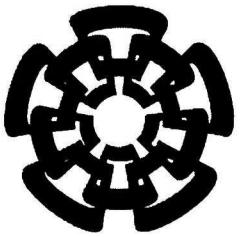
XX(133731.1)



CINVESTAV

Centro de Investigación y de Estudios Avanzados del I.P.N.
Unidad Guadalajara

Segmentación, Reconstrucción 3D y Registro para Neurocirugía

Tesis que presenta:
Jorge Rivera Rovelo

para obtener el grado de:
Doctor en Ciencias

en la especialidad de:
Ingeniería Eléctrica

Director de Tesis
Dr. Eduardo José Bayro Corrochano

Guadalajara, Jalisco, Marzo de 2007.

Segmentación, Reconstrucción 3D y Registro para Neurocirugía

Tesis de Doctorado en Ciencias
Ingeniería Eléctrica

Por:
Jorge Rivera Rovelo
Maestro en Ciencias en Ingeniería Eléctrica
CINVESTAV del IPN, Unidad Guadalajara 2001-2003

Becario de CONACYT, expediente no. 165140

Director de Tesis
Dr. Eduardo José Bayro Corrochano

CINVESTAV del IPN Unidad Guadalajara, Marzo de 2007.

Resumen

En este trabajo de tesis se presentan dos propuestas diferentes para segmentar objetos en imágenes. La primera esta basada en la información de texturas y bordes en la imagen, realizando el proceso en un esquema de crecimiento de regiones. La segunda busca la aproximación de los contornos de los objetos de interés y consiste en una red neuronal auto-organizada que integra la información del flujo vectorial gradiente generalizado dentro de su etapa de adaptación con el fin de determinar las transformaciones, expresadas como operadores del álgebra geométrica conformal, que mueven un punto para definir el contorno del objeto en cuestión. Los resultados experimentales muestran cómo la integración de dicha información en la etapa del aprendizaje, mejora la aproximación de los contornos.

Así mismo, se presentan aplicaciones del algoritmo basado en redes neuronales y álgebra geométrica, en tareas como el modelado y la deformación de modelos virtuales. También se propone un algoritmo basado en *marching cubes* para crear modelos basados en esferas, mostrando mayor eficiencia con respecto al método *unión de esferas* encontrado en la literatura [27].

Aunque los experimentos aplicando los algoritmos antes mencionados se realizaron con imágenes médicas, éstos se pueden emplear en tareas no necesariamente del *procesamiento de imágenes médicas*; especialmente los algoritmos de segmentación y modelado basados en redes neuronales que pueden ser de utilidad en otras tareas de reconocimiento de patrones y procesamiento de imágenes en general.

Finalmente se presenta el problema de *registro* o *registración*, el cual dividimos en dos sub-problemas: la calibración entre el equipo de endoscopía y el sistema de seguimiento óptico (calibración endoscopio-Polaris) y el registro entre modelos virtuales basados en esferas. La calibración endoscopio-Polaris se resuelve utilizando la teoría de la calibración mano-ojo, conocida en problemas de robótica, pero en el marco del álgebra geométrica conformal. Para el registro de modelos basados en esferas, se presenta un algoritmo que sin información previa sobre la correspondencia entre las entidades (esferas) o la deformación sufrida por el modelo, transforma un modelo en tiempo t_1 en otro en tiempo t_2, estimando alternadamente las correspondencias y la transformación.

Los resultados obtenidos se han publicado en congresos internacionales, revistas de divulgación científica y en un capítulo de libro por invitación.

Abstract

The research carried out in this thesis concerns to image segmentation, 3D modeling of volumetric data and non-rigid registration of 3D models. Two approaches for image segmentation are presented. The first approach uses the texture and boundary information in a region growing scheme to extract the object we are interested in. The second one is for contour approximation, and is developed as a self-organizing neural network integrating gradient information extracted from the image. Such information is used during the training stage of the neural network to determine a set of transformations expressed as versors in the geometric algebra, which are applied to a point to define the contour of the object. Experimental results show that gradient information improves the contour approximation.

The same neural net is applied in 3D modeling (surface approximation) and in models deformation, showing that is well suited for such tasks. An algorithm based in *marching cubes* is also presented. This algorithm creates 3D models based on spheres, and its results are compared with the *Union of Spheres*, obtaining more compact models (reduction in the number of primitives needed).

The experiments were carried out taking medical images. However, the algorithms can be used in other tasks different from medical applications, like pattern recognition and others image processing tasks.

Finally, calibration between the neuro-endoscopy equipment and the optical tracking system, and the non-rigid registration of 3D models based on spheres are presented. Calibration of endoscopic equipment with the tracking device is established as a *hand-eye* calibration problem, and it is solved in the geometric algebra framework using the *motor* representation of rigid motions. The non-rigid registration of 3D models is solved in an annealing scheme without prior information about correspondences of the entities, nor information about the transformation.

The results have been published in international conferences, international journals and in a chapter of a book.

Agradecimientos

A mis padres Jorge y Silvia

A mis hermanos David y Jessica

A mi novia Katy

A mis amigos y hermanos de comunidad

Un agradecimiento especial a CONACYT

Índice general

Índice de figuras

Índice de Tablas

Simbología

Tabla 1: Símbolos utilizados a lo largo del documento

Símbolo	Significado
I_C	Pseudo-escalar del álgebra geométrica conformal (AGC) $G_{4,1}$
I_E	Pseudo-escalar del álgebra geométrica G_3
R	Rotor
T	Traslador
$M = TR$	Motor (rotación y traslación)
X	Punto conformal
e_∞	Vector nulo representando el punto en el infinito definido en el AGC
e_0	Vector nulo representando el origen, definido en el AGC
e_{win}	Parámetro de aprendizaje de la neurona ganadora
e_n	Parámetro de aprendizaje de las neuronas vecinas a la ganadora.
∇	Operador gradiente
Z	Matriz de correspondencia binaria
M	Matriz de correspondencias parciales (difusas)
I	Matriz identidad

Capítulo 1

Introducción

Actualmente los sistemas de procesamiento de imágenes y reconocimiento de patrones son de gran importancia por su variada aplicación en áreas como el control de calidad, robótica, procesos industriales y medicina. En las últimas décadas el *procesamiento de imágenes médicas* ha recibido mucha atención por parte de los investigadores, y gracias a esto se han desarrollado sistemas computacionales que ayudan al médico en el diagnóstico y/o tratamiento de enfermedades en el ser humano.

