo pero sobre todo por los conocimientos que me ha compartido. Agradezco también al Dr. Jorge Márquez, Dr. Ernesto Bribiesca, Dr. Frnándo Arámbula y al Dr. Jorge Urrutia, los sinodales que me apoyaron en todo momento y aportaron valiosos comentarios y correcciones en este trabajo.

Gracias al Posgrado en Ciencia e Ingeniería de la Computación, al IIMAS y al CONACYT por el apoyo que recibí y me permitió cursar la maestría. También gracias al DGAPA proyecto IN101108 y al Departamento de Ciencias de la Computación por el equipo y las instalaciones que fueron fundamentales para el desarrollo de este trabajo. También agradezco a Lulú, Diana, Montse, Rosy y Liz por ayudarme con trámites, equipo y demás molestias que les ocasioné.

Agradezco muy especialmente a toda mi familia por su apoyo y comprensión, en especial a mis padres y mi abuelita por hacerse cargo de mis responsabilidades mientras estoy lejos de casa. Gracias a Dios por cuidarlos. Gracias también a mis amigos paty y rodrigo por su compañía y apoyo incondicional, a ireri por formar parte de nuestro equipo y a jorge por compartirme sus conocimientos de graficación y comics, pero sobre todo por su amistad. Por último pero no menos importante gracias a mis amigos de Mérida por acordarse y estar pendientes de mí.

Resumen

En la actualidad existen varias tecnologías que producen imágenes 3D conocidas también como volúmenes. En este trabajo nos interesan aquellas imágenes 3D producidas a partir de proyecciones por medio de algoritmos basados en expansión de series. Estos métodos asumen que la imagen 3D es una función que se aproxima por medio de una combinación lineal de funciones base. Cuando las funciones base son "suaves" y continuas, los métodos producen una función continua, que posteriormente es discretizada para producir la imagen 3D. A partir la imagen 3D se pueden obtener mallas que representan superficies implícitas. Estas mallas son desplegadas usando técnicas de graficación por computadora, en particular de visualización por superficie y utilizan las normales para generar imágenes 2D que preservan las pistas de profundidad de la malla. Estas técnicas requieren de un algoritmo para determinar precisamente la superficie de los objetos. Uno de los métodos más populares es el llamado *Marching Cubes* [34]. Otro método es el algoritmo de rastreo de superficies propuesto por Artzy et al [2], en rejillas cúbicas simples. Algunas de las ventajas del Algoritmo de Artzy frente al de *Marching Cubes* es que no recorre todos los vóxeles y que produce superficies que son el equivalente discreto de una superficie de Jordan. Sin embargo, produce mallas compuestas por rectángulos, lo cual resulta en una apariencia "cuboide".

En esta tesis proponemos usar el hecho de que para imágenes 3D producidas por medio de una combinación de funciones base suaves se pueden tener las normales de dicha aproximación suave e incorporarlas a la malla producida por el Algoritmo de Artzy para generar una representación visualmente más suave.

Índice general

Índice de figuras

Capítulo 1

Introducción

Se considera que el sentido más desarrollado de los humanos es la vista, por lo que la visualización es la manera más fácil de comunicar información, especialmente cuando es compleja o es dada en grandes cantidades. Esta es una de las razones por las cuales la graficación por computadora se ha convertido en el medio que permite aprovechar la abstracción matemática y modelado para comunicar información visualmente [23]. La visualización científica es una rama interdisciplinaria de la ciencia, que permite la transformación de datos científicos y abstractos en imágenes. Un proceso típico en visualización trata de encontrar, para una serie de datos, una representación visual adecuada que permita analizarlos o evaluarlos [11].

Usualmente, cuando se obtienen los datos, éstos incluyen varios parámetros, que en ocasiones poseen un alto grado de dimensionalidad. El almacenarlos en sistemas de administración de bases de datos puede ocasionar que sólo sea posible verlos por partes o pequeñas porciones, lo que puede ser un inconveniente a la hora de analizarlos pues podría ser complicado a simple vista distinguir relaciones entre ellos. A diferencia de estos sistemas, la visualizaciónfacilita el proceso de análisis y comprensión de los datos (algo relacionado a la geometría) a una mayor escala; lo que permite reconocer patrones de comportamiento o sus relaciones (algo relacionado a la topología) [23]. Por estos motivos se han desarrollado técnicas para transformar la información en imágenes, las cuales son la base de la visualización científica.En este trabajo nos interesan ciertos tipos de visualización científica que permiten obtener representaciones visuales de un tipo específico de funciones conocidas como funciones

de densidad. Existenherramientas computacionales que generan discretizaciones de este tipo de funciones por lo que es posible transformar en imágenes datos provenientes de aparatos de medición como telescopios, microscopios, satélites y, en el caso de la biomedicina, de aparatos que usan técnicas de imaginología médica tales como tomografías (CT), resonancia magnética (MRI), radiografía, ecografía, o microscopía electrónica tomográfica.

Los aparatos de medición o adquisición de datos muestrean la función de densidad en tres dimensiones y, como se verá mas adelante, en la práctica producen un conjunto de vóxeles que representan una discretización de la función de densidad, la cual se puede visualizar por medio de varias técnicas de graficación por computadora. Estas técnicas consisten en generar una imagen bidimensional a partir de proyectar un modelo tridimensional creado por medio de programas de cómputo [43]. Existen dos técnicas principales que permiten visualizar las funciones de densidad: la visualización directa de volúmenes (*volume rendering*) y la visualización por superficies (*surface rendering*). La primera, como su nombre lo indica, permite visualizar directamente o proyectar toda la función discretizada, mientras que la segunda; como se verá más adelante, visualiza o proyecta, solamente la frontera del objeto de interés.

Existen varios métodos para realizar visualización por superficie, los cuales proporcionan diferentes niveles de detalle. Algunos suelen producir resultados más reales pero también suelen ser más lentos debido a que tanto el *hardware* como el *software* gráfico han sido optimizados para producir, graficar, proyectar e iluminar polígonos. Por esa razón las técnicas de visualización por superficie usando proyección de polígonos son ampliamente usadas en visualización científica y son las que interesan en este trabajo. Existen varios algoritmos que permiten realizar visualización por superficie y difieren básicamente en la forma en la que determinan los límites o frontera del objeto de interés. Uno de los más populares es el llamado *Marching Cubes*, publicado originalmente en 1987 por Lorensen y Cline [34]. Otro algoritmo para obtener la superficie es el propuesto originalmente por Artzy, Frieder y Herman [2] (nombrado a lo largo de ese trabajo como Algoritmo de Artzy et al) pero que a diferencia del *Marching Cubes* produce superficies visiblemente menos suaves; sin embargo, posee

propiedades matemáticas más adecuadas como se verá más adelante.

Una vez generada la malla que representa la superficie del objeto ésta se puede visualizar usando técnicas estándares de graficación por computadora. Como explicaremos más adelante, las normales a esta superficie juegan un papel muy importante en la generación de la imagen final. Una forma de alterar la apariencia final de la representación de los objetos consiste en modificar "apropiadamente" las normales a las mallas [15].

En este trabajo proponemos un método que permite producir representaciones más suaves de las mallas generadas por el Algoritmo de Artzy por medio de la incorporación de normales "suaves". Estas normales son obtenidas del modelo "suave" utilizado por algunas técnicas que aproximan la función de densidad original.

Existen varios métodos que permiten obtener una estimación de la función de densidad cuando ésta no se puede medir directamente. Estos métodos reconstruyen (o estiman) la función de densidad a través de sus proyecciones (estas técnicas son usadas en microscopía, astronomía e imaginología médica).

Existe una familia de algoritmos de reconstrucción que aproximan la función de densidad por medio de una combinación lineal de funciones base. Cuando estas funciones son suaves, el resultado es una función de densidad suave. Posteriormente para su manipulación y almacenamiento en computadoras, esta aproximación es discretizada.

Debido a que se conoce el método de reconstrucción utilizado, se posee entonces, información de la función de densidad contínua, que combinada con técnicas de graficación tradicionales deben permitir obtener una apariencia suave de la superficie a pesar de usar la malla producida por el Algoritmo de Artzy et al.

La organización de este trabajo es la siguiente. En el Capítulo 2 se describe la técnica de reconstrucción que se usa en este trabajo para obtener el modelo suave. En el Capítulo 3, se abordan tanto los algoritmos *Marching Cubes* y el de Artzy, como las técnicas de graficación por computadora que permitirán obtener una visualización suave de la superficie. El Capítulo 4 contiene la metodología propuesta para suavizar la superficie obtenida con el algortimo de Artzy y los resultados obtenidos, así como las conclusiones y propuestas para trabajos futuros.

Capítulo 2

Reconstrucción a partir de Proyecciones

La imaginología biomédica es el campo que se ocupa del desarrollo y uso de dispositivos de imágenes, y técnicas para obtener imágenes anatómicas internas [4]. Algunas técnicas como la tomografía computarizada (CT), resonancia magnética (MRI), radiografía, ecografía (también llamada ultrasonido), imaginología molecular y microscopía electrónica desempeñan un papel cada vez más importante en la práctica clínica y la investigación básica de ciencias de la vida; en parte porque permiten visualizar y analizar la estructura de objetos que pueden ser muy complicados o difíciles de observar de otra forma y en muchos casos con la ventaja de no alterar, al menos no de una manera drástica, su naturaleza.

El esquema general de un proceso de imaginología biomédica en 3D se muestra en la Figura 2.1. En cada paso se pueden emplear diferentes técnicas y la elección e implementación de cada una depende de la aplicación biomédica para la cual se está empleando. Para el desarrollo de este trabajo nos interesa en particular la visualización así como el método empleado para la reconstrucción. En este capítulo se describe brevemente cómo se obtienen los datos que se usan en la reconstrucción. Incluye también una descripción básica del algoritmo que permite obtener el muestreo de una función tridimensional a partir de sus proyecciones; la cual, será empleada en el proceso de obtención de superficies visualmente suaves.

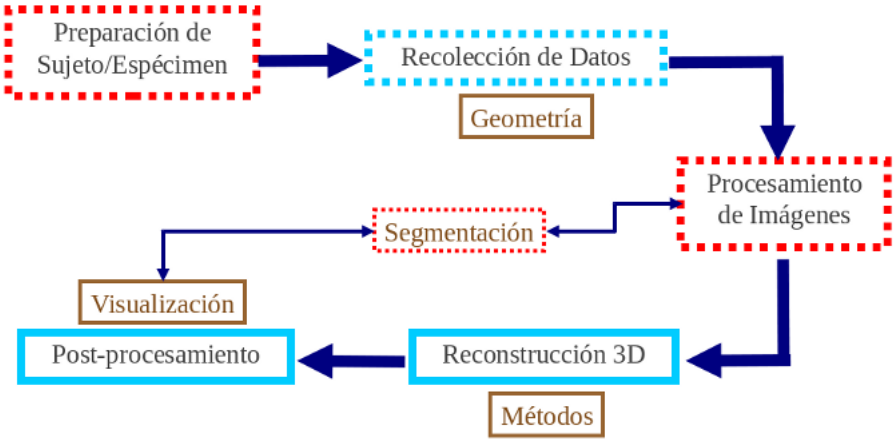


Figura 2.1: Esquema general de los procesos involucrados en imaginología biomédica 3D.

2.1. Adquisición de Datos

Los aparatos de medición utilizados en imaginología biomédica funcionan capturando, por medio de detectores, la energía emitida por alguna fuente después de que ésta ha interactuado con el objeto biológico (o espécimen). En general, algunos instrumentos utilizan radiación electromagnética cuya energía es proporcional a su frecuencia; entre mayor su frecuencia, mayor su energía. El diseño de los instrumentos se puede categorizar en tres esquemas básicos: transmisión, reflexión y emisión.

Algunas técnicas como la ecografía que emplea ondas de presión, son ejemplo de esquemas de reflexión ya que en su modalidad de operación principal se emiten ondas en el intervalo ultrasónico en dirección del espécimen a estudiar y, pasado un lapso corto de tiempo, un arreglo de detectores recibe la energía reflejada por los diferentes tejidos del cuerpo; la energía regresada está relacionada a las diferentes interfaces entre tejidos de distintas densidades.

Los equipos que hacen uso de radioisótopos representan ejemplos típicos de los esquemas de emisión. En estos equipos se introduce un radioisótopo ligado a un compuesto específico para cierta función biológica. Una vez que el compuesto es ingerido, los fotones de alta energía y las partículas subatómicas producidas por el proceso de decaimiento nuclear interactúan con la materia de los tejidos. Dicha interacción es

detectada por sensores alrededor del cuerpo. Otro ejemplo de este esquema es el MRI (imagen por resonancia magnética) usando radiación electromagnética (EM).

Finalmente, instrumentos como los radiógrafos, los tomógrafos computarizados (también conocidos como TAC) y los fluoroscopios, son ejemplos clásicos del esquema de transmsión. En este esquema el objeto a examinar se coloca entre una fuente de energía (por ejemplo, un emisor de rayos x) y un arreglo de detectores. La energía viaja en dirección del objeto e interactúa con el tejido biológico. Los detectores capturan el resultado acumulado de dicha interacción.

La radiación utilizada por varios de estos instrumentos posee un gran contenido energético y por ello, las partículas o fotones que son parte de esta radiación tienden a viajar en trayectorias rectilíneas.

2.2. Proyecciones

Por todo lo anterior, los instrumentos detectan la acumulación de la radiación con la materia a lo largo de las trayectorias rectilíneas. De manera más formal, este proceso se puede expresar como:

$$[\mathcal{P}\nu]\left(\vec{o},\bar{x}\right) = \int_{\mathbb{R}^1} \nu\left(\bar{x}+\tau\vec{o}\right) d\tau, \tag{2.1}$$

donde ν representa la función de densidad, $\bar{x}$ es un punto en el espacio que se representa por medio de un vector y $\vec{o}$ es un vector de dirección (vector normalizado). Entonces, el operador $\mathcal{P}$ representa la integral de línea recta sobre la trayectoria definida por la recta $\bar{x}+\tau\vec{o}$, con $\tau \in \mathbb{R}$. La transformada (2.1) es mejor conocida como la *Transformada del Rayo* [30].

En este trabajo también representaremos a los vectores $\bar{x} \in \mathbb{R}^n$, alternativamente $\bar{k} \in \mathbb{Z}^n$, por medio de una tupla de n números reales (o n-tupla) representada por $(x_1, x_2, ..., x_n)$, alternativamente $(k_1, k_2, ..., k_n)$.

A un conjunto de integrales de línea se le conoce como *proyección*. Una notación alternativa para expresar una proyección es por medio de la *Transformada de Radón*

[42], definida como

$$[\mathcal{R}\nu]\,(\vec{o}, s) = \int_{\mathbb{R}^n} \nu\,(\bar{x})\,\delta\,(s - \langle \bar{x},\, \vec{o} \rangle)\,d\bar{x}, \tag{2.2}$$

donde $\langle \bar{x},\, \bar{y} \rangle$ representa el producto interno entre los vectores $\bar{x}$ e $\bar{y}$. Claramente el operador de Radón representa la integral sobre el hiperplano con orientación definida por $\vec{o}$ y a distancia s del origen.

En la práctica, los detectores son un arreglo de n detectores individuales. Por lo tanto, un detector y_i, para $1 \leq i \leq n$, captura

$$y_i = \int_A [\mathcal{P}\nu]\,(\vec{o},\, \bar{x})\,dA, \tag{2.3}$$

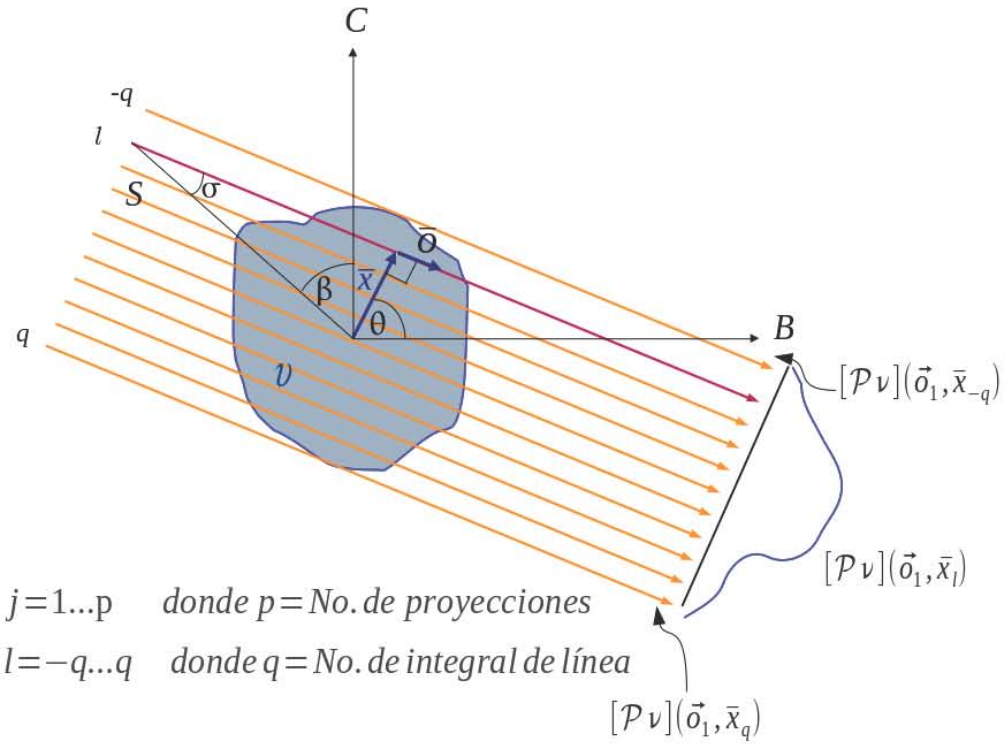
donde $A \subset \mathbb{R}^2$. En el caso más simple se asume que $y_i = [\mathcal{P}\nu]\,(\vec{o}_i,\, \bar{x}_i)$.

Existen varios esquemas básicos de proyección; en la Figura 2.2 se muestran la proyección en paralelo y en cono.

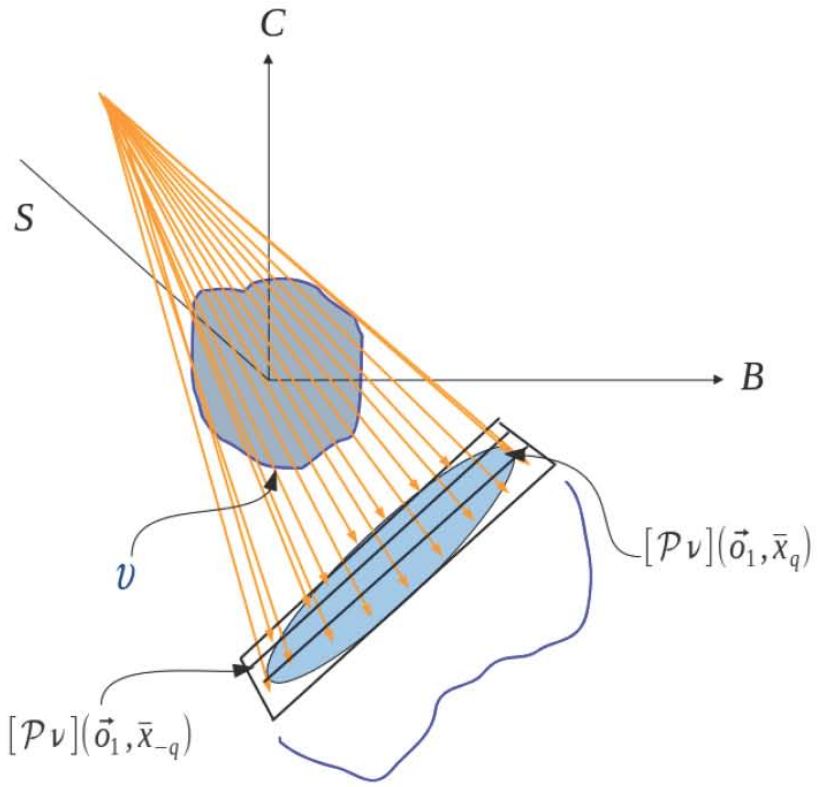
2.3. Métodos de reconstrucción

En las modalidades cuyos detectores miden proyecciones, generalmente interesa recuperar la distribución espacial de la función de densidad. Por lo que se necesita resolver el problema inverso, un proceso que se conoce como *reconstrucción a partir de proyecciones* (por simpleza, de aquí en adelante nos referiremos a él como reconstrucción) [27].

Para resolver el problema inverso existen dos familias básicas: métodos basados en transformadas y métodos basados en expansión de series. Los métodos basados en transformadas pueden usar: a) la relación que existe entre la transformada de Rayo (o Radón) y la transformada de Fourier, conocida matemáticamente como Teorema del Plano Central [10], b) la transformada inversa de la transformada de Radón, o c) la relación que existe entre la transformada de Rayo (o Radón) y el operador de retroproyección (conocido como método de retroproyección [25]).



(a)



(b)

Figura 2.2: Dos tipos de proyecciones. (a) proyección en paralelo y (b) proyección en cono.

Por otra parte, los métodos basados en expansión de series asumen que cualquier función de densidad puede aproximarse por medio de la combinación lineal de funciones base como sigue

$$\nu\left(\bar{x}\right) \approx \sum_{j=1}^{J} c_j b_j\left(\bar{x} - \bar{p_j}\right), \tag{2.4}$$

donde $\bar{x}$ es un vector de posición; es decir, un punto en el espacio, $\{b_j\}$ es el conjunto de funciones base, cada una ponderada por el correspondiente coeficiente c_j. El conjunto $\{\bar{p_j}\}$ representa los puntos donde se encuentran los orígenes de las funciones base $\{b_j\}$. En esta aproximación, las funciones base son conocidas de forma anticipada, mientras que los valores de los coeficientes c_j son determinados por el algoritmo de reconstrucción. En este trabajo estamos interesados en este tipo de algoritmos, en particular en los métodos conocidos como Técnicas de Reconstrucción Algebraica (ART por sus siglas en inglés) [27].

