Extracción de información sobre el manguito rotador a partir de resonancias magnéticas usando técnicas determinísticas y aleatorias

Fabio Andrés Gómez De Los Ríos
Estadístico, M.Sc.(c)

Universidad Nacional de Colombia
Facultad de Ciencias
Escuela de Estadística
Medellín
Febrero 2014

Extracción de información sobre el manguito rotador a partir de resonancias magnéticas usando técnicas determinísticas y aleatorias

Fabio Andrés Gómez De Los Ríos
Estadístico, M.Sc.(c)

Disertación presentada para optar al título de
Magister en Ciencias - Estadística

Director
Marco Paluszny Kluczynsky, Ph.D.
Profesor Titular

Universidad Nacional de Colombia
Facultad de Ciencias
Escuela de Estadística
Medellín
Febrero 2014

Título en español

Extracción de información sobre el manguito rotador a partir de resonancias magnéticas usando técnicas determinísticas y aleatorias.

Title in English

Extracting information about the rotator cuff from magnetic resonance images using deterministic and random techniques.

Resumen: En este trabajo se consideran metodologías para extraer información de imágenes por resonancia del tejido en el manguito rotador; el estudio tiene como objetivo definir un método de visualización alternativo que facilite la detección de rupturas parciales en el tendón supraespinoso. Específicamente, vamos a usar familias de parches triangulares elipsoidales para cubrir la cabeza del húmero cerca de la zona afectada, estos parches se texturizaran y desplegaran con la información de la resonancia usando la técnica de interpolación trilineal. En la generación de puntos para texturizar cada parche se propone una nueva metodología que garantiza la distribución aleatoria sobre la superficie y cuyo costo computacional es significativamente menor en comparación a metodologías alternativas de naturaleza tanto determinística como aleatoria.

Abstract: In this work, we consider some methods to extract information about the rotator cuff based on magnetic resonance images; the study aims to define an alternative method of display that might facilitate the detection of partial tears in the supraspinatus tendon. Specifically, we are going to use families of ellipsoidal triangular patches to cover the humerus head near the affected area, these patches are going to be textured and displayed with the information of the magnetic resonance images using the trilinear interpolation technique. For the generation of points to texture each patch, we propose a new method that guarantees the random distribution of its points using a random statistical method, the computational cost is significatively lower as compared with deterministic as well other standard statistical techniques.

Palabras clave: Manguito rotador, tendón supraespinoso, ruptura parcial, imágenes por resonancia magnética, extracción de textura, parches triangulares, simulación.

Keywords: Rotator cuff, supraspinatus tendon, partial thickness tear, magnetic resonance image, texturing, triangular patches, simulation.

Dedicado a

A mi madre María E. y a mi novia Mónica por su apoyo irrestricto.

Agradecimientos

Quiero agradecer especialmente al profesor Marco Paluszny por su entrega, dedicación, y entusiasmo durante la elaboración de este trabajo. Deseo agradecer también al profesor Norman Giraldo por su valiosa asesoría y disposición.

Índice general

Índice de tablas

Introducción

El manguito rotador es un conjunto de músculos y tendones conectados a la cabeza del húmero cuya función es la movilidad y estabilidad del hombro. La anatomía y fisiología del manguito rotador es compleja y está interconectada con otros grupos de músculos en el hombro. Este grupo de músculos realiza múltiples funciones y a menudo se somete a estrés durante diversas actividades. Las afecciones en el manguito rotador son las causas más comunes de dolor y disfunción en el hombro del adulto. En particular, una ruptura del manguito rotador consiste en la ruptura de uno o más de los tendones ligados a los cuatro músculos en el manguito rotator y de acuerdo a su profundidad la lesión puede ser clasificada en completa o parcial [1]. Las imágenes por resonancia magnética juegan un papel importante en la identificación de rupturas en el manguito rotador, de hecho, muchos cirujanos confian en imágenes por resonancia magnética para apoyar la toma de decisiones y la planificación prequirúrgica de pacientes con dolor en el hombro [2].

El proyecto «The Visible Human Proyect» puso a disposición de la comunidad científica una base de datos que permite la visualización de la anatomía humana. La base de datos se conforma por 1871 imágenes asociadas a cortes transversales para un hombre particular; a partir de este volumen médico el proyecto pone a disposición del usuario una herramienta de software para reconstruir rodajas oblicuas con ubicación y dirección arbitraria. El proceso de reconstrucción se realiza mediante la técnica matemáticamente bien conocida de interpolación trilineal, ver por ejemplo [3].

Un problema posiblemente más relevante en imágenes médicas consiste en desplegar información sobre superficies curvas arbitrarias que se adapten o reproduzcan aproximadamente una estructura del cuerpo humano particular, por ejemplo sobre una superficie adaptada a una arteria para determinar calcificaciones u otras malformaciones, ver [4]. El despliegue de la superficie se hace con la técnica usada para imágenes digitales de texturización de superficies, autores como [5, 6, 7] han tratado el problema de visualización plana de las superficies curvas texturizadas aplicando en medicina.

En este trabajo se exploran metodologías para extraer información de imágenes por resonancia del tejido en el manguito rotador, en particular se enfatiza en la detección de rupturas parciales en el tendón supraespinoso. Se consideran dominios sobre esferas y elipsoides, y en particular familias de parches triangulares[1] de Bézier racionales especiales para ajustar la cabeza del húmero cerca de la zona afectada. Estos parches se texturizan con la información de la resonancia usando la técnica de interpolación trilineal [9]. La generación de puntos para texturizar un parche, se puede realizar de forma determinística o de forma aleatoria; la generación de puntos de forma aleatoria tiene una ventaja en términos de visualización pues evita la acumulación sistemática de puntos en áreas específicas. En este sentido, se definen familias de parches elipsoidales especiales, se construye su respectivo algoritmo para generación de puntos de forma aleatoria [10], y se compara su eficiencia computacional con respecto a la evaluación equivalente mediante el algoritmo de de Casteljau y el algoritmo de evaluación directa con polinomios de Bernstein para parches triangulares de Bézier racionales [8, 11, 12, 13, 14, 15, 16].

Este trabajo se organiza de la siguiente manera. En los capítulos 1 y 2 se presentan conceptos básicos asociados a curvas de Bézier y parches de Bézier triangulares. En los capítulos 3 y 4 se estudian definiciones y resultados elementales sobre variables aleatorias y simulación. En el capítulo 5 se propone y se describe una técnica para generar puntos uniformemente distribuidos sobre parches elipsoidales especiales.

Por ultimo en el capítulo 6 se muestra como aplicación en medicina, un procedimiento para extraer información sobre el manguito rotador a partir de un volumen médico usando la técnica de generación propuesta, y adicionalmente se comparan a través del tiempo de cómputo promedio técnicas alternativas tanto determinísticas como aleatorias.

[1] La palabra parche triangular se refiere al hecho de que la frontera dada por la parametrización consiste en tres curvas, esto es: describe un triángulo curvo en 3D [8]

CAPÍTULO 1

Curvas de Bézier

En 1959, la compañía automotriz francesa Citroën contrató al joven matemático Paul de Faget de Casteljau quien había terminado recientemente su tesis de doctorado en la École Normale Supérieure de Paris. Él comenzó a desarrollar un sistema el cual principalmente pretendía el diseño *ab initio* de curvas y superficies en lugar de enfocarse en la reproducción de los blueprints [1] ya existentes. Él adoptó desde el principio el uso de los polinomios de Bernstein junto con el hoy conocido algoritmo de de Casteljau para la definición de sus curvas y superficies. Citroën mantuvo en secreto el trabajo de de Casteljau por largo tiempo.
Por otro lado, en la compañía automotriz francesa Renault, Pierre Bézier encabezó el departamento de diseño y también se percató de la necesidad de representar en computador piezas mecánicas. El trabajo de Bézier fue influenciado por el conocimiento de desarrollos similares en Citroën, pero él procedió de manera independiente. Sus resultados para la construcción de curvas fueron idénticos a la representación de de Casteljau, solamente que la matemática involucrada fue diferente.
El trabajo de Bézier fue ampliamente publicado y pronto llamó la atención de A.R Forrest. Él se dio cuenta que las curvas de Bézier podrían ser representadas en términos de polinomios de Bernstein, es decir, en la misma forma que de Casteljau había usado desde los años 50 [17].

Una curva de Bézier de grado n con polígono de control o polígono de Bézier $\mathbf{b}_i \in \mathbb{E}^3$ y pesos asociados $w_i \in \mathbb{R}$, es una curva paramétrica $\mathbf{b}_0^n(t) \in \mathbb{E}^3$ de parámetro $t \in [0,1]$ donde el índice i varía entre 0 y n. Tener en cuenta o no los pesos w_i, clasifica la curva de Bézier como racional o polinomial.

1.1. Curvas de Bézier polinomiales de grado n

Toda curva polinómica admite una representación mediante su polígono de Bézier. Más aún, los algoritmos más rápidos y numéricamente más estables para desplegar una curva polinómica se basan en su representación de Bézier [11].

[1] Reproducción en papel de un dibujo técnico, un plano cartográfico o un diseño de ingeniería

1.1.1. Algoritmo de de Casteljau

La principal atracción de este algoritmo es la bonita interacción entre geometría y algebra: una construcción geométrica muy intuitiva que conduce a una teoría potente [8].

En el caso polinomial, mediante el algoritmo de de Casteljau cada punto sobre la curva $\mathbf{b}_0^n(t)$ se construye mediante interpolación lineal repetida, haciendo:

$$\mathbf{b}_i^r(t) = (1-t)\,\mathbf{b}_i^{r-1}(t) + t\mathbf{b}_{i+1}^{r-1}(t), \tag{1.1}$$

donde

$$r = 1, \ldots, n; \quad i = 0, \ldots n-r; \quad \mathbf{b}_i^0 = \mathbf{b}_i.$$

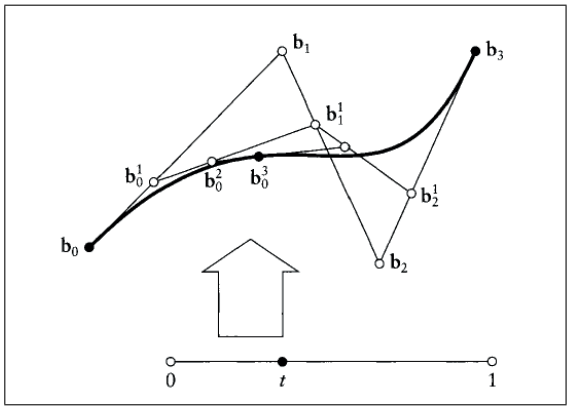


FIGURA 1.1. Representación gráfica del algoritmo de de Casteljau: interpolación lineal repetida [8].

La figura 1.1 ilustra, la construcción de puntos sobre una curva de Bézier cúbica usando el algoritmo de de Casteljau; se destacan los puntos de control intermedios.

1.1.2. Evaluación directa con polinomios de Bernstein

Los polinomios de Bernstein univariados de grado n se definen por:

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}; \quad i = 0, 1, \ldots, n; \tag{1.2}$$

donde

$$\binom{n}{i} = \frac{n!}{(n-i)!i!}.$$

Por convención $B_i^n(t) = 0$ sí $i \notin [0, n]$. Los polinomios de Bernstein son términos de la expansión binomial de $1 = (t + (1-t))^n$.

Los puntos de control intermedios $\mathbf{b}_i^r(t)$ en el algoritmo de de Casteljau se pueden expresar en términos de polinomios de Bernstein y los puntos de control directamente [8]:

$$\mathbf{b}_i^r(t) = \sum_{j=0}^{r} \mathbf{b}_{i+j} B_j^r(t); \quad i = 0, 1, \ldots, n-r. \tag{1.3}$$

En particular, tomando $r = n$ en 1.3 podemos escribir un punto sobre una curva de Bézier polinomial como:

$$\mathbf{b}_0^n(t) = \sum_{j=0}^{n} \mathbf{b}_j B_j^n(t), \tag{1.4}$$

Propiedades

Se puede mostrar que las curvas de Bézier polinomiales de grado n, satisfacen las siguientes propiedades [8]:

- **Invariancia afín:** Al aplicar un mapeo afín al polígono de control, y posteriormente evaluar la curva; se obtiene una curva equivalente a aplicar el mismo mapeo afín a la curva original.
- **Invariancia bajo transformación afín del parámetro:** A menudo es necesario pensar en el intervalo de definición de una curva de Bézier de forma más general, esto es, $u \in [a, b]$. Es común en la práctica definir $t = \dfrac{u-a}{b-a}$, para así proceder con la construcción usual.
- **Envolvente convexa:** Para $t \in [0, 1]$, la curva yace dentro de la envolvente convexa del polígono de control.
- **Interpolación de extremos:** Toda curva de Bézier interpola los extremos $\mathbf{b}_0$ y $\mathbf{b}_n$.
- **Simetría:** Si se considera una curva de Bézier para cada polígono $\mathbf{b}_0, \mathbf{b}_1, \ldots, \mathbf{b}_n$ o $\mathbf{b}_n, \mathbf{b}_{n-1}, \ldots, \mathbf{b}_0$, la apariencia gráfica será la misma; ellas solamente difieren en la orientación con la cual se recorren. Algebraicamente:

$$\sum_{j=0}^{n} \mathbf{b}_j B_j^n(t) = \sum_{j=0}^{n} \mathbf{b}_{n-j} B_j^n(1-t).$$

1.2. Curvas de Bézier racionales de grado n

Para $i = 0, 1, \ldots, n$, sean $\mathbf{b}_i \in \mathbb{R}^3$, y $w_i \in \mathbb{R}$. En particular sea $\mathbf{b}_i = (b_i^x, b_i^y, 1)$. Definiendo $\bar{\mathbf{b}}_i = w_i \mathbf{b}_i$, se considera una curva de Bézier polinomial de grado n denotada $\bar{\mathbf{b}}_0^n(t)$, con polígono de control $\bar{\mathbf{b}}_i$. De acuerdo a 1.4, cada punto sobre la curva se puede escribir como:

$$\begin{aligned}
\bar{\mathbf{b}}_0^n(t) &= \sum_{j=0}^{n} \bar{\mathbf{b}}_j(t) B_j^n(t) \\
&= \sum_{j=0}^{n} \left(w_j b_j^x, w_j b_j^y, w_j \right) B_j^n(t) \\
&= \left(\sum_{j=0}^{n} w_j b_j^x B_j^n(t), \sum_{j=0}^{n} w_j b_j^y B_j^n(t), \sum_{j=0}^{n} w_j B_j^n(t) \right), \qquad (1.5)
\end{aligned}$$

por lo tanto, es posible escribir sus coordenadas homogéneas (ver [17]) como:

$$\mathbf{b}_0^n(t) = \left(\frac{\sum_{j=0}^{n} w_j b_j^x B_j^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)}, \frac{\sum_{j=0}^{n} w_j b_j^y B_j^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)}, 1 \right). \qquad (1.6)$$

La curva $\mathbf{b}_0^n(t)$ que yace sobre plano $Z = 1$, se denomina curva de Bézier racional de grado n, y se construye como la proyección de una curva de Bézier polinomial de grado n en 3D; ver la ilustración 1.2. Las curvas de Bézier racionales en 3D se construyen como la proyección de una curva de Bézier polinomial de grado n en 4D.

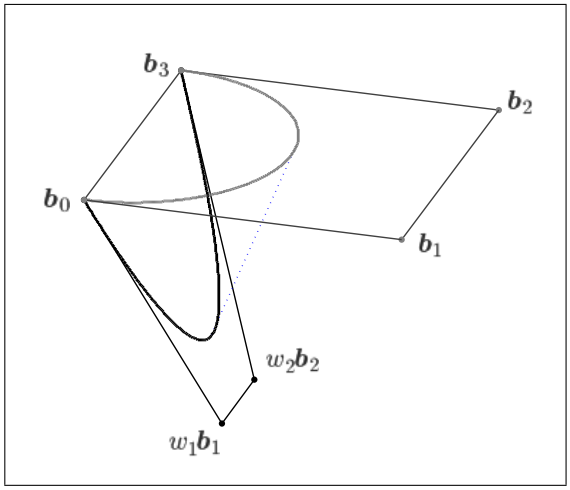


FIGURA 1.2. Representación gráfica de la proyección de una curva en 3D sobre el plano $Z = 1$.

1.2.1. Evaluación directa con polinomios de Bernstein

Replicando el procedimiento de proyectar una curva polinomial en 3D para obtener una curva racional en 2D, se puede mostrar que una curva de Bézier racional de grado n con polígono de control $\mathbf{b}_i \in \mathbb{R}^3$, y pesos asociados $w_i \in \mathbb{R}$; de acuerdo a [8] está dada por:

$$\mathbf{b}_0^n(t) = \frac{\sum_{j=0}^{n} w_j \mathbf{b}_j B_j^n(t)}{\sum_{j=0}^{n} w_j B_j^n(t)}, \tag{1.7}$$

donde

$$B_i^n(t) = \binom{n}{i} t^i (1-t)^{n-i}. \tag{1.8}$$

Esta curva es específicamente la proyección del polígono de control $[w_i\mathbf{b}_i \quad w_i]'$, de la preimagen polinomial en 4D.

Si todos lo pesos w_i son iguales a 1, se obtiene la curva de Bézier polinomial estándar. Si algunos w_i son negativos, podrían ocurrir que $\sum_{j=0}^{n} w_j B_j^n(t) = 0$ para algún $t \in [0,1]$, y por ende la curva se escaparía a infinito; por lo tanto en la práctica se considera cada w_i positivo. Las curvas de Bézier racionales gozan de todas las propiedades que sus contrapartes polinomiales poseen: invariancia afín, invariancia bajo transformación afín del parámetro, envolvente convexa, interpolación de extremos, y simetría [8].

1.2.2. Algoritmo de de Casteljau

Una curva racional de Bézier puede ser evaluada aplicando el algoritmo de de Casteljau tanto al numerador como al denominador y finalmente dividiendo. Este método aunque es simple y fácil de usar, no es numéricamente estable para pesos que varían considerablemente en magnitud. La razón de la inestabilidad es que si algunos w_i son grandes, los puntos intermedios en 3D $[w_i\mathbf{b}_i]^r$ ya no están en la envolvente convexa de los puntos de control originales; esta situación puede generar perdida de precisión. Una técnica geométrica aún más costosa consiste en proyectar cada punto de Casteljau intermedio $[w_i\mathbf{b}_i \quad w_i]'$; $\mathbf{b}_i \in \mathbb{R}^3$ al hiperplano $Z = 1$. Esto produce el algoritmo de de Casteljau racional [8]:

$$\mathbf{b}_i^r(t) = (1-t)\frac{w_i^{r-1}}{w_i^r}\mathbf{b}_i^{r-1} + t\frac{w_{i+1}^{r-1}}{w_i^r}\mathbf{b}_{i+1}^{r-1}, \tag{1.9}$$

donde

$$w_i^r(t) = (1-t)\, w_i^{r-1}(t) + t w_{i+1}^{r-1}(t), \tag{1.10}$$

y

$$r = 1, \ldots, n; \quad i = 0, \ldots n-r; \quad \mathbf{b}_i^0 = \mathbf{b}_i.$$

Una fórmula explicita para los puntos intermedios $\mathbf{b}_i^r$ está dada por

$$\mathbf{b}_i^r(t) = \frac{\sum_{j=0}^{r} w_{i+j}\mathbf{b}_{i+j}B_j^r(t)}{\sum_{j=0}^{r} w_{i+j}B_j^r(t)}. \tag{1.11}$$

En 1.11, se nota que para pesos positivos, los $\mathbf{b}_i^r$ están todos en la envolvente convexa de los $\mathbf{b}_i$ originales; esto asegura la estabilidad numérica [8].

CAPÍTULO 2

Parches triangulares de Bézier

Los parches triangulares de Bézier se consideraron por primera vez en los años 60 por Paul de Casteljau; son no solamente la generalización más natural de las curvas de Bézier, también ofrecen variedad de aplicaciones en animación, medicina, renderización y video juegos, entre otros. Estos parches tienen la capacidad de modelar superficies de topología arbitraria y son útiles para interpolar o ajustar dispersiones de puntos. Es posible producir fácilmente parches de grado bajo que yacen sobre esferas o cuádricas [8].

Un parche triangular de Bézier de grado n con red de control $\mathbf{b_i} \in \mathbb{E}^3$ y pesos asociados $w_\mathbf{i} \in \mathbb{R}$, es una superficie paramétrica $\mathbf{b_0}^n(\mathbf{u}) \in \mathbb{E}^3$. El parámetro $\mathbf{u} = (u_1, u_2, u_3)$ representa un punto en un dominio triangular de coordenadas baricéntricas[1] y el índice múltiple $\mathbf{i} = (i_1, i_2, i_3)$ se asocia con una posición particular en un arreglo triangular[2] de grado n ; por ejemplo, ver figura 2.1. Tener en cuenta o no los pesos $w_\mathbf{i}$, clasifica el parche triangular de Bézier como racional o polinomial.

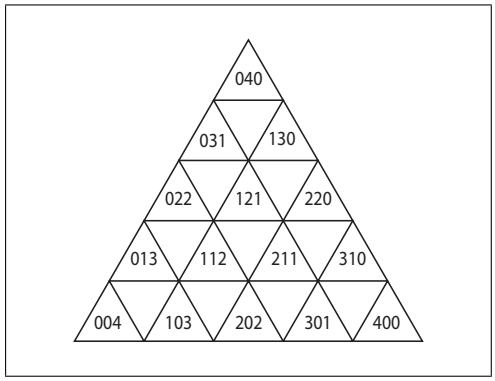


FIGURA 2.1. Representación gráfica de los índices en un arreglo triangular de grado 4.