El presente trabajo es parte del Proyecto del Fondo Sectorial de Salud SEP - CONACYT #49 titulado "Neuronavegador económico de uso múltiple" La figura 1.1 muestra el esquema general del proyecto, el cual consta de dos etapas principales:

- Etapa prequirúrgica: Comienza con la obtención de imágenes ya sea por Resonancia Magnética (RM), Tomografía Computarizada (TC), Tomografía por Emisión de Positrones (TEP), o alguna otra técnica[1] Comprende todo el procesamiento digital de las imágenes médicas antes de que el paciente entre a cirugía, el cual consiste en la segmentación de las diferentes estructuras cerebrales, la representación volumétrica de dichos datos y la integración de información brindada por un atlas digital. Todo ésto con el fin de brindar al médico cirujano un modelo virtual con el cual hacer la planeación de la cirugía y aumentar con ello la probabilidad de éxito de la misma.

- Etapa intraquirúrgica: comienza una vez que el paciente ingresa al quirófano. Comprende el proceso de registro entre el modelo virtual (creado en la etapa anterior) y el paciente, el monitoreo y seguimiento del instrumental quirúrgico, la fusión de información de diferentes fuentes (modelo virtual, endoscopio, ultrasonido, sistema de seguimiento visual) y la actualización del modelo virtual de acuerdo a lo que sucede en el quirófano.

Es importante subrayar que aunque existen sistemas de este tipo alrededor del mundo, desarrollados en países como Estados Unidos de Norteamérica, Alemania, Suecia y Japón,

[1] El apéndice A proporciona información sobre cómo se obtienen algunas de estas imágenes.

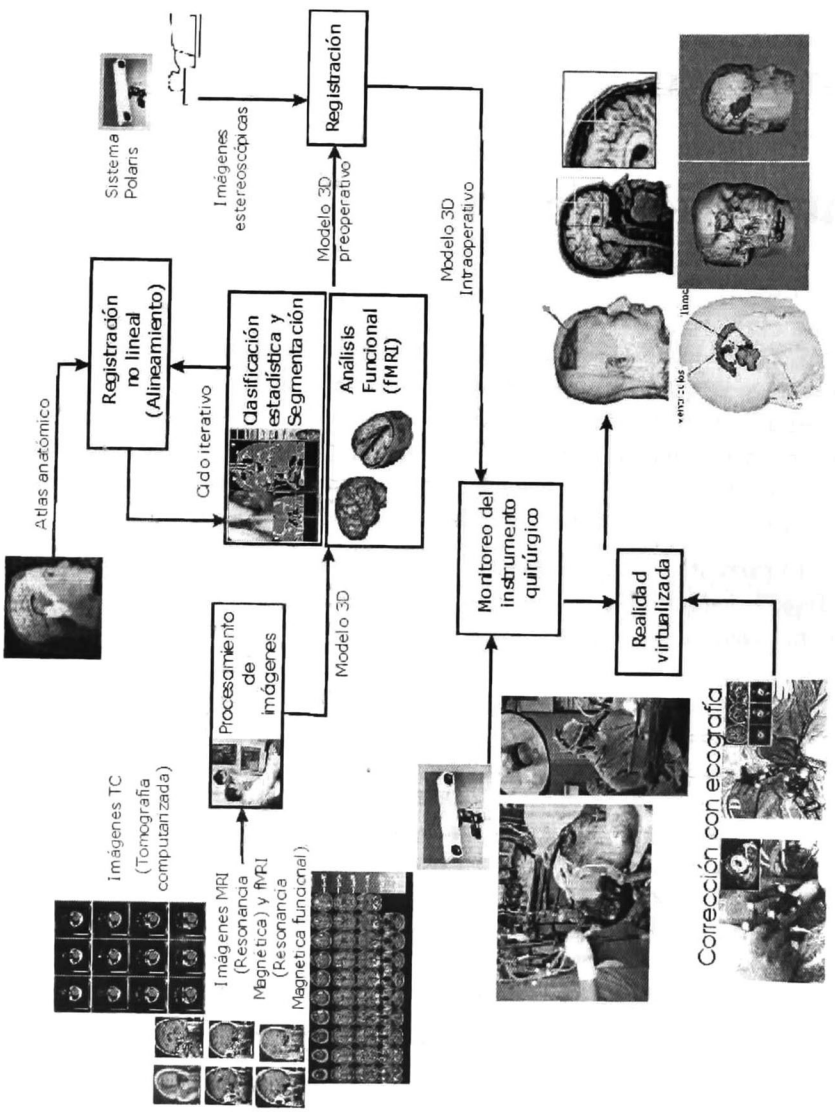


Figura 1.1: Esquema general del proyecto Salud-Conacyt #49

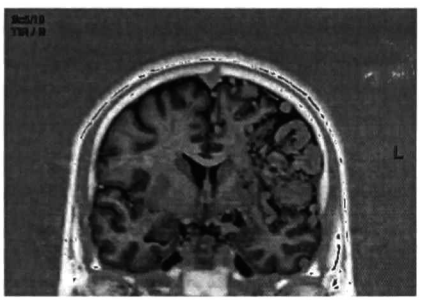
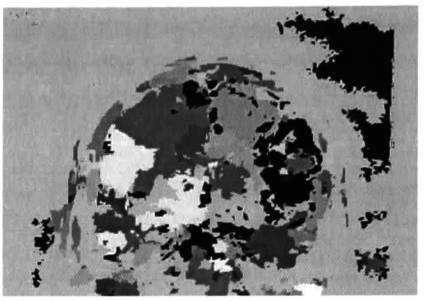
a)  b) 

Figura 1.2: Ejemplo de los resultados obtenidos con el software Analyze. a) Imagen original (MR); b) Resultado arrojado por Analyze con segmentación automática.

todavía hay mucho campo de investigación en las diferentes tareas involucradas; prueba de ello es la gran cantidad de trabajos que se enfocan en estos tópicos actualmente.