2.4. Técnicas de Reconstrucción Algebraica (ART)

Sustituyendo (2.4) en (2.1) se tiene

$$\left[\mathcal{P}\nu\right]\left(\vec{o},\, \bar{x}\right) \approx \int_{\mathbb{R}} \sum_{j=1}^{J} c_j b_j\left(\bar{x} + \tau\vec{o} - \bar{p_j}\right) d\tau \approx \sum_{j=1}^{J} c_j \int_{\mathbb{R}} b_j\left(\bar{x} + \tau\vec{o} - p_j\right) d\tau,$$

que resulta en la aproximación

$$\left[\mathcal{P}\nu\right]\left(\vec{o},\, \bar{x}\right) \approx \sum_{j=1}^{J} c_j \left[\mathcal{P}b_j\right]\left(\vec{o},\, \bar{x}\right). \tag{2.5}$$

Es decir, dado que se puede definir una proyección para todo $(\vec{o},\, \bar{x})$, entonces la ecuación (2.5) indica que la transformada de rayo de la función de densidad $\nu\left(\bar{x}\right)$ es similar a la suma de las transformadas de rayo de las funciones $\{b_j\}$ sopesadas por los coeficientes $\{c_j\}$.

Si suponemos que tenemos en la práctica una forma de medir M líneas (las cuales

se caracterizan por medio de $(\vec{o}_m,\, \bar{x}_m)$, para $1 \leq m \leq M$), entonces tenemos:

$$y_m \approx \sum_{j=1}^{J} c_j \left[\mathcal{P} b_j\right](\vec{o}_m,\, \bar{x}_m) \approx \sum_{j=1}^{J} c_j \ell_{mj}, \tag{2.6}$$

donde y_m es la m-ésima medición y ℓ_{mj} representa la integral de línea de la función base j en la dirección de la línea m, es decir, representa la intersección de una función base y una línea en particular. Entonces para cada línea M se tiene

$$\begin{aligned} y_1 &= c_1\ell_{11} + c_2\ell_{12} + ... + c_J\ell_{1J} \\ y_2 &= c_1\ell_{21} + c_2\ell_{22} + ... + c_J\ell_{2J} \\ &... \\ &... \\ y_M &= c_1\ell_{M1} + c_2\ell_{M2} + ... + c_J\ell_{MJ}. \end{aligned}$$

Claramente se puede expresar el problema por medio de un sistema de ecuaciones lineales $\bar{y} = \mathbf{L}\bar{c}$.

En este momento hacemos una pausa para ilustrar esta aproximación en una función de densidad bidimensional. Para este ejemplo usaremos la Figura 2.3 como referencia. Por simplicidad usaremos píxeles cuadrados como funciones base, definidos como

$$b_j(\bar{x}) = \begin{cases} 1, & si\ -\frac{1}{2} \leq |\bar{x}| < \frac{1}{2}, \\ 0, & en\ otro\ caso, \end{cases} \tag{2.7}$$

donde $\bar{x} \in \mathbb{R}^2$.

Para este ejemplo usamos 100 funciones base ($J = 100$) para representar la función. Cada medición y_m se puede expresar como $y_m = c_1\ell_{m1} + c_2\ell_{m2} + \ldots + c_N\ell_{mJ}$; en particular para la medición y_{m+1} de la Figura 2.3), la suma queda como $y_{m+1} = c_{51}\ell_{m+1,51} + c_{52}\ell_{m+1,52} + c_{61}\ell_{m+1,61} + c_{62}\ell_{m+1,62} + c_{63}\ell_{m+1,63} + c_{73}\ell_{m+1,73} + c_{74}\ell_{m+1,74} + c_{75}\ell_{m+1,75} + c_{84}\ell_{m+1,84} + c_{85}\ell_{m+1,85} + c_{86}\ell_{m+1,86} + c_{96}\ell_{m+1,96} + c_{97}\ell_{m+1,97} + c_{98}\ell_{m+1,98}$. Como se puede observar en la Figura 2.3 los elementos c_j para y_{m+1} corresponden a las celdas verdes. Cada una de las mediciones $\ell_{m,j}$ se puede calcular fácilmente

ya que es la intersección del píxel j y la línea (o rayo) $(\vec{o}_m, \bar{x}_m)$. Dichos segmentos de línea pueden ser precalculados rápidamente. Podemos notar que esta explicación puede extenderse sin perder generalidad al caso de $\mathbb{R}^3$ y vóxeles cúbicos.

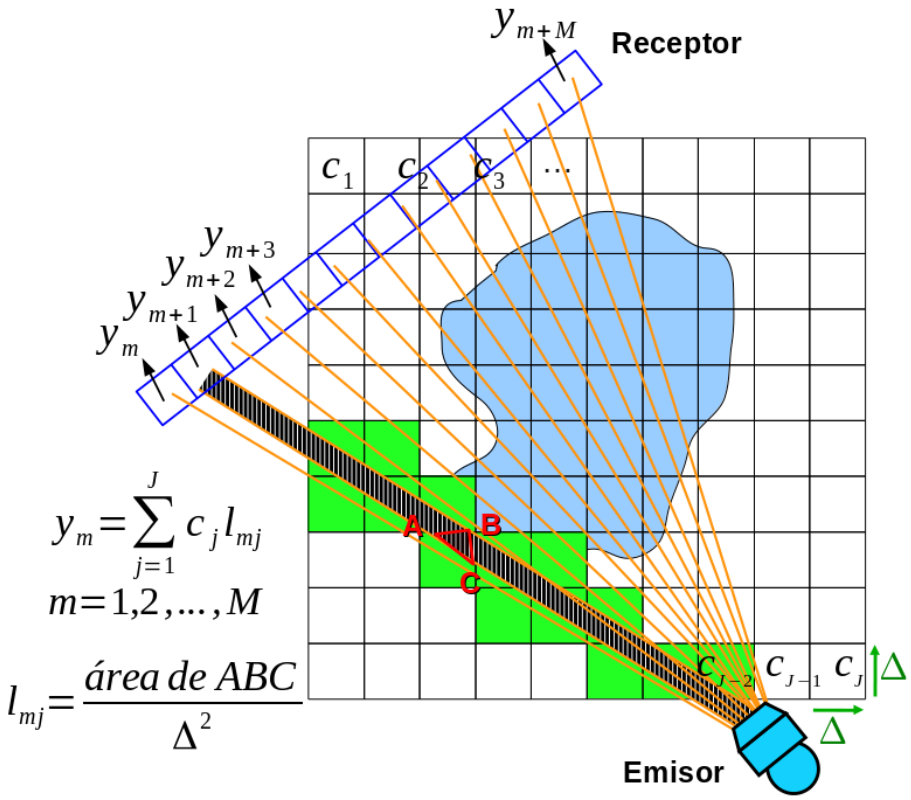


Figura 2.3: Esquema que muestra el proceso de ART para una función en 2D y píxeles.

En la práctica el sistema $\bar{y} = \mathbf{L}\bar{c}$ está sobredeterminado debido a la necesidad de obtener muchas mediciones para compensar el ruido (aunque recientemente se estudian métodos donde el sistema está subdeterminado [13, 17]). Además, si los valores de M y J fueran pequeños entonces se podría utilizar cualquier método de inversión de matrices para resolver $\bar{y} = \mathbf{L}\bar{c}$; sin embargo, en la práctica estos valores suelen ser grandes, por ejemplo, en una rejilla de tamaño 256×256 (considerado moderado), el valor de J es igual a 65,536. También en la práctica las líneas m son vistas como delgadas bandas de ancho σ que en muchas ocasiones son aproximadamente iguales al ancho de los píxeles, así considerando que el ancho de las líneas debe ser aproximadamente el ancho de los píxeles, se requieren entonces J rayos, es decir, $M = J$.

Así, se tienen $M \times J$ ecuaciones, o sea, $65,536 \times 65,536 \approx 4.2 \times 10^9$ y más aún, si se requieren cálculos de doble precisión cada elemento de la matriz ocuparía 8 bytes por lo que una sola matriz requerirá de 8×4.2 gigabytes de almacenamiento [3].

Entonces, debido a lo anterior y principalmente a que en el sistema $\bar{y} = \mathbf{L}\bar{c}$ la matriz $\mathbf{L}$ es dispersa (la mayor parte de elementos son cero), este sistema se puede resolver por medio de la generalización del método de Kaczmarz [31], el cual es un método iterativo definido como sigue:

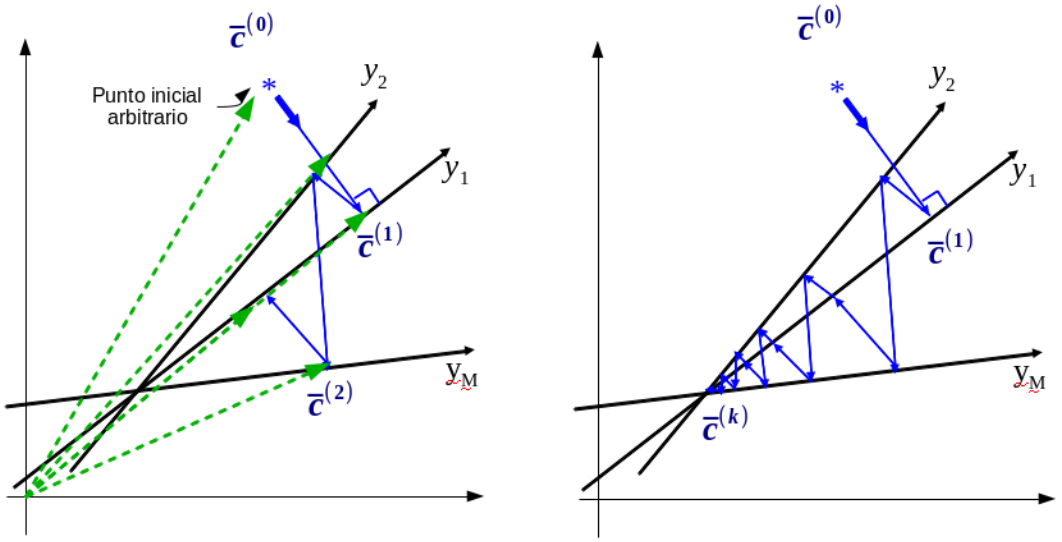
$$\bar{c_j}^{(k+1)} = \bar{c}_j^{(k)} + \lambda^{(k)} \frac{y_m - \sum_{i=1}^{J} \ell_{m,i} c_i^{(k)}}{\sum_{i=1}^{J} (\ell_{m,i})^2} \ell_{m,j} \,, \tag{2.8}$$

donde $\lambda^{(k)}$ es un parámetro de relajación que determina la cantidad de actualización para la iteración k.

Una imagen formada por el conjunto de coeficientes (o valores de densidad) de los píxeles (en el caso de dos dimensiones) representados por $\{c_j\}$ puede verse como un punto en el espacio J-dimensional, por lo tanto cada ecuación y_m puede verse como un hiperplano. De esta forma si la solución al sistema de ecuaciones $\bar{y} = \mathbf{L}\bar{c}$ es única, ésta será el punto de intersección de dichos planos. En general el método propuesto por Kaczmarz consiste en dar un punto inicial a partir del cual se realizan una serie de proyecciones perpendiculares, es decir, este punto inicial (denotado por $c_1^{(0)}$, $c_2^{(0)}$, ..., $c_J^{(0)}$ y representado por el vector $\vec{c}^{(0)}$en la Figura 2.4) se proyecta a la primera linea (y_1 en la Figura 2.4), el punto resultante ($\vec{c}^{(1)}$) se proyecta a la segunda linea (y_2). Cuando se llega a la última línea, se proyecta nuevamente a la primera y así sucesivamente; por lo que, si la solución es única, el método converge al punto de intersección de los hiperplanos.

Algunos instrumentos de adquisición de datos tales como PET [29], producen la reconstrucción usando la metodología anterior. En microscopía electrónica se ha demostrado que usar el algortimo ART produce mejores resultados que cuando se usa alguno de los basados en transformadas [18].

Las funciones base usadas en el ejemplo anterior no son suaves, sin embargo, muchos objetos se pueden aproximar empleando funciones base como las que se verán a continuación.



(a) Proyectar los puntos sobre los hiperplanos.

(b) Si existe la solución el método converge a ese punto.

Figura 2.4: Esquema que muestra el funcionamiento básico del método de Kaczmarz [3].

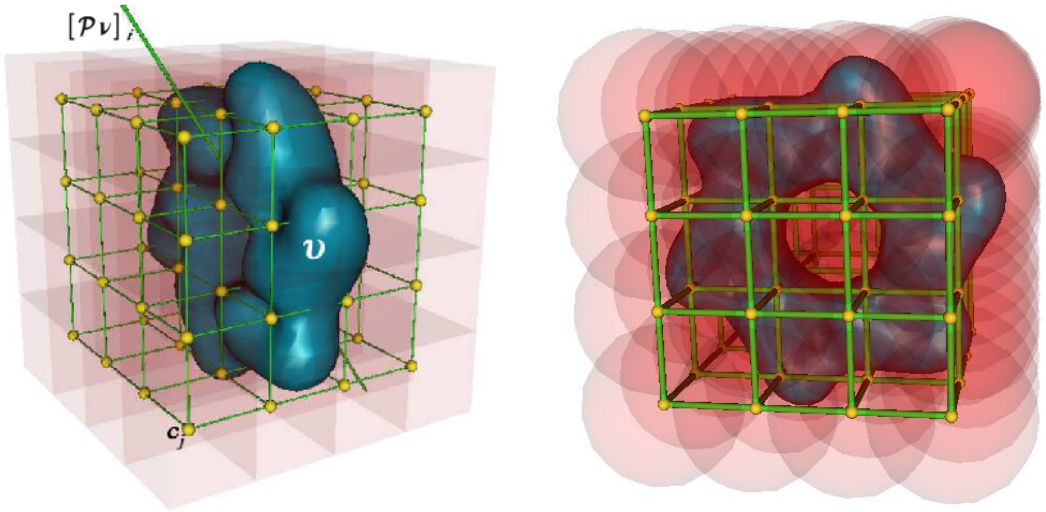
2.5. Funciones Kaiser-Bessel Generalizadas (*blobs*)

Las funciones base de (2.7) son no-continuas (lo cual resulta en una función $\nu(\bar{x})$ discontínua) y aunque ellas han sido utilizadas en aplicaciones prácticas, para algunas aplicaciones biomédicas se usan funciones base contínuas (tal como en TEM [46]), con las cuales se han producido mejores resultados que con las funciones de (2.7).

En este trabajo usamos funciones base radialmente simétricas que son contínuas y con una transición suave en el intervalo $[1, 0]$, definidas como

$$b\left(m,a,\alpha;r\right)=\begin{cases}\frac{I_m\left(\alpha\sqrt{1-\left(\frac{r}{a}\right)^2}\right)}{I_m(\alpha)}\left(\sqrt{1-\left(\frac{r}{a}\right)^2}\right)^m, & \text{si } 0\leq r\leq a,\\ 0, & \text{en otro caso,}\end{cases} \tag{2.9}$$

donde r es la distancia radial desde el centro de la función, I_m es la función de Bessel de orden m, a es el radio de la función (o soporte, es decir donde la función no vale cero) y α es el parámetro que controla su forma. En el resto de este trabajo nos referiremos a la función de (2.9) como *blob*. Entonces los parámetros m (un entero no negativo), a y α (números reales no negativos) controlan la suavidad y forma del blob por lo que la elección de estos valores es importante ya que afectan los resultados del algoritmo de reconstrucción. Estas funciones fueron propuestas para reconstrucción por Lewitt [33] y son una generalización de las funciones Kaiser-Bessel usadas en procesamiento digital de señales.



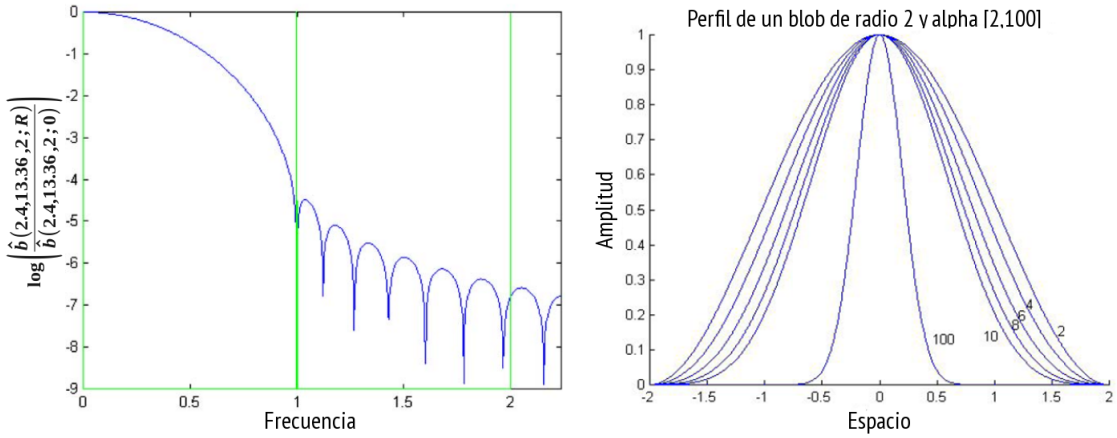
(a) Cubos como funciones base. (b) *Blobs* como funciones base.

Figura 2.5: Diferentes funciones base para (2.4) de [18].

En el ejemplo visto anteriormente de reconstrucción ART para el caso de dos dimensiones (ver Figura 2.3) las funciones base fueron celdas cuadradas, para el caso de tres dimensiones como se puede observar en la Figura 2.5 (a), las celdas resultan ser

celdas cúbicas; sin embargo, la función reconstruida resulta no tener una transición suave debido a los cambios bruscos de las celdas. La Figura 2.5 (b) muestra el caso en el que se usan los *blobs* como funciones base y debido a su característica de suavidad permiten obtener reconstrucciones más suaves que al usar celdas cúbicas.

Estas funciones son ampliamente usadas en procesamiento de imágenes pues poseen soporte compacto al ser acotadas en el intervalo [1,0] y al poseer, en la práctica, ancho de banda limitado tal y como puede observarse en la Figura 2.6.



(a) (b)

Figura 2.6: (a) Espectro normalizado de un blob. (b) Perfil de un blob con diferentes valores de α.

Como se mencionó anteriormente el parámetro m del blob, es un entero positivo que determina el orden o número de derivadas que posee en el punto a. El parámetro α se relaciona con la tasa de cambio de la función por lo que en visualización generalmente es asociado con la suavidad de la función. Cuando $m = 2$ el blob es una función suave que tiene derivada en todo el soporte y como áun no se ha demostrado que valores de m mayores a 2 sean mas efectivas, entonces tal y como se hace en [19], usaremos dicho valor. En la Figura 2.6(b) podemos observar que el parámetro α afecta la forma del blob, por lo que, mientras más pequeño sea este valor la función decae más lentamente.

2.6. Parámetros para Reconstrucción

Cuando se usa la función (2.9) para la aproximación de (2.4) es necesario seleccionar apropiadamente la distribución $\{\bar{p}_j\}$ de los centros de los *blobs* y los parámetros α y a de los mismos. En [36] se propone un criterio para seleccionar estos parámetros. A continuación hacemos un resumen de este criterio. Primero, definimos la transformada de Fourier de una función g en $\mathbb{R}^n$ como

$$\widehat{g}\left(\bar{\xi}\right) = (2\pi)^{-\frac{n}{2}} \int_{\mathbb{R}^n} g\left(\bar{x}\right) e^{-i\left\langle \bar{x},\bar{\xi}\right\rangle} d\bar{x}, \tag{2.10}$$

para todo $\bar{\xi} \in \mathbb{R}^n$. Una relación importante entre la convolución de dos funciones y sus transformadas de Fourier está dada por el siguiente teorema.

Teorema 1 *(Convolución). Sean f y g dos funciones en $\mathbb{R}^n$, entonces*

$$\widehat{f * g} = (2\pi)^{\frac{n}{2}}\, \widehat{f} \times \widehat{g} \; y \; \widehat{f \times g} = (2\pi)^{\frac{n}{2}}\, \widehat{f} * \widehat{g}. \tag{2.11}$$

Las distribuciones de puntos que son de interés para este trabajo son las siguientes. El primer conjunto es el más usado en procesamiento de imágenes y está definido por

$$G_\Delta = \left\{\Delta\bar{k} | \bar{k} \in \mathbb{Z}^3\right\}, \tag{2.12}$$

donde Δ es un número real positivo que se conoce como la distancia de muestreo (ver Figura 3.1 (a)). Es fácil ver que para cualquier punto $\bar{k} \in G_\Delta$ su vecindario de Voronoi es un vóxel cúbico definido por

$$\textit{vóxel}\left(\bar{k}\right) = \left\{\bar{x} \in \mathbb{R}^3 | \Delta\left(k_i - \frac{1}{2}\right) \leq x_i < \Delta\left(k_i + \frac{1}{2}\right), \textit{para}\; 1 \leq i \leq 3\right\}. \tag{2.13}$$

Es fácil ver que los voxeles de (2.12) son cubos y por ello el conjunto G_Δ se conoce como rejilla cúbica simple (sc). Cabe mencionar que para referirnos a un vóxel cúbico es suficiente con referirnos al punto $\bar{k}$ que le da origen.

Otro conjunto de gran interés en procesamiento de imágenes es el definido por

$$B_\Delta = \left\{ \Delta\bar{k} | \bar{k} \in \mathbb{Z}^3 \; y \; k_1 \equiv k_2 \equiv k_3 \, (mod\, 2) \right\}. \tag{2.14}$$

A este conjunto se le conoce como la rejilla cúbica centrada en el cuerpo (bcc) y el vecindario de Voronoi asociado a cada punto de esta rejila es un octahedro truncado (ver Figura 3.1(b)).