[1]Dado un punto $t \in \mathbb{R}^2$, las coordenadas baricéntricas respecto a un triángulo de vértices $\mathbf{t}_1$, $\mathbf{t}_2$, $\mathbf{t}_3 \in \mathbb{R}^2$ son $\mathbf{u} = (u_1, u_2, u_3)$ tal que $t = u_1\mathbf{t}_1 + u_2\mathbf{t}_2 + u_3\mathbf{t}_3$.

[2]Un arreglo triangular de grado n, se define como una secuencia de objetos doblemente indexada $S_\mathbf{k}$ para $\mathbf{k} = (k_1, k_2)$, donde, $k_1 = 0, 1, \ldots n$ y $k_2 = 0, 1, \ldots k_1$. La correspondencia $\mathbf{i} = (k_2, n - k_1, k_1 - k_2)$ es una biyección; en este sentido, podemos representar el arreglo triangular de grado n en función del índice-múltiple $\mathbf{i}$.

2.1. Parches triangulares de Bézier polinomiales de grado n

2.1.1. Algoritmo de de Casteljau

En el caso polinomial, mediante el algoritmo de de Casteljau cada punto sobre el parche triangular $\mathbf{b_0}^n(\mathbf{u})$ se construye mediante interpolación lineal repetida, haciendo:

$$\mathbf{b}_{\mathbf{i}}^{r}(\mathbf{u}) = u_1 \mathbf{b}_{\mathbf{i}+\mathbf{e_1}}^{r-1}(\mathbf{u}) + u_2 \mathbf{b}_{\mathbf{i}+\mathbf{e_2}}^{r-1}(\mathbf{u}) + u_3 \mathbf{b}_{\mathbf{i}+e_3}^{r-1}(\mathbf{u}), \tag{2.1}$$

donde

$$r = 1, \ldots, n; \quad |\mathbf{i}| = n - r; \quad \mathbf{b}_{\mathbf{i}}^{0} = \mathbf{b}_{\mathbf{i}}.$$

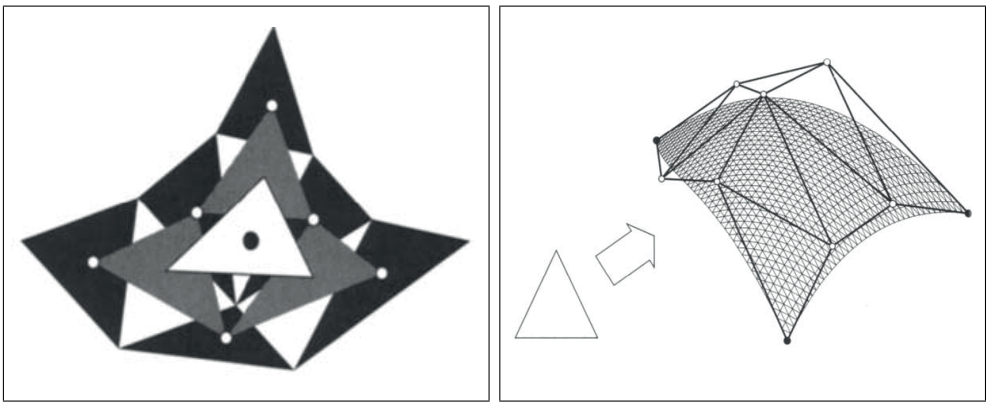


FIGURA 2.2. Algoritmo triangular de de Casteljau: interpolación lineal repetida [8].

La figura 2.2 ilustra: en el lado izquierdo la construcción de un punto sobre un triángulo de Bézier cúbico y en el lado derecho el despliegue completo destacando sus puntos de control.

2.1.2. Evaluación directa con polinomios de Bernstein

Los polinomios de Bernstein de grado n sobre un triángulo se definen por:

$$B_{\mathbf{i}}^{n}(\mathbf{u}) = \binom{n}{\mathbf{i}} u_1^{i_1} u_2^{i_2} u_3^{i_3}; \quad |\mathbf{i}| = n; \tag{2.2}$$

donde

$$\binom{n}{\mathbf{i}} = \frac{n!}{i_1! i_2! i_3!}.$$

Por convención $B_{\mathbf{i}}^{n}(\mathbf{u}) = 0$ sí para algún k, $i_k \notin [0, n]$. Los polinomios de Bernstein son términos de la expansión trinomial de $1 = (u_1 + u_2 + u_3)^n$.

Los puntos de control intermedios $\mathbf{b}_{\mathbf{i}}^{r}(\mathbf{u})$ en el algoritmo de de Casteljau se pueden expresar en términos de polinomios de Bernstein y los puntos de control directamente [8]:

$$\mathbf{b}_{\mathbf{i}}^{r}(\mathbf{u}) = \sum_{|\mathbf{j}|=r} \mathbf{b}_{\mathbf{i}+\mathbf{j}} B_{\mathbf{j}}^{r}(\mathbf{u}); \quad |\mathbf{i}| = n - r. \tag{2.3}$$

En particular, tomando $r = n$ en 2.3 podemos escribir un punto sobre un triángulo de Bézier polinomial como:

$$\mathbf{b}_{\mathbf{0}}^{n}(\mathbf{u}) = \sum_{|\mathbf{i}|=n} \mathbf{b}_{\mathbf{i}} B_{\mathbf{i}}^{n}(\mathbf{u}). \tag{2.4}$$

Propiedades

Se puede mostrar que los parches triangulares de Bézier polinomiales de grado n, satisfacen las siguientes propiedades [8]:

- **Invariancia afín:** Al aplicar un mapeo afín a la red de control, y posteriormente evaluar el parche; se obtiene una superficie equivalente a aplicar el mismo mapeo afín al parche original.
- **Invariancia bajo transformación afín del parámetro:** Un punto $\mathbf{u}$ tendrá las mismas coordenadas baricéntricas $\mathbf{u}$ después de una transformación afín del dominio triangular.
- **Envolvente convexa:** Para $0 \leq u_1, u_2, u_3 \leq 1$, el parche yace dentro de la envolvente convexa de la red de control.
- **Curvas frontera:** Para un parche triangular, las curvas frontera están determinadas por los puntos de control en la frontera respectiva.

2.2. Parches triangulares de Bézier racionales de grado n

Un triángulo de Bézier racional, se define como la proyección de un triángulo de Bézier polinomial en 4D [18]. El parche triangular se puede evaluar en términos del algoritmo de de Casteljau o la evaluación directa con polinomios de Bernstein, según se presenta a continuación.

2.2.1. Algoritmo de de Casteljau

Considerando la evaluación de de Casteljau, se puede interpretar cada punto intermedio $\mathbf{b}_{\mathbf{i}}^{r}(\mathbf{u})$ como la proyección del correspondiente punto intermedio en el algoritmo de de Casteljau polinomial de la pre-imagen 4D de nuestro parche [18]. El procedimiento recursivo para computar un punto sobre el parche triangular, establece:

$$\mathbf{b}_{\mathbf{i}}^{r}(\mathbf{u}) = \frac{u_1 w_{\mathbf{i}+\mathbf{e_1}}^{r-1} \mathbf{b}_{\mathbf{i}+\mathbf{e_1}}^{r-1}(\mathbf{u}) + u_2 w_{\mathbf{i}+\mathbf{e_2}}^{r-1} \mathbf{b}_{\mathbf{i}+\mathbf{e_2}}^{r-1}(\mathbf{u}) + u_3 w_{\mathbf{i}+\mathbf{e_3}}^{r-1} \mathbf{b}_{\mathbf{i}+e_3}^{r-1}(\mathbf{u})}{w_{\mathbf{i}}^{r}} \tag{2.5}$$

donde

$$w_{\mathbf{i}}^{r}(\mathbf{u}) = u_1 w_{\mathbf{i}+\mathbf{e_1}}^{r-1}(\mathbf{u}) + u_2 w_{\mathbf{i}+\mathbf{e_2}}^{r-1}(\mathbf{u}) + u_3 w_{\mathbf{i}+\mathbf{e_3}}^{r-1}(\mathbf{u}) \tag{2.6}$$

y

$$r = 1, \ldots, n; \quad |\mathbf{i}| = n - r; \quad \mathbf{b}_{\mathbf{i}}^{0} = \mathbf{b}_{\mathbf{i}}.$$

2.2.2. Evaluación directa con polinomios de Bernstein

La expresión para evaluar directamente el parche triangular haciendo uso de los puntos de control y sus pesos asociados es [8]:

$$\mathbf{b}^{n}(\mathbf{u}) = \mathbf{b}_{\mathbf{0}}^{n}(\mathbf{u}) = \frac{\sum_{|\mathbf{i}|=n} w_{\mathbf{i}} \mathbf{b}_{\mathbf{i}} B_{\mathbf{i}}^{n}(\mathbf{u})}{\sum_{|\mathbf{i}|=n} w_{\mathbf{i}} B_{\mathbf{i}}^{n}(\mathbf{u})} \tag{2.7}$$

En el capítulo 6 vamos a considerar la representación como un parche triangular de Bézier racional de superficies especiales que yacen sobre un elipsoide de revolución o una esfera.

CAPÍTULO 3

Variables aleatorias

3.1. Variable aleatoria

Considere un espacio de probabilidad $(\Omega, \mathcal{F}, P)$, donde Ω es un conjunto no vacío, $\mathcal{F}$ es un $\sigma -$ algebra de subconjuntos de Ω, y P es una medida finita sobre el espacio $(\Omega, \mathcal{F})$ con $P(\Omega) = 1$. Intuitivamente, Ω representa el conjunto de todos los posibles resultados de un experimento aleatorio, real o conceptual, para alguna codificación dada de los resultados del experimento. El conjunto Ω es referenciado como espacio muestral y los elementos $\omega \in \Omega$ como puntos muestrales o posibles resultados. El $\sigma-$ algebra $\mathcal{F}$ comprende eventos $A \subseteq \Omega$ cuya probabilidad de ocurrencia $P(A)$ está bien definida [19].

3.1.1. Definición

Una variable aleatoria X, es un mapeo medible sobre un espacio de probabilidad $(\Omega, \mathcal{F}, P)$ en un espacio medible $(S, \mathcal{S})$[1,2]. X es medible, si $X^{-1}(B)$ pertenece a $\mathcal{F}$ para todo $B \in \mathcal{S}$. En particular; si $(S, \mathcal{S}) = \left(\mathbb{R}^k, \mathcal{R}^k\right)$, $\mathbb{R}^k$ es el espacio Euclidiano de dimensión k y $\mathcal{R}^k$ es el $\sigma -$ algebra de conjuntos Borel de dimensión k; el mapeo X es llamado vector aleatorio [19].

3.1.2. Distribución

Las cantidades de interés para un mapeo aleatorio X sobre un espacio de probabilidad $(\Omega, \mathcal{F}, P)$, son las probabilidades con las cuales X toma conjuntos de valores. Luego, P determina el aspecto más relevante de X, llamado, su distribución $Q \equiv P \circ X^{-1}$. La distribución de X se define sobre el espacio imagen $(S, \mathcal{S})$ por $Q(B) = P\left(X^{-1}(B)\right)$, $B \in \mathcal{S}$ [19]. Suponga que f es una función medible no negativa, y sea Q una medida dada por: $Q(B) = \int_B f dP, \quad B \in \mathcal{S}$, entonces se dice que Q tiene densidad f respecto a P. Más aún, si Q es una probabilidad, f debería satisfacer: $\int_S f dP = 1$ [20].

[1]Dado un espacio $(S, \mathcal{S})$, $A \subseteq S$ se denomina medible si $A \in \mathcal{S}$.

[2]Si $(S_i, \mathcal{S}_i)$, $i = 1, 2$, es un par de espacios medibles, entonces una función $f : S_1 \to S_2$ se denomina un mapeo medible si $f^{-1}(B) = \{x \in S_1 : f(x) \in B\} \in \mathcal{S}_1$ para todo $B \in S_2$.

Se observa que, dada cualquier medida Q sobre un espacio medible $(S, \mathcal{S})$ se puede construir un espacio de probabilidad $(\Omega, \mathcal{F}, P)$ y un mapeo aleatorio X sobre $(\Omega, \mathcal{F})$ con distribución Q. La construcción más simple del espacio en mención, se obtiene con $\Omega = S$, $\mathcal{F} = \mathcal{S}$, $P = Q$ y $X(\omega) = \omega$ (ocasionalmente es llamada construcción canónica) [19].

3.2. Vector aleatorio

3.2.1. Distribución

Considerando un vector aleatorio $X = (X_1, X_2, \ldots, X_k)$ y un vector de constantes $\mathbf{x} = (x_1, x_2, \ldots, x_k)$ de dimensión k, la distribución Q (una medida sobre $\mathcal{R}^k$) y la función de distribución F (una función real sobre $\mathbb{R}^k$) se definen por [20]:

$$\begin{aligned} Q(B) &= P\left(X^{-1}(B)\right), \quad B \in \mathcal{R}^k, \\ F(x_1, x_2, \ldots, x_k) &= Q(X_1 \leq x_1, X_2 \leq x_2, \ldots, X_k \leq x_k), \end{aligned} \tag{3.1}$$

a menudo, Q y F se denominan la distribución conjunta y la función de distribución conjunta de $X_1, X_2, \ldots, X_k$, respectivamente [20].

Si X es un vector aleatorio de dimensión k y el mapeo $g : \mathbb{R}^k \to \mathbb{R}^l$ es medible, entonces g es un vector aleatorio de dimensión l; si la distribución de X es Q, la distribución de $g(X)$ es $Q \circ g^{-1}$. Si $g_j : \mathbb{R}^k \to \mathbb{R}^1$ está definida por $g_j(x_1, x_2, \ldots, x_k) = x_j$, entonces $g_j(X) = X_j$ y su distribución $Q_j = Q \circ g_j^{-1}$ está dada por $Q_j(B) = Q(\{(x_1, x_2, \ldots, x_k) : x_j \in B\})$, $B \in \mathcal{R}^1$. Los Q_j son las distribuciones marginales de Q [20].

Se puede mostrar que F es continua desde arriba[3], y satisface que[4] $\triangle_A F \geq 0$ para rectángulos acotados A. Adicionalmente, si F es una función real sobre $\mathbb{R}^k$ continua desde arriba, y tal que $\triangle_A F \geq 0$ para rectángulos acotados A, entonces existe una única medida de probabilidad Q sobre $\mathcal{R}^k$ que satisface $Q(A) = \triangle_A F$ para rectángulos acotados A [20].

3.2.2. Densidad

La distribución del vector X, podría incluso tener una densidad f con respecto a la medida de Lebesgue de dimensión k, esto es [20]:

$$\begin{aligned} Q(B) &= \int_B f(\mathbf{x})\, \mathbf{dx}, \quad B \in \mathcal{R}^k, \\ F(x_1, x_2, \ldots, x_k) &= \int_{-\infty}^{x_1} \int_{-\infty}^{x_2} \cdots \int_{-\infty}^{x_k} f(y_1, y_2, \ldots, y_k)\, dy_1 dy_2 \cdots dy_k. \end{aligned} \tag{3.2}$$

[3] En el sentido que $\lim_{h \to 0^+} F(x_1 + h, x_2 + h, \ldots, x_k + h) = F(x_1, x_2, \ldots, x_k)$.

[4] Para una función real F sobre $\mathbb{R}^k$, la diferencia de F a través de los vértices de un rectángulo acotado A es $\triangle_A F = \sum (\operatorname{sgn}_A \mathbf{x}) \cdot F(\mathbf{x})$, la suma se extiende sobre los 2^k vértices de A.

Como Q tiene una densidad f en $\mathbb{R}^k$, entonces Q_j tiene densidad sobre la recta:

$$f_j(x) = \int_{\mathbb{R}^{k-1}} f(x_1, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_k)\, dx_1 \cdots dx_{j-1} dx_{j+1} \cdots dx_k, \quad (3.3)$$

pues el lado derecho de la ecuación 3.3 integrado sobre B, por el teorema de Fubini es $Q(\{(x_1, x_2, \ldots, x_k) : x_j \in B\})$ [20].

3.2.3. Cambio de variable

Suponga que g es un mapeo[5] uno a uno, continuamente diferenciable de V en U, donde U y V son conjuntos[6] abiertos de $\mathbb{R}^k$. Sea T la inversa, y suponga que su Jacobiano $J(\mathbf{x})$ nunca es nulo. Si X tiene densidad f en el soporte V, entonces usando el teorema de cambio de variable para integrales múltiples, dado $B \subseteq U$:

$$\begin{aligned} Q(B) &= P(\{\omega : g(X(\omega)) \in B\}) \\ &= P(\{\omega : X(\omega) \in T(B)\}) \\ &= \int_{T(B)} f(\mathbf{y})\, \mathbf{dy} \\ &= \int_B f(T(\mathbf{x}))\, |J(\mathbf{x})|\, \mathbf{dx}. \end{aligned} \quad (3.4)$$

Por lo tanto, $g(X)$ tiene densidad [20]:

$$d(\mathbf{x}) = \begin{cases} f(T(\mathbf{x}))\, |J(\mathbf{x})| & \text{si } \mathbf{x} \in U \\ 0 & \text{si } \mathbf{x} \notin U \end{cases} \quad (3.5)$$

3.2.4. Independencia

Sea $X = (X_1, X_2, \ldots, X_k)$ un vector aleatorio con función de distribución F. Sea X_j la j-ésima componente del vector aleatorio X, y sea F_j la función de distribución marginal asociada a X_j. Se define que $X_1, X_2, \ldots, X_k$ son mutuamente independientes; si y sólo si:

$$F(x_1, x_2, \ldots, x_k) = \prod_{j=1}^{k} F_j(x_i). \quad (3.6)$$

Si X tiene densidad f y densidades marginales f_j, la independencia equivale a:

$$f(x_1, x_2, \ldots, x_k) = \prod_{i=1}^{k} f_i(x_i). \quad (3.7)$$

[5] Si g es un mapeo continuo de $\mathbb{R}^k \to \mathbb{R}^l$, entonces g es medible [20].
[6] Los conjuntos abiertos, son conjuntos Borel [20].

CAPÍTULO 4

Simulación

4.1. Distribución uniforme

4.1.1. Definición

Sea X una variable aleatoria sobre un espacio de probabilidad $(\Omega, \mathcal{F}, P)$ en un espacio medible $\left(\mathbb{R}^1, \mathcal{R}^1\right)$, cuya función de distribución $Q(B)$ para $B \in \mathcal{R}^1$, se puede expresar mediante una densidad f sobre la medida de Lebesgue como $\int_B f(x)\,dx$. En particular, si $f(x) = \mathbf{1}_{\{x\in(0,1)\}}(x)$, se dice que X tiene distribución uniforme en el intervalo $(0,1)$.

4.1.2. Número aleatorio

Hoy en día, la mayoría de lenguajes de computador contienen un generador de números aleatorios. La función ideal del generador es producir de manera artificial, una secuencia infinita de variables aleatorias independientes con distribución uniforme. El concepto de secuencia infinita es una abstracción matemática imposible de implementar mediante un computador. Lo máximo que se podría lograr en la práctica es generar una secuencia de números «aleatorios» con propiedades estadísticas indistinguibles respecto a una secuencia verdadera de variables aleatorias independientes con distribución uniforme. A pesar de que los métodos de generación basados en modelos físicos para radiación y mecánica cuántica ofrecen fuentes estables de verdadera aleatoriedad, en la práctica los generadores de números aleatorios se basan en algoritmos simples que pueden ser fácilmente implementados en un computador y aproximan adecuadamente el carácter verdaderamente aleatorio. En `MATLAB`, por ejemplo, se generan números aleatorios usando la función `rand` [21].

Producir números aleatorios, es fundamental para generar vectores aleatorios $X \in \mathbb{R}^k$ con distribución arbitraria, toda vez que el procedimiento se realiza en general, extrayendo d números aleatorios $U_1, U_2, \ldots, U_d$, y retornando $h(U_1, U_2, \ldots, U_d)$, donde h es una función de $(0,1)^d$ a $\mathbb{R}^k$ [21].

4.2. Método de la transformada inversa

4.2.1. Transformada inversa en $\mathbb{R}^1$

Sea X una variable aleatoria sobre un espacio de probabilidad $(\Omega, \mathcal{F}, P)$ en un espacio medible $\left(\mathbb{R}^1, \mathcal{R}^1\right)$, con función de distribución F. En este caso $\triangle_{(a,b]} F = F(b) - F(a) \geq 0$ para $a < b$, por lo tanto F es una función no decreciente; así que la función inversa F^{-1} se puede definir como $F^{-1}(y) = \text{ínf}\{x : y \leq F(x)\}$, para $y \in (0,1)$. Sea U una variable aleatoria sobre el espacio de probabilidad $(\Omega, \mathcal{F}, P)$ en el espacio medible $\left(\mathbb{R}^1, \mathcal{R}^1\right)$, con distribución uniforme, la función de distribución de la variable $F^{-1}(U)$ está dada por[1]:

$$\begin{aligned} P\left(\left\{w : F^{-1}(U(w)) \leq x\right\}\right) &= P\left(\{w : U(w) \leq F(x)\}\right) \\ &= \int_{(0,F(x)]} \mathbf{1}_{\{y \in (0,1)\}}(y)\, dy \\ &= F(x). \end{aligned}$$

De tal manera que, para generar una variable aleatoria $X \in \mathbb{R}^1$ con función de distribución arbitraria F [21], se genera un número aleatorio U uniforme, y se retorna $X = F^{-1}(U)$.