1.1. Motivación

La segmentación de imágenes es una tarea común en las aplicaciones de procesamiento digital de imágenes y tiene además, una gran importancia en aplicaciones médicas. En las últimas décadas han sido muchos los esfuerzos y las propuestas que se han hecho a este respecto [25, 13, 7, 15, 1, 35, 31, 22, 24, 23, 21] y se han desarrollado grandes sistemas de software para realizar el procesamiento de imágenes médicas. Un ejemplo de estos sistemas es el llamando *Analyze* [8] de la *Mayo Foundation*, el cual es el resultado de 15 años de investigación y desarrollo en el Mayo Clinic College of Medicine, en USA, y que puede ser adquirido por la cantidad de USD$10,000 aproximadamente (incluye Analyze Biomedical Imaging Software, Developers Add-On y Brain Atlas Add-On). La figura 1.2 muestra un ejemplo de los resultados obtenidos con este software al realizar la segmentación automática con uno de sus algoritmos para el caso de una imagen de resonancia magnética. Note como existe una sobre-segmentación de las estructuras cerebrales.

Una vez que se tienen las imágenes médicas segmentadas, la siguiente tarea es la representación de los datos volumétricos, es decir, la reconstrucción tridimensional de la cabeza del paciente y de las diferentes estructuras cerebrales. Podríamos decir que la reconstrucción tridimensional es una tarea estándar resuelta, pero si vamos más allá de la sola reconstrucción para su visualización en un monitor y pensamos en sus aplicaciones posteriores, encontramos que también hay representaciones que pueden ser útiles para determinar el tratamiento que se dará al paciente (por ejemplo, la cantidad de radiación que se emitirá en una determinada área del cerebro para el tratamiento de tumores, etc). Por ello, también abordamos la construcción de modelos basados en esferas.

Resuelta la tarea del modelado tridimensional, muchas veces es necesario realizar el registro (otras veces llamado alineamiento) de modelos que corresponden a un mismo

objeto, pero que se ha deformado con el tiempo. Una vez más nos topamos con un área que recibe gran atención en la investigación [27, 20, 11, 6], pero dada la propuesta de modelado 3D que se presenta en este trabajo, es necesario proponer también la manera de hacer el registro entre dos modelos obtenidos mediante el algoritmo propuesto basado en esferas.

1.2. Antecedentes

Cuando estamos tratando con el problema de la segmentación de tumores en imágenes del cerebro, una forma de resolver el problema es usando imágenes de resonancia magnética (MRI por sus siglas en inglés), para tomar ventaja de las características que resalta en la imagen cada una de sus variantes: T1, T1-weighted, T2 y T2-weighted, lo cual hace el problema de la segmentación del tumor más fácil de tratar. Ésto se debe a que algunas de estas variantes resaltan en la imagen la masa tumoral, diferenciándola de otras estructuras, lo cual se logra al aplicar al paciente un agente contrastante antes de la adquisición de las imágenes. Por tanto, tomando estas imágenes (con y sin contraste), se pueden combinar, diferenciar, etc. y puede desarrollarse un algoritmo para la segmentación automática de tumores [31]. Otros métodos usan un atlas probabilístico del cerebro y buscan anormalidades (*outliers*) entre los datos del paciente y el atlas ([22, 24, 23, 21]).

El uso de redes neuronales artificiales en el procesamiento de imágenes médicas es un área que está recibiendo mucha atención con una gran variedad de aplicaciones tales como la segmentación o la clasificación de tejidos, entre otras. Las redes neuronales auto-organizadas como los Mapas de Kohonen o Mapas Auto-organizados (SOM por sus siglas en inglés: *Self-Organizing Maps*), *Neural Gas* (NG) y *Growing Neural Gas* (GNG, [12]) han sido usadas ampliamente cuando se necesita preservar la topología de los datos [12, 18]. Un ejemplo reciente es el trabajo de Angelopoulou et al. [1], donde se utiliza la red GNG para encontrar marcas (*landmarks*) en imágenes médicas bidimensionales.

Los principales trabajos sobre segmentación que sirvieron como base al trabajo desarrollado en esta tesis son los siguientes:

- Prastawa et al [22] presenta un método para la segmentación de tumores cerebrales basado en la detección de irregularidades al comparar la imagen del paciente contra la obtenida de un atlas un atlas.
- Andrade [2] preprocesa la imagen a segmentar con el fin de eliminar ruido (suavizarla) pero al mismo tiempo resaltar los contornos. La segmentación se hace en un esquema de crecimiento de regiones, pero el usuario debe especificar las semillas.
- Xu [35] propone un modelo deformable 2D llamado *GGVF-Snake* y presenta un tipo nuevo de fuerza externa llamada *Generalized Gradient Vector Flow*(GGVF) calculada con los datos de la imagen y que sirve para guiar la evolución de la

curva hacia los bordes. Se muestra su aplicación para la reconstrucción de la corteza cerebral.

- Angelopoulou et al [1] propone el uso de redes neuronales para la extracción automática de "marcadores" en los contornos de imágenes MR. Presenta una comparación del desempeño de la red *"Growing Neural Gas"* (GNG) contra las redes *Self-Organized Maps* (SOM) y *Neural Gas* (NG), siendo mejor la primera. Su desventaja es que el usuario debe dar un pre-procesamiento a las imágenes para resaltar el objeto de interés (aplicar un umbral que remueva todo excepto los ventrículos). Después extrae los contornos del objeto de interés y de éstos selecciona manualmente las entradas; las posiciones finales de las neuronas corresponden a los marcadores buscados.

Por otro lado, la representación de objetos volumétricos usando primitivas geométricas como puntos, lineas o planos, es una tarea común. Tal vez el algoritmo más famoso y ampliamente usado para la representación tridimensional es el propuesto por W. Lorensen y H. Cline en su artículo de 1987 [20], donde se presentó un algoritmo nuevo para crear modelos tridimensionales basados en triángulos representando las superficies de objetos. Tiene una tabla de 15 casos básicos para determinar los triángulos y ha sido ampliamente usado desde entonces en toda clase de aplicaciones. Posteriormente, F. Triquet, P. Meseure y C. Chaillou en [11], presentaron una manera de optimizar la implementación del bien conocido *"Marching Cubes"* [20] evitando visitar los mismos vértices varias veces y utiliza solo cubos útiles (aquellos que intersecan a la superficie). Propone usar 20 casos base.