Finalmente, nos interesa el conjunto

$$F_\Delta = \left\{ \Delta\bar{k} | \bar{k} \in \mathbb{Z}^3 \; y \; k_1 + k_2 + k_3 \equiv 0 \; (mod\, 2) \right\}. \tag{2.15}$$

A este conjunto de puntos de le conoce como rejilla cúbica centrada en la cara (fcc) y el vecindario de Voronoi asociado a cada uno de sus puntos es un rombo dodecahedro (ver Figura 3.1(c)).

Un tren de pulsos sobre una rejilla Γ se define como $\text{Ш}_\Gamma = \sum_{\gamma \in \Gamma} \delta_{\bar{\gamma}}$. Por lo tanto podemos tener trenes de pulsos sobre las rejillas G_Δ, B_Δ y F_Δ definidas como Ш_{G_Δ}, Ш_{B_Δ} y Ш_{F_Δ}.

En [5, 18] se ha demostrado que las transformadas de Fourier de estos trenes de pulsos se definen como sigue

$$\text{Ш}_{G_\Delta} = \left(\frac{\sqrt{2\pi}}{\Delta} \right)^3 \text{Ш}_{G_{\frac{2\pi}{\Delta}}}, \tag{2.16}$$

$$\text{Ш}_{B_\Delta} = \frac{1}{\sqrt{2}} \left(\frac{\sqrt{\pi}}{\Delta} \right)^3 \text{Ш}_{F_{\frac{\pi}{\Delta}}}, \tag{2.17}$$

$$\text{Ш}_{F_\Delta} = \sqrt{2} \left(\frac{\sqrt{\pi}}{\Delta} \right)^3 \text{Ш}_{B_{\frac{\pi}{\Delta}}}. \tag{2.18}$$

Se ha demostrado en [37, 14] que la rejilla bcc es la más "eficiente" para muestrear el espacio Euclideano 3D cuando una función de ancho de banda limitado y espectro radialmente simétrico. Para reconstrucción, Matej y Lewitt [36] demostraron que la rejilla más deseable para $\{\bar{p}_j\}$ es la bcc. Entonces, con lo definido anteriormente queda seleccionar los parámetros α, a y $\triangle$. Para la combinación lineal de $blobs$ con coeficientes $c_j = 1$ para $1 \leq j \leq J$ se debe tener una aproximación de una función

constante. En tal caso, la parte derecha de (2.4) se convierte en la convolución del *blob* de 2.9 con una versión truncada de Ш_{B_Δ}. Por lo tanto, dicha convolución es aproxidamente

$$\widehat{b} \times \frac{1}{\sqrt{2}} \left(\frac{\sqrt{\pi}}{\Delta} \right)^3 \text{Ш}_{F_{\frac{\pi}{\Delta}}}. \tag{2.19}$$

Para que esta ecuación aproxime de la mejor forma la transformada de Fourier de una función constante, la cual es un pulso en el origen, es apropiado seleccionar la función del *blob* de tal forma que $\widehat{b}\,(2, \alpha, a; R)$sea cero en las posiciones de $F_{\frac{\pi}{\Delta}}$ que se encuentren más cercanas al origen; en otras palabras $\xi = \frac{\sqrt{2}\pi}{\Delta}$. La transformada de Fourier de un *blob* está definida como sigue[33]

$$\widehat{b}\,(2, \alpha, a; R) = \frac{a^3 \alpha^2}{I_2\,(\alpha)} \begin{cases} \frac{I_{\frac{7}{2}}\left(\sqrt{\alpha^2 - (aR)^2}\right)}{\left(\sqrt{\alpha^2 - (aR)^2}\right)^{\frac{7}{2}}}, & si\ aR \leq \alpha, \\ \frac{J_{\frac{7}{2}}\left(\sqrt{(aR)^2 - \alpha^2}\right)}{\left(\sqrt{(aR)^2 - \alpha^2}\right)^{\frac{7}{2}}}, & si\ aR > \alpha, \end{cases} \tag{2.20}$$

donde $I_{\frac{7}{2}}$ nunca se hace cero y $J_{\frac{7}{2}}\left(\sqrt{(aR)^2 - \alpha^2}\right)$ se hace cero por primera vez cuando $\sqrt{(aR)^2 - \alpha^2} = 6.987832$. Por lo tanto, es fácil ver que se tiene la siguiente relación

$$\alpha = \sqrt{2\pi \left(\frac{a}{\Delta} \right)^2 - 6.987932^2}. \tag{2.21}$$

Con esta relación es posible obtener los parámetros α y a bajo el siguiente procedimiento. Fijando Δ y variando el radio a se obtienen varios pares (α, a) para la Δ seleccionada. Por lo tanto, es posible evaluar qué tan bien se aproxima la función constante con (2.4) y cada uno de los parámetros (α, a) por medio del error cuadrático medio. Así, es posible seleccionar parámetros α y a que produzcan una buena aproximación y no sean costosos computacionalmente (un radio grande y un α pequeño abarcan más puntos y por lo tanto se llevan a cabo más cálculos).

Cabe hacer notar que el criterio recién descrito no toma en cuenta la resolución. Los parámetros obtenidos con el proceso anterior han sido utilizados para obtener reconstrucciones vía ART [20]. Sin embargo, en [18] se demostró qué parámetros seleccio-

nados de esta forma pueden producir artefactos en las reconstrucciones y observables en visualizaciones de alta resolución. Para resolver ese problema, en [18] se realiza un procedimiento parecido al descrito arriba para la aproximación de una función constante, pero en esta ocasión se calcula el error cuadrático medio entre las normales de la aproximación (2.4) a una esfera con densidad uno y las normales analíticas. Para seleccionar un solo conjunto de parámetros α y a se realizó el siguiente procedimiento. Se selecciona una aproximación $\nu(\bar{x}) = c_1 b(2, \alpha, m; |\bar{x} - \bar{p}_1|) * c_2 b(2, \alpha, m; |\bar{x} - \bar{p}_2|)$ para $c_{1,2} = 1$ y una separación $|\bar{p}_1 - \bar{p}_2| = 2\Delta$. Para dicha aproximación se busca la isosuperficie para $\nu(\bar{x}) = \frac{1}{2}$ que sea apenas convexa. Para hallar dicha superficie se utiliza los parámetros obtenidos para aproximar las normales de la esfera de densidad igual a uno. Bajo dichas circunstancias se halló que los parámetros que obtienen una superficie apenas convexa son los siguientes $\alpha = 13.36$, $a = 2.4$, $m = 2$ y $\Delta = \frac{1}{\sqrt{2}}$. Por lo tanto, estos son los valores que usaremos en este trabajo.

Capítulo 3

Visualización por Superficies

Una vez que se obtiene la aproximación de una función de densidad ν por medio de (2.4) con las funciones base definidas por (2.9) es posible visualizarla por dos métodos básicos. El primero utiliza el método *raycasting* para hallar el conjunto

$$\mathcal{S}_\tau = \{\bar{x} | \nu(\bar{x}) = \tau\}, \tag{3.1}$$

donde τ es un valor real. La hipótesis detrás de esta ecuación es que existe un umbral (o isovalor) τ para el que los valores de ν mayores a τ definen los puntos dentro de los objetos de interés, mientras que valores de ν menores a τ definen los puntos en el exterior de los objetos de interés. Por lo tanto, el conjunto de (3.1) define puntos en la frontera o *superficie* de los objetos de interés. El conjunto $\mathcal{S}_\tau$ también es conocido como *isosuperficie* o *superficie implícita* y representa puntos de valor constante en un espacio volumétrico, es decir, es un conjunto de nivel de la función continua ν cuyo dominio es el espacio tridimensional. En el caso de imaginología biomédica representa regiones de las imágenes en donde hay un valor particular de densidad dado por τ. La selección de umbral o isovalor en la ecuación (3.1) es en sí un proceso tradicional de segmentación conocido como umbralización (o *thresholding*).

En la práctica, el método de *raycasting* se puede usar para visualizar la aproximación a (3.1) usando 2.4 directamente. El método consiste, básicamente, en "lanzar" varios rayos desde el plano de proyección (típicamente la pantalla de la computadora) hacia el arreglo de puntos $\{\bar{p}_j\}$ que tienen asociados los valores del conjunto $\{c_j\}$. Estos

rayos pueden ser perpendiculares al plano de proyección (perspectiva ortogonal) o en la dirección de un punto fijo en el espacio (proyección en perspectiva). Para cada rayo se encuentran las M funciones que interceptan dicho rayo. Para dicha línea se encuentra el punto más cercano al plano de proyección tal que $\sum_{m=1}^{M} c_m b_m \left(\bar{x} - \bar{p_m}\right) - \tau = 0$. Existen varios métodos que realizan este proceso de manera eficiente como en [15] y en [44]. Cabe mencionar que en graficación por computadora el modelado de objetos por medio de (2.4) se conoce como *modelo blobby*, propuesto por [8] y se han sugerido varias funciones base tales como las propuestas por [36].

Un segundo método para visualizar la función contínua ν consiste en, primero, discretizarla de la siguiente forma

$$V_\Gamma\left(\bar{k}\right) = \nu\left(\bar{x}\right) \times \text{Ш}_\Gamma, \tag{3.2}$$

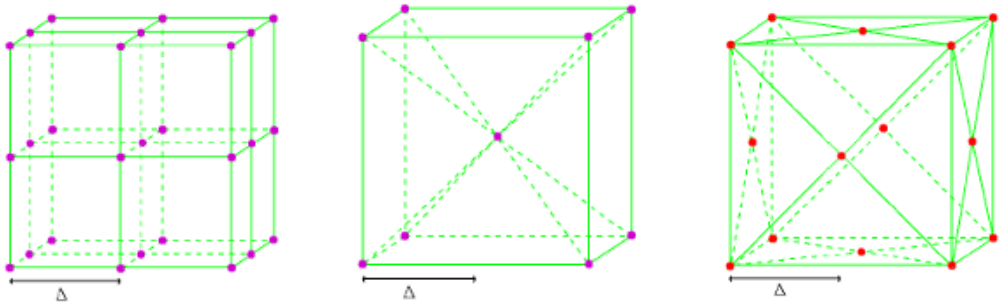
donde Ш_Γ representa el tren de pulsos distribuido o arreglado de una forma específica [18]. Arreglos típicos son los conjuntos 2.12, 2.14 y 2.15 que se describieron en el capítulo anterior.

Sin embargo, la forma más común de realizar la discretización de la función ν de (2.4) es

$$V_{G_\Delta}\left(\bar{k}\right) = \nu\left(\bar{x}\right) \times \text{Ш}_{G_\Delta}. \tag{3.3}$$

El uso de las rejillas (2.14) y (2.15) es poco común a pesar de poseer mejores propiedades para el procesamiento de imágenes y de ser más óptimas que la definida en (2.12) [14]. En este trabajo usaremos la discretización de ν definida por (2.12).

Después de un proceso de adquisición de imágenes y, generalmente de un proceso de conversión de datos y filtrado, se obtiene una imagen 3D. A partir de aquí se pueden seguir varios caminos para visualizar un objeto pero generalmente se clasifican en dos grupos. El primer grupo asume que la imagen 3D V_{G_Δ} de (2.4) se puede entender como un conjunto de imágenes 2D de regiones contiguas apiladas. Por lo tanto, se puede representar el objeto en la imagen 3D a partir de los contornos del objeto presentes en cada imagen 2D. El segundo grupo representa a los objetos en



(a) Rejilla SC. (b) Rejilla BCC. (c) Rejilla FCC.

Figura 3.1: Diferentes tipos de Rejilla

V_{G_Δ} tomando en cuenta toda la imagen 3D. Entre los métodos del primer grupo se encuentra el algoritmo presentado por E. Keppel [32] donde se reconstruye a partir de una pila de contornos de las imágenes transversales en dos dimensiones. Varios autores han hecho alguna modificación de este algoritmo como los presentados en [9] y [16] pero básicamente consisten en definir primero los contornos de los objetos de interés en cada una de las imágenes ya sea de forma interactiva o utilizando algún método de detección de bordes. Posteriormente los contornos de imágenes contiguas se conectan para poder formar una estructura tridimensional, ésto generalmente se realiza por medio de triangulación. Así, el paso principal de estos métodos es la técnica empleada para conectar los contornos, pero en la práctica éstos pueden ser formas extremadamente complejas, tanto, que de una imagen a la siguiente los cambios pueden variar ampliamente. Por esa razón, la conexión de los contornos generalmente se basa en heurísticas donde hay que definir cuál se aplica a cada caso, por lo que esos métodos casi no son empleados actualmente [24].

En el caso de los métodos que emplean toda la imagen 3D, el siguiente paso en el proceso de visualización consiste en identificar los diferentes objetos representados en el volumen de tal forma que se puedan seleccionar aquellos vóxeles que representan los objetos que se desea visualizar. Este paso se lleva a cabo por medio de un proceso conocido como segmentación. La forma más simple consiste en binarizar los datos con un umbral (o isovalor) de intensidad para distinguir, por ejemplo, los huesos de otros tejidos (como los músculos) en el caso de tomografías compuarizadas. Este método de

umbralización asume que se tiene una distribución binomial en los valores de densidad de la función V_{G_Δ} . Esta situación no siempre se cumple y por lo tanto existen técnicas más sofisticadas, así que la investigación en esta área es abundante.

Para los métodos que trabajan directamente en toda la imagen V_{G_Δ} existen dos métodos básicos para realizar su visualización.. El primero de ellos se conoce como visualización directa de volúmenes (*volume rendering*). El segundo es la visualización indirecta o por superficie (*surface rendering*); haciendo énfasis en que éste puede ser ejecutado por métodos estándares de graficación por computadora. Los métodos basados en superficie y los basados en vóxeles tienen sus méritos; sin embargo, la decisión de cuál emplear en una aplicación específica depende de los recursos computacionales disponibles así como de los objetivos de la visualización [24].

La visualización directa de volúmenes consiste en asignar un valor de opacidad a cada vóxel de V_{G_Δ} (proceso conocido como clasificación o función de transferencia [48]). Posteriormente, todos los vóxeles son proyectados y sus opacidades se acumulan a lo largo de la dirección de proyección. Cabe mencionar que existen otros esquemas además del de acumulación; tales como seleccionar el valor máximo, *Maximum Intensity Projection* (MIP) [22, 38].

La visualización por superficies se basa en (3.1); sin embargo, sobre la función discretizada V_{G_Δ}. Por lo tanto, los diversos métodos que emplean esta técnica encuentran el conjunto $\mathcal{F}_\tau$ que aproxima al conjunto $\mathcal{S}_\tau$ de (3.1). Existen muchos métodos en la literatura para hallar el conjunto $\mathcal{F}_\tau$ pero básicamente pueden ser clasificados en dos categorías: basados en descomposición de vóxeles y los basados en crecimiento de regiones de la superficie [1]. Dentro de la primera categoría el método más conocido es el *Marching Cubes* propuesto por Lorensen y Cline en 1987 [34], más adelante se describe con un poco más de detalle pero básicamente consiste en definir un cubo formado por ocho vóxeles contiguos y determinar si las esquinas de éste cubo (centros de los ocho vóxeles) se encuentran dentro o fuera de la superficie. De este modo, dependiendo de la configuración del cubo, se realiza una triangulación generando así por lo menos un triángulo por cada vóxel que atravieza la superficie. Como la superficie queda definida por pequeños triángulos entonces generalmente se requiere de

algoritmos de postprocesado y simplificación. También se han realizado variantes al *Marching Cubes* o modificaciones para resolver ambiguedades o prevenir agujeros. En [39] se presenta un resumen de los diferentes trabajos realizados a partir de *Marching Cubes.*

Con respecto a los algoritmos basados en crecimiento de regiones de la superficie para encontrar $\mathcal{F}_\tau$ se encuentra el propuesto por Stoddart, Illingworth y Windeatt [28]. Este algoritmo es conocido como *Marching Triangles* y produce una malla a partir de una nube de puntos. Básicamente consiste en encontrar un triángulo perteneciente a la superficie e ir añadiendo vértices que formen triángulos con las aristas pertenecientes a la malla y que cumplan con los requisitos de una triangulación de Delaunay. Comparado con los métodos basados en vóxeles éste algoritmo puede generar mallas con mayor calidad y mejor equiespaciadas; sin embargo, para superficies con grandes curvaturas puede generar un gran número de triángulos para lo cual se han propuesto algoritmos como el presentado en [1].

Sin embargo, el *Marching Cubes*, se ha establecido como el estándar de visualización por superficies [39]. Este algoritmo produce un conjunto $\mathcal{M}_\tau$ que aproxima una isosuperficie $\mathcal{S}_\tau$. Por otra parte, propuesto por Artzy, Frieder y Herman [2], es un algoritmo que produce un conjunto $\mathcal{A}_\tau$ que aproxima al conjunto $\mathcal{S}_\tau$ burdamente (crea representaciones cuboides) pero posee ciertas propiedades interesantes que se describirán más adelante.

Así, una vez que se obtiene el conjunto $\mathcal{F}_\tau$ (ya sea $\mathcal{M}_\tau$ o $\mathcal{A}_\tau$) por cualquiera de los métodos antes mencionados, la representación final se produce usando técnicas estándares de graficación por computadora [12, 15]. Como veremos más adelante, los vectores normales a la superficie juegan un papel muy importante en la generación de la representación final. Por ejemplo, las normales asociadas al conjunto $A\tau$ producido por la implementación original del Algoritmo de Artzy et al contribuyen a la apariencia cuboide de las representaciones obtenidas por éste método. Sin embargo, en este trabajo debido a sus propiedades matemáticas estamos interesados en usar el Algoritmo de Artzy para producir el conjunto $\mathcal{F}_\tau$ pero suavizaremos su representación final por medio de asignar normales "apropiadas" a los vértices de la malla $\mathcal{A}_\tau$. Sabe-

mos que $V_{G_\triangle}$ es la discretización de (2.4) y que poseemos los conjuntos $\{c_j\}$ y $\{\bar{p}_j\}$ así como los parámetros del *blob* (ver sección 2.6) porque asumimos que se usó ART para aproximar la función $\nu(\bar{x})$. Por lo tanto podemos conocer el gradiente, y por lo tanto la normal en cualquier punto de la aproximación dada por (2.4). Debido a que la aproximación (2.4) con las funciones de (2.9) producen una función contínua y de variación suave, esperamos que la asignación de las normales a los vértices de $\mathcal{A}_\tau$ a apartir de dicha aproximación produzca una representación final suave y comparable con la producida con el conjunto $\mathcal{M}_\tau$.

3.1. Algoritmo de Extracción de Superficie de Lorensen y Cline

Este algoritmo es conocido en la literatura como *Marching Cubes* y su objetivo es extraer la malla poligonal $\mathcal{M}_\tau$ que aproxime al conjunto $\mathcal{S}_\tau$ de (3.1), dicha malla es representada por un conjunto de triángulos. Para obtener $\mathcal{M}_\tau$, el algoritmo requiere de un τ y de la discretización $V_{G_\triangle}$ [34]. El algoritmo realiza los siguientes tres pasos básicos

1. Segmentación.
2. Creación de triángulos.
3. Cálculo de normales.

Para localizar la superficie, el algoritmo recorre los vóxeles tomando 8 a la vez, es decir, crea un cubo lógico formado con los centros de 8 vóxeles tomados de la siguiente forma, ver Figura 3.2. Dado un punto $\bar{k}^0 \in V_{G_\triangle}$, el cubo lógico se forma con otros siete puntos $\bar{k}^1 = (k_1^0+1, k_2^0, k_3^0)$, $\bar{k}^2 = (k_1^0, k_2^0-1, k_3^0)$, $\bar{k}^3 = (k_1^0+1, k_2^0-1, k_3^0)$, $\bar{k}^4 = (k_1^0, k_2^0, k_3^0+1)$, $\bar{k}^5 = (k_1^0+1, k_2^0, k_3^0+1)$, $\bar{k}^6 = (k_1^0, k_2^0-1, k_3^0+1)$, $\bar{k}^7 = (k_1^0+1, k_2^0-1, k_3^0+1)$. Nos referiremos a este conjunto de puntos como $\mathcal{C}_{\bar{a}}$ donde $\bar{a} = \bar{k}^0$. Si al conjunto $\mathcal{C}_{\bar{a}}$ lo dividimos en un subconjunto interior $\mathcal{I}_{\bar{a}} = \{\bar{p} \in \mathcal{C}_{\bar{a}} | \nu(\bar{p}) > \tau\}$, que representa puntos en el interior de la isosuperficie de interés, y en otro subconjunto exterior $\mathcal{E}_{\bar{a}} =$

$\{\bar{p} \in \mathcal{C}_{\bar{a}} | \nu(\bar{p}) < \tau\}$, que representa puntos en el exterior de la isosuperficie, entonces claramente existen 2^8 posibles casos. Dada la configuración anterior existen casos en los que todos los vértices en $\mathcal{C}_{\bar{a}}$ también pertenecen a $\mathcal{E}_{\bar{a}}$, lo que significa que se encuentran fuera de la superficie, mientras que si todos pertenecen a $\mathcal{I}_{\bar{a}}$ significa que todos se encuentran dentro. Éstos casos no interesan, los casos de interés son aquellos en los que $\mathcal{I}_{\bar{a}} \neq \mathcal{C}_{\bar{a}}$ y $\mathcal{E}_{\bar{a}} \neq \mathcal{C}_{\bar{a}}$ porque significa que la isosuperficie atraviesa dicho cubo lógico. Claramente esos casos pueden ser codificados en un *byte* asumiento que un valor de 1 representa a los vértices en $\mathcal{I}_{\bar{a}}$, mientras que un valor 0 a los vértices en $\mathcal{E}_{\bar{a}}$. El algoritmo precalcula un arreglo indizado (*lookup table*) con las posibles configuraciones. Cada índice de ese arreglo es un *byte* que se forma con el valor (0 o 1) de cada punto del cubo lógico y cada valor del arreglo corresponde a la configuración de polígonos (un conjunto de triángulos) que representan la porción de la isosuperficie que atraviesta dicho cubo lógico, ver Figura 3.2.