4.2.2. Transformada inversa en $\mathbb{R}^k$

Sea $X = (X_1, X_2, \ldots, X_k)$ un vector aleatorio de dimensión k, cuyas componentes son mutuamente independientes, y para el cual la función de distribución marginal de la j-ésima componente se denota por F_{X_j}. Sea $U = (U_1, U_2, \ldots, U_k)$ otro vector aleatorio de dimensión k, cuyas componentes son también mutuamente independientes, para el cual la función de distribución marginal de la j-ésima componente se denota por F_{U_j}, y además suponga que cada componente U_j se distribuye de manera uniforme. Sea $F_{X_j}^{-1}(y) = \text{ínf}\left\{x : y \leq F_{X_j}(x)\right\}$, para $y \in (0,1)$. La función de distribución conjunta del vector aleatorio $h(U) := \left(F_{X_1}^{-1}(U_1), F_{X_2}^{-1}(U_2), \ldots, F_{X_k}^{-1}(U_k)\right)$ es:

$$\begin{aligned} & P\left(\left\{w : F_{X_1}^{-1}(U_1(\omega)) \leq x_1, F_{X_2}^{-1}(U_2(\omega)) \leq x_2, \ldots, F_{X_k}^{-1}(U_k(\omega)) \leq x_k\right\}\right) \\ = \; & P\left(\left\{w : U_1(\omega) \leq F_{X_1}(x_1), U_2(\omega) \leq F_{X_2}(x_2), \ldots, U_k(\omega) \leq F_{X_k}(x_k)\right\}\right) \\ = \; & \int_{\times_{j=1}^k \left(0, F_{X_j}(x_j)\right]} \mathbf{1}_{\{\mathbf{y} \in (0,1)^k\}}(\mathbf{y})\, \mathbf{dy} \\ = \; & \prod_{j=1}^k F_{X_j}(x_j). \end{aligned}$$

En este sentido, generar un vector aleatorio $X \in \mathbb{R}^k$ con función de distribución arbitraria F, y de componentes independientes es posible generando un vector aleatorio U con distribuciones marginales uniformes e independientes, y retornando $X = h(U)$.

[1] El mapeo $F^{-1}(U)$, es medible [22].

CAPÍTULO 5

Distribución uniforme en elipsoides especiales

Sea $s(x_1, x_2) = (x_1, x_2, x_3(x_1, x_2))$, una superficie paramétrica que yace sobre un elipsoide, centrado en el origen, y con semi-ejes[1] $a_1 \geq a_2 \geq a_3 > 0$. Específicamente, s está definida como el gráfico de la función x_3 sobre el dominio D:

$$\begin{aligned} x_3(x_1, x_2) &= a_3\sqrt{1 - \left(\frac{x_1}{a_1}\right)^2 - \left(\frac{x_2}{a_2}\right)^2}, \\ D &= \left\{(x_1, x_2) : \alpha_2 a_2\sqrt{1 - \left(\frac{x_1}{a_1}\right)^2} < x_2 < \alpha_1 a_2\sqrt{1 - \left(\frac{x_1}{a_1}\right)^2},\ \beta_2 < x_1 < \beta_1\right\}, \end{aligned} \tag{5.1}$$

donde, para $i = 1, 2$ se cumple que $|\alpha_i| \leq 1$ y $|\beta_i| \leq a_1$, ver en 5.1 una ilustración de D.

El objetivo es generar puntos uniformemente distribuidos sobre la imagen del dominio[2] D. Es decir, generar valores de un vector aleatorio $X = (X_1, X_2)$ sobre un espacio de probabilidad $(\Omega, \mathcal{F}, P)$, de manera que, $s(X)$ se distribuya uniformemente sobre $s(D)$. La uniformidad significa que dado $A \in \mathcal{R}^2$ contenido en D, y denotando λ la medida de Lebesgue de dimensión 2 para la superficie; la probabilidad de $s(X) \in s(A)$ es igual a $\lambda(s(A))/\lambda(s(D))$. Teniendo en cuenta que, x_3 es continuamente diferenciable[3], de acuerdo a [24] el requerimiento de uniformidad implica que:

$$\begin{aligned} P\{\omega : s(X(\omega)) \in s(A)\} &= P\{\omega : X(\omega) \in A\} \\ &= \lambda(s(A))/\lambda(s(D)) \\ &= \int_A (\lambda(s(D)))^{-1}\sqrt{1 + \left(\frac{\partial x_3}{\partial x_1}\right)^2 + \left(\frac{\partial x_3}{\partial x_2}\right)^2}\,\mathbf{dx}. \end{aligned} \tag{5.2}$$

[1] Escogemos las coordenadas x_1, x_2, y x_3 de manera tal que los semi-ejes correspondientes satisfacen $a_1 \geq a_2 \geq a_3 > 0$.

[2] D es abierto, por lo tanto $D \in \mathcal{R}^2$.

[3] Ver, por ejemplo [23].

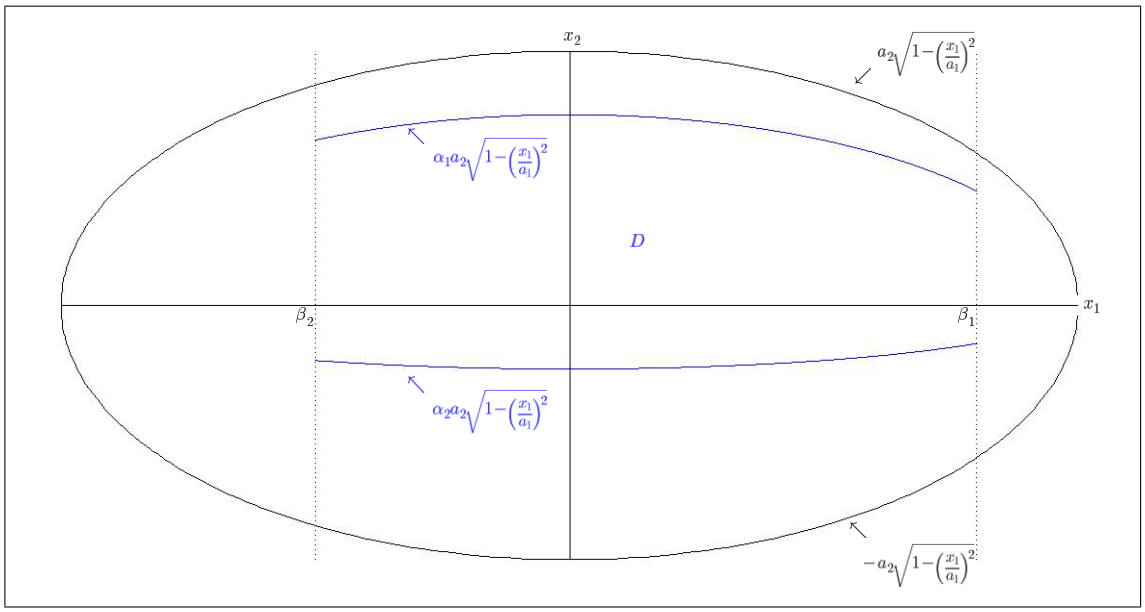


FIGURA 5.1. Ilustración del dominio D .

La ecuación 5.2 implica que X debe tener densidad respecto a la medida de Lebesgue de dimensión 2 dada por: $f(x_1, x_2) = (\lambda(s(D)))^{-1}\sqrt{1+\left(\frac{\partial x_3}{\partial x_1}\right)^2+\left(\frac{\partial x_3}{\partial x_2}\right)^2}$, si $\mathbf{x} \in D$, y 0 en otro caso. Las derivadas parciales de x_3 son:

$$\frac{\partial x_3}{\partial x_1} = -\frac{a_3}{a_1^2}x_1\left(\sqrt{1-\left(\frac{x_1}{a_1}\right)^2-\left(\frac{x_2}{a_2}\right)^2}\right)^{-1}, \quad \frac{\partial x_3}{\partial x_2} = -\frac{a_3}{a_2^2}x_2\left(\sqrt{1-\left(\frac{x_1}{a_1}\right)^2-\left(\frac{x_2}{a_2}\right)^2}\right)^{-1},$$

si además se denota $\delta_1 = 1-(a_3/a_1)^2$, $\delta_2 = 1-(a_3/a_2)^2$ y $\kappa = (\lambda(s(D)))^{-1}$; la función de densidad f para $\mathbf{x} \in D$ se puede escribir como:

$$\begin{aligned} f(x_1, x_2) &= \kappa\sqrt{1+\left(\frac{\partial x_3}{\partial x_1}\right)^2+\left(\frac{\partial x_3}{\partial x_2}\right)^2} \\ &= \kappa\sqrt{\frac{1-\delta_1\left(\frac{x_1}{a_1}\right)^2-\delta_2\left(\frac{x_2}{a_2}\right)^2}{1-\left(\frac{x_1}{a_1}\right)^2-\left(\frac{x_2}{a_2}\right)^2}}, \end{aligned} \tag{5.3}$$

donde $1 > \delta_1 \geq \delta_2 \geq 0$ debido a que $a_1 \geq a_2 \geq a_3 > 0$.

En conclusión, para satisfacer el requerimiento de uniformidad es necesario generar el vector aleatorio X con densidad conjunta f dada por la ecuación 5.3.

La estrategia consiste en hacer un cambio de variable que transforma D en un rectángulo. Sea $\mathbf{y} = (y_1, y_2) = g(x_1, x_2)$ un cambio de variable tal que su inversa $(x_1, x_2) = g^{-1}(y_1, y_2)$ está dada por:

$$\begin{aligned} x_1 &= a_1 y_1, \\ x_2 &= a_2 y_2 \sqrt{1 - y_1^2}, \text{en el dominio} \\ D_g &= \left\{ (y_1, y_2) : \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1} \text{ y } \alpha_2 < y_2 < \alpha_1 \right\}. \end{aligned} \tag{5.4}$$

El mapeo g, es uno a uno, continuamente diferenciable, y su Jacobiano está dado por:

$$\begin{aligned} J(y) &= \left| \begin{bmatrix} \frac{\partial x_1}{\partial y_1} & \frac{\partial x_1}{\partial y_2} \\ \frac{\partial x_2}{\partial y_1} & \frac{\partial x_2}{\partial y_2} \end{bmatrix} \right| \\ &= \left| \begin{bmatrix} a_1 & 0 \\ -a_2 y_1 y_2 \left(\sqrt{1 - y_1^2} \right)^{-1} & a_2 \sqrt{1 - y_1^2} \end{bmatrix} \right| \\ &= a_1 a_2 \sqrt{1 - y_1^2}, \end{aligned}$$

de tal forma que utilizando el resultado en 3.5, se obtiene que la densidad de la variable aleatoria $Y = g(X)$ es:

$$\begin{aligned} f_g(\mathbf{y}) &= |J(\mathbf{y})| \, f\left(g^{-1}(\mathbf{y})\right) \\ &= \kappa a_1 a_2 \sqrt{1 - y_1^2} \sqrt{\frac{1 - \delta_1 y_1^2 - \delta_2 \left(1 - y_1^2\right) y_2^2}{\left(1 - y_1^2\right) - \left(1 - y_1^2\right) y_2^2}} \\ &= \kappa a_1 a_2 \sqrt{1 - \delta_1 y_1^2} \sqrt{\frac{1 - m(y_1) y_2^2}{1 - y_2^2}}, \end{aligned} \tag{5.5}$$

para $\mathbf{y} \in D_g$, y $m(y_1) = \frac{\delta_2 \left(1 - y_1^2\right)}{1 - \delta_1 y_1^2}$.

Retomando el problema de generar el vector aleatorio X, se nota que es posible resolverlo generando el vector aleatorio Y, y considerando su imagen inversa $X = g^{-1}(Y)$.

A continuación se presentan dos procedimientos para generar el vector aleatorio Y, dependiendo de si el elipsoide sobre D es una esfera o un elipsoide de revolución.

5.1. Esfera

En el caso de la esfera $a_1 = a_2 = a_3$, y $\delta_1 = \delta_2 = 0$, por lo tanto la densidad conjunta de Y en D_g es:

$$f_g(\mathbf{y}) = \kappa a_1^2 \sqrt{\frac{1}{1-y_2^2}}, \tag{5.6}$$

las densidades marginales de Y, de acuerdo a 3.3 son:

$$\begin{aligned} f_{g,1}(y_1) &= \int_{\mathbb{R}^1} \kappa a_1^2 \sqrt{\frac{1}{1-y_2^2}} dy_2 \\ &= \int_{\alpha_2}^{\alpha_1} \kappa a_1^2 \sqrt{\frac{1}{1-y_2^2}} dy_2 \\ &= \kappa a_1^2 \left(\arcsin(\alpha_1) - \arcsin(\alpha_2)\right), \text{ para } \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1}, \end{aligned} \tag{5.7}$$

$$\begin{aligned} f_{g,2}(y_2) &= \int_{\mathbb{R}^1} \kappa a_1^2 \sqrt{\frac{1}{1-y_2^2}} dy_1 \\ &= \int_{\frac{\beta_2}{a_1}}^{\frac{\beta_1}{a_1}} \kappa a_1^2 \sqrt{\frac{1}{1-y_2^2}} dy_1 \\ &= \kappa a_1 (\beta_1 - \beta_2) \sqrt{\frac{1}{1-y_2^2}}, \text{ para } \alpha_2 < y_2 < \alpha_1. \end{aligned} \tag{5.8}$$

Las funciones de distribución, se pueden escribir como[4]:

$$\begin{aligned} F_{g,1}(y_1) &= \begin{cases} 0 & \text{si } y_1 \le \frac{\beta_2}{a_1} \\ \int_{\frac{\beta_2}{a_1}}^{y_1} f_{g,1}(y_1)\, dy_1 & \text{si } \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1} \\ 1 & \text{si } y_1 \ge \frac{\beta_1}{a_1} \end{cases} \\ &= \begin{cases} 0 & \text{si } y_1 \le \frac{\beta_2}{a_1} \\ \dfrac{a_1\left(y_1 - \frac{\beta_2}{a_1}\right)}{\beta_1 - \beta_2} & \text{si } \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1} \\ 1 & \text{si } y_1 \ge \frac{\beta_1}{a_1}, \end{cases} \end{aligned} \tag{5.9}$$

[4]Dado que, $\int_{\frac{\beta_2}{a_1}}^{\frac{\beta_1}{a_1}} f_{g,1}(y_1)\, dy_1 = \int_{\alpha_2}^{\alpha_1} f_{g,2}(y_2)\, dy_2 = 1$; $\kappa = 1/\left(a_1(\beta_1 - \beta_2)\left(\arcsin(\alpha_1) - \arcsin(\alpha_2)\right)\right)$.

$$
\begin{aligned}
F_{g,2}\left(y_2\right) &= \begin{cases} 0 & \text{si } \ y_2 \leq \alpha_2 \\ \displaystyle\int_{\alpha_2}^{y_2} f_{g,2}\left(y_2\right) dy_2 & \text{si } \ \alpha_2 < y_2 < \alpha_1 \\ 1 & \text{si } \ y_2 \geq \alpha_1 \end{cases} \\
&= \begin{cases} 0 & \text{si } \ y_2 \leq \alpha_2 \\ \dfrac{\arcsin\left(y_2\right) - \arcsin\left(\alpha_2\right)}{\arcsin\left(\alpha_1\right) - \arcsin\left(\alpha_2\right)} & \text{si } \ \alpha_2 < y_2 < \alpha_1 \\ 1 & \text{si } y_2 \geq \alpha_1. \end{cases}
\end{aligned} \tag{5.10}
$$

Las variables Y_1, Y_2 son independientes[5], por lo tanto, es posible utilizar el método de la transformada inversa para generar el vector aleatorio Y. Es decir, para generar Y, se genera el vector $U = (U_1, U_2)$ de componentes independientes y uniformemente distribuidas, y se retorna $Y = h\left(U\right) = \left(F_{g,1}^{-1}\left(U_1\right), F_{g,2}^{-1}\left(U_2\right)\right)$. Explícitamente:

$$
\begin{aligned}
F_{g,1}^{-1}\left(U_1\right) &= \frac{\beta_2 + \left(\beta_1 - \beta_2\right) U_1}{a_1}, \\
F_{g,2}^{-1}\left(U_2\right) &= \sin\left(U_2 \arcsin\left(\alpha_1\right) + \left(1 - U_2\right) \arcsin\left(\alpha_2\right)\right).
\end{aligned} \tag{5.11}
$$

5.2. Elipsoide de revolución

Considerando el elipsoide de revolución, se tiene que $a_2 = a_3$, y $\delta_2 = 0$, por lo tanto la densidad conjunta de Y en D_g es:

$$
f_g\left(\mathbf{y}\right) = \kappa a_1 a_2 \sqrt{1 - \delta_1 y_1^2} \sqrt{\frac{1}{1 - y_2^2}}, \tag{5.12}
$$

las densidades marginales de Y, de acuerdo a 3.3 son:

$$
\begin{aligned}
f_{g,1}\left(y_1\right) &= \int_{\mathbb{R}^1} \kappa a_1 a_2 \sqrt{1 - \delta_1 y_1^2} \sqrt{\frac{1}{1 - y_2^2}} dy_2 \\
&= \int_{\alpha_2}^{\alpha_1} \kappa a_1 a_2 \sqrt{1 - \delta_1 y_1^2} \sqrt{\frac{1}{1 - y_2^2}} dy_2 \\
&= \kappa a_1 a_2 \left(\arcsin\left(\alpha_1\right) - \arcsin\left(\alpha_2\right)\right) \sqrt{1 - \delta_1 y_1^2}, \ \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1},
\end{aligned} \tag{5.13}
$$

[5]Claramente, $f_g\left(\mathbf{y}\right) = f_{g,2}\left(y_1\right) f_{g,2}\left(y_2\right)$.

$$\begin{aligned} f_{g,2}(y_2) &= \int_{\mathbb{R}^1} \kappa a_1 a_2 \sqrt{1-\delta_1 y_1^2}\sqrt{\frac{1}{1-y_2^2}}dy_1 \\ &= \int_{\frac{\beta_2}{a_1}}^{\frac{\beta_1}{a_1}} \kappa a_1 a_2 \sqrt{1-\delta_1 y_1^2}\sqrt{\frac{1}{1-y_2^2}}dy_1 \\ &= \kappa a_1 a_2 \sqrt{\frac{1}{1-y_2^2}} \int_{\frac{\beta_2}{a_1}}^{\frac{\beta_1}{a_1}} \sqrt{1-\delta_1 y_1^2}dy_1 \\ &= \kappa a_1 a_2 \left(\phi\left(\frac{\beta_1}{a_1};\delta_1\right) - \phi\left(\frac{\beta_2}{a_1};\delta_1\right)\right)\sqrt{\frac{1}{1-y_2^2}},\ \alpha_2 < y_2 < \alpha_1, \end{aligned} \tag{5.14}$$

donde

$$\phi(t;\delta_1) = \int \sqrt{1-\delta_1 t^2}dt = \left(2\sqrt{\delta_1}\right)^{-1}\left(\sqrt{\delta_1}t\sqrt{1-\delta_1 t^2} + \arcsin\left(\sqrt{\delta_1}t\right)\right). \tag{5.15}$$

Las funciones de distribución, se pueden escribir como[6]:

$$\begin{aligned} F_{g,1}(y_1) &= \begin{cases} 0 & \text{si } \ y_1 \le \frac{\beta_2}{a_1} \\ \int_{\frac{\beta_2}{a_1}}^{y_1} f_{g,1}(y_1)\,dy_1 & \text{si } \ \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1} \\ 1 & \text{si } \ y_1 \ge \frac{\beta_1}{a_1} \end{cases} \\ &= \begin{cases} 0 & \text{si } \ y_1 \le \frac{\beta_2}{a_1} \\ \dfrac{\phi(y_1;\delta_1) - \phi\left(\frac{\beta_2}{a_1};\delta_1\right)}{\phi\left(\frac{\beta_1}{a_1};\delta_1\right) - \phi\left(\frac{\beta_2}{a_1};\delta_1\right)} & \text{si } \ \frac{\beta_2}{a_1} < y_1 < \frac{\beta_1}{a_1} \\ 1 & \text{si } \ y_1 \ge \frac{\beta_1}{a_1}, \end{cases} \end{aligned} \tag{5.16}$$

$$\begin{aligned} F_{g,2}(y_2) &= \begin{cases} 0 & \text{si } \ y_2 \le \alpha_2 \\ \int_{\alpha_2}^{y_2} f_{g,2}(y_2)\,dy_2 & \text{si } \ \alpha_2 < y_2 < \alpha_1 \\ 1 & \text{si } \ y_2 \ge \alpha_1 \end{cases} \\ &= \begin{cases} 0 & \text{si } \ y_2 \le \alpha_2 \\ \dfrac{\arcsin(y_2) - \arcsin(\alpha_2)}{\arcsin(\alpha_1) - \arcsin(\alpha_2)} & \text{si } \ \alpha_2 < y_2 < \alpha_1 \\ 1 & \text{si } y_2 \ge \alpha_1. \end{cases} \end{aligned} \tag{5.17}$$

[6]Dado que, $\int_{\frac{\beta_2}{a_1}}^{\frac{\beta_1}{a_1}} f_{g,1}(y_1)\,dy_1 = \int_{\alpha_2}^{\alpha_1} f_{g,2}(y_2)\,dy_2$=1; $\kappa = 1/\left(a_1 a_2\left(\phi\left(\frac{\beta_1}{a_1};\delta_1\right) - \phi\left(\frac{\beta_2}{a_1};\delta_1\right)\right)(\arcsin(\alpha_1) - \arcsin(\alpha_2))\right)$.