Un algoritmo alternativo lo encontramos en [6], donde J. Burguet e I. Bloch proponen un método para obtener un modelo de la cabeza basado en tetraedros que mantenga la topología lo mejor posible, por lo cual dichos tetraedros deben cumplir ciertos requisitos o características. Finalmente dichos tetraedros son etiquetados de acuerdo a las imágenes segmentadas de las que se construyeron.

La unión de esferas (UoS, por sus siglas en inglés) propuesta por V. Ranjan and A. Fournier en [27] trata la obtención de modelos basados en esferas usando la tetraedrización de Delaunay para describir la forma de un objeto, desafortunadamente necesitan en general una gran cantidad de primitivas. La complejidad de este algoritmo es $O(n^2)$ tanto en tiempo como en número de primitivas.

También en [27] encontramos el problema del registro o alineamiento de modelos 3D. Para resolverlo, el autor indica la correspondencia exacta de cierto número de esferas y luego usa mínimos cuadrados para estimar la transformación. La mayoría de los métodos de registración suponen que la correspondencia entre los puntos es conocida y solamente resuelven para la transformación (o viceversa), pero en [14] se propone un algorimtmo llamado *Thin-Plate Robust Point Matching (TPS-RPM)*, el cual encuentra la correspondencia entre los dos conjuntos de puntos e iterativamente transforma uno de esos conjuntos de puntos para ajustarlo al otro. Este algoritmo se propuso como una forma de resolver el problema de registro no rígido entre dos conjuntos de puntos en el espacio

2D/3D.

Es importante notar que los métodos de registro se pueden clasificar de múltiples maneras

- Dimensionalidad: es decir, si el registro es entre imágenes (2D:2D), o entre una imagen y un modelo tridimensional (2D:3D), o entre modelos tridimensionales (3D:3D).

  Naturaleza de la información: dependiendo si para realizar el registro se utilizan *landmarks*, propiedades de los voxels, o alguna otra información.

- Naturaleza de la transformación: depende si la transformación que nos interesa es rígida, afina, proyectiva u otra no-rígida.

- Dominio: si la transformación es local o global.

- Nivel de interacción: si el algoritmo es interactivo, semiautomático o totalmente automático.

  Modalidad: si el registro es entre un solo tipo de imágenes se llama monomodal; en caso que sea entre imágenes de diferente naturaleza se llama multimodal: CT, MR, fMR, PET)

En el trabajo sobre registro presentado en esta tesis, limitaremos nuestra atención al registro automático 3D:3D de modelos cuya transformación puede ser no-rígida.

1.3. Problemas a resolver

Analizando los resultados en la literatura, resumidos brevemente en la sección anterior, se puede concluir que

- La segmentación de imágenes médicas es un área que esta recibiendo mucha atención y aunque existen muchas propuestas con diferentes técnicas, aún es un área abierta en la cual se pueden hacer propuestas novedosas o mejoras a propuestas existentes.

- El modelado de objetos tridimensionales utilizando otras entidades geométricas diferentes a los puntos, líneas o planos resulta interesante y son muy pocos los trabajos que tratan de modelos basados en esferas, pero se sospecha que podrían ser de gran utilidad y tener impacto positivo en la comunidad de procesamiento de imágenes médicas.

  Existen gran variedad de métodos para realizar el registro de imágenes médicas y de modelos tridimensionales, aunque la mayoría de éstos toma como punto de partida ciertos marcadores para realizar la deformación del modelo inicial hacia el esperado, y aunque existen propuestas al respecto, se puede presentar alguno que sirva para los modelos basados en otras entidades geométricas como las esferas.

Por tanto, los problemas que se abordan en este trabajo y las propuestas realizadas son:

- **Segmentación** de imágenes bidimensionales y modelos tridimensionales. Para el caso 2D se presentan dos métodos, uno basado en información de textura y bordes de la imagen y otro basado en redes neuronales e información gradiente dentro del marco del álgebra geométrica conformal (AGC). Para el caso 3D se seguirá el mismo método usando redes neuronales e información gradiente.

  Modelado de las estructuras cerebrales tanto en su representación estándar para visualización y análisis por parte del médico (modelos realizados también con la propuesta basada en redes neuronales), como con base en esferas. Para este segundo tipo de modelos, se presenta una propuesta desarrollada en el marco del álgebra geométrica conformal, dividiendo el espacio en pequeños cubos con los cuales se determinan las esferas necesarias para crear el modelo.

  Registro del paciente y el instrumental quirúrgico presente en la sala de operaciones con el modelo virtual creado, así como el registro de modelos tridimensionales. Para el registro entre instrumental y modelo virtual se utilizan marcadores de referencia y el sistema de seguimiento óptico *Polaris*; las transformaciones se estiman en el marco del AGC como *motores* (transformaciones de cuerpo rígido). En la calibración endoscopio-Polaris se utilizan las líneas definidas por los motores para encontrar la transformación necesaria. Para el registro entre modelos, se muestra la utilidad del mismo método propuesto usando redes neuronales y además otro basado en TPS-RPM que resuelve sin conocimiento previo acerca de la correspondencia entre las entidades usadas o la transformación que las afecta.

El abordar estos problemas, además de contribuir al avance del proyecto Salud-Conacyt No. 49, permite hacer propuestas novedosas publicables en conferencias internacionales o revistas de divulgación científica.

Capítulo 2

Fundamentos

2.1. Álgebra Geométrica

El álgebra geométrica o álgebra de Clifford fue introducida por William Kingdom Clifford (1845-1879). Desde entonces, han habido muchas propuestas interesantes para aplicarla en áreas como robótica y visión computacional, entre otras. De manera general, el álgebra geométrica (AG) es un sistema matemático que permite manejar los objetos geométricos como entidades algebraicas que pueden ser fácilmente tratadas con operaciones algebraicas (suma, resta, multiplicación, división).