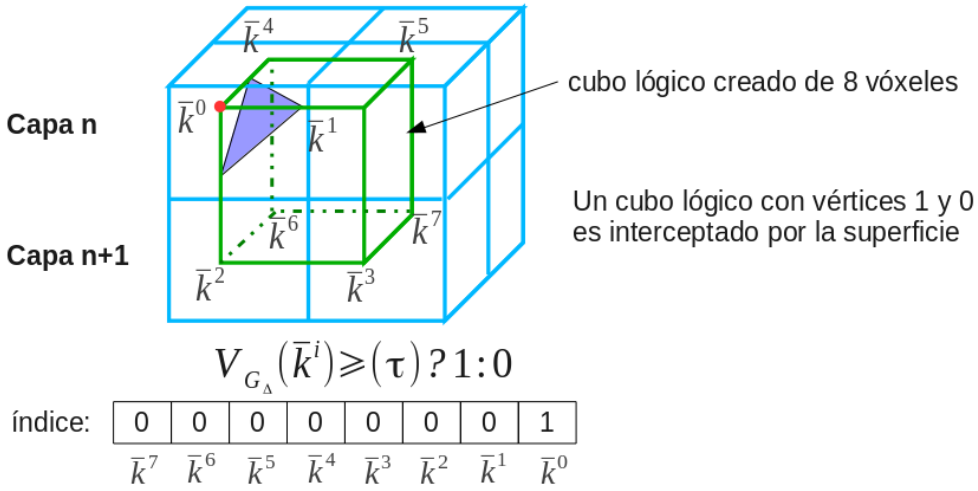


Figura 3.2: Cubo lógico del algoritmo *Marching Cubes*. Los vértices con puntos rojos representan al conjunto $\mathcal{I}_{\bar{a}}$ y los demás al conjunto $\mathcal{E}_{\bar{a}}$. A cada vértice se le asigna el valor 0 o 1 dependiento si su valor de densidad es mayor o menor que τ. El conjunto de ceros y unos forman un *byte* que indica el índice en un arreglo que contiene la configuración correspondiente de triángulos.

El arreglo de 256 posibles configuraciones puede simplificarse debido a complementariedad y simetrías. De esta forma las configuraciones se reducen a solamente 15 diferentes casos los cuales pueden verse en la Figura 3.3. En esa figura los vértices con

puntos rojos también representan al conjunto $\mathcal{I}_{\bar{a}}$, mientras que los otros representan al conjunto $\mathcal{E}_{\bar{a}}$.

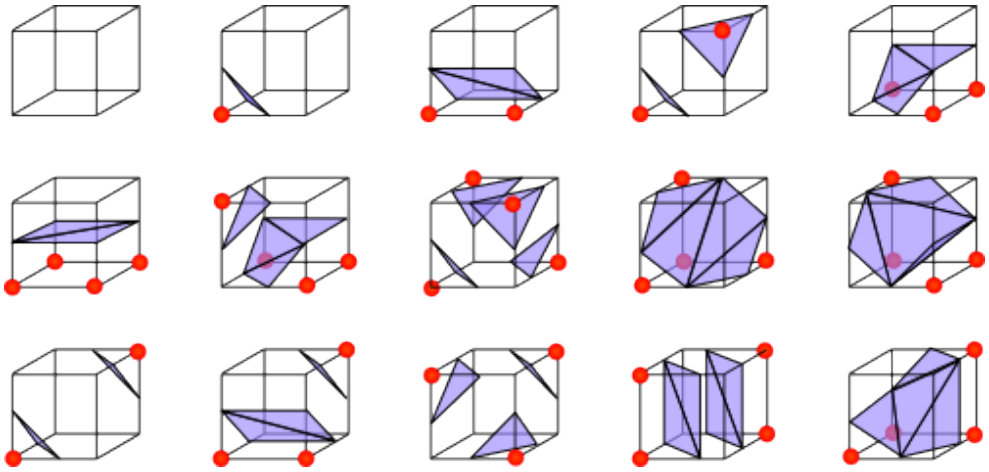


Figura 3.3: Las 15 configuraciones base de *Marching Cubes*.

Para dos puntos $\bar{a}$ y $\bar{b}$ que pertenecen a $\mathcal{C}$ tales que $\bar{a} \in \mathcal{I}$, $\bar{b} \in \mathcal{E}$ y $\left|\bar{a} - \bar{b}\right| = \triangle$ se tiene una arista que cruza la superficie por lo que en ésta se halla un vértice de un triángulo. Para determinar el punto donde se encuentra un vértice en la arista se realiza una interpolación, generalmente lineal, de la siguiente manera. Dado el umbral τ, entonces el vértice del triángulo está dado por

$$\bar{r} = \bar{a} + \rho\left(\bar{b} - \bar{a}\right), \tag{3.4}$$

donde

$$\rho = \frac{\tau - V_{G_\triangle}\left(\bar{a}\right)}{V_{G_\triangle}\left(\bar{b}\right) - V_{G_\triangle}\left(\bar{a}\right)}. \tag{3.5}$$

Así, al terminar de recorrer todos los vóxeles del volúmen, los triángulos encontrados para cada cubo lógico forman la malla $\mathcal{M}_\tau$ que aproxima la superficie $\mathcal{S}_\tau$ que se desea representar.

Típicamente para estimar las normales en los vértices de la malla $\mathcal{M}_\tau$ se puede realizar el promedio de los productos vectoriales de las aristas comunes a cada vértice.

3.2. Algoritmo de Sequimiento de Superficies de Artzy *et al*

Este algoritmo propuesto por Artzy, Frieder y Herman [2] es conocido en la literatura simplemente como Algoritmo de Artzy y también permite obtener una malla o un conjunto de vértices $\mathcal{A}_\tau$ dado un umbral τ para un volumen discretizado V_{G_Δ} tal que $\mathcal{A}_\tau$ aproxima $\mathcal{S}_\tau$. A diferencia del *Marching Cubes* los polígonos producidos por este algoritmo son caras cuadradas y ortogonales a los ejes coordenados en vez de un conjunto de triángulos.

El Algortimo de Artzy asume que el volumen dado por V_{G_Δ}se binariza, obteniendo $B\left(G_\Delta\right)$, es decir, que los vóxeles que lo componen tienen asignado uno de dos posibles valores 0 o 1. Para hacer esta clasificación se puede emplear cualquier método de clasificación o segmentación; sin embargo, para este trabajo se emplea la umbralización ya que es el empleado por *Marching Cubes*, algoritmo con el que se desea comparar resultados.

Preliminares del Algoritmo de Artzy

En [26] se presenta tanto el Algoritmo de Artzy como los conceptos preliminares necesarios para su entendimiento, los cuales describimos brevemente a continuación.

Los elementos pertenecientes a la superficie o frontera (también llamados *bels* por el inglés *boundary elements*) son caras comunes a vóxeles *adyacentes,* uno interior y otro exterior. Por lo anterior vemos que es necesario definir el término adyacencia; sin embargo, antes observemos que podemos hacer referencia a un vóxel por el punto en V_{G_Δ}al cual se asocia, por ejemplo, el vóxel $\bar{c} \in V_{G_\Delta}$.

La adyacencia entre dos vóxeles $\bar{c}$, $\bar{d}$ es una relación binaria simétrica sobre el conjunto de todos los vóxeles del espacio y se dice que $\bar{c}$ y $\bar{d}$ son vecinos cuando son adyacentes. Existen varias definiciones de adyacencia para los vóxeles cúbicos pero para este trabajo vamos a emplear los siguientes dos.

Los vóxeles $\bar{c}$, $\bar{d}$ en $\mathbb{Z}^3$ son adyacentes por cara, o ω-adyacentes, si difieren sola-

mente en una de sus coordenadas, es decir:

$$\left(\bar{c},\, \bar{d}\right) \in \omega \iff \left\|\bar{c}_n - \bar{d}_n\right\| = 1, \tag{3.6}$$

donde $\left\|\bar{k}\right\|$ representa la norma ℓ_2 del vector $\bar{k}$ definida como $\sqrt{\sum_{i=1}^{n} k_i^2}$. A esta adyacencia también se le conoce como 6-vecindad de un vóxel pues puede ser adyacente a otros 6 vóxeles como se observa en la Figura 3.4.

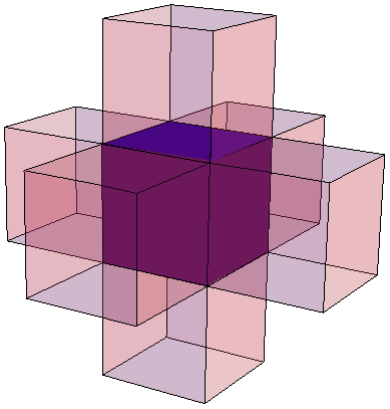


Figura 3.4: Se muestra la ω-adyacencia del vóxel azul con sus 6 vóxeles adyacentes en rosa.

Los vóxeles $\bar{c}$, $\bar{d}$ en V_{G_Δ} son adyacentes por arista, o δ-adyacentes, si comparten una cara o una arista; es decir

$$\left(\bar{c},\, \bar{d}\,\right) \in \delta \iff 1 \leq \left|\bar{c}_n - \bar{d}_n\right| \leq \sqrt{2}, \tag{3.7}$$

también conocida como 18-vecindad por tener precisamente 18 vecinos como se puede observar en la Figura 3.5. Como se puede observar, la ω-adyacencia es una subrelación de la δ-adyacencia.

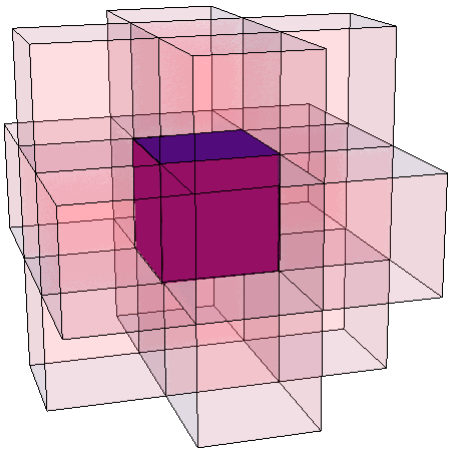


Figura 3.5: Se muestra la δ-adyacencia del vóxel azul con sus 18 vóxeles adyacentes en rosa.

A partir de las definiciones anteriores se dice que dos vóxeles $\bar{c}$, $\bar{d}$ estan conectados si existe un γ-camino, es decir, si existe una secuencia $\langle \bar{c}^0, \bar{c}^1, ..., \bar{c}^N \rangle$ de vóxeles en V_{G_Δ} tal que $\bar{c}^0 = \bar{c}$ y $\bar{c}^N = \bar{d}$, para $1 \leq n \leq N$, si $\bar{c}^n$ y $\bar{c}^{n-1}$ son γ-adyacentes. Así, se tienen ω-caminos con vóxeles ω-conexos y δ-caminos con vóxeles δ-conexos.

Como se mencionó anteriormente, el Algoritmo de Artzy supone que los vóxeles se encuentran clasificados en dos conjuntos $O = \left\{ \bar{k} \in \mathbb{Z}^3 \left| V_{G_\Delta} \left(\bar{k} \right) \leq \tau \right. \right\}$ y $Q = \left\{ \bar{k} \in \mathbb{Z}^3 \left| V_{G_\Delta} \left(\bar{k} \right) > \tau \right. \right\}$, donde O representa a los vóxeles exteriores y Q a los interiores, se define la frontera ∂ entre O y Q como

$$\partial (O, Q) = \left\{ \left(\bar{c}, \bar{d} \right) | \bar{c} \in O, \bar{d} \in Q, \left(\bar{c}, \bar{d} \right) \in \omega_N \right\}. \tag{3.8}$$

Entonces la función $\partial (O, Q)$ esta formada por todos los *bels* $\beta = \left(\bar{c}, \bar{d} \right)$ tales que $\bar{c} \in O$, $\bar{d} \in Q$ y $\left(\bar{c}, \bar{d} \right) \in \omega$, es decir los *bels* representan una cara rectangular entre dos vóxeles, uno interior y el otro exterior.

Así, el objetivo de este algoritmo es encontrar el conjunto $\mathcal{A}_\tau$ de *bels* que aproximan la superficie $\mathcal{S}_\tau$. Para ello es necesario definir un punto de inicio o semilla $\beta_0 \in \partial (O, Q)$ a partir del cuál se encontrarán las demás caras que aproximan ∂. Teniendo el punto de inicio ahora es necesario definir los movimientos necesarios para encontrar

la siguiente cara. Estos movimientos se ejemplifican en la Figura 3.6 tomando un vóxel de referencia, en donde las flechas indican la dirección del recorrido (no el orden) y atravesando sólamente una arista del vóxel a la vez. Es importante mencionar que en [26] se indica que hay configuraciones simétricas para esos recorridos; sin embargo, no se pierde generalidad al restringirlas a esta configuración. También se puede observar en la Figura 3.6 que cada vez que se atraviesa una arista y se obtiene una cara, para ir a la siguiente sólo se tienen dos direcciones posibles en las que puede continuar el recorrido. De esta forma se obtienen sólamente seis posibles direcciones al recorrer un vóxel.

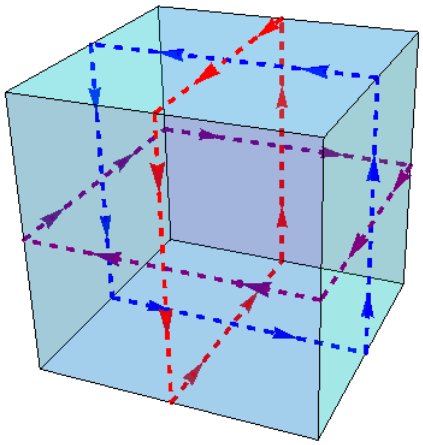
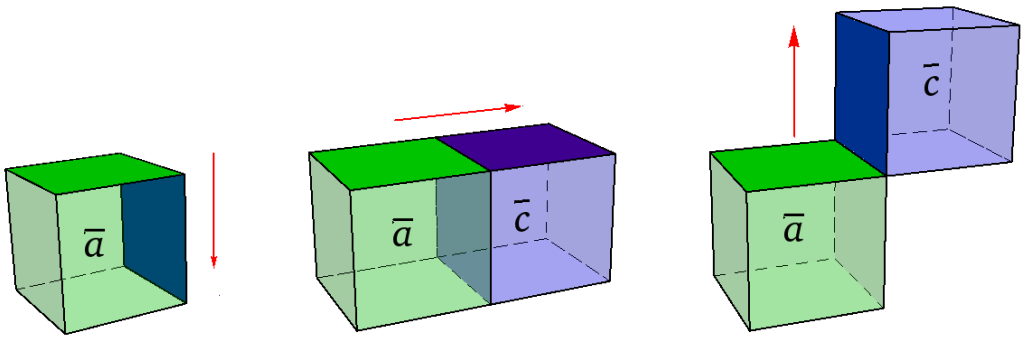


Figura 3.6: Configuracion de direcciones válidas en el Algoritmo de Artzy.



(a) (b) (c)

Figura 3.7: Direcciones válidas en el Algoritmo de Artzy.

Para un *bel* $\beta_i = \left(\bar{a}, \bar{b}\right)$ el algortimo debe encontrar los *bels* β_1 y β_2 tales que se encuentran en alguna de las direcciones de la Figura 3.6 y $\beta_i = \left(\bar{a}, \bar{b}\right)$ cumple sólo alguna de las siguientes condiciones

1. $\bar{c} = \bar{a}$ y $\left(\bar{b}, \bar{d}\right) \in \delta$, ver Figura 3.7(a),

2. $(\bar{a}, \bar{c}) \in \omega$ y $\left(\bar{b}, \bar{d}\right) \in \omega$, ver Figura 3.7(b),

3. $(\bar{a}, \bar{c}) \in \delta$ y $\bar{b} = \bar{d} \in \omega$, ver Figura 3.7(c).

Claramente, la descripción anterior funciona de forma distribuida. Para una máquina secuencial en [2] se presentó un algoritmo eficiente que simula el funcionamiento en paralelo, almacenando para cada *bel* visitado las dos caras (según las dos direcciones correspondientes) a explorar en una estructura de datos Q. El algoritmo trabaja extrayendo iterativamente caras de esta estructura de datos y finaliza cuando no hay más caras que visitar, es decir, Q está se vacía. Para no incluir indefinidamente caras visitadas, el algoritmo mantiene una estructura L de caras visitadas (en se propone un árbol binario [2]). Por lo tanto, si para cada *bel* β_i para $i = \{1, 2\}$, éste ya ha sido incluido en L entonces no es incluido en Q. En el algoritmo1 se reproduce el Algoritmo de Artzy presentado en [2].

La primer ventaja del Algoritmo de Artzy es que no se recorre necesariamente todo el volumen $B\left(G_\Delta\right)$. La segunda es que la frontera o superficie satisface el análogo digital de lo que se conoce como el Teorema de la Curva de Jordan, es decir, dicha superficie divide el volumen en interior y exterior, además de que es conexa y cerrada. Esto se describe en [26] donde se demuestra que este algoritmo es topológicamente correcto por lo que no posee agujeros como podría suceder en la implementación original de *Marching Cubes*.

Algoritmo 1 Algoritmo de Artzy.

Entradas: Semilla o cara inicial β_0 y el volumen discretizado y binarizado B_{G_Δ} con respecto a τ.

Salidas: El conjunto $\mathcal{A}_\tau$ con las caras pertenecientes a ∂.

1. Poner la cara β_0 en $\mathcal{A}_\tau$, agregar β_0 en Q y poner dos copias de β_0 en L.
2. **while** $Q \neq \emptyset$ **do**
3. Saca una cara β de Q
4. Encontrar las caras β_1 y β_2 en B_{G_Δ} a las que se puede ir desde β, siguiendo las dos direcciones indicadas en la Figura 3.6 y de acuerdo a alguno de los casos mostrados en la Figura 3.7.
5. **if** $\beta_1 \in L$ **then**
6. Quitar β_1 de L
7. **else**
8. Poner β_1 en $\mathcal{A}_\tau$, Q y L
9. **end if**
10. **if** $\beta_2 \in L$ **then**
11. Quitar β_2 de L
12. **else**
13. Poner β_2 en $\mathcal{A}_\tau$, Q y L
14. **end if**
15. **end while**

3.3. Visualización de Mallas

Los dos algoritmos anteriores encuentran una aproximación de la isosuperficie por medio de una malla poligonal. La visualización de la malla se realiza a través de técnicas de graficación por computadora. Estas técnicas consisten en generar una imagen bidimensional a partir de la proyección de un modelo tridimensional, usando programas de cómputo [15].

Existen varias técnicas de renderizado, entre ellas la rasterización es la más rápida para producir gráficos tridimensionales en tiempo real. Por ese motivo es el método de renderizado implementado en las tarjetas gráficas. A todo el proceso de renderizado por el que pasa la malla, es decir, desde su representación en polígonos hasta el despliegue en pantalla, se le conoce como *graphics rendering pipeline*. En la Figura 3.8 se muestra de forma simplificada, un ejemplo de este proceso.

Básicamente, la rasterización toma la malla de una escena tridimensional y proyecta los polígonos que la componen sobre una superficie bidimensional o ventana de visualización (el monitor, por ejemplo). Los polígonos que componen la malla son representados por triángulos en el espacio tridimensional y son transformados en triángulos en el espacio bidimensional con posiciones correspondientes en la ventana de visualización. Posteriormente se determina si esos triángulos bidimensionales quedan dentro de los límites de la ventana y finalmente son rellenados, es decir, se le asigna un color a cada uno de los píxeles que le corresponden a cada triángulo, ver Figura 3.9.

Al proceso de cortar los triángulos con puntos fuera de la ventana se le llama *clipping* o recorte, mientras que al proceso de rellenar los píxeles de los triángulos bidimensionales se le conoce como *scanline rendering* o conversión de escaneo o barrido. Otros procesos de graficación por computadora son realizados para determinar si un píxel perteneciente a un triángulo debe ser dibujado o no, pero para propósitos de éste trabajo interesa principalmente ver de forma básica cómo se determina el color de los píxeles sin considerar texturas.

Para lograr la percepción de tridimensionalidad es necesario asignar color a cada

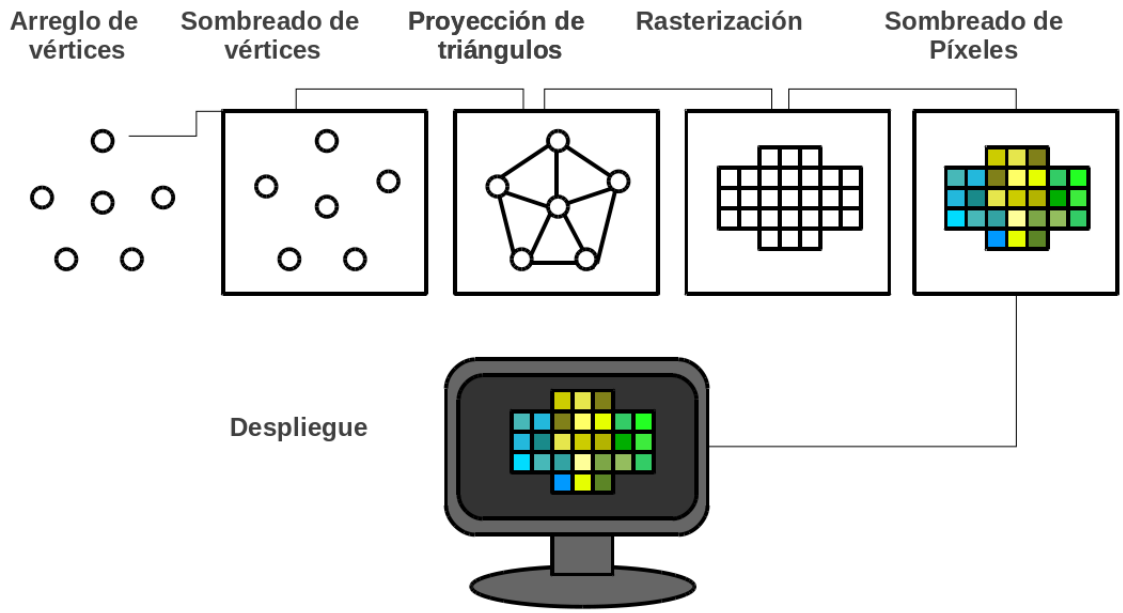


Figura 3.8: Proceso de *renderizado* en graficación por computadora consiste en pasar de la representación en polígonos hasta el despliegue en pantalla.