Note que las variables Y_1, Y_2 son independientes[7], de tal manera que es posible utilizar el método de la transformada inversa para generar el vector aleatorio Y. Es decir, se genera el vector $U = (U_1, U_2)$ de componentes independientes y uniformemente distribuidas, y se retorna $Y = h(U) = \left(F_{g,1}^{-1}(U_1), F_{g,2}^{-1}(U_2)\right)$. Explícitamente:

$$\begin{aligned} F_{g,1}^{-1}(U_1) &= \phi^{-1}\left(U_1\phi\left(\frac{\beta_1}{a_1};\delta_1\right) + (1-U_1)\,\phi\left(\frac{\beta_2}{a_1};\delta_1\right);\delta_1\right), \\ F_{g,2}^{-1}(U_2) &= \sin\left(U_2 \arcsin(\alpha_1) + (1-U_2)\arcsin(\alpha_2)\right), \end{aligned} \tag{5.18}$$

la función ϕ es estrictamente monótona en el intervalo $\left[\frac{\beta_2}{a_1}, \frac{\beta_1}{a_1}\right]$, por lo tanto existe su función inversa ϕ^{-1}, también estrictamente monótona. Dado que para ϕ^{-1} no se tiene una expresión cerrada, la función se puede aproximar numéricamente.

Consideremos el cambio de variable $z = z(y_1) = \sqrt{\delta_1}y_1$. La variable aleatoria Z tiene función de distribución: $F_z(z) = (\phi(z;1) - \phi(r_2;1)) / (\phi(r_1;1) - \phi(r_2;1))$ definida en el intervalo (r_2, r_1), donde $r_i = (\beta_i/a_1)\sqrt{\delta_1}$. En particular, si Z_* es una variable aleatoria con función de distribución $F_{z_*}(z) = F_z(z)$ donde $(r_2, r_1) = (-1, 1)$, se dice que Z_* tiene densidad semicircular.

De acuerdo a [25], es posible muestrear Z a partir de la función inversa de $F_{z_*}^{-1}(u)$ pues:

$$\begin{aligned} F_z^{-1}(u) &= F_{z_*}^{-1}\left(F_{z_*}(r_2) + (F_{z_*}(r_1) - F_{z_*}(r_2))\,u\right) \\ &= F_{z_*}^{-1}\left(\frac{\phi(r_2;1) - \phi(-1;1)}{\phi(1;1) - \phi(-1;1)} + \left(\frac{\phi(r_1;1) - \phi(r_2;1)}{\phi(1;1) - \phi(-1;1)}\right)u\right) \\ &= \phi^{-1}\left(\phi(r_2;1) + (\phi(r_1;1) - \phi(r_2;1))\,u;1\right). \end{aligned} \tag{5.19}$$

Es decir para generar Z, podríamos simular un número aleatorio U con densidad uniforme en el intervalo $(\phi(r_2;1), \phi(r_1;1))$, y posteriormente evaluar $\phi^{-1}(U;1)$.

5.2.1. Aproximación interpolante locamente monótona

Se propone en el proceso de generación de Y_1, aproximar la función ϕ^{-1} dividiendo el intervalo I en $s_0 < s_1 < \ldots < s_{n-1}$, computando $\phi(s_k)$ para $k = 0, 1, \ldots, n-1$, y realizando interpolación cúbica de Hermite por trozos[8] monótonos a el conjunto de datos $\{\phi(s_k), s_k\}_{k=0}^{n-1}$ de acuerdo a la metodología descrita en [26]. Allí los autores derivan las condiciones suficientes y necesarias para que una cúbica de Hermite sea localmente monótona, y desarrollan un algoritmo para construir una interpolante cúbica de Hermite por trozos monótonos para datos monótonos. En `MATLAB`, por ejemplo, se realiza la interpolación usando la función `pchip` la cual garantiza la monotonicidad.

[7] Claramente, $f_g(\mathbf{y}) = f_{g,2}(y_1) f_{g,2}(y_2)$.

[8] Para una breve ilustración de la interpolación cúbica de Hermite por trozos ver el apéndice A.

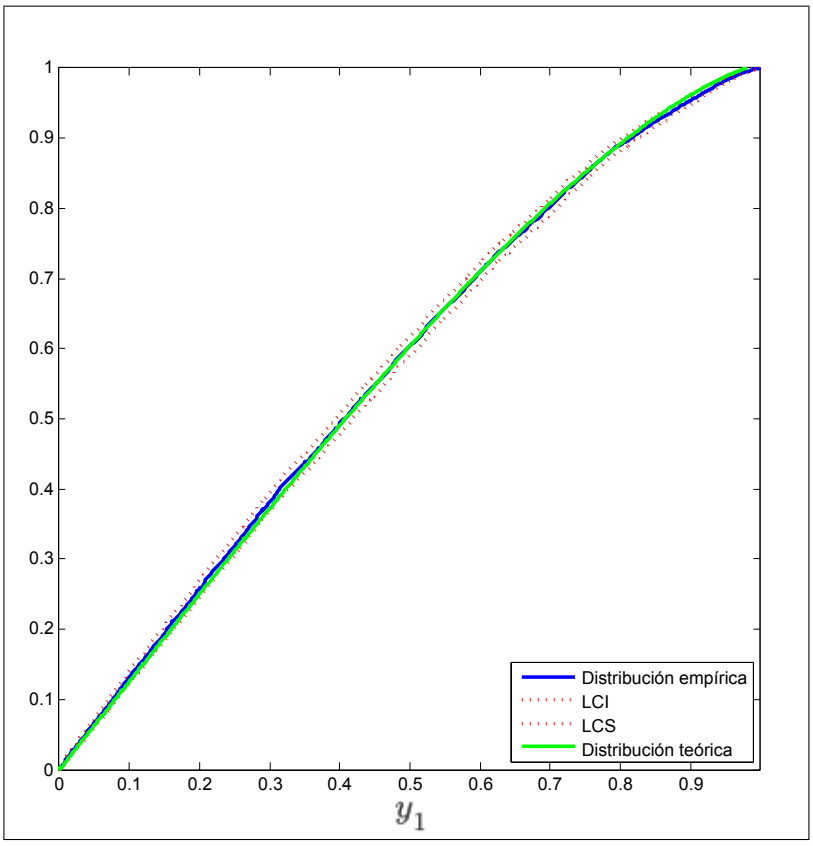


FIGURA 5.2. Función de distribución teórica vs. empírica generación de Y_1.

El gráfico 5.2 resulta al generar una secuencia al azar de tamaño $N = 5000$ para la variable aleatoria Y_1 con densidad $f_{g,1}$ mediante la aproximación de la función ϕ con una interpolante cúbica monótona. La elección particular de parámetros es $a_1 = 5, a_2 = 1, \beta_1 = 5, \beta_2 = 0$.

A continuación, como referentes se presentan metodologías alternativas para generar el vector aleatorio Y_1; los procedimientos presentados no dependen de la aproximación de la función ϕ^{-1}.

5.2.2. Método de aceptación y rechazo

El método de aceptación y rechazo es unos de los métodos más útiles para muestrear variables aleatorias con distribuciones arbitrarias. Este método es aplicable a distribuciones multivariadas de naturaleza discreta o continua; cabe anotar que el método pierde eficiencia a medida que la dimensión de la variable simulada crece [21].

Deseamos simular una variable aleatoria X con densidad $f(x)$. Supongamos que tenemos un método para muestrear eficientemente una variable aleatoria Y con densidad $g(y)$. Adicionalmente asumamos que es posible acotar la razón $f(x)/g(x)$ por una constante $c > 0$; $c = \sup_x\{f(x)/g(x)\}$.

El método de aceptación y rechazo se basa en el siguiente algoritmo [21]:

1. Genere X con densidad $g(x)$.
2. Genere U con densidad uniforme en el intervalo $(0,1)$.
3. Si $U \leq f(x)/(cg(x))$ retorne X; en otro caso retorne al paso 2.

Es decir generar X con densidad $g(x)$ y aceptarlo con probabilidad $f(x)/(cg(x))$; de lo contrario rechazar X e intentar nuevamente. La eficiencia del método de aceptación y rechazo se define como $P(U \leq f(X)/(cg(X)))$, y resulta ser: $1/c$, ver [21].

Específicamente si queremos simular la variable aleatoria Y_1 cuya densidad hemos denotado por $f_{g,1}(y_1)$, podríamos usar el algoritmo de aceptación y rechazo tomando $f(x) = f_{g,1}(x)$ y $g(x) = (a_1/(\beta_1 - \beta_2))\,\mathbf{1}_{\{x\in(\beta_2/a_1,(\beta_1/a_1)\}}(x)$. Es decir la variable que muestrearemos eficientemente es uniforme en el intervalo $(\beta_2/a_1, \beta_1/a_1)$.

Para determinar la constante c en el algoritmo, veamos que el máximo de la función $f_{g,1}(x)$ que denotaremos f^* es $f_{g,1}(0)$ si $0 \in (\beta_2/a_1, \beta_1/a_1)$ o máx$(f_{g,1}(\beta_2/a_1), f_{g,1}(\beta_1/a_1))$ en caso contrario. Dado $c = ((\beta_1 - \beta_2)/a_1) f^*$ teniendo en cuenta que $g(x) = (a_1/(\beta_1 - \beta_2))$ en el intervalo $(\beta_2/a_1, \beta_1/a_1)$, es claro que $f(x) \leq cg(x)$.

5.2.3. Algoritmo de Hastings

El algoritmo de Hastings es una de las técnicas más populares usadas por estadísticos. Principalmente se usa para simular variables aleatorias con distribuciones poco manejables. Dada una densidad de probabilidad particular $\pi(\mathbf{x})$ definida sobre una región S, y una densidad de probabilidad arbitraria $q(\mathbf{x} \mid \mathbf{y})$ sobre $S \times S$; considere el siguiente algoritmo, ver [27]:

1. Escoja un punto inicial $X_0 \in S$.
2. Sobre la iteración t:
   (a) Genere $\dot{X}$ de la distribución $q(\mathbf{x} \mid \mathbf{x_{t-1}})$.
   (b) Compute
   $$\rho\left(X_{t-1}, \dot{X}\right) = \text{mín}\left\{1, \frac{\pi\left(\dot{X}\right) q\left(X_{t-1} \mid \dot{X}\right)}{\pi\left(X_{t-1}\right) q\left(\dot{X} \mid X_{t-1}\right)}\right\}.$$
   (c) Tome
   $$X_t = \begin{cases} \dot{X} & \text{con probabilidad } \rho\left(X_{t-1}, \dot{X}\right) \\ X_{t-1} & \text{con probabilidad } 1 - \rho\left(X_{t-1}, \dot{X}\right). \end{cases}$$
3. Repita el paso 2 hasta alcanzar el equilibrio.

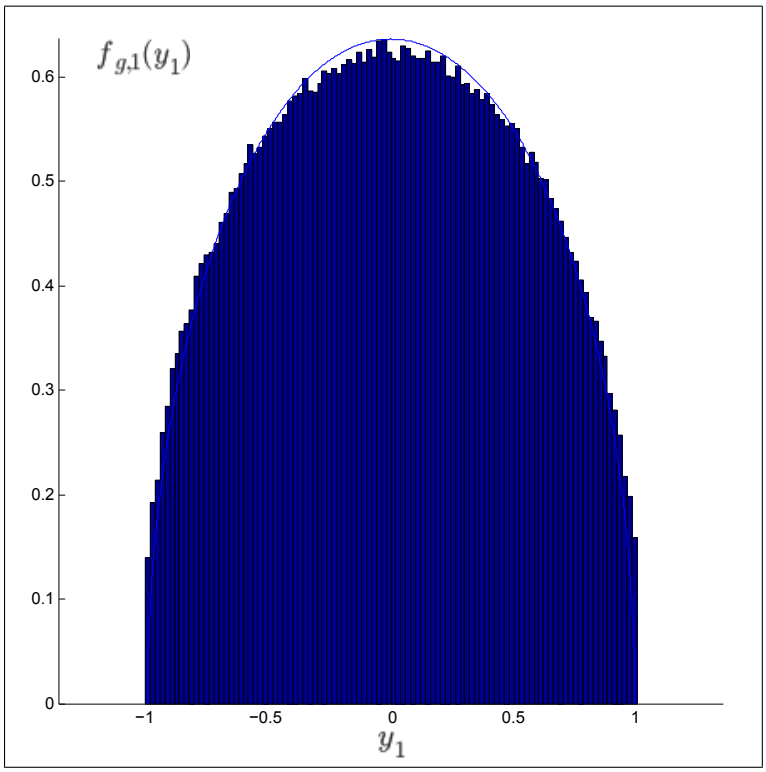


FIGURA 5.3. Generación de puntos con distribución $f_{g,1}$.

Es decir, en la iteración t, el muestreador de Hastings escoge una muestra $\dot{X}$ de la distribución instrumental q, la cual puede depender de la última variable muestreada X_{t-1}. El algoritmo decide aceptar la nueva variable aleatoria $\dot{X}$, si en algún sentido es más factible que provenga de la distribución estacionaria $\pi(\mathbf{x})$ que la muestra anterior X_{t-1}. El algoritmo de Hastings induce una cadena de Markov con distribución estacionaria $\pi(\mathbf{x})$, sin importar la escogencia de la distribución instrumental q. En la práctica se escoge q fácil de muestrear, y cercana a la distribución estacionaria $\pi(\mathbf{x})$.

En `MATLAB`, por ejemplo, se realiza la simulación mediante el algoritmo de Hastings usando la función `mhsample`.

El gráfico 5.3, muestra un histograma para la generación de la variable Y_2 cuya distribución se ha representado con $f_{g,2}$, y se superpone la densidad teórica; la función instrumental es una distribución normal truncada. Los parámetros específicos de la muestra son $a_1 = 5, a_2 = 1, \beta_1 = 5, \beta_2 = -5$.

En el siguiente capítulo realizará aplicaciones médicas de estos algoritmos y se presentarán tablas de comparación de su desempeño computacional.

CAPÍTULO 6

Aplicación en imágenes médicas

La imageneología médica es el conjunto de técnicas y procesos usados para crear imágenes digitales del cuerpo humano o partes de él, con propósitos clínicos o para la ciencia médica. Las imágenes digitales se componen de pixeles individuales, para los cuales los valores de color o luminosidad son dados de forma discreta. Ellas pueden ser eficientemente procesadas, objetivamente evaluadas, y puestas a disposición en varios lugares al mismo tiempo por medio de las redes de comunicación y protocolos apropiados, tales como el Picture Archiving and Communication Systems (PACS) y el protocolo Digital Imaging and Communications in Medicine (DICOM). Específicamente, DICOM es el estándar mundialmente reconocido para fines de intercambio de imágenes médicas; creado para su manejo, almacenamiento, impresión y transmisión [28].

Desde el descubrimiento de los rayos X por Wilhelm Conrad Röntgen en 1895, las imágenes médicas se han convertido en un componente principal de diagnóstico, planificación de tratamientos clínicos y procedimientos, así como también estudios de seguimiento.
Debido al uso creciente de sistemas de imagen digital directa para diagnóstico médico, el procesamiento de imágenes médicas ha llegado a ser preponderante en el cuidado de la salud [28].

Las imágenes médicas se adquieren usando una variedad de técnicas y dispositivos, incluyendo radiografía convencional, tomografía computarizada, imágenes por resonancia magnética, ultrasonido, y medicina nuclear [28]. En este trabajo enfatizamos en el uso de imágenes por resonancia magnética, ver figura 6.1.

6.1. Imagen por resonancia magnética (IRM)

El proceso de obtención de imagen por resonancia magnética es una técnica de imageneología médica no invasiva, basada en el fenómeno de la resonancia magnética nuclear, y usada en radiología para visualizar detalladamente las estructuras internas del cuerpo humano.

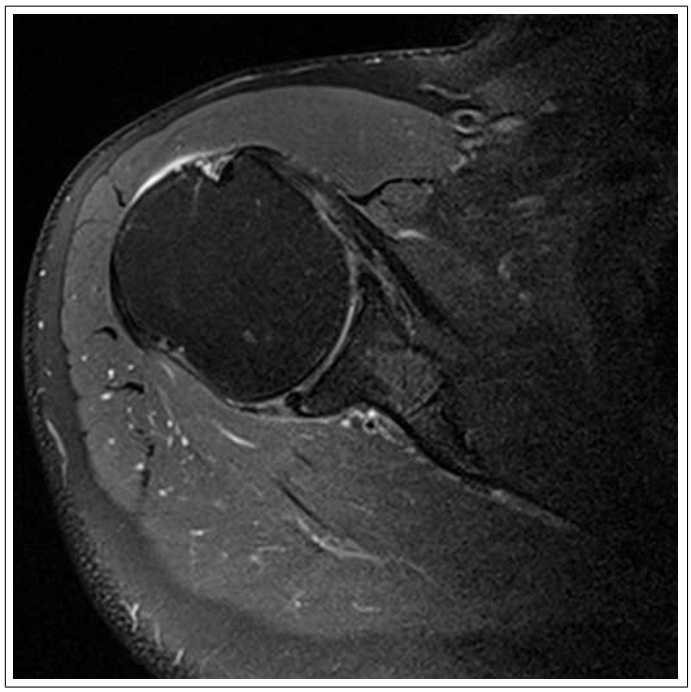


FIGURA 6.1. Imagen por resonancia magnética de un hombro.

Descripción del proceso de RM

Las imágenes por resonancia magnética usan las propiedades naturales del hidrogeno que, como parte de agua o lípidos constituye el 75-80 % del cuerpo humano. Las propiedades más importantes son la densidad de protones (a menudo abreviado en ingles PD), y dos tiempos característicos denominados tiempo de relajación espín-medio y tiempo de relajación espín-espín, denotados T1 y T2 respectivamente. La densidad de protones está relacionada con el número de átomos de hidrógeno en un volumen determinado; fluidos tales como LCR y sangre tienen mayor PD que el tendón y el hueso. Los tiempos de relajación describen el tiempo del tejido necesario para volver al equilibrio después de un pulso de radio frecuencias. En las imágenes de PD sobre las cuales centraremos nuestra atención, PD altos dan altas intensidades de señal que a su vez se asocian con píxeles brillantes en la imagen, ver [29].

En un equipo de IRM, el componente principal es un imán capaz de generar un campo magnético constante de gran intensidad. El campo magnético constante tiene la función de alinear los momentos magnéticos de los núcleos atómicos en dos direcciones, paralela (vectores de igual dirección), y anti-paralela (vectores en sentido opuesto) con respecto al campo magnético. La intensidad del campo y el momento magnético del núcleo determinan la frecuencia de resonancia de los núcleos, así como la proporción de núcleos que se encuentran en cada uno de los dos estados.

La proporción en cada estado está gobernada por la ley estadística de Maxwell-Boltzmann. Esto quiere decir que, para un átomo de hidrógeno y un campo magnético de 1.5 teslas a temperatura ambiente, solamente un núcleo adicional por cada millón se orientará paralelamente en relación a los núcleos orientados anti-paralelamente.

La gran cantidad de núcleos presente en un pequeño volumen hace que esta pequeña diferencia estadística sea suficiente para ser detectada.

El siguiente paso consiste en emitir la radiación electromagnética a una frecuencia de resonancia. Algunos de los núcleos que se encuentran en el estado paralelo o de baja energía cambiarán al estado anti-paralelo o de alta energía, y al cabo de un corto período de tiempo, reemitirán la energía que podrá ser detectada usando el instrumental adecuado. Debido a que el rango de frecuencias es el de las radiofrecuencias para los imanes del equipo, el instrumental suele consistir en una bobina que hace las veces de antena, receptora y transmisora, un amplificador y un sintetizador de radiofrecuencias.

Teniendo en cuenta que el imán principal genera un campo constante, todos los núcleos que posean el mismo momento magnético tendrán la misma frecuencia de resonancia. Esto significa que una señal que ocasione una resonancia magnética en estas condiciones podrá ser detectada, sin información espacial, o sea, sobre la ubicación del fenómeno dónde se produce la resonancia.