Tratando de hacer una analogía que nos ayude a comprender intuitivamente por qué resulta interesante esta forma de tratar las entidades geométricas algebraicamente, recordemos cómo ayudó al desarrollo de la lógica el hecho de que George Boole (1815-1864) se haya dado a la tarea de captar las estructuras del pensamiento de un modo algebraico. Ésto hizo que se cambiara el modo de pensar en relación a los patrones de la lógica, los cuales se trasladaron a estructuras algebraicas pero sin alterarlos de manera intrínseca, facilitando su comprensión y manipulación con esta nueva forma de verlos.

De forma similar, el álgebra geométrica nos facilita la comprensión y manipulación de entidades geométricas al ser tratadas con herramientas algebraicas. Algunos autores se refieren a este sistema como *álgebra Geométrica (AG)* porque estan más interesados en la interpretación geométrica de las entidades algebraicas, mientras que otros se refieren a éste como *álgebra de Clifford (AC)* porque estan más interesados en los aspectos algebraicos. En este trabajo adoptamos el término "álgebra geométrica" y daremos una breve descripción, aunque el lector interesado puede consultar excelente material en [4, 26, 29, 9]

El álgebra geométrica $G_{p,q,r}$ se construye sobre el espacio vectorial $\mathcal{V}^{p,q,r}$, donde p es el número de elementos de la base vectorial que elevados al cuadrado resultan en 1, q es el número de elementos cuyo cuadrado es -1 y r es el número de elementos cuyo cuadrado es 0. Si $p \neq 0$ y $q = r = 0$, la métrica es Euclidiana; si solamente $r = 0$, la métrica es pseudoeuclidiana; si $p \neq 0, q \neq 0, r \neq 0$, la métrica es degenerada. La dimensión de

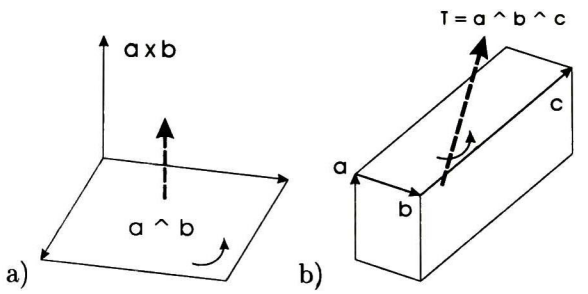


Figura 2.1: Interpretación geométrica del producto wedge a) entre dos vectores; b) entre tres vectores

$G_{n=p,q,r}$ es 2^n, y G_n se construye por la aplicación del "producto geométrico" (o producto Clifford) sobre la base vectorial e_i. El producto geométrico entre dos vectores a, b se define como en (2.1) y tiene dos partes: el producto interior $a \cdot b$ es la parte simétrica, mientras que el producto *wedge* $a \wedge b$ es la parte antisimétrica.

$$ab = a \cdot b + a \wedge b. \tag{2.1}$$

Por tanto, el producto geométrico de dos vectores e_i, e_j que pertenecen a la base del espacio vectorial esta dado por

$$e_i e_j = \begin{cases} 1 & i = j \in \{1, ..., p\} \\ -1 & i = j \in \{p+1, ..., p+q\} \\ 0 & i = j \in \{p+q+1, ..., n = p+q+r\} \\ e_{ij} = e_i \wedge e_j = -e_j \wedge e_i & i \neq j \end{cases} \tag{2.2}$$

El resultado del producto wedge tiene magnitud $|a|\,|b| \sin\theta$ pero no es ni un escalar ni un vector, sino lo que llamaremos *bivector* o segmento de plano orientado. El producto wedge tiene la misma magnitud que el producto cruz y comparte su propiedad anticonmutativa: $a \wedge b = -(b \wedge a)$; el producto wedge de un vector con otro paralelo a él es igual a cero. Una forma de visualizar el producto wedge es imaginarlo como el segmento de plano resultante de deslizar el vector a a lo largo del vector b (ver figura 2.1.a). De manera similar, si el bivector $a \wedge b$ es deslizado a lo largo de otro vector c, entonces obtenemos un *trivector* o elemento de volumen orientado (ver figura 2.1.b).

Ésto resulta en una base para G_n que contiene elementos de diferente grado llamados *blades*; por ejemplo: escalar (1), vectores $(e_1, ...e_n)$, bivectores $(e_{12}, e_{23}...)$, trivectores (e_{123}), etc. Esta base es llamada *base de blades*, sus elementos son los *blades base* y el elemento de mayor grado $I = e_1 \wedge e_2 ... \wedge e_n$ es llamado pseudoescalar. Una combinación lineal de *blades* de la base, todos del mismo grado k se llama k-vector,; por ejemplo

$$\begin{aligned} A &= a_1 e_1 + a_2 e_2 + a_3 e_3 : \text{es un 1-vector} \\ B &= b_1 e_{12} + b_2 e_{23} + b_3 e_{31} : \text{es un 2-vector} \end{aligned}$$

donde $a_i, b_i, i = 1, 2, 3$ son escalares. La combinación lineal de k-vectores se llama *multivector*; por ejemplo

$$M = m_1 e_1 + m_2 e_3 + m_3 e_{23} + m_3 e_{31} + m_4 e_{123} \tag{2.3}$$

y estos multivectores son los que representan diferentes objetos geométricos (dependiendo de sus características), como puntos, líneas, planos, círculos, esferas, etc. La interpretación de los multivectores dependerá del álgebra en la cual se este trabajando. Por ejemplo, un punto es representado en el álgebra $G_{3,0,0}$ (el álgebra del espacio Euclidiano 3D) como

$$x = ae_1 + be_2 + ce_3$$

Sin embargo, un círculo no puede ser definido en $G_{3,0,0}$ como un simple multivector (aunque si es posible definirlo paramétricamente), pero es posible definirlo en $G_{4,1}$ como un 4-vector $Z = S_1 \wedge S_2$ (la intersección de dos esferas en el mismo espacio).