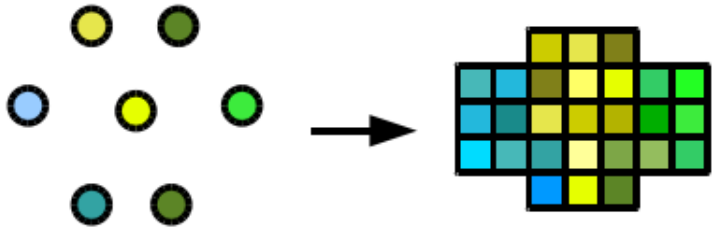


Figura 3.9: Si cada vértice posse un valor diferente de color, el proceso de rasterización se encarga de interpolar los colores y asignarlos a los píxeles correspondientes.

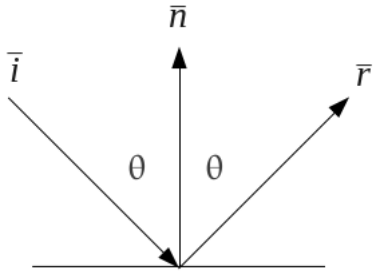


Figura 3.10: Ley de reflexión.

uno de los píxeles apropiadamente reflejando los efectos de iluminación y sombreado, conceptos que se describen a continuación.

3.3.1. Iluminación

La luz generalmente se modela como un conjunto de rayos (vectores) que inciden o se dirigen hacia los objetos. Supongamos que existe un rayo $\bar{i}$ incidente sobre un espejo, como sabemos, ésta es una superficie que refleja, por lo que el rayo rebota dando origen a un rayo $\bar{r}$ que sale de la superficie. Al punto $\bar{p}$ sobre el objeto donde rebota el rayo generalmente se le conoce como punto de incidencia. Si en este punto definimos un vector normal $\bar{n}$, entonces dicho vector divide el ángulo formado entre el rayo incidente y el rayo reflejado en otros dos ángulos. Al ángulo θ_i entre $\bar{n}$ e $\bar{i}$ se le conoce como ángulo de incidencia, mientras que al ángulo θ_r entre $\bar{n}$ y $\bar{r}$ se le conoce como ángulo de reflexión. Dados los términos anteriores, la *ley de reflexión* (o iluminación) indica que cuando un rayo $\bar{i}$ incide sobre una superficie y éste se refleja, entonces los ángulos θ_i y θ_r son iguales, ver Figura 3.10.

El tipo de material de los objetos tiene gran impacto en la forma en la que se reflejan los rayos; sin embargo, cumplen con la ley de reflexión. En el caso de materiales suaves (tales como espejos, agua) los rayos reflejados generalmente tienen la misma concentración que los incidentes, ver Figura 3.11(a) lo que ocasiona que la superficie tenga una apariencia brillosa. A este tipo de iluminación o reflexión se le conoce como *reflexión especular*. En cambio, en el caso de superficies rugosas (como el asfalto o la tela) los rayos son reflejados en diferentes direcciones, ver Figura 3.11(b), a este

(a) (b)

Figura 3.11: (a) Reflexión Especular, (b) Reflexión Difusa.

modelo se le conoce como *reflexión difusa*. En la realidad no todos los rayos son reflejados y dependiendo de la superficie de los objetos, éstos pueden ser también absorbidos o pueden reflejar las propiedades de la luz de cierto color.

Cuando la luz rebota, ésta tiende a disminuir su intensidad con respecto al tiempo y la distancia, por lo que objetos lejanos parecen menos iluminados que objetos cercanos al observador, a este fenómeno se le conoce como *atenuación*. Entonces, el color y la intensidad de la luz que se refleja en los objetos es lo que percibimos como el color de una superficie,ver Figura 3.12.

La iluminación de una escena se consigue mediante la creación de una o varias fuentes de luz, las cuales interactúan con la superficie de los objetos. Por lo tanto, la iluminación en gráficas por computadora consiste básicamente en modelar las fuentes de luz (direccional, puntual o focal [15]) y en determinar un modelo de iluminación (o reflexión) que se encargue de la relación entre la luz y los materiales de los objetos en la escena. Así, el principal objetivo de un modelo de iluminación consiste en calcular la intensidad de luz en cada punto de la superficie de los objetos en la escena, lo que depende de las propiedades del objeto (o material); los colores, tipos y posiciones de las fuentes de luz; y la posición del observador. En gráficas por computadora, la posición del observador es un plano de proyección, generalmente la ventana de visualización, ver Figura 3.12. Existen varias técnicas que realizan el cálculo de la iluminación aproximando la ley de reflexión y así poder obtener el color de los píxeles.

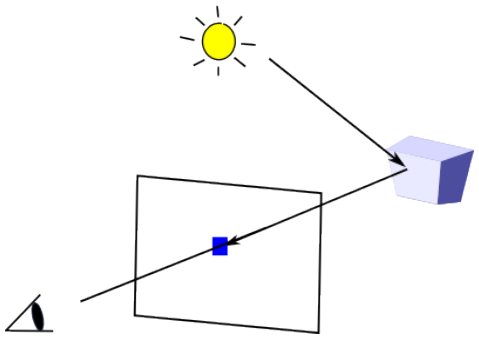


Figura 3.12: La visualización de una escena depende de la posición del observador, las propiedades del objeto así como del tipo y posición de las fuentes de luz.

La técnica o modelo de iluminación más empleado es el de iluminación de Phong [41]. A pesar de que se se sabe que los objetos pueden reflejar o emitir luz, en este modelo se considera que sólamente reflejan la luz que les llega de las fuentes de luz presentes en la escena o la reflejada por otros objetos y que dichas fuentes de luz son de tipo puntual. Más aún, este método define la forma en que una superficie refleja la luz como una combinación de la reflexión difusa de superficies rugosas con la reflexión especular de superficies brillantes. Bui Tuong Phong observó que las superficies brillantes poseen pequeñas luces o brillos especulares y, por lo tanto, definió su modelo como la suma de tres componentes de luz: un componente ambiental, uno difuso y uno especular (ver Figura 3.16).

La iluminación ambiental es aquella que no proviene de una dirección concreta, sino que incide sobre todas las partes del objeto por igual y está dada por

$$c_a = k_a i_a^{in},$$

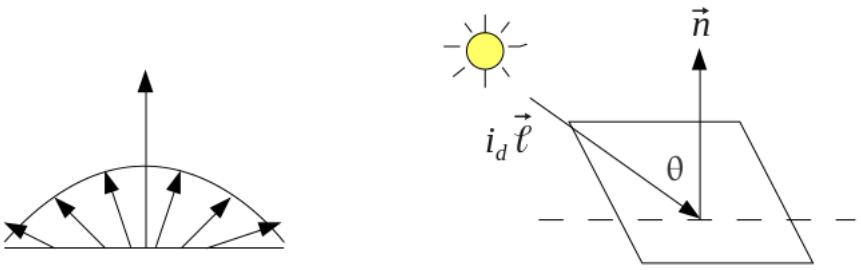
donde k_a es un coeficiente de reflexión ambiental, i_a^{in} que es el valor de intensidad de luz ambiental que incide en el objeto; el cual es el mismo en todo punto de la escena, y c_a es la intensidad reflejada. Este tipo de iluminación evita que aquellas zonas que no posean una fuente directa de luz se vean completamente negras y produce una superficie visualmente plana.

La iluminación difusa considera que los objetos no tienen iluminación uniforme en

toda su superficie; es decir, considera que la luz proviene de una dirección particular y es reflejada en todas direcciones (ver Figura 3.13(a)) pero afecta sólo a aquellas partes del objeto en las que incide. A las superficies que reflejan este tipo de luz se les conoce también como superficies Lambertianas debido a que la intensidad de la reflexión varía dependiendo del ángulo de incidencia (ver Figura 3.13(b)), pero es independiente del punto de vista del observador. Este tipo de iluminación se modela

$$c_d = k_d i_d \cos\theta = k_d i_d \left\langle \vec{\ell}, \vec{n} \right\rangle_+ ,$$

donde k_d es un coeficiente de reflexión difusa (propiedad física del tipo de material), i_d es el valor de intensidad de luz que incide en el objeto, $\vec{n}$ es el vector normal a la superficie en el punto $\bar{p}$, θ es el ángulo entre $\vec{n}$ y el vector representando la luz incidente $i_d\vec{\ell}$, ver Figura 3.14. Cabe hacer notar que este tipo de iluminación produce superficies opacas. También cabe mencionar que para objetos opacos la contribución sólo es válida para $\left\langle \vec{\ell}, \vec{n} \right\rangle \geq 0$, ya que de lo contrario significa que las caras están ocultas. Esta condición se indica con la notación $\left\langle \vec{\ell}, \vec{n} \right\rangle_+$.



(a) (b)

Figura 3.13: Iluminación difusa.

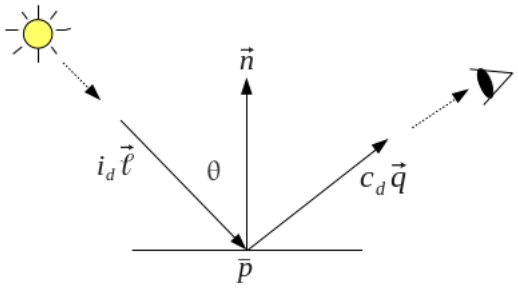


Figura 3.14: Iluminación difusa.

Por otra parte la iluminación especular procede de una dirección particular y se refleja en una dirección única. Este tipo de iluminación (efecto) se observa generalmente en superficies brillantes (como los metales pulidos) y ocasiona un brillo en la superficie. Como la iluminación difusa, afecta solamente las partes del objeto sobre las que incide y depende, además del ángulo de incidencia, del punto de vista del observador. Esta iluminación se modela como sigue

$$c_e = k_e i_e \cos\varphi = k_e i_e \left(\langle \vec{q}, \vec{r} \rangle_+\right)^\alpha,$$

donde $\bar{r} = 2\left\langle \vec{\ell}, \vec{n} \right\rangle \vec{n} - \vec{\ell}$ y k_e es el coeficiente de reflexión especular, i_e es la intencidad de la luz que incide en el objeto, $\vec{n}$ es el vector normal a la superficie en el punto $\bar{p}$, θ es el ángulo entre $\vec{n}$ y el rayo de luz incidente $i_e\vec{\ell}$, $\vec{r}$ es el vector de reflexión simétrico a $\vec{\ell}$, $\vec{q}$ es el vector de dirección del observador y α es un índice que indica que tan brillante es un objeto, ver Figura 3.15. Esta iluminación produce los puntos luminosos (brillos) que se observan en las superficies brillantes.

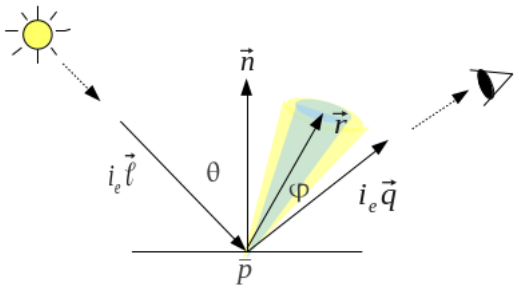


Figura 3.15: Iluminación especular.

Así, de acuerdo al modelo de iluminación de Phong, la intensidad luminosa en un punto $\bar{p}$ visto desde una determinada distancia a lo largo de una dirección $\vec{q}$ está dada por la suma de los tres componentes anteriores

$$\begin{aligned} c = & \quad c_a + c_e + c_d \\ = & \quad k_a i_a + k_d i_d \left\langle \vec{\ell}, \vec{n} \right\rangle_+ + k_e i_e \left\langle \vec{q}, \vec{r} \right\rangle_+^\alpha , \end{aligned}$$

Para evitar que dos superficies con orientación igual queden igualmente iluminadas se emplea un factor de atenuación. La intensidad de la luz se decrementa tanto como la distancia de la fuente de luz incremente, por ello en la ecuación anterior también se considera un factor de atenuación generalmente dado en función de la distancia d entre la fuente de luz y el objeto, típicamente el factor es $\frac{1}{d^2}$. Si consderamos además que en la escena generalmente hay más de una fuente de luz, entonces los efectos de todas ellas sobre un punto $\bar{p}$ está determinado por la suma de las intensidades calculadas de forma individual. Para L luces, el efecto final es

$$c = k_a i_a + \frac{1}{d^2} \sum_{m=1}^{L} k_d i_{d,m} \left\langle \vec{\ell}_m, \vec{n} \right\rangle_+ + k_e i_{e,m} \left(\langle \vec{q}, \vec{r}_m \rangle_+ \right)^\alpha , \tag{3.9}$$

donde $\bar{r}_m = 2 \left\langle \vec{\ell}_m, \vec{n} \right\rangle \vec{n} - \vec{\ell}_m$.

Cabe mencionar que al calcular la ecuación anterior para un punto $\bar{p}$ se deben considerar situaciones como el caso en el que éste punto y la fuente de luz se encuentran en puntos opuestos de la superficie, por lo cual no tiene caso realizar el cálculo de iluminación para ese punto con respecto a esa fuente de luz. Otro caso es cuando el factor de atenuación no representa la distancia, sino se trata se simular la atmósfera entre el objeto y el observador.

La ecuación (3.9) permite obtener un sólo valor de intensidad dado un punto, considerando superficies en tonos de grises; sin embargo, la luz en realidad consta de componentes de color de los cuales depende el color final de los puntos en la escena. Existen varios modelos de color pero considerando el modelo rojo, verde y azul, conocido también como RGB (por las palabras en inglés *red, green blue*); entonces,

la ecuación (3.9) se puede representar de la siguiente forma

$$c_\lambda = k_{a,\lambda} i_{a,\lambda} + \frac{1}{d^2} \sum_{m=1}^{L} k_{d,\lambda} i_{d,m,\lambda} \left\langle \vec{\ell}_m, \vec{n} \right\rangle_+ + k_{e\lambda} i_{e,m,\lambda} \left(\langle \vec{q}, \vec{r}_m \rangle_+ \right)^\alpha , \tag{3.10}$$

para $\lambda = \{r, g, b\}$ lo que produce un vector de color o posición en el espacio de color R,G,B.En la Figura 3.16 se muestra la iluminación para un objeto con cada uno de los componentes anteriores que dan como resultado la iluminación de Phong.

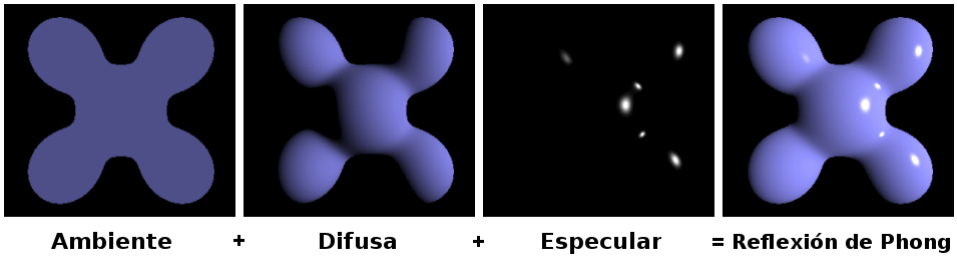


Figura 3.16: Ilustración del modelo de iluminación de Phong [45], mostrando los componentes de iluminación ambiental, difusa, especular y el efecto de la suma de todos ellos.

3.3.2. Sombreado

El sombreado consiste en la asignación de intensidades a cada punto de los polígonos que forman la malla del objeto. Se puede iluminar un objeto calculando para cada punto visible su normal y evaluando la ecuación de algún modelo de iluminación, como el dado por (3.10), sin embargo esto resulta bastante costoso, por lo que se han desarrollado técnicas que permitan hacer este proceso menos costoso.

El algoritmo más rápido de sombreado es el plano y simplemente sombrea todos los píxeles sobre algún triángulo dado, tomando en cuenta solamente un valor de iluminación; es decir, calcula un único valor de intensidad para cada polígono el cual se asigna a todos sus puntos internos. Esta técnica produce visualizaciones con cambios súbitos en el color de los polígonos debido a que se usa la misma normal para todo el polígono, la cual puede variar de forma drástica con respecto a las normales

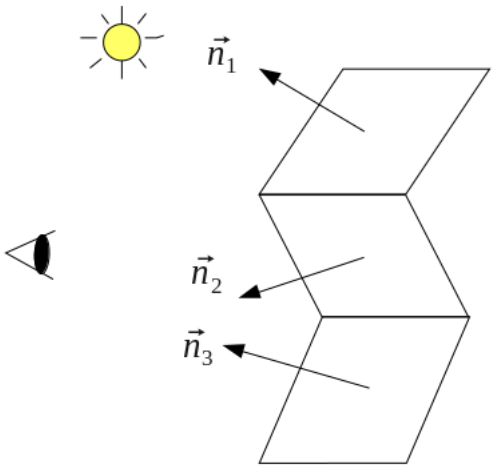


Figura 3.17: Sombreado Plano.

de los polígonos vecinos (ver Figura 3.17); por lo que solamente permite suavizar la superficie si se subdivide en triángulos más pequeños.

Un sombreado más suave; es decir, tratando de eliminar las discontinuidades que se producen con la técnica anterior, se puede conseguir con el sombreado de Gouraud. Con esta técnica se calculan las normales en cada vértice del polígono usando una descripción analítica, si se tiene, o estimándolas a partir de las normales a las caras comunes. Después de obtener las normales se hace el cálculo del modelo de iluminación (por ejemplo, con la ecuación 3.10) para obtener el color en esos vértices. A partir de esos colores se usa una técnica de interpolación para obtener el color de los puntos interiores del polígono a sombrear.

El algoritmo que proporciona resultados más suaves aunque es más lento que los anteriores es el llamado sombreado de Phong [41]. Funciona mejor que los mencionados anteriormente cuando se aplica un modelo de iluminación con un componente especular como el de iluminación de Phong. A diferencia del sombreado de Gouroud donde se interpolan los colores en el polígono, en el sombreado de Phong se interpola el vector normal a la superficie en cada píxel. La normal entonces es evaluada en el modelo de reflexión o iluminación de Phong para obtener finalmente el color en cada píxel.Este modelo de sombreado es implementado usando píxel o *fragment shaders*, pues por defecto las tarjetas gráficas no lo implementan.

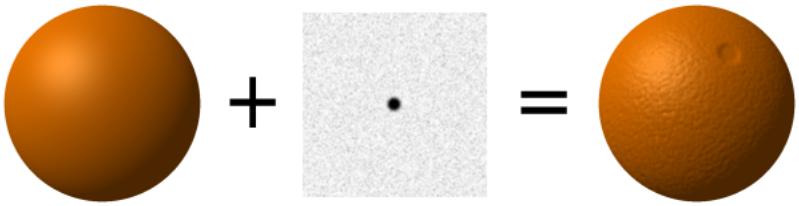


Figura 3.18: Una malla mas un mapa de alturas producen el *Bump Mapping*.

3.3.3. Mapeo de Relieves (*Bump Mapping*)

Con lo visto anteriormente podemos observar que para visualizar la malla poligonal de forma suave dependemos del modelo de iluminación y en particular de la normal. En graficación por computadora existe una técnica llamada *bump mapping* [7] que podemos traducir como mapeo de relieves; la cual, aprovecha la importancia de las normales en el modelo de iluminación y las modifica para dar un efecto de relieve a la supercie de los objetos sin modificar su geometría ni topología. Sus resultados pueden ser bastante cercanos a texturas reales.

Esta técnica consiste básicamente en tener un mapa de alturas a partir del cual se perturban los vectores normales a la superficie del objeto. Modificar los vértices a partir del mapa de alturas es un proceso algo costoso por lo que ésta técnica trata de simular este proceso calculando solamente el vector normal que tendrían los vértices perturbados. Las normales encontradas son posteriormente asignadas a los vértices correspondientes, los cuales permanecen intactos. La modificación de las normales junto con el modelo de iluminación permiten crear una apariencia diferente de los vértices sin haber modificado su posición, ver Figura 3.18.

En resumen, en *Marching Cubes* las normales son calculadas utilizando los vértices de los triángulos generados, típicamente a través del producto vectorial. Por otro lado, en el algortimo original de Artzy las normales son asignadas por cuadrilatero tomando en cuenta a sus cuatro vecinos arista y son iguales sobre todos los píxeles que representan un polígono. Ésta es la principal razón por la que las imágenes producidas a partir del algoritmo de Artzy tengan apariencia más cuboide que las producidas por *Marching Cubes*.

Una forma de lograr que la superficie obtenida por el algortimo de Artzy sea suave podría ser aumentanto el número de cuadriláteros ya sea por medio de sobremuestreo de las imágenes tridimensionales o de los polígonos de la malla; sin embargo, esto sería a costa de un mayor uso de recursos computacionales. La otra opción es proveer normales "suaves" sobre los vértices de las caras siguiendo una idea similar a la propuesta por Blinn con el *Bump Mapping*.