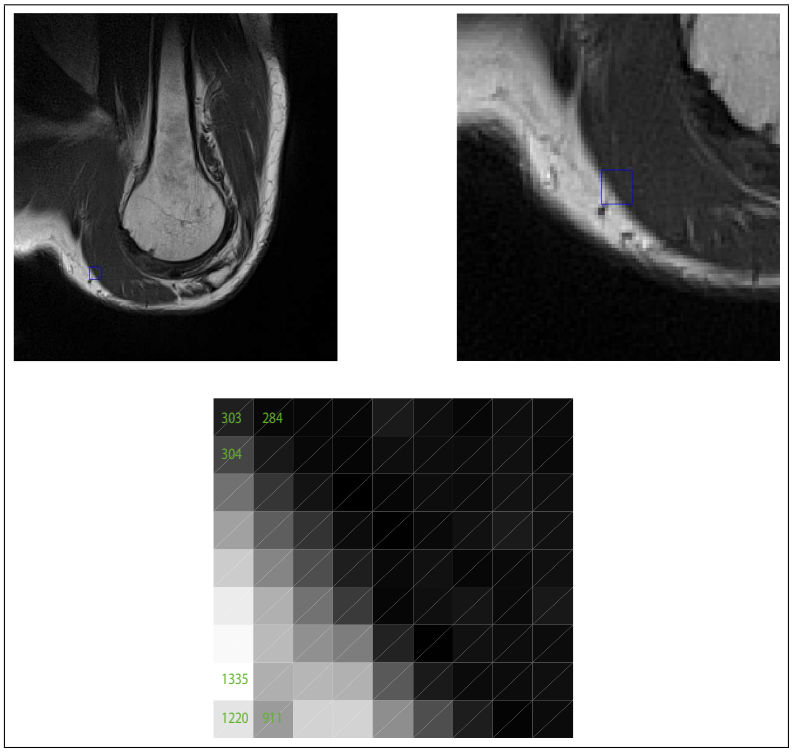


FIGURA 6.2. Acercamiento y codificación de una imagen por resonancia magnética de un hombro.

Para resolver este problema se añaden bobinas, llamadas bobinas de gradiente. Cada una de las bobinas genera un campo magnético de una cierta intensidad con una frecuencia controlada. Estos campos magnéticos alteran el campo magnético ya presente, y por tanto, la frecuencia de resonancia de los núcleos. Utilizando tres bobinas ortogonales es posible asignarle a cada región del espacio una frecuencia de resonancia diferente, de manera que cuando se produzca una resonancia a una frecuencia determinada será posible determinar la región del espacio de la que proviene.

Finalmente, por medio del proceso de resonancia magnética se obtiene una matriz M de números, codificados como niveles de gris para conformar los píxeles de la imagen. Al mínimo, y al máximo nivel de gris en M; se le asocia los colores negro y blanco respectivamente. La figura 6.2 muestra un ejemplo para codificación de colores en una imagen por resonancia magnética. En `MATLAB,` por ejemplo, la función `dicominfo` aplicada sobre un archivo DICOM permite obtener información detallada sobre el paciente y la imagen asociada, y la función `dicomread` genera la matriz M con la codificación de niveles de gris.

Interpolación trilineal

El método de interpolación trilineal consiste en interpolar linealmente el valor de una función C en un punto (x, y, z) dentro de un cubo unitario, usando los valores de la función en los vértices. Consideremos un cubo unitario representado así:

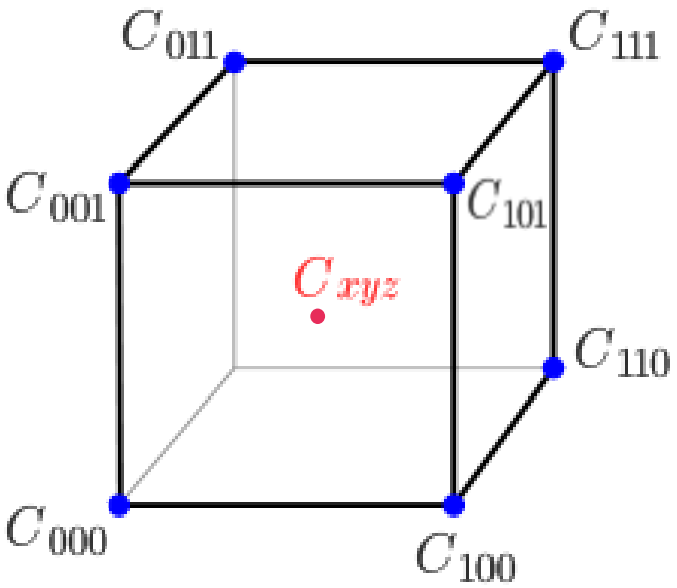


FIGURA 6.3. Interpolación trilineal.

Si se denota C_{000}, $C_{001}, \cdots, C_{111}$ los valores de la función en cada vértice, el valor de la función en la posición (x, y, z) dentro del cubo será denotado C_{xyz}, y de acuerdo a este método estará dado por, ver [30]:

$$C_{xyz} = (1-x)(1-y)(1-z)C_{000} + (1-x)(1-y)zC_{001} + \cdots + xyzC_{111}.$$

En la aplicación particular a imágenes médicas, la función C eventualmente es el nivel de gris de la imagen en una posición espacial.

6.2. Extracción de información sobre el manguito rotador

El manguito rotador es un conjunto de músculos y tendones conectados a la cabeza del húmero cuya función es la movilidad y estabilidad del hombro. Cuando uno o más de los tendones del manguito rotador está desgarrado, el tendón ya no se adhiere plenamente a la cabeza del húmero, y se cataloga como una ruptura en el manguito rotador. Una ruptura en el manguito rotador ocasiona debilidad en el hombro. Esto implica que muchas de las actividades diarias, como peinarse o vestirse, pueden llegar a ser dolorosas y difícil de hacer. La mayoría de los desgarros ocurren en el músculo y en el tendón supraespinoso, pero eventualmente otras partes del manguito rotador pueden también verse involucradas, ver [31].
Creemos que estudiar este problema es pertinente pues como se anota en [32], el dolor de hombro es una queja frecuente en los servicios de salud, se estima que dicha afección constituye cerca del 16 % de todas las quejas músculoesqueléticas y representa la tercera causa de consulta en centros de atención primaria. La prevalencia de dicho síntoma en estudios poblacionales puede oscilar entre el 12 % y 22,3 %, siendo las mujeres, personas mayores de 50 años, diabéticos, fumadores y obesos aquellas personas con mayor riesgo de padecerlo.

Anatomía del manguito rotador

Los tendones del supraespinoso, infraespinoso, subescapular y redondo menor comprenden el manguito rotador, ver [2].

- El músculo **subescapular** se origina de la fosa subescapular a lo largo de la cara anterior de la escapula. Las fibras del músculo del subescapular convergen para formar varios tendones con apariencia de abanico en la unión miotendinosa. Las fibras profundas del tendón subescapular se combinan y refuerzan la cápsula anterior de la articulación glenohumeral. La mayoría de las fibras del tendón subescapular se insertan encima de la tuberosidad menor. Sin embargo, las fibras más superficiales se extienden a la tuberosidad mayor y en conjunto con las fibras del ligamento humeral transverso forman la raíz del surco bicipital o intertubercular. El subescapular es inervado por los nervios subescapular superior e inferior (C5, C6 y C7) y es irrigado por la arteria subescapular. El subescapular se separa de la cavidad glenoidea por el receso subescapular.

- El músculo **supraespinoso** consta de los vientres anterior y posterior. El vientre supraespinoso anterior es una estructura fusiforme que se origina por completo de la fosa supraespinosa, y que se convierte en un tendón tubular grueso comprendiendo el 40 % anterior del tendón supraespinoso. El vientre supraespinoso posterior es una estructura unipenada más pequeña que se origina principalmente de la espina escapular y el cuello glenoideo, y que se convierte en un tendón delgado y plano que comprende el 60 % posterior del tendón supraespinoso. Las fibras del músculo, diferente al subescapular, están orientadas lateralmente. El tendón se inserta sobre la cara superior de la tuberosidad mayor. En tendón supraespinoso es bordeado por el acromion, bursa subacromial y subdeltoidea (superiormente), por la cápsula de la articulación (inferiormente).

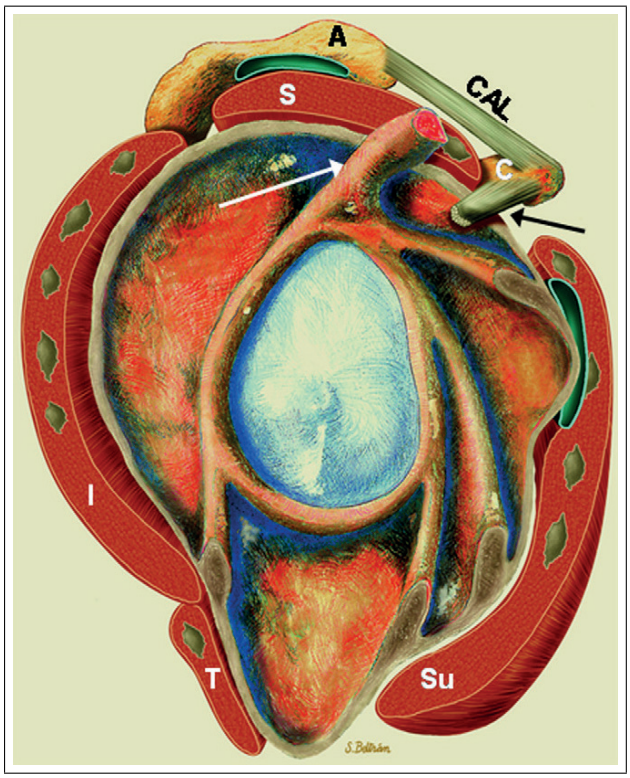


FIGURA 6.4. Representación de una vista lateral del hombro: manguito rotador, S: supraespinoso, I: infraespinoso, Rm: redondo menor, Su: subescapular); A: acromion; C: ligamento coracohumeral (flecha negra); CAL: ligamento coracoacromial; cabeza del tendón del biceps (flecha blanca).

Anteriormente, el tendón supraespinoso se combina con el ligamento coracohumeral y posteriormente se une con el tendón infraespinoso. El supraespinoso es irrigado por la arteria supraescapular y es inervado por el nervio supraescapular (C5 y C6), que pasa a través de la muesca supraescapular y que cursa oblicua y lateralmente a lo largo de la superficie inferior del supraespinoso.

- El músculo **infraespinoso** se origina de la fosa infraespinosa de la escapula. Al igual que el subescapular, las fibras del músculo convergen para formar varios tendones con apariencia de abanico en la unión miotendinosa. El tendón se inserta encima de la cara media de la tuberosidad mayor. El infraespinoso se separa del deltoides suprayacente por capa fascial. El margen inferior del infraespinoso se combiana con la cápsula articular. Las fibras distales del nervio subescapular, una vez que pasan a través de un túnel óseo fibroso en la muesca espinoglenoidea inervan el músculo infraespinoso. Se irriga por las arterias supraescapular y la arteria escapular circunfleja.

- El músculo **redondo menor** se origina entre los dos tercios superiores del borde lateral de la escapula y se une en la porción posterior de la cápsula de la articulación glenohumeral. Este músculo inserta en la cara inferior de la tuberosidad mayor. El redondo menor es inervado por el nervio axilar (C5, C6) y es irrigado por las arterias subescapular y escapular circunfleja.

Evaluación mediante RM de anomalías en el manguito rotador

Las imágenes por resonancia magnética proveen la información más completa sobre la estructura del tendón. Pueden exhibir el tamaño del desgarro, la forma del desgarro, el grado de retracción del tendón, extensión del desgarro a estructuras adyacentes, la cualidad de los fragmentos restantes del tendón y la atrofia muscular asociada o anormalidades óseas, si los hubiere. Estos factores pueden influir en la selección del tratamiento y ayudar a determinar el pronóstico. Las imágenes por resonancia magnética también pueden determinar otras causas de dolor en el hombro las cuales podrían ser similares a los desgarros en el manguito rotador.

Las imágenes oblicuas coronales y sagitales son ideales para la evaluación de rupturas parciales de los tendones supraespinoso e infraespinoso. Del mismo modo, las imágenes oblicuas axiales y sagitales son las mejores para la evaluación de los tendones subescapular y redondo menor. El diagnóstico de este tipo de desgarro se sugiere por el aumento de la intensidad de la señal dentro del manguito rotador atravesando parcialmente la sustancia del manguito rotador. Por lo general, hay alteraciones morfológicas en la porción distal de los tendones, ver [33].

Las imágenes por resonancia magnética juegan un papel importante en la identificación de rupturas en el manguito rotador, de hecho muchos cirujanos confían en imágenes por resonancia magnética para apoyar la toma de decisiones y la planificación prequirúrgica de pacientes con dolor en el hombro, ver [2]. Sin embargo, la resonancia magnética requiere un análisis cuidadoso de los patrones de intensidad de la señal y de las alteraciones morfológicas en la porción distal de los tendones, en especial del tendón supraespinoso. Se ha enfatizado el valor de los signos secundarios de las anomalías del manguito rotador en la RM de estos casos, pero estos signos carecen de especificidad. Incluso con este análisis los resultados de los exámenes con RM no son tan buenos desde el punto de vista diagnóstico como en los casos de ruptura total del manguito rotador, ver [33].

Las metodologías de generación de puntos sobre superficies tanto aleatorias como determinísticas pueden ser aplicadas para detectar una ruptura parcial del tendón supraespinoso, por ejemplo, dado un volumen DICOM[1] de 20 rodajas asociadas al escaneo de un hombro derecho mediante una resonancia magnética simple, y dada una segmentación del húmero; con el propósito de visualizar la ruptura, se texturizan con base en los niveles de gris en el volumen médico, superficies tanto esféricas como elipsoidales. Los parámetros de la superficie se estiman mediante la técnica de mínimos cuadrados ordinarios aplicada sobre los puntos de segmentación. Más aún, fijando el centro de la superficie elipsoidal se puede variar la longitud de los ejes para flexibilizar la visualización en el volumen médico. Dar textura a una superficie requiere las siguientes etapas:

1. Generar puntos de forma determinística o aleatoria sobre la superficie.
2. Asignar a cada punto un nivel de gris mediante la técnica interpolación trilineal.
3. Desplegar la superficie sombreada.

[1]El paciente asociado al volumen médico utilizado es el profesor Marco Paluszny.

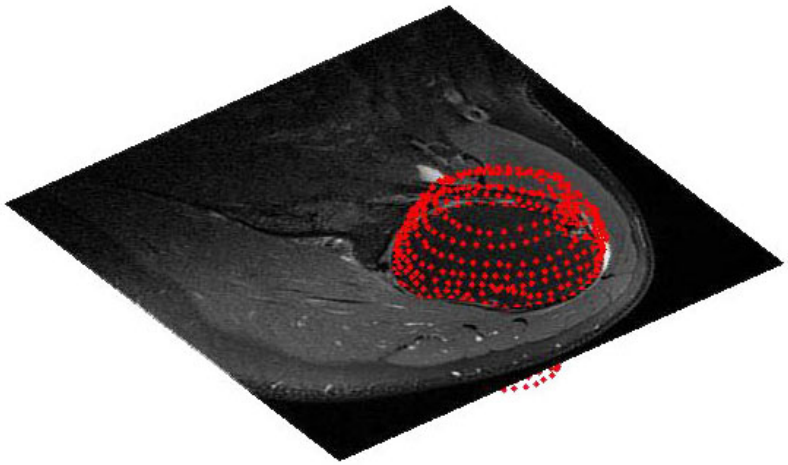


FIGURA 6.5. Segmentación del húmero.

En la etapa de generación de puntos se dispone de varias opciones. Desde el punto de vista determinístico, se pueden generar puntos sobre la superficie representada como parche de Bézier racional usando tanto el algoritmo de de Casteljau como la evaluación directa con polinomios de Bernstein. Desde el punto de vista aleatorio, si la superficie es la esfera los puntos se pueden generar usando el método de la transformada inversa para 2 dimensiones; por otra parte si la superficie es un elipsoide de revolución se pueden generar puntos al combinar el método de la transformada inversa en una dimensión con métodos como: aproximación interpolante localmente monótona, algoritmo de aceptación y rechazo, y algoritmo de Hastings. En el caso del elipsoide arbitrario se podría resolver usando el algoritmo de Hastings para el vector conjunto Y, pero nuestros experimentos indican que no habría ganancia en términos de costo computacional respecto a las metodologías determinísticas por lo tanto no se presentaran comparaciones.

Creemos que es sensato medir el tiempo de cómputo promedio pues las metodologías de visualización se implementan generalmente en software (en lugar de hardware). Consideramos que el tiempo que tarda el proceso para generar un número de puntos razonable sobre las superficies estudiadas, medido en un ordenador de especificaciones estándar, refleja la habilidad del algoritmo para responder eficientemente a plataformas potencialmente desarrollables sin la necesidad de utilizar recursos especiales.

En [8] se proveen expresiones específicas para los pesos $w_{\mathbf{i}}$ y la red de control $\mathbf{b_i}$ en la representación de una esfera unitaria como un parche triangular de Bézier racional de grado 4, en general un parche esférico tendrá grado mayor, por lo tanto tiempo de computo mayor. Teniendo en cuenta la propiedad de invariancia afín de un parche triangular de Bézier, el elipsoide que resulta al escalar arbitrariamente la esfera unitaria se puede representar con los mismos pesos $w_{\mathbf{i}}$ y con la red de control que resulta al escalar $\mathbf{b_i}$. En particular, con la transformación apropiada, es posible generar puntos sobre la superficie del ajuste de mínimos cuadrados a la segmentación del húmero, sea el caso esférico o el caso elipsoidal.

En la tabla 6.1 se presenta la comparación del tiempo promedio de cómputo medido en segundos para la generación de puntos sobre el octante de una esfera.

Método	***n=1035***	***n=10011***	***n=50086***
Transformada inversa	0.00017	0.00099	0.00399
Evaluación directa	0.00349	0.02603	0.12588
Algoritmo de de Casteljau	0.00289	0.04363	0.21224

TABLA 6.1. Tiempo promedio de cómputo medido en segundos en las metodologías para generar puntos sobre el octante la esfera.

Considerando el octante de un elipsoide de revolución, en la tabla 6.2 se muestra la comparación del tiempo promedio de cómputo medido en segundos para la generación de puntos sobre dicha superficie.

Método	***n=1035***	***n=10011***	***n=50086***
Transformada inversa aproximada	0.00132	0.00335	0.01089
Aceptación rechazo	0.01396	0.12518	0.63603
Hastings	0.40632	2.22516	10.3851
Evaluación directa	0.00352	0.02610	0.12654
Algoritmo de de Casteljau	0.00290	0.04405	0.21291

TABLA 6.2. Tiempo promedio de cómputo medido en segundos en las metodologías para generar puntos sobre el octante del elipsoide de revolución.

Para asignar un nivel de gris haciendo uso de la técnica de interpolación trilineal a cada punto generado sobre la superficie, en `MATLAB` disponemos de la función interna `interp3`. En la tabla 6.3 se muestran los tiempos de cómputo promedio medido en segundos en la asignación del nivel de gris variando el número de puntos interpolados.

Método	***n=1035***	***n=10011***	***n=50086***
Interpolación trilineal	0.08295	0.08791	0.09304

TABLA 6.3. Tiempo promedio de cómputo medido en segundos en la asignación del nivel de gris de acuerdo al número de puntos.

Finalmente en la etapa de despliegue de la superficie sombreada se puede usar en `MATLAB` la función interna `trisurf`. Los tiempos de despliegue promedio medido en segundos variando el número de puntos considerados se muestran en la tabla 6.4.

Método	***n=1035***	***n=10011***	***n=50086***
Trisurf	0.01023	0.04110	0.20923

TABLA 6.4. Tiempo promedio medido en segundos de despliegue de la superficie sombreada de acuerdo al número de puntos.

A continuación presentamos un gráfico esquemático que muestra cómo se sombrean los puntos de acuerdo al tipo de superficie elegido.

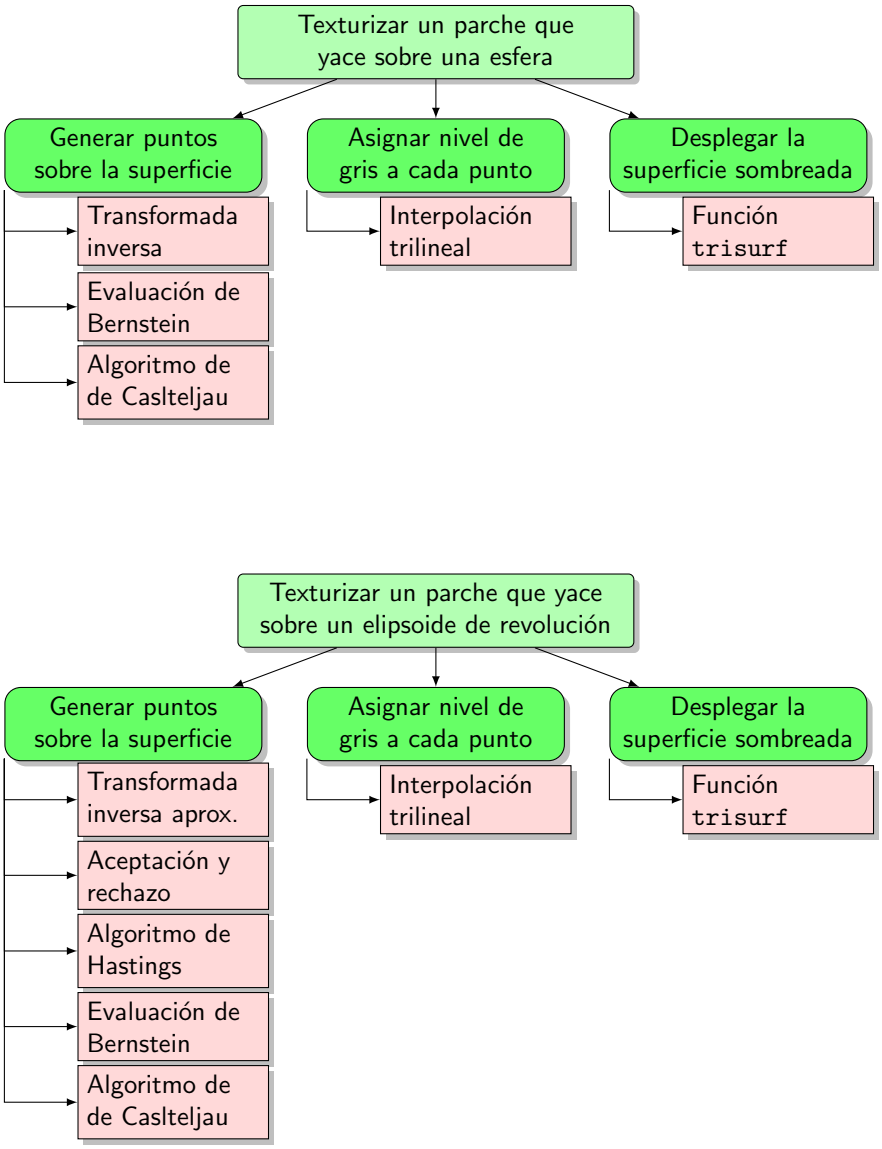


FIGURA: Metodologías para texturizar un parche que cubre la cabeza humeral dependiendo de si el parche yace sobre una esfera o un elipsoide de revolución.