Dado un multivector M, si estamos interesados en extraer solamente los blades de un grado determinado, escribimos $\langle M \rangle_r$ donde r es el grado de los blades que deseamos extraer, con lo cual obtenemos un multivector homogéneo M', o un r-vector. Por ejemplo, sea M el multivector de (2.3); entonces podríamos obtener

$$\begin{aligned} \langle M \rangle_1 &= m_1 e_1 + m_2 e_3 \\ \langle M \rangle_2 &= m_3 e_{23} + m_3 e_{31} \\ \langle M \rangle_3 &= m_4 e_{123} \end{aligned}$$

Los movimientos de cuerpo rígido se definen en el álgebra geométrica usando entidades que se llaman *versores*, de los cuales existen unos especiales llamados *rotores, trasladores y motores* que rotan, trasladan y aplican ambas transformaciones, respectivamente, a las entidades que se aplican. Estos versores se aplican multiplicativamente a las entidades que se desean transformar. Sin embargo, es importante notar que en $G_{3,0,0}$, aunque las rotaciones se aplican por multiplicación, la traslación se aplica por suma directa; así que para aplicarla también de forma multiplicativa necesitamos usar otra álgebra. Todas estas transformaciones se llevan a cabo de manera ideal en el álgebra geométrica Conformal $G_{4,1,0}$, que se explica enseguida.

2.1.1. El álgebra geométrica conformal

Trabajar en el álgebra geométrica Conformal (AGC) $G_{4,1,0}$ significa embeber el espacio Euclidiano en un espacio de dimensión superior con dos vectores base extra, los cuales tienen un significado particular; como veremos más adelante, de esta forma podemos representar objetos particulares del espacio Euclidiano con subespacios del espacio conformal (expresados en los multivectores).

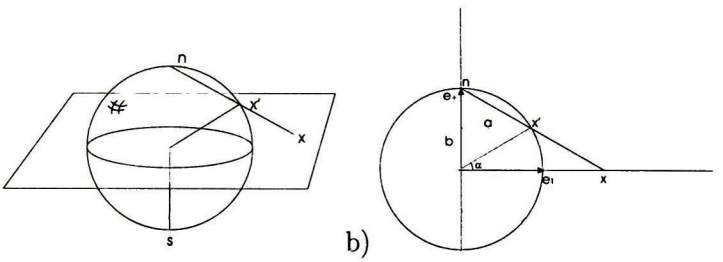


Figura 2.2: a) Proyección estereográfica; b) Proyección de puntos del círculo al plano y viceversa

La idea detrás de la geometría conformal es interpretar los puntos como puntos proyectados estereográficamente[1] Consideremos la figura 2.2.a. Imaginemos que colocamos una fuente de luz en el polo norte (marcado como **n**); entonces cada punto en la esfera proyecta una sombra en el papel (plano en la figura). Para simplificar los cálculos analicemos cómo se proyectan puntos de la esfera al plano y viceversa para el caso 1D, como se muestra en la figura 2.2.b (la esfera ahora es un círculo; asumimos radio 1 por simplicidad).

Además del vector del espacio 1D e_1, se añade el vector e_+ perpendicular a éste. La proyección sobre el eje e_1 de un punto $x' = ae_1 + be_+$ del círculo esta dada por [2]

$$x = \left(\frac{a}{1-b}\right) e_1 \tag{2.4}$$

y para proyectar un punto ce_1 ($c \in \mathbb{R}$) al círculo hay que estimar los factores apropiados a, b para el vector $x' = ae_1 + be_+$, lo cual nos lleva a

$$x' = \frac{2c}{c^2+1} e_1 + \frac{c^2-1}{c^2+1} e_+ \tag{2.5}$$

y si utilizamos como coordenada homogénea a e_-, la representación homogenea del punto en el círculo es

$$x' = ce_1 + \frac{1}{2}\left(c^2-1\right) e_+ + \frac{1}{2}\left(c^2+1\right) e_- \tag{2.6}$$

Así, al elevar x' al cuadrado tenemos

$$\begin{aligned} (x')^2 &= c^2 + \left(\frac{1}{2}\left(c^2-1\right)\right)^2 - \left(\frac{1}{2}\left(c^2+1\right)\right)^2 \\ &= c^2 + \frac{1}{4}\left[c^4 - 2c^2 + 1 - c^4 - 2c^2 - 1\right] \\ &= 0 \end{aligned}$$

[1] Proyección estereográfica es una forma de hacer un mapa plano de la tierra

[2] Es fácil deducir la expresión si utilizamos semejanza de triángulos.

De esta manera, los punto Euclidianos x, proyectados estereográficamente sobre el círculo son representados por el conjunto de vectores nulos de este nuevo espacio.

Si observamos nuevamente lo anterior, vemos que los vectores que añadimos son e_+ y e_-, que al elevarlos al cuadrado resultan en $1, -1$, respectivamente. Con estos dos vectores se definen los vectores nulos e_0 y e_∞:

$$e_0 = \frac{1}{2}(e_- - e_+); \quad e_\infty = (e_- + e_+), \tag{2.7}$$

interpretados como el origen y el punto en el infinito, respectivamente. Además, definimos

$$E = e_\infty \wedge e_0 = e_+ \wedge e_-$$

Con estos elementos, se pueden conprobar fácilmente las siguientes relaciones

$$\begin{array}{ccccc} e_0^2 = e_\infty^2 = 0 & & e_\infty \wedge e_0 = -1 & & E = e_+ e_- \\ E^2 = 1 & & E e_\infty = -e_\infty & & E e_0 = e_0 \\ I_C = I_E \wedge E & & \text{donde} & & I_E = e_1 \wedge e_2 \wedge e_3 \end{array} \tag{2.8}$$

De ahora en adelante, los puntos en el espacio euclidiano 3D los representaremos con letras minúsculas, mientras que los puntos en el espacio conformal con letras mayúsculas; así mismo, las entidades en el AGC serán expresadas en lo que se llama el *Inner Product Null Space* (IPNS), no en el *Outer Product Null Space* (OPNS), a menos que se especifique lo contrario explícitamente (ver características de estos espacios en el Apéndice B.3). Para mapear un punto $x \in G_3$ al espacio conformal en $G_{4,1}$, usamos

$$X = x + \frac{1}{2}x^2 e_\infty + e_0 \tag{2.9}$$

(lo cual se puede deducir fácilmente utilizando las ecuaciones (2.5) y (2.7)).