Con lo visto en el Capítulo 2, al hacer la reconstrucción por medio de ART se tiene la función ν que es contínua y representa todo el volumen, por lo que se pueden obtener las normales en cualquier punto y debido al uso de las funciones blob que son suaves, entonces estas normales también lo son. Por lo tanto, el principal objetivo es calcular las normales a partir de los datos obtenidos de la reconstrucción y colocarlos en los vértices de las caras dadas por el algortimo de Artzy, de forma que se intente generar una visualización más suave. En el siguiente capítulo se describe la metodología que se siguió para obtener dichas normales.

3.4. Normales de una Combinación lineal de *Blobs*

Para una función g, la normal en cualquiera de sus puntos se calcula como ∇g y el vector normal normalizado $-\frac{\nabla g}{\|\nabla g\|}$. Por lo tanto, nos interesa calcular el gradiente de (2.4). Debido a que ART con *blobs* aproxima ν como una combinación lineal de funciones base continuas, entonces esta aproximación es también contínua. Por lo tanto, se puede obtener el gradiente en cualquier punto de dicha función. Para la combinación lineal de *blobs* podemos calcular el gradiente de la siguiente forma

$$\nabla \nu \left(\bar{x} \right) \approx \sum_{j=1}^{J} c_j \nabla b_j \left(\| \bar{x} - \bar{p_j} \| \right). \tag{3.11}$$

Para calcular $\nabla b(r)$ podemos expresar la función $b\left(r\right)$ radialmente simétrica como $b(r) = b(r(\bar{x}))$ para $r(\bar{x}) = \|\bar{x}\|$. Usando la regla de la cadena tenemos que $\nabla b(r) =$

$\frac{\partial}{\partial r}b(r(\bar{x}))\nabla r(\bar{x})$. Es fácil ver que

$$\nabla r(\bar{x}) = \frac{\bar{x}}{\|\bar{x}\|}.$$

En [33] se define $\frac{\partial}{\partial r}b$ como sigue

$$\frac{\partial b}{\partial r}(m, a, \alpha; r) = \begin{cases} \left[\frac{-1/\alpha}{\alpha^{m-2}I_m(\alpha)}\right](r/a)z^{m-1}I_{m-1}(z), & \text{si } 0 \leq r \leq a, \\ 0, & \text{en otro caso,} \end{cases} \tag{3.12}$$

donde $z = \sqrt{\alpha[1-(r/a)^2]}$. Por lo tanto, el gradiente de un blob se define como

$$\nabla b(m, a, \alpha; \|\bar{x} - \bar{p}\|) = \frac{\partial b}{\partial r}(m, a, \alpha; \|\bar{x} - \bar{p}\|)\frac{(\bar{x} - \bar{p})}{\|\bar{x} - \bar{p}\|}.$$

Así, el gradiente de la función 2.4 como se describe en [33] se obtiene por medio de la siguiente ecuación:

$$\nabla \nu(\bar{x}) \approx \sum_{j=1}^{J} c_j \frac{\partial b}{\partial r}\frac{\partial b}{\partial r}(m, a, \alpha; \|\bar{x} - \bar{p_j}\|)\frac{(\bar{x} - \bar{p}_j)}{\|\bar{x} - \bar{p}_j\|}, \tag{3.13}$$

lo anterior quiere decir que como las funciones *blob* son radialmente simétricas, entonces la dirección del gradiente de las funciones base centradas en $\bar{p}_j$ es la misma que la dirección de $(\bar{x} - \bar{p}_j)$.

Capítulo 4

Metodología propuesta para el Cálculo de Normales

Como se vió en el Capítulo 3, en la implementación original del Algoritmo de Artzy las normales de la superficie $\mathcal{A}_\tau$ se obtienen en cada cara a partir de las caras adyacentes. Esto significa que cuando se aproxima la función original f por medio de un algoritmo como ART, se desaprovecha información que proviene del método de reconstrucción y que es útil para el cálculo de las normales; las cuales, como se vió en la sección 3.3.1, son importantes para la visualización.

En el Capítulo 2 también se vió que el propósito de usar ART con las funciones *blob* es producir reconstrucciones más suaves; es decir, en las que se reduzca en la medida de lo posible transiciones bruscas ocasionadas ya sea durante la adquisición de los datos o aquellas que se pueden producir por el propio método de reconstrucción. A pesar de ello, durante el proceso de visualización del volumen, los algoritmos de rastreo de superficie vistos anteriormente pierden un poco esta característica de suavidad que se había logrado durante la reconstrucción. Por lo anterior, en el presente capítulo se describe el proceso propuesto que permite asignar normales a los vértices de la malla $\mathcal{A}_\tau$, a partir de los datos obtenidos de la reconstrucción con ART.

Para obtener las normales a partir de una reconstrucción hecha por ART es necesario obtener primero las proyecciones en paralelo de funciones de densidad que posteriormente sean procesadas por una implementación de ART con *blobs*. Dicha implementación permite obtener el conjunto $\{c_j\}$ que junto con los parámetros de los

blobs definidos en la sección 2.6, son empleados para evaluar la función (2.4). Lo anterior permite obtener la aproximación de la función ν que debe ser muestreada usando la ecuación (3.3). Para muestrear la aproximación de la función (2.4) empleamos el procedimiento que se describe a continuación.

4.1. Discretización del volumen obtenido por ART

Como vimos antes, la discretización de un volumen se lleva a cabo por medio de la ecuación (3.3). Para el caso del volumen obtenido por ART que aproxima la función 2.4 tenemos

$$V_{G_\Delta} = \nu\left(\bar{x}\right) \times \text{III}_{G_\Delta} = \sum_{j=1}^{J} c_j b_j\left(\bar{x} - \bar{p}_j\right) \times \text{III}_{G_\Delta}.$$

Recordando que un tren de pulsos consiste en deltas de Dirac trasladadas a posiciones definidas por $\Delta\bar{k}$, donde $\bar{k} \in \mathbb{Z}^3$; entonces, para una muestra tenemos que

$$V\left(\bar{k}\right) = \partial_{\Delta\bar{k}}\nu = \nu\left(\Delta\bar{k}\right) = \sum_{j=1}^{J} c_j b_j\left(\Delta\bar{k} - \bar{p}_j\right).$$

Claramente este proceso es poco eficiente computacionalmente, por lo que hemos utilizado el siguiente método.

Para obtener V_{G_Δ} usaremos el hecho de que la función ν se obtiene por medio de ponderar la función base $b\left(m, a, \alpha; r\right)$ por el conjunto de coeficientes $\{c_j\}$ y centrada en los puntos $\{\bar{p}_j\}$. Para cada punto $\bar{p} \in \{\bar{p}_j\}$ obtenemos el subconjunto de G_Δ como sigue

$$K_j = \left\{\bar{k} \,\middle|\, \bar{k} \in G_\Delta \text{ y } \left\|\Delta\bar{k} - \bar{p}_j\right\| \leq a\right\}, \tag{4.1}$$

donde a es el radio del *blob* b. Para cada punto $\bar{k} \in K_j$ calculamos

$$V_{G_\Delta}\left(\bar{k}\right) \leftarrow V_{G_\Delta}\left(\bar{k}\right) + c_j b_j\left(m, a, \alpha; \left\|\Delta\bar{k} - \bar{p}_j\right\|\right), \tag{4.2}$$

este proceso se ilustra en la Figura 4.1 y lo presentamos como algoritmo a continuación.

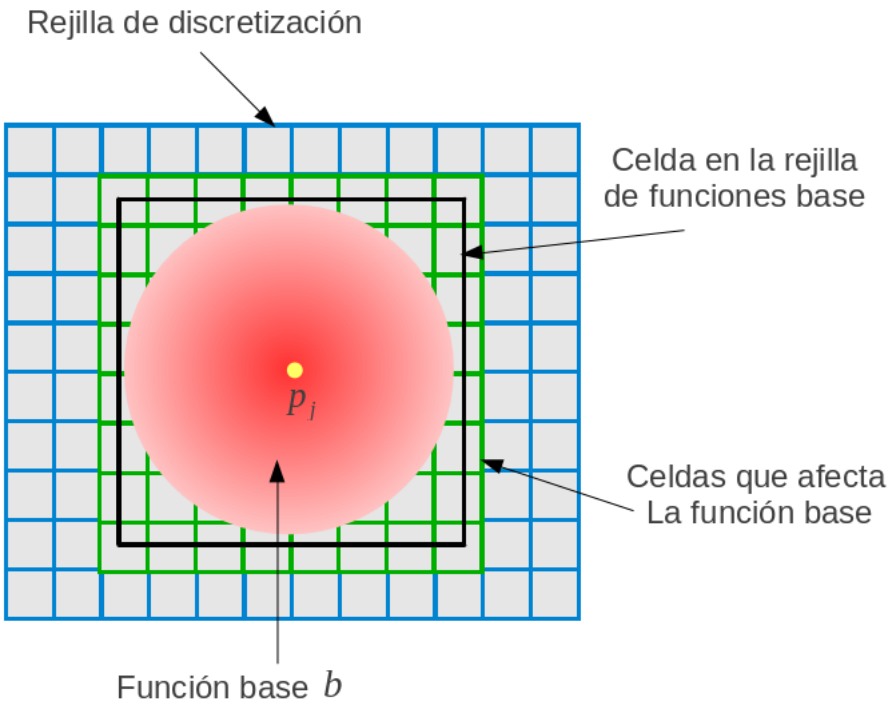


Figura 4.1: Proceso de discretización para el punto p_i y el coeficiente c_j.

Algoritmo 2 Proceso de discretización.

Entradas: Conjunto $\{p_j\}$, conjunto de coeficientes $\{c_j\}$, función $b(m, a, \alpha; r)$ y G_Δ.
Salidas: $\mathrm{V_{G.}}$.

1. **for** $\bar{k} \in G_\Delta$ **do**
2. $V_{G_\Delta}\left(\bar{k}\right) \leftarrow 0$
3. **end for**
4. **for** $j \leftarrow 1$ **to** J **do**
5. **if** $c_j \neq 0$ **then**
6. obtener K_j usando (4.1)
7. **for all** $\bar{k} \in K_j$ **do**
8. calcular ecuación *4.2*
9. **end for**
10. **end if**

4.2. Adquisición de la malla poligonal y cálculo de las normales

Como podemos observar del proceso de discretización propuesto en el Algoritmo 2, si al momento de calcular la ecuación (4.2) en el paso 8, determinamos si los vértices del vóxel $\bar{k}$ pertenecen a $\mathcal{A}_\tau$, entonces es posible calcular el vector normal correspondiente a cada vértice.

Como vimos en el Capítulo 3, podemos referenciar un vóxel como los centros de las celdas de discretización, es decir, $\bar{k} \in V_{G_\triangle}$. Entonces, para cada vóxel $\bar{k} \in V_{G_\Delta}$ podemos definir Q_k como el conjunto de vértices $\{q_i\}$ como se muestra en la Figura

4.2.

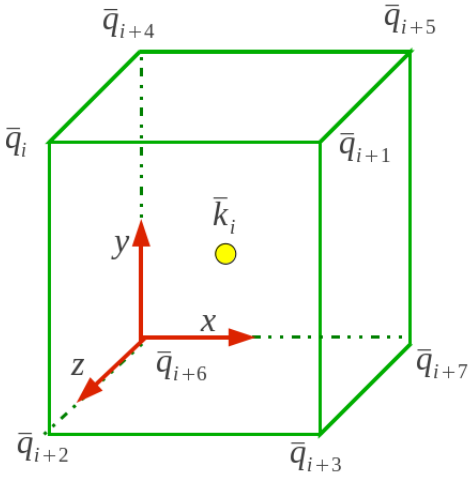


Figura 4.2: Esquinas para el vóxel $\bar{k}_i$.

Para cada uno de los puntos $\{q_i\}$ se puede calcular el gradiente $\nabla V(\bar{q}_i)$. Entonces el conjunto de normales $\mathcal{N}_{\mathcal{A}_\tau}$ para los vértices de $\mathcal{A}_\tau$ se obtiene al normalizar ∇V:

$$\mathcal{N}_{\mathcal{A}_\tau} = \frac{\nabla V(\bar{x})}{\|\nabla V(\bar{x})\|} \tag{4.3}$$

Para obtener ∇V debemos obtener para cada $\bar{q}_i$

$$\nabla V(\bar{q}_i) \leftarrow \nabla V(\bar{q}_i) + c_j \frac{\partial b}{\partial r}(m, \alpha, a; \|\bar{q}_i - \bar{p}_j\|) \frac{(\bar{q}_i - \bar{p}_j)}{\|\bar{q}_i - \bar{p}_j\|}, \tag{4.4}$$

y, por lo tanto, podemos obtener la normal $n_q \in \mathcal{N}_{\mathcal{A}_\tau}$ en el vértice $\bar{q}_i$, ver Figura 4.3.

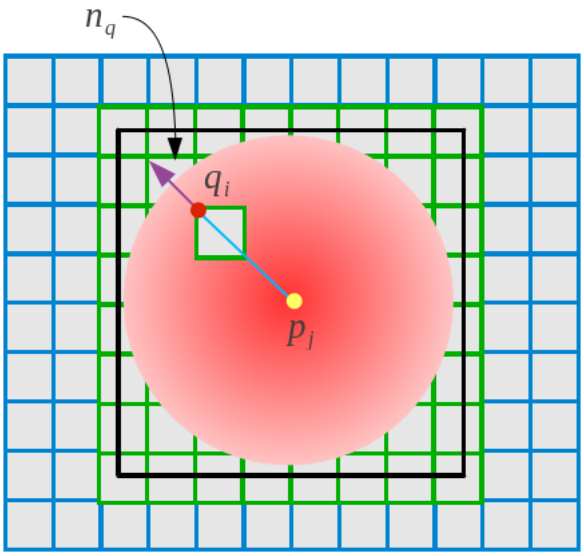


Figura 4.3: Normal n_q para el vértice q_i.

Proponemos la siguiente modificación al algoritmo 2, de forma que se obtengan las normales $\mathcal{N}_\tau$ para la malla $\mathcal{A}_\tau$.

Algoritmo 3 Cálculo de las normales para vértices de la malla obtenida por el Algoritmo de Artzy obtenidos a partir de los datos de una reconstrucción usando ART..

Entradas: Conjunto $\{p_j\}$, conjunto$\{c_j\}$, función $b(m, a, \alpha; r)$, función $\nabla b(m, a, \alpha; r)$, $\mathcal{A}_\tau$ y G_Δ.

Salidas: $\mathcal{N}_{\mathcal{A}_\tau}$

1. **for** $\bar{k} \in G_\Delta$ **do**
2. $V_{G_\Delta}\left(\bar{k}\right) \leftarrow 0$
3. **end for**
4. **for** $j \leftarrow 1$ **to** J **do**
5. **if** $c_j \neq 0$ **then**
6. obtener K_j usando (4.1) y determinar Q_k
7. **for all** $\bar{k} \in K_j$ **do**
8. calcular ecuación *4.2*
9. calcular ecuación *4.4*
10. **end for**
11. **end if**
12. **end for**
13. **for all** $\nabla V(\bar{q}_i) \in N_{A_\tau}$
14. calcular $\bar{n}_q = \frac{\nabla V(\bar{q}_i)}{\|\nabla V(\bar{q}_i)\|}$
15. **end for**

A continuación se presentan una serie de experimentos que se realizaron usando la metodología descrita anteriormente.

4.3. Experimentos

Para probar lo propuesto en el Algoritmo 3 realizamos algunos experimentos usando datos artificiales y datos reales. Para todos ellos se requirieron de las proyecciones en paralelo de las funciones de densidad a reconstruir. Dichas proyecciones fueron procesadas por la implementación de ART con *blobs* de la biblioteca Xmipp [47], la cual permite obtener el conjunto de coeficientes $\{c_j\}$. A partir de dichos coeficientes podemos evaluar la función (2.4) y posteriormente muestrearla usando la ecuación (3.3). Es importante mencionar que para calcular la función (2.4) de todos los experimentos, usamos la rejilla B_Δ con $\Delta = \frac{1}{\sqrt{2}}$ para los puntos $\{p_j\}$ y los siguientes parámetros de los *blobs*: $\alpha = 13.36$, $a = 2.4$, $m = 2$, los cuales se definieron en la sección 2.6. La visualización de las mallas en todos los experimentos se realizó por medio de OpenGL [40], el cual implementa por defecto el modelo de iluminación de Blinn–Phong [6] que es una modificación propuesta por Blinn, al modelo de iluminación de Phong [41]. Para el sombreado, OpenGL implementa el de Gouraud [21].

4.3.1. Maniquís (Phantoms)

En la primera parte de estos experimentos se crearon dos maniquís o *phantoms* empleando el formato de archivo con extensión ".descr" de la biblioteca Xmipp [47] y a partir de los cuales se generó su correspondiente reconstrucción con ART empleando también funciones de esta bliblioteca.

Una vez obtenidas las funciones base que representan a la función ν, es decir el conjunto de coeficientes $\{c_j\}$ y el de puntos$\{p_j\}$, se empleó el conjunto de programas que desarrollamos a partir del algoritmo 3 y que permiten muestrear la función a diferentes tamaños y obtener las normales $\mathcal{N}_\tau$ de los vértices de las mallas del Algoritmo de Artzy$\mathcal{A}_\tau$.

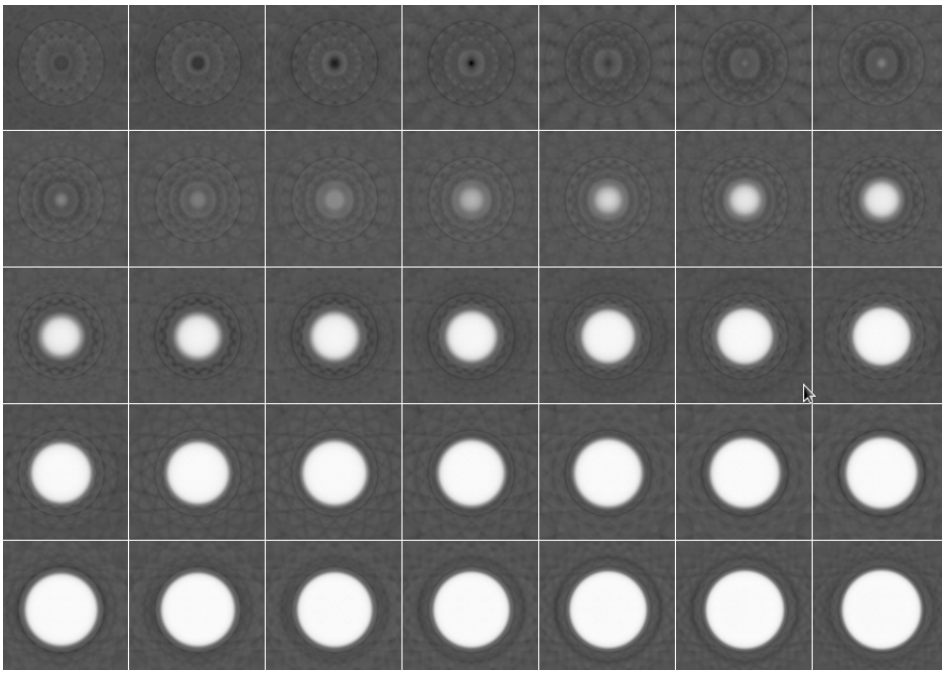
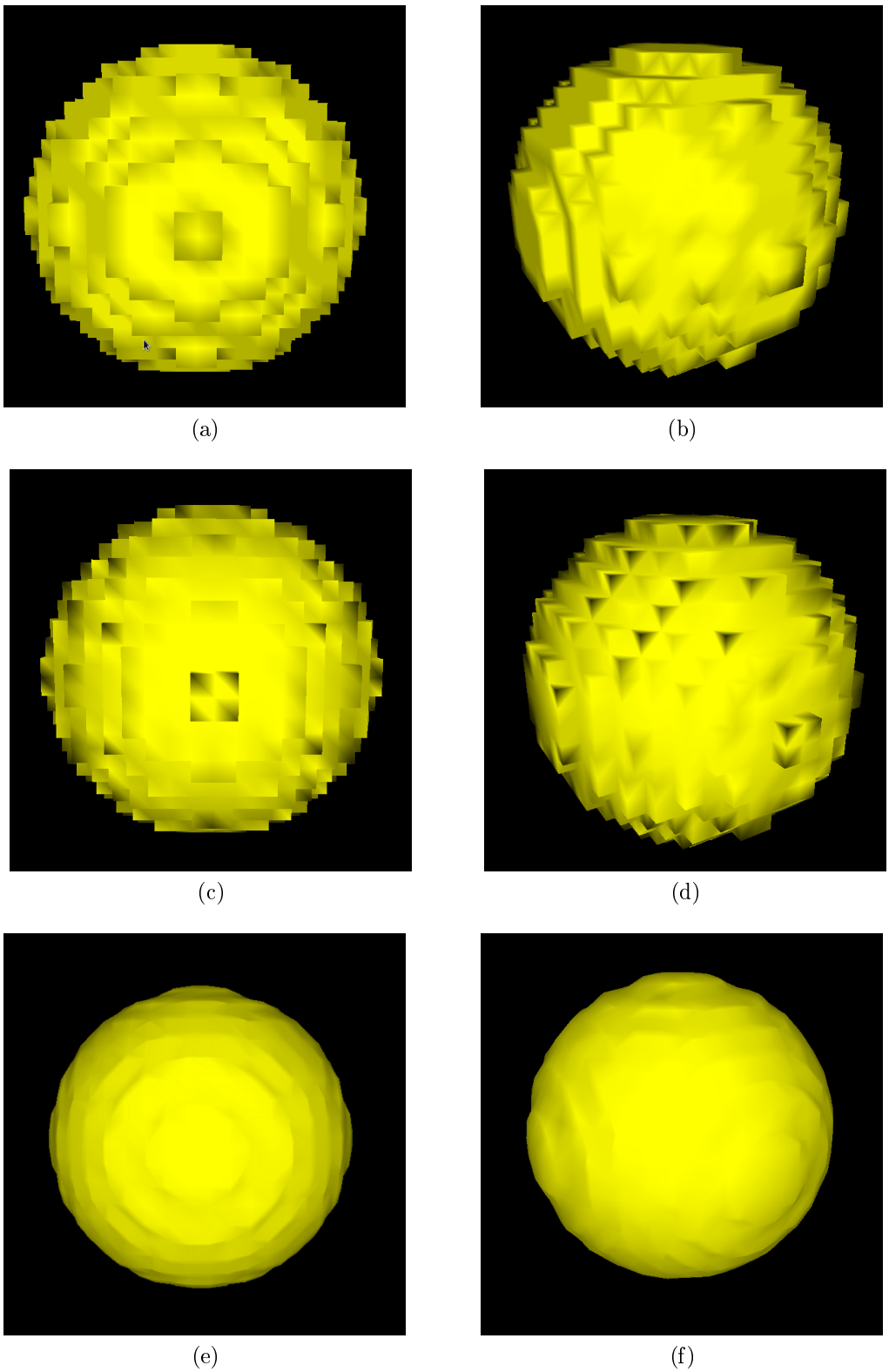


Figura 4.4: Algunas rebanadas de las proyecciones de una esfera vistas con la biblioteca Xmipp.