El despliegue de una superficie esférica sobre la cabeza del húmero, sombreada con los niveles de gris provistos en el volumen DICOM, se muestra en la figura 6.6. Esta superficie representa un esquema de visualización en 3D que permite de manera alternativa diagnosticar la ruptura parcial en el tendón supraespinoso[2].

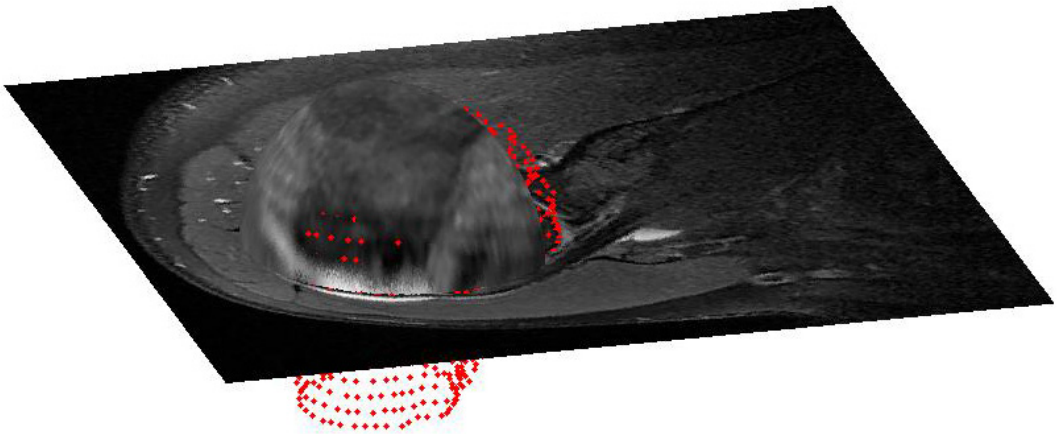


FIGURA 6.6. Superficie sombreada sobre la cabeza del húmero.

[2]El tipo exacto de imagen utilizado es una resonancia magnética simple del hombro derecho, cuyo protocolo fue el siguiente; **posición:** head first supine , **contraste:** proton density, **secuencia de pulso:** turbo spin echo, **técnica de saturación:** fat-saturation. Para información detallada sobre protocolos y posicionamiento del paciente ver [34]

Conclusiones

Se estudiaron metodologías para extraer información de imágenes por resonancia del tejido en el manguito rotador; el estudio fue particularmente motivado en la idea de definir un método de visualización alternativo a la inspección de las imágenes planas que es lo usual en la mayoría de los casos y así contribuir en la detección de rupturas parciales en el tendón supraespinoso que ocasionalmente resulta difícil. Hasta donde sabemos, estas metodologías aún no han sido consideradas por especialistas del área. Básicamente se propuso considerar familias de parches triangulares elipsoidales para cubrir la cabeza del húmero cerca de la zona afectada, generar puntos sobre los parches, posteriormente asignar un nivel de gris a cada punto en el parche con la información de la resonancia usando la técnica de interpolación trilineal, y así desplegar la superficie sombreada.

En la generación de puntos para texturizar cada parche se propuso una nueva metodología que garantiza la distribución uniforme pero aleatoria sobre la superficie evitando la acumulación sistemática de puntos en áreas específicas. Adicionalmente el costo computacional de la técnica propuesta en este trabajo es significativamente menor en comparación a metodologías alternativas de naturaleza tanto determinística como aleatoria. La metodología propuesta se limita a parches sobre esferas y elipsoides de revolución, en principio no se ha considerado la extensión de la técnica a otras superficies, y para definir la superficie a texturizar partimos de una segmentación del húmero dada.

Las rutinas que permiten reproducir cada una de las metodologías puestas a consideración y cada una de las gráficas incorporadas fueron implementadas en `MATLAB` y se anexan en este trabajo. La consecución de imágenes médicas específicas puede ser laboriosa y en este trabajo usamos solamente una secuencia de imágenes del hombro. La finalidad este trabajo no es la construcción de una herramienta de evaluación clínica ni tampoco validar la eficiencia de la propuesta en la detección de una enfermedad particular, sino más bien, proponer una alternativa de visualización que podría eventualmente ser útil desde el punto de vista médico. Tanto en la aplicación de la metodología propuesta a la secuencia dada como en el despliegue de sus imágenes planas asociadas, la ruptura parcial existente en el tendón supraespinoso queda en evidencia.

A futuro con la ayuda de especialistas en el área de imágenes diagnosticas esperamos obtener nuevas imágenes por resonancia magnética del manguito rotador, reproducir el esquema de visualización propuesto en este trabajo y evaluar su utilidad desde el punto de vista médico.

APÉNDICE A

Interpolación de Hermite cúbica por trozos

Consideramos primero el problema de interpolación general. Para la construcción del interpolante de Hermite nos basamos en [11].

Una función $f(u) \in \mathcal{C}^n$ se puede aproximar por un polinomio p de grado n interpolando f en $n+1$ abscisas $u_0 < u_1 < \ldots < u_n$. Un resultado clásico en análisis numérico establece que la diferencia entre p y f en el intervalo $[u_0, u_n]$ se puede expresar como:

$$p(u) - f(u) = \frac{f^{(n)}(v)}{(n+1)!}(u_0 - u)(u_1 - u)\ldots(u_n - u),$$

donde $v = v(u)$ está en $[u_0, u_n]$. Por lo tanto usualmente el error se hace pequeño cuando cada $|u_{k+1} - u_k|$ se acerca a 0. Sin embargo un interpolador de grado más alto de la función $f(u)$ no necesariamente resulta en una mejor aproximación de esta función, este problema se conoce como fenómeno de Runge. Para ilustrar, consideremos la interpolación en $n+1$ puntos equidistantes de la función $f(u) = (1 + 25u^2)^{-1}$; con un polinomio de grado $k \leq n$ el error oscila cerca de -1 y 1. Se puede mostrar que el error de aproximación aumenta sin cota cuando el grado del polinomio crece [35]. Por esta razón es común el uso de funciones polinómicas a trozos de grado bajo.

Por ejemplo, las cúbicas son usadas con mucha frecuencia debido a que tienen grado bajo, y a que permiten suficiente flexibilidad para muchas aplicaciones. A continuación se describe la interpolación de Hermite con polinomios cúbicos a trozos.

Considerando m puntos $\mathbf{p}_1, \mathbf{p}_2, \ldots, \mathbf{p}_m$, y derivadas $\mathbf{d}_1, \mathbf{d}_2, \ldots, \mathbf{d}_m$, correspondientes a valores del parámetro $u_1 < u_2 < \ldots < u_m$, entonces existe una única cúbica por trozos, continuamente diferenciable $\mathbf{s}(u)$ sobre $[u_1, u_m]$ tal que

$$\mathbf{s}(u_i) = \mathbf{p}_i, \quad y \quad \mathbf{s}'(u_i) = \mathbf{d}_i.$$

Su representación está dada por

$$\mathbf{s}(u) \quad = \quad \sum_{0}^{3} \mathbf{b}_{3i+j} B_j^3(t_i), \quad u \in [u_i, u_{i+1}],$$

donde $t_i = (u - u_i) / (u_{i+1} - u_i)$ representa el parámetro local sobre $[u_i, u_{i+1}]$ para $i = 1, 2, \ldots, m-1$, y

$$\begin{aligned} \mathbf{b}_{3i} &= \mathbf{p}_i \\ \mathbf{b}_{3i+1} &= \mathbf{p}_i + \mathbf{d}_i \Delta u_i / 3 \\ \mathbf{b}_{3i+2} &= \mathbf{p}_{i+1} - \mathbf{d}_{i+1} \Delta u_i / 3. \end{aligned}$$

Frecuentemente las derivadas $\mathbf{d}_i$ no están dadas directamente, sino deben ser estimadas de los datos.

APÉNDICE B

Implementación MATLAB

En este anexo presentamos los códigos de las rutinas implementadas en MATLAB para realizar nuestra experimentación.

B.1. SphereRandomC.m

Es la implementación de la metodología descrita en 5.1.

Función principal

```
function[X1,X2,X3]=SphereRandomC(n,a1,alpha2,alpha1,beta2,beta1,c)
%FUNCION: Considerando el sistema coordenado x1 x2 x3, tomamos la semi-esfera de
%radio a1  en el dominio D definida por: x3=a1*sqrt(1-(x1/a1)^2-(x2/a1)^2) en D=
%{(x1,x2);beta2<x1<beta1,alpha2*a1*sqrt(1-(x1/a1)^2)<x2<alpha1*a1*sqrt(1-(x1/a1)^2)}.
%Los parámetros satisfacen 0<a1,-a1<=beta2<beta1<=a1,-1<=alpha2<alpha1<=1.
%Esta función genera n puntos uniformemente distribuidos sobre dicha superficie
%y la traslada con el vector c. Usa el método de la trasformada inversa.
%
%ENTRADAS
%n  =     Cantidad de puntos por generar.
%a1 =     Radio de la esfera.
%alpha2 = Parámetro de la curva límite inferior en el dominio D.
%alpha1 = Parámetro de la curva límite superior en el dominio D.
%beta2 =  Parámetro de la curva límite izquierdo en el dominio D.
%beta2 =  Parámetro de la curva límite derecho en el dominio D.
%c     =  centro de la esfera.
%
%SALIDAS
%[X1,X2,X3] = Coordenadas X1,X2,X3 para n vectores aleatorios con distribución
%uniforme sobre la esfera en el dominio D.
%
%Chequeo de parámetros.
if((round(n)>0)&&(a1>0)&&(abs(beta2)<=a1)&&(abs(beta1)<=a1)&&(beta2<beta1)&&...
  (abs(alpha2)<=1)&&(abs(alpha2)<=1)&&(alpha2<alpha1))
%Si los parámetros están bien especificados...
    %Genera dos vectores aleatorios de tamaño n uniformes en el intervalo (0,1).
```

```
    U=rand(n,2);
    %Genera  n muestras de  las variables aleatorias auxiliares Y1,Y2 con
    %distribuciones fg1 y fg2.
    Y1=((beta1-beta2)*U(:,1)+beta2)/a1;
    Y2=sin(asin(alpha2)*(1-U(:,2))+asin(alpha1)*U(:,2));
    %Genera  n muestras de  las variables aleatorias X1,X2,X3.
    X1=a1*Y1;X2=a1*Y2.*sqrt(1-Y1.^2);X3=sqrt(a1^2-X1.^2-X2.^2);
    %Traslada de acuerdo al centro c.
    X1=X1+c(1);X2=X2+c(2);X3=X3+c(3);
%Si los parámetros no están bien especificados, muestre mensaje de error.
else msgbox('Error ! Verificar Condiciones en los Parametros')
end
end
```

B.2. EllipsoidRandomC.m

Es la implementación de la metodología descrita en 5.2.1.

Funciones auxiliares

```
function [y]=Fsemicircle(t)
y=0.5+(1/pi)*t.*sqrt(1-(t.^2))+(1/pi)*asin(t);
end
```

Función principal

```
function [X1,X2,X3,Y1]=EllipsoidRandomC(n,a2,a1,beta2,beta1,alpha2,alpha1,npart,c)
%FUNCION: Considerando el sistema coordenado x1 x2 x3, tomamos el semi-elipsoide
%de semiejes a1,a2,a2 en el dominio D dado por: x3=a2*sqrt(1-(x1/a1)^2-(x2/a1)^2)
%en D={(x1,x2); beta2<x1<beta1 alpha2*a2*sqrt(1-(x1/a1)^2)<x2<alpha1*a2*...
%sqrt(1-(x1/a1)^2)}. Los parámetros satisfacen 0<a2=a3<=a1,-a1<=beta2<beta1<=a1,
%-1<=alpha2<alpha1<=1. Esta función genera n puntos uniformemente distribuidos
%sobre dicha superficie y la traslada con el vector c. Usa una aproximación al
%método de la transformada inversa en Y1, y transformada inversa en Y2.
%
%ENTRADAS
%n  =     Cantidad de puntos por generar.
%a1 =     Semi-eje x1.
%a2 =     Semi-eje x2,x3.
%alpha2 = Parámetro de la curva límite inferior en el dominio D.
%alpha1 = Parámetro de la curva límite superior en el dominio D.
%beta2 =  Parámetro de la curva límite izquierdo en el dominio D.
%beta2 =  Parámetro de la curva límite derecho en el dominio D.
%npart =  Número de particiones para Phi^-1
%c     =  Centro del elipsoide de revolución.
%
%SALIDAS
%[X1,X2,X3] = Coordenadas X1,X2,X3 para n vectores aleatorios con distribución
%uniforme sobre el elipsoide de revolución en el dominio D.
%
%Chequeo de parámetros.
%Si los parámetros están bien especificados...
    if((round(n)>0)&&(a2>0)&&(a1>0)&&(a1>a2)&&(abs(beta2)<=a1)&&...
       (abs(beta1)<=a1)&&(beta2<beta1)&&(abs(alpha2)<=1)&&(abs(alpha2)<=1)&&...
```

```
      (alpha2<alpha1))
    %Cálculo del parámetro delta1.
    delta1=1-(a2/a1)^2;
    %Límites de la variable auxiliar Y1*.
    r2=(sqrt(delta1)*beta2)/a1;r1=(sqrt(delta1)*beta1)/a1;
    %Evalúa la función de distribución semicircular en los límites de Y1*.
    Fr2=Fsemicircle(r2);Fr1=Fsemicircle(r1);
    %Discretiza el intervalo (r2,r1).
    t=transpose(linspace(r2,r1,npart));
    %Evalúa la función de distribución semicircular  en la discretización de
    %(r2,r1).
    Ft= Fsemicircle(t);
    %Genera n números aleatorios con distribución uniforme en (0,1).
    U=rand(n,1);
    %Genera n números aleatorios con distribución uniforme en (Fr2,Fr1).
    V=rand(n,1)*(Fr1-Fr2)+Fr2;
    %Realiza aproximación de la función cuantil Phi^-1 con la interpolante de
    %Hermite cúbica monótona. Con la muestra V, se genera el vector aleatorio de
    %tamaño n para Y1*.
    Phiaprox=pchip(Ft,t,V);Phiaprox = max(0,Phiaprox-r2)-max(0,Phiaprox-r1)+r2;
    %Genera  n muestras de  las variables aleatorias auxiliares Y1,Y2 con
    %distribuciones fg1 y fg2.
    Y1=Phiaprox/sqrt(delta1);Y2=sin(asin(alpha2)*(1-U)+asin(alpha1)*U);
    %Genera  n muestras de  las variables aleatorias X1,X2,X3.
    X1=a1*Y1;X2=a2*(Y2.*sqrt(1-(Y1).^2));X3=a2*sqrt(1-(X1/a1).^2-(X2/a2).^2);
    %Traslada de acuerdo al centro c.
    X1=X1+c(1);X2=X2+c(2);X3=X3+c(3);
%Si los parámetros no están bien especificados, muestre mensaje de error.
else msgbox('Error ! Verificar Condiciones en los Parametros')
end
end
```

B.3. EllipsoidAcRejC.m

Es la implementación de la metodología descrita en 5.2.2.

Funciones auxiliares

```
function X = accrejrnd(f,g,grnd,c,m,n)
X = zeros(m,n); % Preallocate memory
for i = 1:m*n
    accept = false;
    while accept == false
        u = rand();
        v = grnd();
        if c*u <= f(v)/g(v)
           X(i) = v;
           accept = true;
        end
    end
end
```

```
function [y]=phi(t,d)
y=(1/(2*sqrt(d)))*(sqrt(d)*t.*sqrt(1-d*(t.^2))+asin(sqrt(d)*t));
end
```

Función principal

```
function [X1,X2,X3,Y1]=EllipsoidAcRejC(n,a2,a1,beta2,beta1,alpha2,alpha1,c)
%FUNCION: Considerando el sistema coordenado x1 x2 x3, tomamos el semi-elipsoide
%de semiejes a1,a2,a2 en el dominio D dado por: x3=a2*sqrt(1-(x1/a1)^2-(x2/a1)^2)
%en D={(x1,x2); beta2<x1<beta1 alpha2*a2*sqrt(1-(x1/a1)^2)<x2<alpha1*a2*...
%sqrt(1-(x1/a1)^2)}. Los parámetros satisfacen 0<a2=a3<=a1,-a1<=beta2<beta1<=a1,
%-1<=alpha2<alpha1<=1. Esta función genera n puntos uniformemente distribuidos
%sobre dicha superficie y la traslada con el vector c.
%Usa el método de aceptación y rechazo en la variable Y1, y trasformada inversa
%en Y2.
%
%ENTRADAS
%n  =     Cantidad de puntos por generar.
%a1 =     Semi-eje x1.
%a2 =     Semi-eje x2,x3.
%alpha2 = Parámetro de la curva límite inferior en el dominio D.
%alpha1 = Parámetro de la curva límite superior en el dominio D.
%beta2 =  Parámetro de la curva límite izquierdo en el dominio D.
%beta2 =  Parámetro de la curva límite derecho en el dominio D.
%c =      Centro del elipsoide de revolución.
%
%SALIDAS
%[X1,X2,X3] = Coordenadas X1,X2,X3 para n vectores aleatorios con distribución
%uniforme sobre el elipsoide de revolución en el dominio D.
%
%Chequeo de parámetros.
%Si los parámetros están bien especificados...
    if((round(n)>0)&&(a2>0)&&(a1>0)&&(a1>a2)&&(abs(beta2)<=a1)&&...
       (abs(beta1)<=a1)&&(beta2<beta1)&&(abs(alpha2)<=1)&&(abs(alpha2)<=1)&&...
       (alpha2<alpha1))
    %Cálculo  del parámetro delta1.
    delta1=1-(a2/a1)^2;
    %Límites de la variable Y1.
    r2=beta2/a1;r1=beta1/a1;
    %Cálculo de las constantes de normalización  k y ku de la función objetivo f
    %y la función instrumental g respectivamente.
    k=1/(phi(r1,delta1)-phi(r2,delta1));ku=1/(r1-r2);
    %Definición de la función objetivo f, la función instrumental g, y el
    %generador de g.
    f = @(x) k*sqrt(1-delta1*(x.^2));g = @(x) ku;grnd = @()r2+(r1-r2)*rand();
    %Cálculo constante de proporcionalidad cons (es tal que f < cons g).
    if(0>r2 && 0<r1);cons=f(0)/ku; else cons=max(f(r2),f(r1))/ku; end
    %Genera  n  muestras de la variable Y1  mediante el método de aceptación y
    %rechazo (accrejrnd, función en mathworks).
    Y1=accrejrnd(f,g,grnd,cons,n,1);
    %Genera n números aleatorios con distribución uniforme en (0,1).
    U=rand(n,1);
    %Genera  n muestras de  la variable aleatoria Y2.
    Y2=sin(asin(alpha2)*(1-U)+asin(alpha1)*U);
    %Genera  n muestras de  las variables aleatorias X1,X2,X3.
    X1=a1*Y1;X2=a2*(Y2.*sqrt(1-(Y1).^2));X3=a2*sqrt(1-(X1/a1).^2-(X2/a2).^2);
    %Traslada de acuerdo al centro c.
    X1=X1+c(1);X2=X2+c(2);X3=X3+c(3);
%Si los parámetros no están bien especificados, muestre mensaje de error.
else msgbox('Error ! Verificar Condiciones en los Parametros')
end
end
```

B.4. EllipsoidHastingC.m

Es la implementación de la metodología descrita en 5.2.3.