Sean X_1, X_2 dos puntos conformales. Si restamos X_2 de X_1, obtenemos

$$X_1 - X_2 = (x_1 - x_2) + \frac{1}{2}(x_1^2 - x_2^2)e_\infty + e_0 \tag{2.10}$$

y si elevamos al cuadrado este resultado, obtenemos

$$(X_1 - X_2)^2 = (x_1 - x_2)^2 \tag{2.11}$$

Por tanto, si deseamos medir la distancia euclideana entre dos puntos expresados en conformal, podemos usar (2.11). Otra forma de obtener esta distancia es usar

$$|x_1 - x_2| = \sqrt{-2(X_1 \quad X_2)} \tag{2.12}$$

Para obtener la magnitud del vector en el espacio 3D, simplemente hacemos

$$|x| = \sqrt{-2(X \cdot e_0)} \tag{2.13}$$

El producto punto entre dos vectores conformales X_1 y X_2 resulta en $X_1 \cdot X_2 = (x_1 \cdot x_2) - \frac{1}{2}x_1^2 - \frac{1}{2}x_2^2$ (de donde se deduce (2.12)); por tanto

$$x_1 \cdot x_2 = X_1 \quad X_2 + \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2 \tag{2.14}$$

El ángulo θ entre dos vectores x_1 y x_2, calculado desde su representación conformal X_1, X_2 se calcula usando (2.14) y (2.13)

$$\theta = \arccos\left(\frac{X_1 \quad X_2 + \frac{1}{2}x_1^2 + \frac{1}{2}x_2^2}{|x_1||x_2|}\right) \tag{2.15}$$

Por otro lado, para calcular el vector $x_3 \in \langle G_3 \rangle_1$ perpendicular a los vectores x_1 y x_2, primero calculamos el producto exterior entre sus representaciones conformales $A = X_1 \wedge X_2$, y después tomamos los coeficientes de los componentes e_{12}, e_{23}, e_{31} (representados como $\langle A \rangle_{e_{12}}, \langle A \rangle_{e_{23}}, \langle A \rangle_{e_{31}}$), los cuales definen el plano $\mathbf{b} = \langle A \rangle_{e_{12}} \wedge e_{12} + \langle A \rangle_{e_{23}} \wedge e_{23} + \langle A \rangle_{e_{31}} \wedge e_{31}$, dual al vector x_3, el cual se obtiene multiplicándolo por $I_E = e_1 \wedge e_2 \wedge e_3$

$$x_3 = I_E \quad \mathbf{b} \quad = \quad I_E(\langle A \rangle_{e_{12}} \wedge e_{12} + \langle A \rangle_{e_{23}} \wedge e_{23} + \langle A \rangle_{e_{31}} \wedge e_{31}) \tag{2.16}$$

Como ya se mencionó, podemos usar el AGC para representar objetos particulares del espacio Euclidiano. Las esferas son especialmente interesantes porque son las entidades básicas en el AGC, a partir de las cuales se derivan las otras entidades. Las esferas con centro en c y radio ρ se representan como

$$S = c + \frac{1}{2}(c^2 - \rho^2)e_\infty + e_0 \tag{2.17}$$

De hecho, se puede pensar en los puntos conformales X como esferas degeneradas de radio $\rho = 0$. Las entidades como los círculos Z y los pares de puntos PP se pueden derivar de las esferas, tal como se muestra en la figura 2.3 y en la Tabla 2.1. Nótese que como muestra la ecuación (2.17), la información sobre el centro y el radio de la esfera se encuentra codificada en un vector 5-dimensional. Para extraer dicha información, usamos (2.18) y (2.19).

$$c = (S \wedge E) \quad E, \tag{2.18}$$

$$\rho = \sqrt{-S \quad S}, \tag{2.19}$$

Para cualquier versión de magnitud arbitraria de la esfera S, aplicamos

$$\rho = \sqrt{-\left(\frac{S}{-S \cdot e_\infty}\right)^2} \tag{2.20}$$

Sean S_1, S_2 dos esferas en el AGC. Si aplicamos el producto interior entre ellas, obtenemos

$$\begin{aligned} S_1 \cdot S_2 &= (c_1 + \frac{1}{2}(c_1^2 - \rho_1^2)e_\infty + e_0) \cdot (c_2 + \frac{1}{2}(c_2^2 - \rho_2^2)e_\infty + e_0) \\ S_1 \cdot S_2 &= c_1 \cdot c_2 - \frac{1}{2}c_1^2 - \frac{1}{2}c_2^2 + \frac{1}{2}\rho_1^2 + \frac{1}{2}\rho_2^2 \\ &= -\frac{1}{2}(c_1 - c_2)^2 + \frac{1}{2}(\rho_1^2 + \rho_2^2). \end{aligned} \tag{2.21}$$

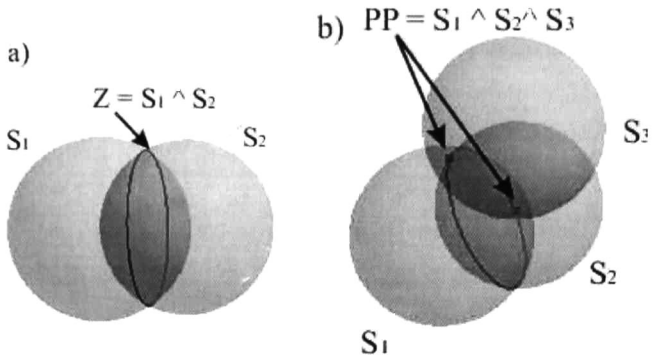


Figura 2.3: Obtención del círculo y el par de puntos a partir de las esferas: a) Círculo Z como la intersección de dos esferas; b) Par de puntos PP como la intersección de tres esferas.