El maniquí más simple generado se trató de una esfera con radio de 40 unidades posicionada en el centro de un espacio de 128 unidades. La Figura 4.4 presenta algunas rebanadas de las imágenes de las proyecciones de este maniquí, vistas con el comando xmipp_show de la bliblioteca Xmipp. Debido a que todos los maniquís presentados en este trabajo son simétricos, en todos los casos sólamente mostramos algunas rebanadas de la primera mitad.

A partir de las las proyecciones mostradas previamente se procedió a la reconstrucción de la esfera que posteriormente fue muestreada con $\Delta_\nu = 30$. Para realizar el rastreo empleando el Algoritmo de Artzy el umbral o isovalor empleado fue de $\tau = 0.340$. En las Figuras 4.5 (a) y (b) se muestra la malla obtenida por el Algoritmo de Artzy original; mientras que las Figuras 4.5 (c) y (d), muestran el resultado obtenido con el método propuesto y finalmente la Figuras (e) y (f) muestran el resultado obtenido por *Marching Cubes*.

Como se puede observar en las Figuras 4.5 (c) y (d), algunas esquinas de la malla calculada con el método propuesto se encuentran obscuras. Podemos observar que en



(a) (b) (c) (d) (e) (f)

Figura 4.5: Muestreo de la esfera con $\triangle_v = 30$. Las Figuras (a) y (b) muestran la malla obtenida por el Algoritmo de Artzty original, (c) y (d) muestran el resultado de aplicar las normales con el método propuesto y (e) y (y) muestran el resultado de la malla obtenida con *Marching Cubes* .

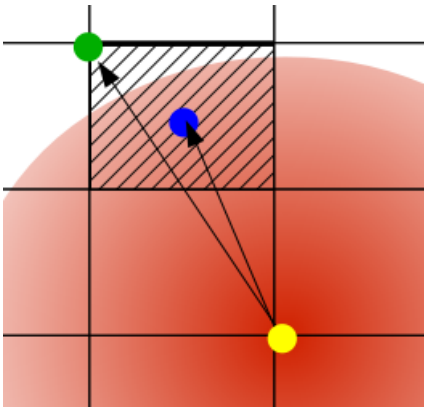


Figura 4.6: Ejemplo de vértice con normal cero. La superficie pasa sobre el centro del vóxel pero no sobre todas sus esquinas, lo que puede producir normales con valor cero.

general son las que se encuentran más al exterior de la malla. Al observar esos detalles comprobamos que los valores de la normal en esos puntos son cero, lo que ocasiona esa apariencia obscura. Esto nos dió un indicio de que es posible que esto ocurra cuando algunos de los vértices de las caras obtenidas por el Algoritmo de Artzy se encuentren fuera de donde pasa realmente la malla. Por ejemplo, en la Figura 4.6, se muestra un caso en el que a pesar de que la distancia del centro del *blob* al centro del vóxel (punto azul) es menor que el radio del *blob*, la distancia de la esquina superior izquierda (punto verde) es mayor que el radio del *blob*. Esto produce que si ningún *blob* pasa por la esquina del vóxel, entonces la normal en ese punto no tendrá valor asignado, lo que, debido a la implementación ocasiona que ésta valgacero.

A continuación se presentan los resultados obtenidos con otros maniquís generados con Xmipp y colocados en un volumen total de 256 unidades.

La Figura 4.7 presenta las imágenes de algunas rebanadas de las proyecciones del maniquí 2, mientras que las figuras (4.8) y (4.9) muestra los resultados una vez hecha la reconstrucción y discretizando a $\triangle_v = 100$ y $\triangle_v = 256$ respectivamente. En ambos casos el umbral $\tau = 0.105$ fue empleado para el rastreo con el Algoritmo de Artzy. Cada figura (a), (b) y (c) muestra la malla obtenida empleando el Algoritmo de Artzy, el método propuesto y Marching Cubes respectivamente.

En la Figura 4.10 se presentan algunas rebanadas correspondientes al maniquí 3,

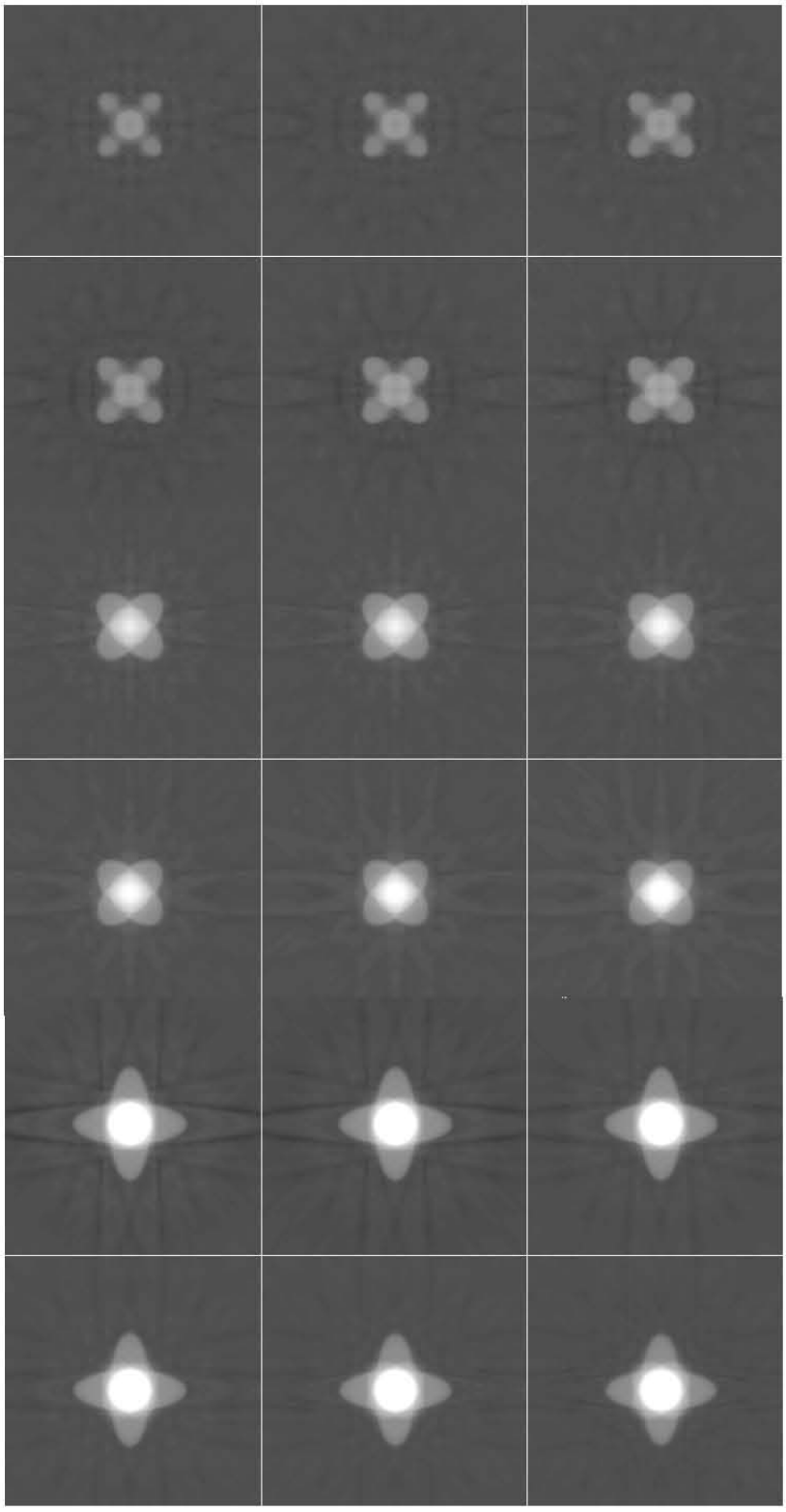
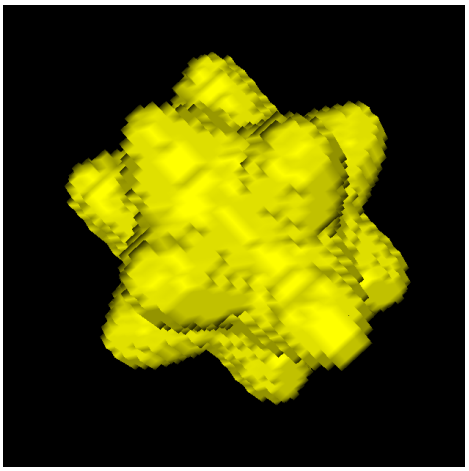
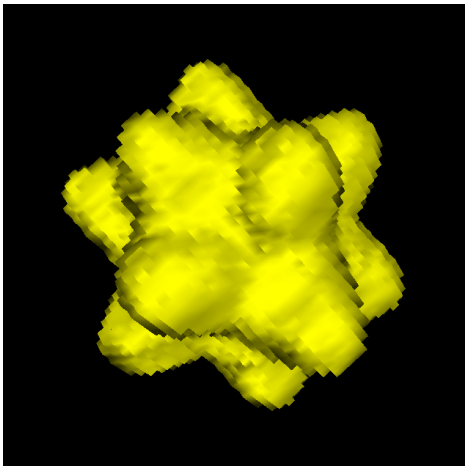


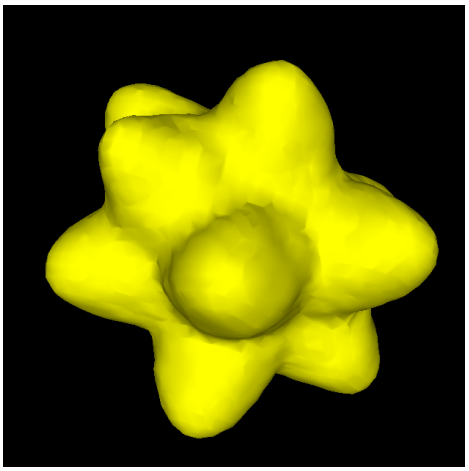
Figura 4.7: Algunas rebanadas de las proyecciones del maniquí 2 vistas con la biblioteca Xmipp.



(a)

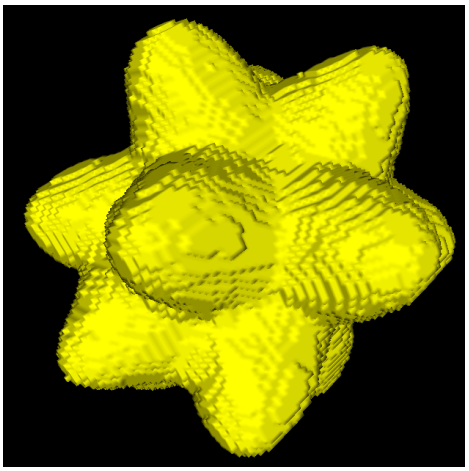


(b)

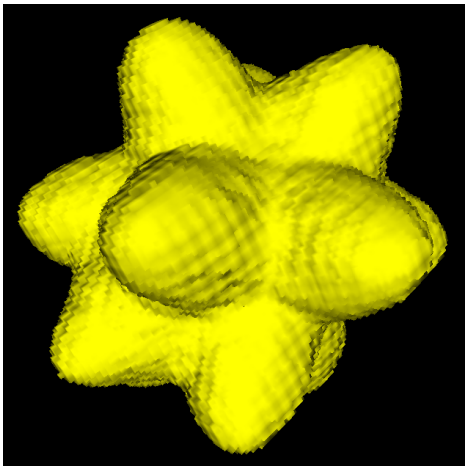


(c)

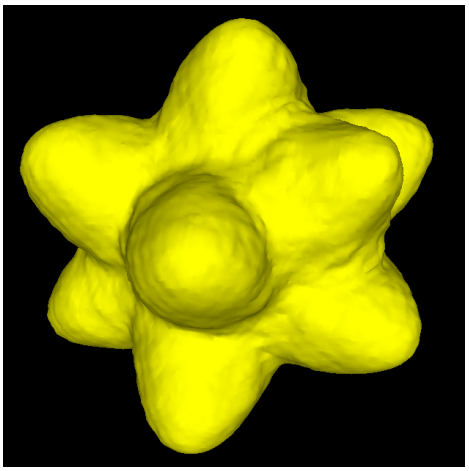
Figura 4.8: Resultados del maniquí muestreado con $\triangle_v = 100$. La Figura (a) muestra la malla obtenida por el Algoritmo de Artzty original, (b) muestra el resultado de aplicar las normales con el método propuesto y (c) muestra el resultado de la malla obtenida con *Marching Cubes*.



(a)



(b)



(c)

Figura 4.9: Resultados del maniquí 2 muestreado con $\triangle_v = 256$. La Figura (a) muestra la malla obtenida por el Algoritmo de Artzty original, (b) muestra el resultado de aplicar las normales con el método propuesto y (c) muestra el resultado de la malla obtenida con *Marching Cubes*.

mientras que la figura (4.11) muestra los resultados una vez hecha la reconstrucción y discretizando a $\triangle_v = 100$. El umbral $\tau = 0.266$ fue empleado para el rastreo con el Algoritmo de Artzy.

4.3.2. Datos reales

A continuación se presentan los resultados de los tres algoritmos aplicados a un conjunto de imágenes TEM [35] obtenidas con el fin de estudiar la proteína compleja DnaB-DnaC. La Figura 4.12 presenta las imágenes de algunas rebanadas de las proyecciones de la molécula. Para estos datos las discretizaciones fueron de $\triangle_v = 100$ y $\triangle_v = 256$ y $\triangle_v = 320$, Los resultados se muestran en las figuras (4.13) y (4.14) respectivamente. En todos los casos el umbral empleado para el rastreo con el Algoritmo de Artzy fue de $\tau = 0.0112$. Como en las imágenes anteriores, cada figura muestra la malla obtenida empleando el Algoritmo de Artzy, el método propuesto y Marching Cubes respectivamente.

De las imágenes previamente presentadas podemos observar que en algunas secciones (particularmente al interior de las imágenes) se logra una mejora visual con respecto a la malla obtenida con el Algoritmo de Artzy, sin embargo en las esquinas no se observa esta mejora debido al problema mencionado anteriormente, donde puede suceder que la superficie de la malla no atraviese la esquina pero sí el centro del vóxel. Comparando los resultados obtenidos a partir de los maniquís con los reales, se puede observar que el método propuesto logró una superficie más suave. Si observamos bien las imágenes de las proyecciones, podemos notar que las que corresponden a la molécula DnaB-DnaC poseen ruido, mientras que las de los maniquís no. Por lo tanto, en los datos reales es menos probable que el gradiente en los vértices de los vóxeles sea cero y en consecuencia es menos probable que hayan normales iguales a cero. A pesar de ello también es fácil observar que la visualización de la malla obtenida con *Marching Cubes* sigue siendo visiblemente más suave que las otras. Sin embargo, este trabajo se da un indicio de que realizando ciertas mejoras al método propuesto se pueden obtener visualizaciones aún más suaves de la superficie obtenida por el Al-

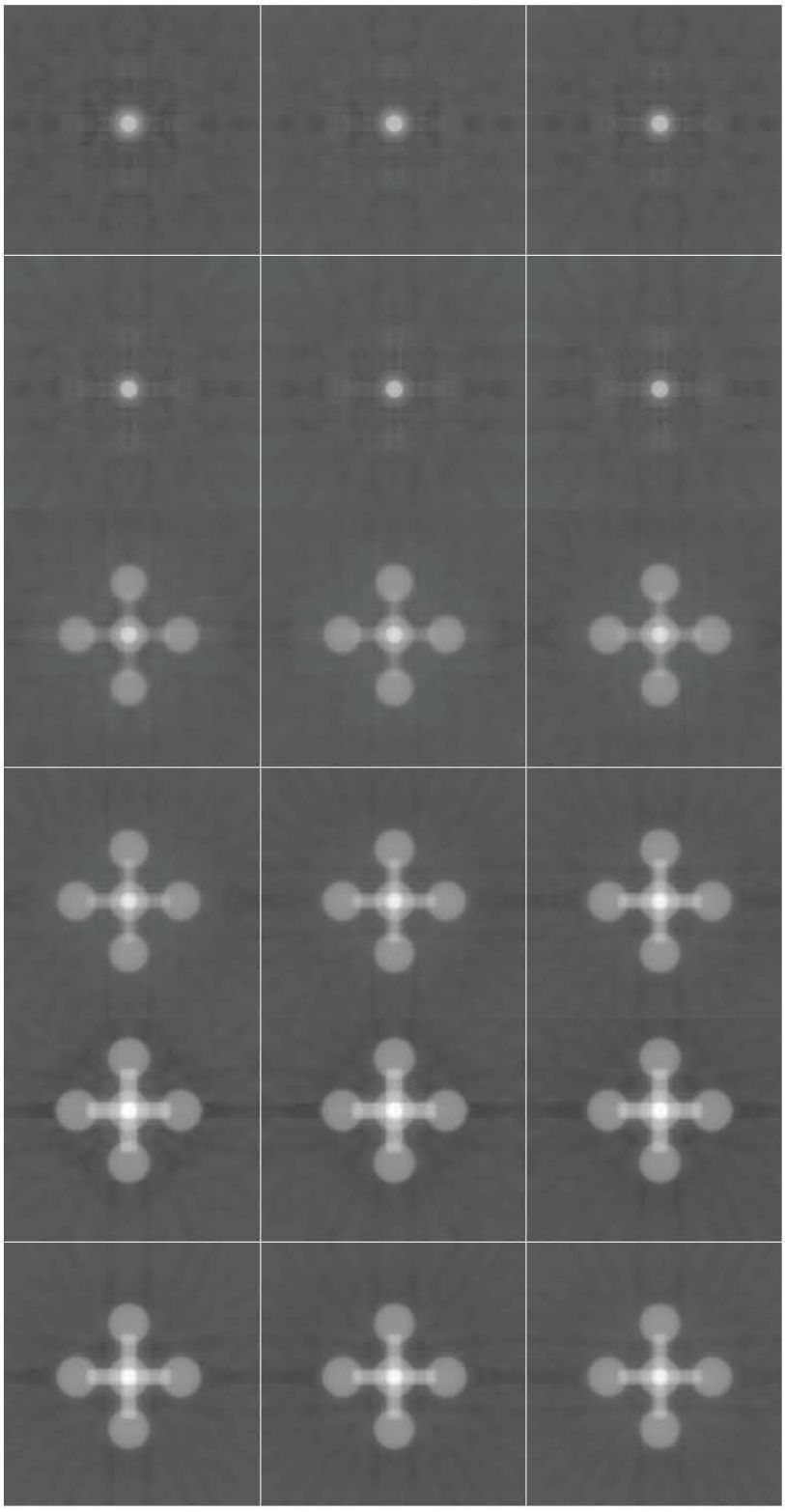
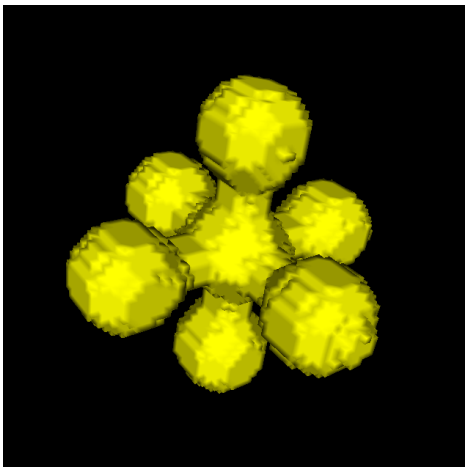
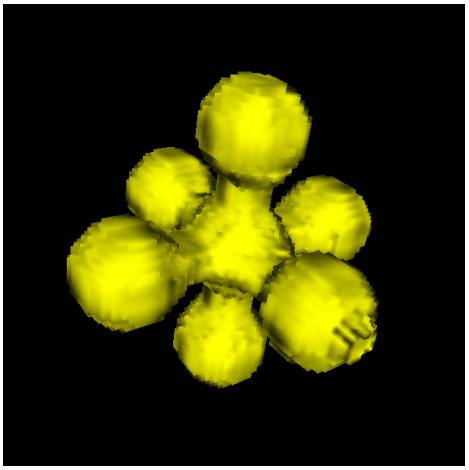


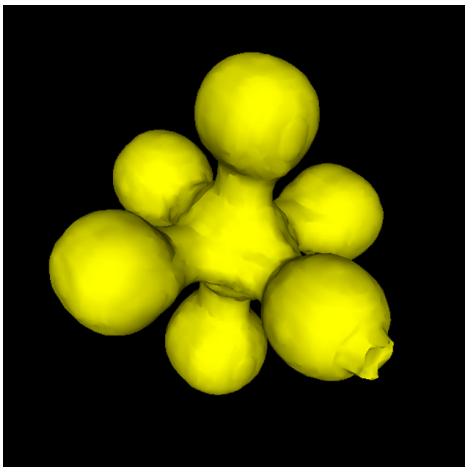
Figura 4.10: Algunas rebanadas de las proyecciones del maniquí 3 vistas con la biblioteca Xmipp.