Funciones auxiliares

```
function pdfnormalt=pdfnormalt(x,r2,r1,sigma,k)
if(x>=r2 && x<=r1)
pdfnormalt=k*normpdf(x,0,sigma);
else
pdfnormalt=0;
end
end
```

```
function X=rndnormalt(Nr2,Nr1,sigma)
z=rand;
X=norminv(Nr2+z*(Nr1-Nr2),0,sigma);
end
```

Función principal

```
function [X1,X2,X3,Y1]=EllipsoidHastingC(n,a2,a1,beta2,beta1,alpha2,alpha1,c)
%FUNCION: Considerando el sistema coordenado x1 x2 x3, tomamos el semi-elipsoide
%de semiejes a1,a2,a2 en el dominio D dado por: x3=a2*sqrt(1-(x1/a1)^2-(x2/a1)^2)
%en D={(x1,x2); beta2<x1<beta1 alpha2*a2*sqrt(1-(x1/a1)^2)<x2<alpha1*a2*...
%sqrt(1-(x1/a1)^2)}. Los parámetros satisfacen 0<a2=a3<=a1,-a1<=beta2<beta1<=a1,
%-1<=alpha2<alpha1<=1. Esta función genera n puntos uniformemente distribuidos
%sobre dicha superficie y la traslada con el vector c.
%Usa el algoritmo de Metropolis-Hastings en la variable Y1, y la trasformada
%inversa en Y2.
%
%ENTRADAS
%n  =     Cantidad de puntos por generar.
%a1 =     Semi-eje x1.
%a2 =     Semi-eje x2,x3.
%alpha2 = Parámetro de la curva límite inferior en el dominio D.
%alpha1 = Parámetro de la curva límite superior en el dominio D.
%beta2 =  Parámetro de la curva límite izquierdo en el dominio D.
%beta2 =  Parámetro de la curva límite derecho en el dominio D.
%c =      Centro del elipsoide de revolución.
%
%SALIDAS
%[X1,X2,X3] = Coordenadas X1,X2,X3 para n vectores aleatorios con distribución
%uniforme sobre el elipsoide de revolución en el dominio D.
%
%Chequeo de parámetros.
%Si los parámetros están bien especificados...
if((round(n)>0)&&(a2>0)&&(a1>0)&&(a1>a2)&&(abs(beta2)<=a1)&&...
   (abs(beta1)<=a1)&&(beta2<beta1)&&(abs(alpha2)<=1)&&(abs(alpha2)<=1)&&...
   (alpha2<alpha1))
    %Cálculo del parámetro delta1.
    delta1=1-(a2/a1)^2;
    %Límites de la variable Y1.
```

```
    r2=beta2/a1;r1=beta1/a1;
    %Definición función objetivo pdf, la función instrumental proppdf , y el
    %generador proprnd. La función instrumental es normal truncada en (r2,r1).
    sigma=1;Nr1=normcdf(r1,0,sigma);Nr2=normcdf(r2,0,sigma);k=1/(Nr1-Nr2);
    pdf = @(x) sqrt(1-(sqrt(delta1)*x)^2); proppdf = @(x,y)...
    pdfnormalt(x,r2,r1,sigma,k); proprnd = @(x) rndnormalt(Nr2,Nr1,sigma);
    %Define punto inicial como punto medio del intervalo
    start=(r1+r2)/2;
    %Genera  n  muestras de la variable Y1  mediante el algoritmo de Metropolis-
    %Hastings. La función mhsample es una función interna de matlab implementada
    %en el toolbox de estadística, en particular funciona para la versión 2011b.
    %Para alcanzar la distribución estacionaria es necesario iterar cierto número
    %de veces en el algoritmo, digamos K, basados en los gráficos de cuadrados
    %acumulados promedio en las muestras generada, vemos como valor factible de
    %K el número 1000, es decir quemaremos las primeras 1000 muestra.
    Y1=mhsample(start,n,'pdf',pdf,'proprnd',proprnd,'proppdf',proppdf,...
    'burnin',1000);
    %Genera n números aleatorios con distribución uniforme en el intervalo (0,1).
    U=rand(n,1);
    %Genera  n muestras de  la variable aleatoria Y2.
    Y2=sin(asin(alpha2)*(1-U)+asin(alpha1)*U);
    %Genera  n muestras de  las variables aleatorias X1,X2,X3.
    X1=a1*Y1;
    X2=a2*(Y2.*sqrt(1-(Y1).^2));
    X3=a2*sqrt(1-(X1/a1).^2-(X2/a2).^2);
    X1=X1+c(1);X2=X2+c(2);X3=X3+c(3);
%Si los parámetros no están bien especificados, muestre mensaje de error.
else msgbox('Error ! Verificar Condiciones en los Parametros')
end
end
```

B.5. PuntosParcheTriang.m

Es la implementación de la metodología descrita en 2.2.1.

Funciones auxiliares

```
function PtosBezierResultantes=IteracionTriangDeCasteljau(PtosBezierEntrada,...
         IndicesTriang,TCoord,n,i)
%FUNCION:
%IteracionTriangDeCasteljau:
%es una función auxiliar en el algoritmo de de Casteljau para calcular puntos
%sobre un parche de Bézier triangular asociados a un conjunto de ternas de
%coordenadas baricéntricas. El papel de la función es recibir una matriz con
%puntos de control intermedios  en la iteración i del algoritmo para cada terna
%de coordenadas baricéntricas (PtosBezierEntrada) y devolver una matriz con
%puntos de control intermedios en la i+1 iteración del algoritmo para cada
%terna de coordenadas baricéntricas (PtosBezierResultantes).
%
%ENTRADAS
%PtosBezierEntrada:
%Matriz con los puntos de control intermedios en la  iteración i del algoritmo
%para cada terna de coordenadas baricéntricas a evaluar. La matriz se compone
%por 3*s filas y m columnas donde s denota el número de ternas de coordenadas
%baricéntricas y m=(n-i+1)*(n-i+2)/2 denota el número de puntos de control
%intermedios. Para cualquier p en 1,2,...,s las filas 3*(p-1)+1,3*(p-1)+2 y
```

```
%3*(p-1)+3 representan los puntos de control intermedios asociados a  la
%coordenada baricéntrica p y para cualquier q en 1,2,...,m la columna q contiene
%el punto de control intermedio q evaluado en todas las coordenadas baricéntricas.
%
%IndicesTriang:
%Matriz que contiene los índices de cada triángulo en la red de control por
%ejemplo para n=4  indizamos la red de control así...
%                1
%               2 3
%              4 5 6
%             7 8 9 10
%          11 12 13 14 15
%los índices(ver ejemplos) que representan cada triángulo serían:
%          3     1     2
%          5     2     4
%          ...   ...   ...
%
%         14     9    13
%         15    10    14
%en todo caso esta matriz tendrá n*(n+1)/2 filas y 3 columnas.
%
%TCoord:
%Arreglo tal que  T(:,:,u) es una matriz con la componente u (u=1,2,3) de cada
%coordenada baricéntricas repetida por cada coordenada cartesiana en dirección
%de las filas y n*(n+1)/2 veces en dirección de las columnas (ver ejemplos).
%
%SALIDAS
%PtosBezierResultantes:
%Matriz con los puntos de control intermedios en la
iteración i+1 del algoritmo
%para cada terna de coordenadas baricéntricas a evaluar. La matriz se compone por
%3*s filas y m columnas donde s denota el número de ternas de coordenadas
%baricéntricas y m=(n-i+0)*(n-i+1)/2 denota el número de puntos de control
%intermedios. Para cualquier p en 1,2,...,s las filas 3*(p-1)+1,3*(p-1)+2 y
%3*(p-1)+3 representan los puntos de control intermedios asociados a  la
%coordenada baricéntrica p y para cualquier q en 1,2,...,m la columna q contiene
%el punto de control intermedio q evaluado en todas las coordenadas baricéntricas.
%
m=(n-i+0)*(n-i+1)/2;
%Cálculo de una iteración adelante en el algoritmo de de Casteljau vectorizado.
%Cada componente en la suma representa la dirección u, v, o 1-u-v.
PtosBezierResultantes=...
    +TCoord(:,1:m,1).*PtosBezierEntrada(:,IndicesTriang(1:m,1))...
    +TCoord(:,1:m,2).*PtosBezierEntrada(:,IndicesTriang(1:m,2))...
    +TCoord(:,1:m,3).*PtosBezierEntrada(:,IndicesTriang(1:m,3));
end


function PtosBezierFinales=TriangDeCasteljauFinal(PtosBezierIniciales,...
         IndicesTriang,TCoord,temp,n)
%FUNCION:
%TriangDeCasteljauFinal:
%Es una función auxiliar en el algoritmo de de Casteljau para calcular puntos
%sobre un parche de Bézier triangular asociados a un conjunto de ternas de
%coordenadas baricéntricas.El papel de la función es recibir una matriz con
%puntos de control intermedios  en la iteración 0 del algoritmo para cada terna
%de coordenadas baricéntricas (PtosBezierIniciales) y devolver una matriz con
%puntos de control intermedios en la i iteración del algoritmo para cada terna
%de coordenadas baricéntricas (PtosBezierResultantes).
%
```

```
%ENTRADAS
%PtosBezierIniciales:
%Matriz con los puntos de control  en la  iteración 0 del algoritmo para cada
%terna de coordenadas baricéntricas a evaluar. La matriz se compone por 3*s filas
%y m columnas donde s denota el número de ternas de coordenadas baricéntricas y
%m=(n+1)*(n+2)/2 denota el número de puntos de control. Para cualquier p en
%1,2,...,s las filas 3*(p-1)+1,3*(p-1)+2 y 3*(p-1)+3 representan los puntos de
%control  asociados a  la coordenada baricéntrica p y para cualquier q en
%1,2,...,m la columna q contiene el punto de control intermedio q evaluado en
%todas las coordenadas baricéntricas.
%
%IndicesTriang:
%Matriz que contiene los índices de cada triángulo en la red de control por
%ejemplo para n=4  indizamos la red de control así...
%                1
%              2 3
%            4 5 6
%          7 8 9 10
%       11 12 13 14 15
%los índices(ver ejemplos) que representan cada triángulo serían:
%          3     1     2
%          5     2     4
%          ...   ...   ...
%
%         14     9    13
%         15    10    14
%en todo caso esta matriz tendrá n*(n+1)/2 filas y 3 columnas.
%
%TCoord:
%Arreglo tal que  T(:,:,u) es una matriz con la componente u (u=1,2,3) de cada
%coordenada baricéntricas repetida por cada coordenada cartesiana en dirección
%de las filas y n*(n+1)/2 veces en dirección de las columnas (ver ejemplos).
%
%SALIDAS
%PtosBezierFinales:
%Matriz con los puntos de control intermedios en la  iteración i del algoritmo
%para cada terna de coordenadas baricéntricas a evaluar. La matriz se compone por
%3*s filas y m columnas donde s denota el número de ternas de coordenadas
%baricéntricas y m=(n-i+0)*(n-i+1)/2 denota el número de puntos de control
%intermedios. Para cualquier p en 1,2,...,s las filas 3*(p-1)+1,3*(p-1)+2 y
%3*(p-1)+3 representan los puntos de control intermedios asociados a  la
%coordenada baricéntrica p y para cualquier q en 1,2,...,m la columna q contiene
%el punto de control intermedio q evaluado en todas las coordenadas baricéntricas.
%
%Aplicación de la función IteracionTriangDeCastelja sobre PtosBezierIniciales i
%veces.
if (temp == 1)
    PtosBezierFinales=IteracionTriangDeCasteljau(PtosBezierIniciales,...
                      IndicesTriang,TCoord,n,0);
else
    PtosBezierFinales=IteracionTriangDeCasteljau...
                     (TriangDeCasteljauFinal(PtosBezierIniciales,...
                     IndicesTriang,TCoord,temp-1,n),IndicesTriang,...
                     TCoord,n,temp-1);
end
end
```

Función principal

```
function [X1 X2 X3] = PuntosParcheTriang(Controles,CoordBari,IndicesTriag,Pesos)
%FUNCION:
%PuntosParcheTriang:
%Programa basado en algoritmo de de Casteljau vectorizado (con el fin de
%aprovechar la orientación a vectores de matlab) para computar puntos sobre un
%parche de Bézier triangular asociados a un conjunto de ternas de coordenadas
%baricéntricas.
%
%ENTRADAS
%Controles:
%Puntos de control para un parche de Bézier triangular como una matriz de
%dimensión  3*r donde r representa el número de puntos de control del parche.
%Aquí cada fila representa cada coordenada x1,x2 o x3 en el espacio del punto
%de control.
%
%CoordBari:
%Coordenadas baricéntricas para un parche de Bézier triangular como una matriz
%de dimensión  3*s donde s representa el número de evaluaciones sobre el parche.
%Aquí cada fila representa cada coordenada u,v o 1-u-v.
%IndicesTriang:
%Matriz que contiene los índices de cada triángulo en la red de control por
%ejemplo para n=4  indizamos la red de control así...
%                1
%               2 3
%              4 5 6
%             7 8 9 10
%          11 12 13 14 15
%los indices(ver ejemplos) que representan cada triángulo serían:
%          3     1     2
%          5     2     4
%          ...   ...   ...
%
%         14     9    13
%         15    10    14
%en todo caso esta matriz tendra n*(n+1)/2 filas y 3 columnas.
%
%SALIDAS
%
%Puntos:
%Puntos calculados para un parche de Bézier triangular de acuerdo a las
%coordenadas baricéntricas ingresadas, como una matriz de dimensión  3*s .
%Aquí cada fila representa cada coordenada x1,x2 o x3 en el espacio del punto
%calculado.
%
%Cálculos apartir de variables de entrada.
%
[s,~]=size(CoordBari);
[~,r]=size(Controles);
%Dimesión del parche.
[d,~]=size(Controles);
[k,~]=size(IndicesTriag);
%Cálculo del grado del parche.
n=(sqrt(9+8*(r-1))-3)/2;

%Cálculo de la matriz PtosBezierIniciales en función de los puntos de control
%para adecuarse a las variables de entrada de la función TriangDeCasteljauFinal.m
%(ver detalle).
```

```
d=d+1;
Controles=[Controles;ones(1,r)];
PesosG=repmat(Pesos,d,1);
Controles=Controles.*PesosG;
PtosBezierIniciales=repmat(Controles,s,1);
%%%PtosBezierIniciales=repmat(Controles,s,1);
%Definición del arreglo TCoord en función de las coordenadas baricéntricas para
%adecuarse a las variables de entrada de la función TriangDeCasteljauFinal.m
%(ver detalle).
TCoord=zeros(d*s,k,3);
TCoord(:,:,1)= repmat(kron(CoordBari(:,1),ones(d,1)),1,k);
TCoord(:,:,2)= repmat(kron(CoordBari(:,2),ones(d,1)),1,k);
TCoord(:,:,3)= repmat(kron(CoordBari(:,3),ones(d,1)),1,k);
%
%Utilización de la rutina  TriangDeCasteljauFinal.m (ver detalle) y posterior
%redimensionamiento de la matriz resultante.
Puntos=reshape(TriangDeCasteljauFinal(PtosBezierIniciales,IndicesTriag,...
               TCoord,n,n),d,s);
X1=Puntos(1,:)./Puntos(4,:);
X2=Puntos(2,:)./Puntos(4,:);
X3=Puntos(3,:)./Puntos(4,:);
end
```

B.6. PuntosParcheBernstein.m

Es la implementación de la metodología descrita en 2.2.2.

Funciones auxiliares

```
function maltura = m_altura(m,k)
maltura=zeros(3,k);
maltura(:,1)=[0;(m-(k-1));(k-1)];
if k==1

else
    for i=1:(k-1)
        maltura(:,(1+i))=maltura(:,i)+[1;0;-1];
    end
end
end
```

```
function mcompleta = m_completa(m)
mcompleta=zeros(3,((m+1)*(m+2)/2));
mcompleta(:,1)= m_altura(m,1);
    for i=2:(m+1)
        mcompleta(:,(((i-1)*i/2)+1):(i*(i+1)/2))= m_altura(m,i);
    end

end
```

Función principal

```
function [X1 X2 X3] = PuntosParcheBernstein(Controles,Pesos,CoordBari)
%FUNCION:
%PuntosParcheBernstein:
%Programa basado en la representación de Bernstein de un parche triangular
%vectorizado (con el fin de aprovechar la orientación a vectores de matlab) para
%computar puntos sobre un parche de Bézier triangular asociados a un conjunto de
%ternas de coordenadas baricéntricas.
%
%ENTRADAS
%Controles:
%Puntos de control para un parche de Bézier triangular como una matriz de
%dimensión  3*r donde r representa el número de puntos de control del parche.
%Aquí cada fila representa cada coordenada x1,x2 o x3 en el espacio del punto
%de control.
%
%CoordBari:
%Coordenadas baricéntricas para un parche de Bézier triangular como una matriz
%de dimensión  3*s donde s representa el número de evaluaciones sobre el parche.
%Aquí cada fila representa cada coordenada u,v o 1-u-v.
%
%SALIDAS
%
%Puntos:
%Puntos calculados para un parche de Bézier triangular de acuerdo a las
%coordenadas baricéntricas ingresadas, como una matriz de dimensión  3*s .
%Aquí cada fila representa cada coordenada x1,x2 o x3 en el espacio del punto
%calculado.
[~,s]=size(CoordBari);
[~,m]= size(Controles);
n=(sqrt(9+8*(m-1))-3)/2;
unosp=ones(1,m);
indice=m_completa(n);
Pg=kron(CoordBari,unosp);Ig=repmat(indice,1,s);Potencia=reshape(prod(Pg.^Ig),m,s);
Pesosg=repmat(Pesos,1,s);
Combinaciones=factorial(n)./prod(factorial(indice));
Combinacionesg=repmat(Combinaciones',1,s);
Puntos=Controles*(Pesosg.*Combinacionesg.*Potencia)./...
       (repmat(sum(Pesosg.*Combinacionesg.*Potencia),3,1));
X1=Puntos(1,:);
X2=Puntos(2,:);
X3=Puntos(3,:);
end
```

B.7. Extraccionhumero.m

Es la implementación para construir la figura 6.6.

Funciones auxiliares

```
% Crea un punto de referencia top_left_pixel_position y dos vectores
% ortonormales que definen el plano en 3D correspondiente a cada rodaja

for k=0:9
info=dicominfo([prefijo 'MR00000'  num2str(k)  '.dcm']);
% la variable "prefijo" fue generada por el script crear_Valores_Imagenes
top_left_pixel_position(:,k+1)=info.ImagePositionPatient;
e1(:,k+1)=info.ImageOrientationPatient(1:3,1);
e2(:,k+1)=info.ImageOrientationPatient(4:6,1);
end

for k=10:length(dir([prefijo '*.dcm']))-1 % ver crear_Valores_Imagenes
info=dicominfo([prefijo 'MR0000'  num2str(k)  '.dcm']);
top_left_pixel_position(:,k+1)=info.ImagePositionPatient;
e1(:,k+1)=info.ImageOrientationPatient(1:3,1);
e2(:,k+1)=info.ImageOrientationPatient(4:6,1);
end
```

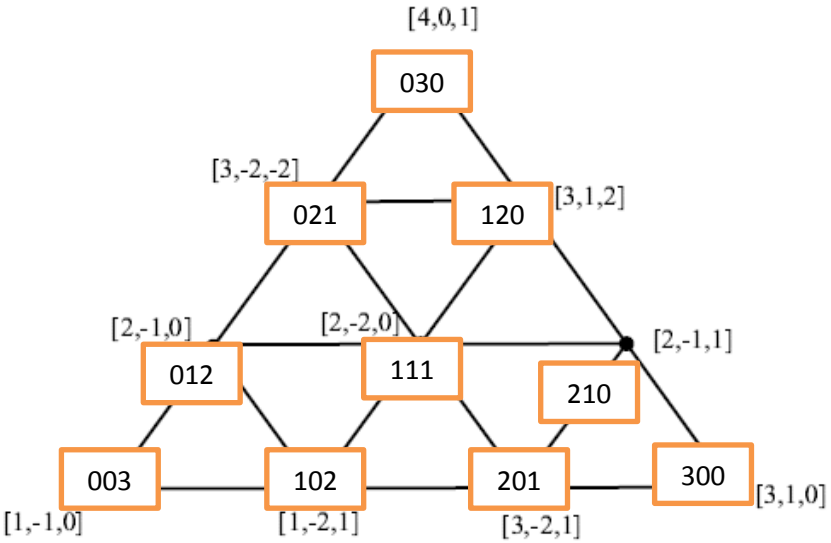
```
% Este archivo lee todos los cortes de una serie y construye un arreglo
% tridimensional, que denominamos volumen logico y denotamos por
% ValoresImagenes.

% NO OLVIDAR dar el valor adecuado a la variable "prefijo"

% ENTRADA: ninguna, el script se debe correr donde estan los archivos
% MR000000, MR000001, etc.
% SALIDA: el arreglo tridimensional ValoresImagenes

prefijo='';

for k=0:9
ValoresImagenes(:,:,k+1)=dicomread([prefijo 'MR00000'  num2str(k) '.dcm']);
end
%
% For k=10 up to 99, there are two digits so have to write 'MR0000'
% num2str(k) instead of 'MR00000'  num2str(k)
%
for k=10:length(dir([prefijo '*.dcm']))-1
    % dir('*.dcm') lista los archivos .dcm del directorio
    % por ejemplo, el directorio SE000000 contiene 20 imagenes en total
ValoresImagenes(:,:,k+1)=dicomread([prefijo 'MR0000'  num2str(k)  '.dcm']);
end
clear k
```

Función principal

```
%Este archivo realiza la gráfica de la  texturización de una superficie esférica o
%elipsoidal con el nivel de gris de una resonancia magnética del manguito rotador
% apartir cubriendo un conjunto de puntos digitados para la frontera del húmero
%(segmentación). La función utiliza los archivos elaborados por el profesor Marco
%Paluszny:crear_Valores_Imagenes y crear_top_left_e1_e2, para definir el volúmen
%lógico y determinar el plano donde yace cada rodaja.
%
%Carga de puntos digitados para la frontera del húmero. Los archivos fueron
```

```
%provistos por Juan Esteban Suarez para la secuencia SE000000.
load('ptos_digitados.mat')
%Escalamiento  de puntos para la frontera del húmero.
SE000000_x=0.293*SE000000_2(1,:);
SE000000_y=0.293*SE000000_2(2,:);
SE000000_z=(1/1.0)*SE000000_2(3,:);
Sg=[SE000000_x' SE000000_y' SE000000_z'];
%Selecciono por inspección visual la cabeza del húmero (lo que quiero ajustar).
Sgh=Sg(601:768,:);
%Computo los parámetros de ajuste de la esfera por mínimos cuadrados con la función
%ellipsoid_fit, del autor Yory Petrov disponible en la web:
%http://www.mathworks.com/matlabcentral/fileexchange/24693-ellipsoid-fit.
[Center_LSE,Radius_LSE]=ellipsoid_fit(Sgh,3);
%Leo todos los cortes de la serie y construyo un arreglo tridimensional,
%que denominamos volumen lógico y denotamos por ValoresImagenes.
crear_Valores_Imagenes
% Crea un punto de referencia top_left_pixel_position y dos vectores ortonormales
%que definen el plano en 3D correspondiente a cada rodaja.
crear_top_left_e1_e2
% tt(1) es el tamaño del eje y, tt(2) del eje x, tt(3) el número de rodajas.
tt=size(ValoresImagenes);
delta_x=info.PixelSpacing(1);
delta_y=info.PixelSpacing(2);
delta_z=info.SpacingBetweenSlices;
%Se define un sistema coordenado de acuerdo a los tamaños de pixel y espacios
%entre rodaja.
CoordX=0:delta_x:((tt(1)-1) * delta_x);CoordY=0:delta_y:((tt(2)-1)*delta_y);
CoordZ=0:delta_z:((tt(3)-1)*delta_z);
[CoordX CoordY CoordZ]=meshgrid(CoordX,CoordY,CoordZ);
%Determinar radio alternativo teniendo en cuenta distancia al centro.
tp=size(Sgh);
centers=repmat(Center_LSE',tp(1),1);
d=sqrt(sum((Sgh-centers).^2,2));
rmin=min(d);
rmax=max(d);
rmed=mean(d);
%Determino las propiedades graficas del fig por generar.
fig=figure;hold on,
xlabel('x'), ylabel('y'), zlabel('z')
daspect([1 1 1]) % la longitud unitaria es igual en los tres ejes.
daspect('manual')
axis off
view([150 45])
%Compúto el mapa de color para la figura.
maxGris=double(max(max(max(ValoresImagenes)))); % debe ser double!
map=transpose(repmat((1:maxGris)/maxGris,3,1));
colormap(map)
set(gcf,'renderer','zbuffer')
%Genero n puntos sobre la esfera o elipsoide con la función SphereRandomC de
%radio a1 y fronteras determinadas por beta2,beta1,alpha2,alpha1.
n=50086;a1=rmed*1.22;beta2=0;beta1=a1;alpha2=0.0;alpha1=1.0;c=Center_LSE;
[X1 X2 X3]=SphereRandomC(n,a1,alpha2,alpha1,beta2,beta1,c);
%Con la siguiente instrucción podemos rotar la superficie.
theta=-pi/4;
M=[cos(theta),sin(theta),0;-sin(theta),cos(theta),0;0,0,1];
A=[X1-c(1) X2-c(2) X3-c(3)]*M;
X1=A(:,1)+c(1);
X2=A(:,2)+c(2);
X3=A(:,3)+c(3);
%Realizamos interpolación trilineal para calcular el nivel de gris de cada punto.
```

```
tic,Vq = interp3(CoordX,CoordY,CoordZ,double(ValoresImagenes),X1,X2,X3);toc
%triangulación para X1, X2.
tri = delaunay(X1,X2);
%Grafíco cada luna teniendo en cuenta el nivel de gris calculado.
h = trisurf(tri, X1, X2, X3,double(Vq/maxGris),'EdgeColor','none');
%Esta instrucción permite sobreponer a la superficie la rodaja k.
k=10;
[q1 q2]=meshgrid(0:delta_x:((tt(1)-1) * delta_x),0:delta_y:((tt(2)-1) * delta_y));
q1=reshape(q1,tt(1)*tt(2),1);
q2=reshape(q2,tt(1)*tt(2),1);
q3=(k*3.3)*ones(tt(1)*tt(2),1);
Vq = interp3(CoordX,CoordY,CoordZ,double(ValoresImagenes),q1,q2,q3);
tri = delaunay(q1,q2);
h = trisurf(tri, q1, q2, q3,double(Vq/maxGris),'EdgeColor','none');
%Grafíco los puntos digitados.
plot3(Sg(:,1),Sg(:,2),Sg(:,3),'r.'),hold on,daspect([1,1,1]);
```



Ejemplo 1 Evaluación directa racional PuntosParcheBernstein.m

El objetivo es calcular puntos sobre un parche de Bézier triangular definido por el triángulo anterior para las siguientes ternas de coordenadas baricéntricas (simultáneamente):

$$u1 = (0.25,0.25,0.5) \ y \ u2 = (0.2,0.5,0.3)$$

Para tal fin necesitamos definir tres matrices: una en función de los puntos de control **(Controles)**, otra en función de las ternas de coordenadas baricéntricas **(CoordBari)** y la última en función de los pesos en el triángulo (si lo consideramos polinomial es un vector de unos):

Controles =

4	3	3	2	2	2	1	1	3	3
0	-2	1	-1	-2	-1	-1	-2	-2	1
1	-2	2	0	0	1	0	1	1	0

CoordBari=

0.25	0.25	0.50
0.20	0.50	0.30

Pesos =

1
1
1
1
1
1
1
1
1
1

Posteriormente ejecutamos en matlab

1. **PuntosParcheBernstein(Controles,Pesos,CoordBari).**

Dicha función hace los siguientes cálculos:

Las constantes básicas:

s=2, representa el número de ternas de coordenadas baricéntricas.

m=10, número de puntos de control del parche.

n=3, grado del parche.

La matriz con los índices (**i**) para los puntos de control

indice =

0	0	1	0	1	2	0	1	2	3
3	2	2	1	1	1	0	0	0	0
0	1	0	2	1	0	3	2	1	0

La matriz Pg (| indica que se juntan de izquierda a derecha):

U1

(u) 0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25|
(v) 0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25|
(w) 0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50|

U2

|(u) 0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
|(v) 0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
|(w) 0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30

Esta matriz contiene para cada punto de control su correspondiente tripleta u,v y w asociada a cada terna de coordenadas baricéntricas .

Ig =

U1

```
(i1) 0     0     1     0     1     2     0     1     2     3|
(i2) 3     2     2     1     1     1     0     0     0     0|
(i3) 0     1     0     2     1     0     3     2     1     0|
```

U2

```
(i1) |0     0     1     0     1     2     0     1     2     3
(i2) |3     2     2     1     1     1     0     0     0     0
(i3) |0     1     0     2     1     0     3     2     1     0
```

Esta matriz contiene para cada punto de control su correspondiente tripleta i1,i2 y i3 asociada a cada terna de coordenadas baricéntricas(utilizando índice).

En este sentido si realizamos la potencia componente a componente de la matriz Pg con la matriz Ig obtendríamos los factores del tipo u^i1, v^i2 y w^i3 asociado a cada punto de control y para cada terna de coordenadas baricéntricas. Si adicionalmente al resultado le calculamos el producto por columnas obtenemos para cada punto de control y para cada terna de coordenadas baricéntricas los productos **u^i1*v^i2 * w^i3** así:

Potencia =

```
0.0156    0.1250
0.0313    0.0750
0.0156    0.0500
0.0625    0.0450
0.0313    0.0300
0.0156    0.0200
0.1250    0.0270
0.0625    0.0180
0.0313    0.0120
0.0156    0.0080
```

Los pesos para cada punto de control asociada a cada terna de coordenadas baricéntricas serían en este caso **wi** :

Pesosg =

1	1
1	1
1	1
1	1
1	1
1	1
1	1
1	1
1	1
1	1

La combinación para cada punto de control asociada a cada terna de coordenadas baricéntricas serían en este caso **nCi** :

Combinacionesg =

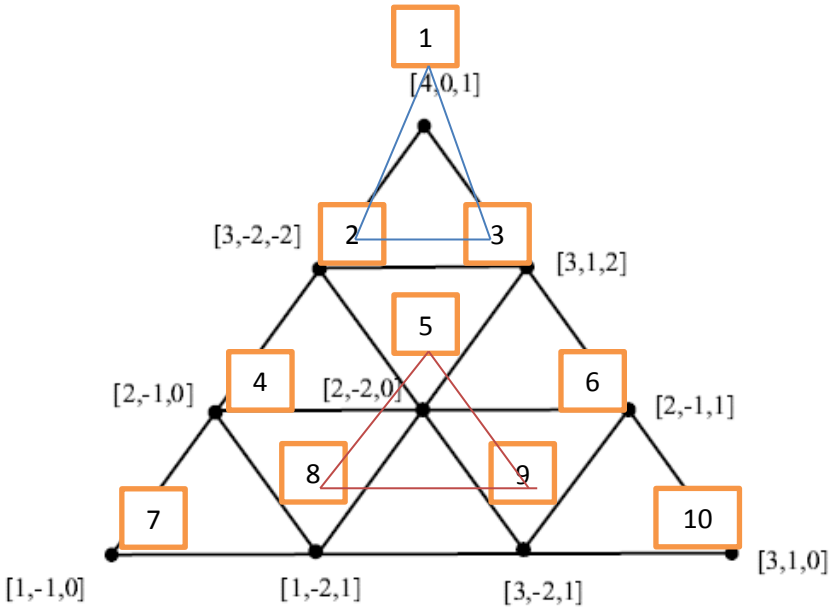
1	1
3	3
3	3
3	3
6	6
3	3
1	1
3	3
3	3
1	1

Así disponemos de Potencia, Pesosg, Combinacionesg y controles que son los elementos necesarios para evaluar directamente un punto sobre el parche triangular de Bézier, la traducción algorítmica de la formula es:

```
Controles*(Pesosg.*Combinacionesg.*Potencia)…
/(repmat(sum(Pesosg.*Combinacionesg.*Potencia),3,1))
```

Este cálculo nos entrega los puntos sobre el parche:

```
[X1 X2] =

    1.9688    2.5880
   -1.4219   -1.0540
    0.2500    0.1250
```



Ejemplo 2 Algoritmo de de Casteljau racional PuntosParcheTriang.m

El objetivo es calcular puntos sobre un parche de Bézier triangular definido por el triángulo anterior para las siguientes ternas de coordenadas baricéntricas (simultáneamente):

$$u1 = (0.25,0.25,0.5) \; y \; u2 = (0.2,0.5,0.3)$$

Para tal fin necesitamos definir cuatro matrices: una en función de los puntos de control **(Controles)** ,otra en función de las ternas de coordenadas baricéntricas (**CoordBari)** , los pesos asociados a cada punto de control (**Pesos)** y la última en función de los índices **(IndicesTriag)** que forman cada triángulo en el parche así:

Controles =

4	3	3	2	2	2	1	1	3	3
0	-2	1	-1	-2	-1	-1	-2	-2	1
1	-2	2	0	0	1	0	1	1	0

CoordBari =

0.2500	0.2500	0.5000
0.2000	0.5000	0.3000

Pesos =

0.50 0.50 0.50 0.50 0.50 1.00 1.00 1.00 1.00 1.00

IndicesTriag =

3	1	2
5	2	4
6	3	5
8	4	7
9	5	8
10	6	9

(Variando las columnas se determina con cual coordenada en la terna es ponderado cada punto)

Posteriormente ejecutamos en matlab

1. **[X1 X2 X3] = PuntosParcheTriang(Controles,CoordBari,IndicesTriag,Pesos)**

Dicha función realiza los siguientes cálculos:

Las constantes básicas:

s=2, representa el número de ternas de coordenadas baricéntricas.

r=10, número de puntos de control del parche.

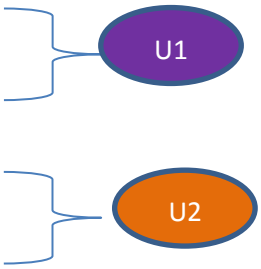
k=6, número de triángulos del parche.

n=3, grado del parche

Las matrices de básicas entrada en la función TriangDeCasteljauFinal: PtosBezierIniciales resulta multiplicar los puntos de control (adicionando una columna de unos) por sus pesos asociados, posteriormente se repite la matriz de acuerdo al número de coordenadas baricentricas por evaluar.

PtosBezierIniciales =

2.0	1.5	1.5	1.0	1.0	2.0	1.0	1.0	3.0	3.0
0.0	-1.0	0.5	-0.5	-1.0	-1.0	-1.0	-2.0	-2.0	1.0
0.5	-1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0
0.5	0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0	1.0
2.0	1.5	1.5	1.0	1.0	2.0	1.0	1.0	3.0	3.0
0.0	-1.0	0.5	-0.5	-1.0	-1.0	-1.0	-2.0	-2.0	1.0
0.5	-1.0	1.0	0.0	0.0	1.0	0.0	1.0	1.0	0.0
0.5	0.5	0.5	0.5	0.5	1.0	1.0	1.0	1.0	1.0



TCoord(:,:,1) =

0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20
0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20	0.20

(Primer componente de las ternas)

TCoord(:,:,2) =

0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25	0.25
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50

(Segundo componente de las ternas)

TCoord(:,:,3)=

0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50	0.50
0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30
0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30	0.30

(Tercer componente de las ternas)

T es un arreglo con cada componente de cada terna de coordenadas baricéntricas repetida como se muestra gráficamente.

Con la información disponible hasta el momento ejecutamos:

1.1 TriangDeCasteljauFinal(PtosBezierIniciales,...
IndicesTriang,TCoord,n,n)

El objetivo de esta instrucción es realizar cada una de las iteraciones (hace el papel del ciclo) en el algoritmo de de Casteljau simultáneamente para todas las ternas de coordenadas baricéntricas consideradas, la función se basa en la iteración de la función IteracionTriangDeCasteljau.

En la primera iteración se computaría:

1.1.1 B1= IteracionTriangDeCasteljau(PtosBezierIniciales,...
IndicesTriang,TCoord,n,0)

1.625	1.125	1.375	1.000	1.500	2.750
-0.375	-0.750	-0.625	-1.125	-1.750	-1.000
-0.125	-0.250	0.500	0.250	0.750	0.750
0.500	0.500	0.625	0.875	0.875	1.000
1.750	1.250	1.450	1.000	1.400	2.500
-0.200	-0.850	-0.250	-0.950	-1.500	-0.900
0.150	-0.500	0.700	0.200	0.500	0.800
0.500	0.500	0.600	0.750	0.750	1.000

En la segunda iteración se computa:

1.1.2 B2= IteracionTriangDeCasteljau(B1,...
IndicesTriang,TCoord,n,1)

1,313	1,156	1,781
-0,625	-1,188	-1,281
-0,031	0,250	0,688
0,531	0,781	0,844
1,540	1,205	1,645
-0,405	-1,010	-0,755
0,065	-0,090	0,660
0,520	0,625	0,725

En la tercera y última iteración se obtiene:

1.1.3 B3= IteracionTriangDeCasteljau(B2,...
IndicesTriang,TCoord,n,2)

1,352
-1,070
0,289
0,734
1,461
-0,657
0,138
0,593

Luego reordenando las componentes del vector resultante en una matriz y dividiendo por la cuarta componente se obtiene:

1.8404	2.4650
-1.4574	-1.1080
0.3936	0.2321.

Bibliografía

[1] O. Opsha, A. Malik, R. Baltazar, D. Primakov, S. Beltran, T. Miller, and J. Beltran, "MRI of the Rotator Cuff and Internal Derangement," *European Journal of Radiology*, vol. 68, pp. 36–56, 2008.

[2] M. Tuite, "Magnetic Resonance Imaging of Rotator Cuff Disease and External Impingement," *Magnetic Resonance Imaging Clinics of North America*, vol. 20, pp. 187–200, 2012.

[3] M. Ackerman, "The Visible Human Project," *Proceedings of the IEEE*, vol. 86, pp. 504–511, 1998.

[4] M. Lentini and M. Paluszny, "Approximation of meander-like regions by paths of lemniscatic sectors and the computation of orthogonal meshes," *Computer Aided Geometric Design*, pp. 729–737, 2008.

[5] C. Gonzáles, "Aproximación de Superficies Desarrollables con Splines de Conos," *Colombia, 2012: Tesis de Maestría. Universidad Nacional de Colombia.*

[6] L. Saroul, "Surface Extraction and Flattening for Anatomical Visualization," *Francia, 2006: Tesis Doctoral. Ecole Polytechnique Federale de Lausanne.*

[7] L. Saroul, O. Figueiredo, and R. Hersch, "Distance Preserving Flattening of Surface Sections," *IEEE Transactions on Visualization and Computer Graphics.*, vol. 12, 2006.

[8] G. Farin, *Curves and Surfaces for CAGD: A Practical Guide*. Academic Press, 2002.

[9] D. Rajon and W. Bolch, " Marching Cube Algorithm: Review and Trilinear Interpolation Adaptation for Image-Based Dosimetric Models," *Computerized Medical Imaging and Graphics*, vol. 27, pp. 411–435, 2003.

[10] R. Hogg, A. Craig, and J. McKean, *Introduction to Mathematical Statistics*. Pearson Education, 2005.

[11] H. Prautzsch, W. Boehm, and M. Paluszny, *Bézier and B-spline Techniques*. Springer, 2002.

[12] W. Boehm and A. Müller, "On de Casteljau's Algorithm," *Computer Aided Geometric Design*, vol. 16, pp. 587–605, 1999.

[13] H. Pottmann, "Rational Curves and Surfaces with Rational Offsets," *Computer Aided Geometric Design*, vol. 12, pp. 175–192, 1995.

[14] R. Farouki, "The Bernstein Polynomial Basis: A Centennial Retrospective," *Computer Aided Geometric Design*, vol. 29, pp. 379–419, 2012.

[15] L. Hernanes and L. Koller, "On Computing Bézier Curves by Pascal Matrix Methods," *Applied Mathematics and Computation*, vol. 217, pp. 10118–10128, 2011.

[16] G. Farin, B. Piper, and A. Worsey, "The Octant of a Sphere As a Nondegenerate Triangular Bézier Patch," *Computer Aided Geometric Design*, vol. 4, pp. 329–332, 1988.

[17] G. Farin, J. Hoschek, and S. Kim, *Handbook of CAGD*. Elseiver, 2002.

[18] G. Farin, "From Conics to NURBS: a Tutorial and Survey," *IEEE Computer Graphics and Application*, pp. 78–86, 1992.

[19] R. Bhattacharya and E. Waymire, *A Basic Course in Probability Theory*. Springer, 2007.

[20] P. Billingsley, *Probability and Measure*. John Wiley and Sons, Inc, 1995.

[21] D. Kroese, T. Taimre, and Z. Botev, *Handbook of Monte Carlo Methods*. John Wiley and Sons, Inc, 2011.

[22] O. Kallenberg, *Foundations of Modern Probability*. Springer, 1997.

[23] M. Taylor, *Measure Theory and Integration*. Americal Mathematical Society, 2006.

[24] R. Smith, "Efficient Monte Carlo Procedures for Generating Points Uniformly Distributed Over Bounded Regions," *Operations Research*, vol. 32, pp. 1296–1308, 1984.

[25] S. Nadarajah and S. Kotz, "R Programs for Computing Truncated Distributions," *Journal of Statistical Software*, 2006.

[26] F. Fritsch and R. Carlson, "Monotone Piecewise Cubic Interpolation," *SIAM Journal on Numerical Analysis*, pp. 238–246, 1980.

[27] R. Levine, Z. Yu, W. Hanley, and J. Nitao, "Implementing Componentwise Hastings Algorithms," *Computational Statistics and Data Analysis*, pp. 363–386, 2005.

[28] T. Deserno, *Biomedical Image Processing*. Springer, 2011.

[29] D. McRobbie, E. Moore, M. Graves, and M. Prince, *MRI From Picture to Proton*. Cambridge University Press, 2006.

[30] B. Preim and D. Bartz, *Visualization in Medicine: Theory, Algorithms, and Applications*. Elsevier, 2007.

[31] American Academy of Orthopaedic Surgeons, "Rotator Cuff Tears: Surgical Treatment Options." [Online; accessed 22-Noviembre-2013].

[32] C. Guevara, "Costo-efectividad de diferentes alternativas diagnósticas de ruptura del manguito rotador," *Revista de la Asociación Colombiana de Medicina Física y Rehabilitación*, pp. 118–128, 2013.

[33] J. Khanna, *MRI for Orthopaedic Surgeons*. Thieme Medical Publishers, 2010.

[34] M. Elmaoğu and A. Çelik, *MRI Handbook: MR Physics, Patient Positioning, and Protocols*. Springer, 2012.

[35] G. Dahlquist and A. Bjorck, *Numerical Methods*. Dover Publications, 1974.

[36] J. Nocedal and S. Wright, *Numerical Optimization*. Springer, 2006.

[37] D. Hitchcock, "A History of The Metropolis Hastings Algorithm," *The American Statistician*, pp. 254–257, 2003.

[38] D. Resnick and H. Kang, *Trastornos Internos de Las Articulaciones*. Editorial Médica Panamericana, 2000.