(a)



(b)



(c)

Figura 4.11: Resultados del maniquí 3 muestreado con $\triangle_v = 100$. La Figura (a) muestra la malla obtenida por el Algoritmo de Artzty original, (b) muestra el resultado de aplicar las normales con el método propuesto y (c) muestra el resultado de la malla obtenida con *Marching Cubes*.

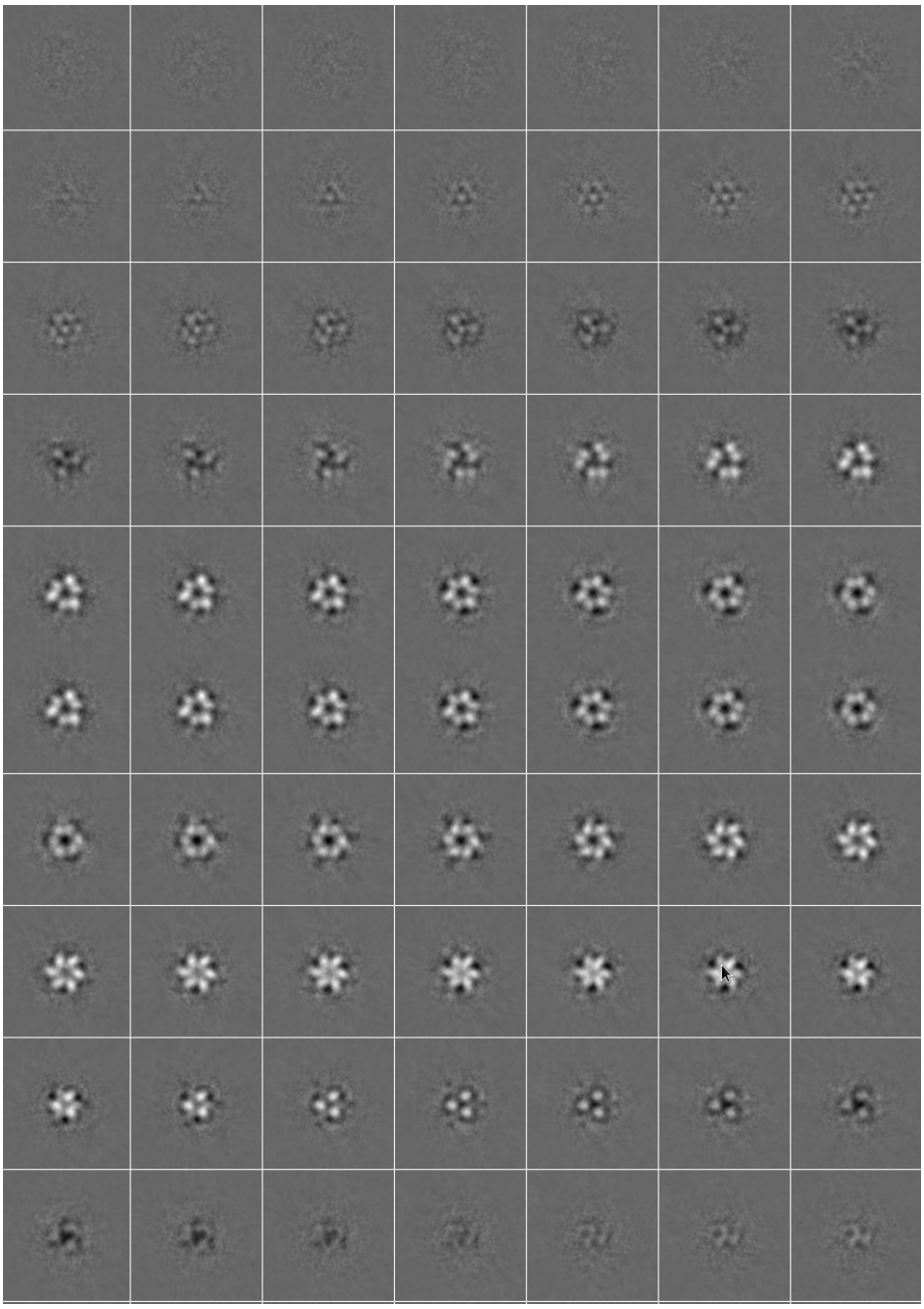
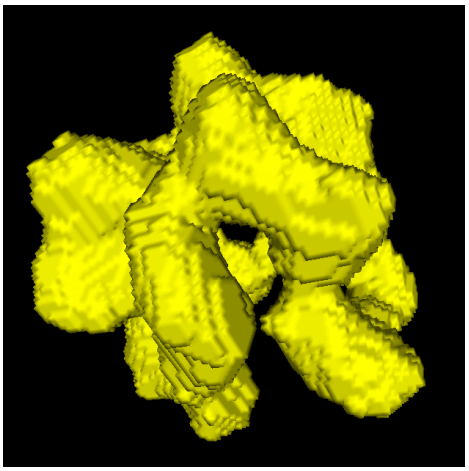
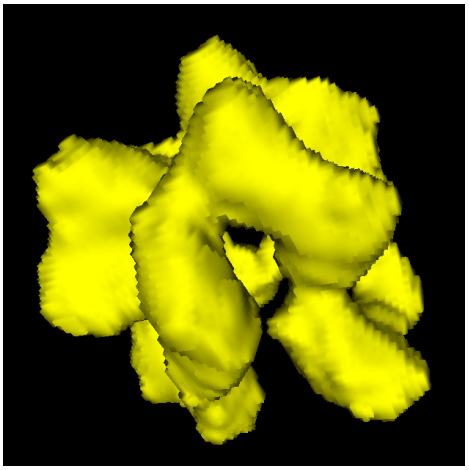


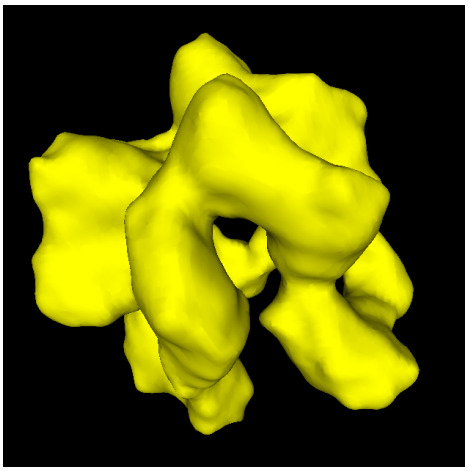
Figura 4.12: Algunas rebanadas de las proyecciones de la molécula DnaB-DnaC vistas con la biblioteca Xmipp.



(a)

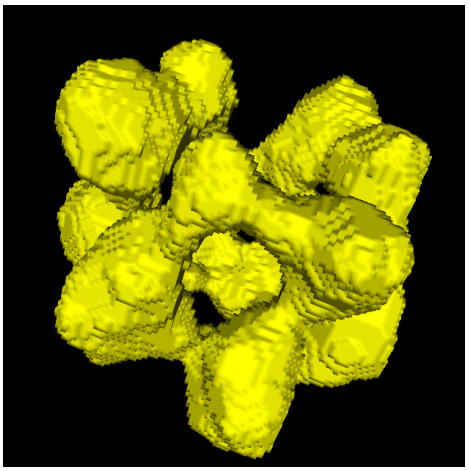


(b)

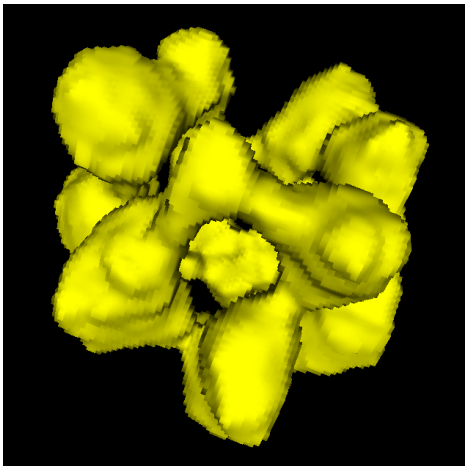


(c)

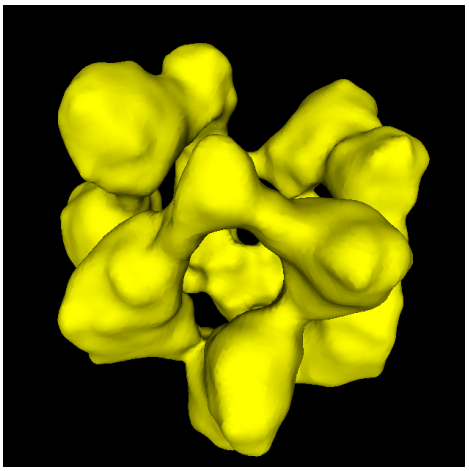
Figura 4.13: Resultados de la molécula DnaB-DnaC muestreada con $\triangle_v = 256$. La Figura (a) muestra la malla obtenida por el Algoritmo de Artzty original, (b) muestra el resultado de aplicar las normales con el método propuesto y (c) muestra el resultado de la malla obtenida con *Marching Cubes*.



(a)



(b)



(c)

Figura 4.14: Resultados de la molécula DnaB-DnaC muestreada con $\triangle_v = 320$. La Figura (a) muestra la malla obtenida por el Algoritmo de Artzty original, (b) muestra el resultado de aplicar las normales con el método propuesto y (c) muestra el resultado de la malla obtenida con *Marching Cubes*.

goritmo de Artzy. Finalmente cabe mencionar que, como se espera, la visualización mejora conforme discretización aumenta, es decir, el valor de $\triangle_v$ es más pequeño, sin embargo uno de los objetivo es lograr visualizaciones suaves sin incrementar demasiado el tamaño de la discretización, por lo que más adelante se proponen algunas mejoras al método propuesto.

4.3.3. Conclusiones

La idea original planteada por la técnica de *bump mapping* nos dió un indicio de que modificando solamente las normales de una malla y dependiendo del modelo de iluminación, se puede modificar la apariencia de un objeto sin modificar su geometría ni topología. Esta idea fue aplicada a mallas obtenidas por medio del Algoritmo de Artzy a las cuales se le asignaron normales diferentes con el propósito de suavizar la visualización de las mallas obtenidas con este algoritmo. Debido a que se emplearon modelos que provienen de reconstrucciones hechas por medio de ART, se utilizó información proveniente de dicha reconstrucción para proponer un método que permita asignar normales a las mallas obtenidas con el Algoritmo de Artzy. Los resultados obtenidos con el método propuesto indican que la información proveniente de la reconstrucción es útil para mejorar la apariencia de la superficies reconstruidas; permitiendo así, aprovechar las ventajas de obtener la superficie por medio del Algoritmo de Artzy. A pesar de que en los resultados visuales conseguidos hasta ahora, el algoritmo *Marching Cubes* consigue una superficie más suave, los experimentos realizados indican que es posible mejorar aún más la apariencia de las mallas del Algoritmo de Artzy empleando los datos de la reconstrucción.

4.3.4. Trabajo Futuro

La propuesta del cálculo de normales presentada en este trabajo, resultó ser un primer acercamiento a la mejora visual de las mallas obtenidas por el Algoritmo de Artzy. Como se indicó previamente, es posible realizar varias mejoras con el fin de lograr ese objetivo. Implementar una técnica que permita asignar normales a puntos

que se encuentren fuera de la superficie ocupada por los *blobs* por ejemplo, por medio de una interpolación, es fundamental para lograr una superficie más suave y mejorar los resultados obtenidos en este trabajo. Como se describió en el Capítulo 3, la normal juega un papel muy importante a la hora de visualizar la superficie y los resultados son visiblemente afectados por el modelo de iluminación y la técnica de sombreado que se utilice, por lo que la elección de ambas técnicas resultan de gran importancia para la visualización. Entonces, implementar una mejor técnica de sombreado como la propuesta por Phong, contribuirá también a la mejora visual de las mallas. El uso de shaders que implementan actualmente las tarjetas gráficas, permite modificar la información que se tiene de los píxeles y por lo tanto, se pueden hacer cambios en sus normales. Debido a eso, una implementación por medio de tarjetas gráficas o GPUs para el cálculo de las normales, permitiría que la idea propuesta en este trabajo resulte más eficiente.

Todos los experimentos presentados en este trabajo, se realizaron empleando la rejilla *sc* para la discretización, sin embargo, como se ha indicado en otros trabajos [18], la rejilla *bcc* podría proporcionar mejores resultados. Si en vez de usar la rejilla *sc*, se colocan los vóxeles obtenidos a partir del muestreo en una rejilla *bcc,* entonces las caras obtenidas por el Algoritmo de Artzy serán caras pertenecientes a un octahedro truncado (el vóxel definido para una rejilla *bcc)* por lo que, según lo presentado en [18], se obtendría una visualización más suave.

Bibliografía

[1] Y. an Xi y Y. Duan, "A region-growing based iso-surface extraction algorithm," in *Computer-Aided Design and Computer Graphics, 2007 10th IEEE International Conference on*, pp. 120 –125, oct. 2007.

[2] E. Artzy, G. Frieder, y G. T. Herman, "The theory, design, implementation and evaluation of a three-dimensional surface detection algorithm," *SIGGRAPH Computer Graphics*, vol. 14, no. 3, pp. 2–9, 1980.

[3] M. S. Avinash C. Kak, *Principles of Computerized Tomographic Imaging*. IEEE Press, 1987.

[4] Bethesda, ed., *Collection development manual of the National Library of Medicine*. National Institutes of Health, Health & Human Services, 4 ed., 2004.

[5] J. S. Blakemore, *Solid State Physics*. Cambridge University Press, 2nd ed., 1985.

[6] J. F. Blinn, "Models of light reflection for computer synthesized pictures," *SIGGRAPH Computer Graphics*, vol. 11, pp. 192–198, July 1977.

[7] J. F. Blinn, "Simulation of wrinkled surfaces," *Proceedings SIGGRAPH*, vol. 12 (3), pp. 286–292, August 1978.

[8] J. F. Blinn, "A generalization of algebraic surface drawing," *ACM Transactions on Graphics*, vol. 1, pp. 235–256, July 1982.

[9] J.-D. Boissonnat, "Shape reconstruction from planar cross sections," *Computer Vision, Graphics, and Image Processing*, vol. 44, pp. 1–29, August 1988.

[10] R. Bracewell, *The Fourier transform and its applications*. electrical and electronic engineering series, McGraw-Hill, 3 ed., 1999.

[11] T. A. D. Bruce H. McCormick y M. D. Brown, "Visualization in scientific computing," *ACM Press*, 1987.

[12] S. R. Buss, *3-D Computer Graphics. A mathematical introduction with OpenGL*. New York: Cambridge, 1 ed., 2003.

[13] E. Candes, J. Romberg, y T. Tao, "Robust uncertainty principles: Exact signal reconstruction from highly incomplete frequency information," *IEEE Transactions on Information Theory*, vol. 52, no. 2, p. 489, 2006.

[14] D. P. Petersen y D. Middleton, "Sampling and reconstruction of wave-numberlimited functions in N-dimensional euclidean spaces," *Information and Control*, vol. 5, pp. 279 – 323, 1962.

[15] J. D. Foley, A. van Dam, S. K. Feiner, y J. F. Hughes, *Computer graphics: principles and practice (2nd ed.)*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1990.

[16] H. Fuchs, Z. M. Kedem, y S. P. Uselton, "Optimal surface reconstruction from planar contours," *Communications of the ACM*, vol. 20, pp. 693–702, October 1977.

[17] E. Garduño, G. T. Herman, y R. Davidi, "Reconstruction from a few projections by ℓ_1-minimization of the Haar transform," *Inverse Problems*, vol. 27, no. 5, p. 055006, 2011.

[18] E. Garduño, *Visualization and extraction of structural components from reconstructed volumes*. PhD thesis, Faculties of the University of Pennsylvania, 2007.

[19] E. Garduño y G. T. Herman, "Optimization of basis functions for both reconstruction and visualization," *Discrete Applied Mathematics*, vol. 139, no. 1-3, pp. 95–111, 2004.

[20] R. Gordon y G. Herman, "Three-dimensional reconstruction from projections: A review of algorithms.," *International Review of Cytology*, vol. 38, pp. 111–151, 1974.

[21] H. Gouraud, "Continuous shading of curved surfaces," *IEEE Transactions on Computers*, vol. 20, pp. 623–629, June 1971.

[22] M. Hadwiger, J. M. Kniss, C. Rezk-salama, D. Weiskopf, y K. Engel, *Real-time Volume Graphics*. Natick, MA, USA: A. K. Peters, Ltd., 2006.

[23] H. Hagen, A. Ebert, R. H. van Lengen, y G. Scheuermann, "Scientific visualization: methods and applications," in *Proceedings of the 19th spring conference on Computer graphics*, SCCG '03, (New York, NY, USA), pp. 23–33, ACM, 2003.

[24] H. Hagen, H. Müller, y G. M. Nielson, eds., *Focus on Scientific Visualization*, (London, UK), Springer-Verlag, 1993.

[25] R. Hartley y A. Zisserman, *Multiple View Geometry in computer vision*. Cambridge University Press, 2a ed., 2003.

[26] G. T. Herman, *Geometry of Digital Spaces*. Birkhäuser, 1998.

[27] G. T. Herman, *Fundamentals of Computerized Tomography. Image Reconstruction from Projections*. Advances in Computer Vision and Pattern Recognition, Academic Press, 1980, 2 ed., October 2010.

[28] A. Hilton, A. Stoddart, J. Illingworth, y T. Windeatt, "Marching triangles: range image fusion for complex object modelling," in *Proceedings of the International Conference on Image Processing*, vol. 1, pp. 381 –384 vol.2, sep 1996.

[29] A. Iriarte, C. O. S. Sorzano, J. M. Carazo, J. L. Rubio, y R. Marabini, "A theoretical model for EM-ML reconstruction algorithms applied to rotating PET scanners," *Physics in Medicine and Biology*, vol. 54, no. 7, p. 1909, 2009.

[30] F. John, "The ultrahyperbolic differential equation with four independent variables," *Duke Mathematical Journal*, vol. 4, pp. 300–322, 1938.

[31] S. Kaczmarz, "Angenäherte auflösung von systemen linearer gleichungen," *Bulletin de l'Académie Polonaise des Sciences et Lettres*, vol. A35, pp. 355–357, 1937.

[32] E. Keppel, "Approximating complex surfaces by triangulation of contour lines," *IBM Journal of Research and Development*, vol. 19, pp. 2–11, January 1975.

[33] R. M. Lewitt, "Multidimensional digital image representations using generalized Kaiser-Bessel window functions," *Journal of the Optical Society of America*, vol. A7, pp. 1834–1846, 1990.

[34] W. E. Lorensen y H. E. Cline, "Marching cubes: A high resolution 3D surface construction algorithm," *Computer Graphics*, vol. 21, pp. 163–169, July 1987.

[35] M. Bárcena, T. Ruiz, L.E. Donate, S.E. Brown, N. E. Dixon, M. Radermacher, y J.M. Carazo, "The DnaB-DnaC complex: A structure based on dimert assembled around an occluded channel," *European Molecular Biology Organization Journal*, vol. 20, pp. 1462–1468, 2001.

[36] S. Matej y R. M. Lewitt, "Efficient 3D grids for image-reconstruction using spherically-symmetrical volume elements," *IEEE Transactions on Nuclear Science*, vol. 42, pp. 1361–1370, 1995.

[37] H. Miyakawa, "Sampling theorem of stationary stochastic variables in multidimensional space," *Journal of the Institute of Electronic and Communication Engineers of Japan*, vol. 42, pp. 421–âĂŞ427, 1959.

[38] L. Mroz, A. König, y E. Gröller, "Maximum intensity projection at warp speed," *Computers & Graphics*, vol. 24, no. 3, pp. 343 – 352, 2000.

[39] T. S. Newman y H. Yi, "A survey of the marching cubes algorithm," *Computers & Graphics*, vol. 30, no. 5, pp. 854 – 879, 2006.

[40] Opengl, D. Shreiner, M. Woo, J. Neider, y T. Davis, *OpenGL(R) Programming Guide : The Official Guide to Learning OpenGL(R), Version 2 (5th Edition)*. Addison-Wesley Professional, Aug. 2005.

[41] B. T. Phong, "Illumination for computer generated pictures," *Communications of the ACM*, vol. 18, pp. 311–317, June 1975.

[42] J. Radon, "Über die Bestimmung von Funktionen durch ihre Integralwerte längs gewisser Mannigfaltigkeiten (the radon transform and some of its applications)," *Journal of Mathematical Physics*, vol. 69, pp. 262–277, April 1917.

[43] E. Rodríguez, *Computer Graphic Artist*. Global Media, 2007.

[44] S. D. Roth, "Ray casting for modeling solids," *Computer Graphics and Image Processing*, vol. 18, no. 2, pp. 109 – 144, 1982.

[45] B. Smith, "Image of Phong components," August 7 2006.

[46] C. O. S. Sorzano, R. Marabini, G. T. Herman, Y. Censor, y J. M. Carazo, "Transfer function restoration in 3d electron microscopy via iterative data refinement," *Physics in Medicine and Biology*, vol. 49, no. 4, p. 509, 2004.

[47] C. Sorzano, R. Marabini, J. V. Muriel, J. B. Castro, S. Scheres, J. Carazo, y A. P. Montano, "Xmipp: A new generation of an open-source image processing package for electron microscopy," *Journal of Structural Biology*, vol. 142(2), pp. 194–204, 2004.

[48] Q. Zhang, R. Eagleson, y T. Peters, "Rapid voxel classification methodology for interactive 3D medical image visualization," in *Medical Image Computing and Computer-Assisted Intervention - MICCAI 2007* (N. Ayache, S. Ourselin, y A. Maeder, eds.), vol. 4792 of *Lecture Notes in Computer Science*, pp. 86–93, Springer Berlin / Heidelberg, 2007.