Por otro lado, si restamos S_2 de S_1, obtenemos

$$S_1 - S_2 = (c_1 - c_2) + \frac{1}{2}(c_1^2 - c_2^2 - \rho_1^2 + \rho_2^2)e_\infty + e_0 \tag{2.22}$$

y si elevamos al cuadrado este resultado, obtenemos

$$(S_1 - S2)^2 = (c_1 - c_2)^2 \tag{2.23}$$

Por tanto, si necesitamos medir la distancia Euclidiana entre los correspondientes centros de las dos esferas, podemos aplicar (2.23) o bien (2.24), la cual resulta de (2.21).

$$(c_1 - c_2)^2 = -2S_1 \cdot S_2 + \rho_1^2 + \rho_2^2. \tag{2.24}$$

La tabla 2.1 muestra las representaciones en IPNS y OPNS de las entidades en el AGC. Como en el caso de la esfera, se puede obtener la información de las diferentes entidades (como la normal de un plano, la dirección de una línea, etc) mediante simples operaciones algebraicas; además, todas estas entidades y sus transformaciones pueden ser manejadas fácilmente usando los operadores de movimiento rígido descritos más delante.

Transformaciones en el AGC

En el AG, existen operadores específicos llamados versores para modelar las rotaciones, traslaciones y dilataciones. En general, un versor G es un vector que puede ser expresado como el producto geométrico de vectores no singulares.

$$G = \pm a_1 a_2 ... a_k \tag{2.25}$$

Tabla 2.1: Entidades en el AGC en representación estándar (IPNS) y dual (OPNS)

Entidad	Representación estándar (IPNS)	Representación dual (OPNS) (*)
Esfera	$S = p + \frac{1}{2}(p^2 - \rho^2)e_\infty + e_0$	$S^* = A \wedge B \wedge C \wedge D$
Punto	$X = x + \frac{1}{2}x^2 e_\infty + e_0$	$X^* = (-Ex - \frac{1}{2}x^2 e_\infty + e_0)I_E$
Plano	$P = nI_E - de_\infty$ $n = (a-b) \wedge (a-c)$ $d = (a \wedge b \wedge c)I_E$	$P^* = e_\infty \wedge A \wedge B \wedge C$
Línea	$L = rI_E + e_\infty mI_E$ $r = a - b$ $m = a \wedge b$	$L^* = e_\infty \wedge A \wedge B$
Círculo	$Z = S_1 \wedge S_2$	$Z^* = A \wedge B \wedge C$
Par de puntos	$PP = S_1 \wedge S_2 \wedge S_3$	$PP^* = A \wedge B$, $X* = e_\infty \wedge X$

En el AGC las rotaciones son representadas por los *rotores*

$$\begin{aligned} R &= e^{\frac{1}{2}\mathbf{b}\theta} \\ &= \cos(\frac{\theta}{2}) + \mathbf{b}\sin(\frac{\theta}{2}) \end{aligned} \tag{2.26}$$

donde $\mathbf{b}$ es el bivector dual al eje de rotación y θ es el ángulo de rotación. La rotación de una entidad se realiza multiplicándola por la izquierda con el rotor R y por la derecha con la reversión[3] del rotor $\tilde{R}$. Por ejemplo, la rotación del punto X se escribe $RX\tilde{R}$.

Si deseamos trasladar una entidad con respecto a un vector de traslación $t \in \langle G_3 \rangle_1$, se usa el *traslador*

$$\begin{aligned} T &= e^{-\frac{te_\infty}{2}} \\ T &= 1 + \frac{1}{2}e_\infty t \end{aligned} \tag{2.27}$$

el cual puede ser interpretado como un rotor especial dado en un espacio nulo puesto que $e_\infty^2 = 0$. Al igual que las rotaciones, la traslación de entidades se realiza al multiplicarlas por la izquierda con el traslador T y por la derecha con su reversión $\tilde{T}$: $X' = TX\tilde{T}$

Para expresar movimientos de cuerpo rígido, se pueden aplicar los rotores y los trasladores consecutivamente. El operador resultante se llama *motor*

$$M = TR \tag{2.28}$$

[3]La reversión se define en el apéndice B.2

y es un multivector especial de grado par. Para ver cómo esta formado, realicemos la multiplicación de R y T

$$\begin{aligned} M &= TR \\ &= (1+\frac{1}{2}e_\infty t)(\cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2})) \\ &= \cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2})+\frac{1}{2}e_\infty(t(\cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2}))) \\ &= R+R' \end{aligned} \tag{2.29}$$

Dado que la multiplicación de un vector $t \in \langle G_3\rangle_1$ por un bivector $\mathbf{b} \in \langle G_3\rangle_2$ resulta en un multivector de la forma $\lambda_1 e_1 + \lambda_2 e_2 + \lambda_3 e_3 + \lambda_4 e_{123}$, y puesto que $t\cos(\frac{\theta}{2}) \in \langle G_3\rangle_1$ podemos reescribir (2.29) como

$$\begin{aligned} M &= \cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2})+\frac{1}{2}e_\infty(t\cos(\frac{\theta}{2})+t\mathbf{b}\sin(\frac{\theta}{2})) \\ &= \cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2})+\frac{1}{2}e_\infty(t\cos(\frac{\theta}{2})+\lambda_1 e_1+\lambda_2 e_2+\lambda_3 e_3+\lambda_4 e_{123} \\ &= \cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2})+e_\infty(t'+\lambda e_{123}) \\ &= \cos(\frac{\theta}{2})+\mathbf{b}\sin(\frac{\theta}{2})+e_\infty t'+\lambda e_{\infty 123}) \end{aligned} \tag{2.30}$$

donde $t' \in \langle G_3\rangle_1$ y $\lambda = \frac{1}{2}\lambda_4$. Note que $e_\infty t'$ es un bivector con componentes $e_{\infty 1}, e_{\infty 2}, e_{\infty 3}$.

Si tomamos solamente las partes bivectoriales del motor M, obtenemos

$$\begin{aligned} \langle M\rangle_2 &= \langle R\rangle_2 + \langle R'\rangle_2 \\ &= \mathbf{m}+\mathbf{m}' \\ &= \sin(\frac{\theta}{2})\mathbf{b}+e_\infty t' \end{aligned} \tag{2.31}$$

Por tanto, si expresamos el vector t' en términos de su bivector dual $t' = t''I_E$, podemos reescribir (2.31) como

$$\langle M\rangle_2 = b'I_E + e_\infty t''I_E \tag{2.32}$$

Y si vemos la representación de las líneas en la tabla 2.1, observamos que la parte bivectorial del motor M es de hecho una línea y corresponde al eje del tornillo en el cual se realiza la rotación y traslación del objeto. El movimiento de cuerpo rígido para un punto X se puede escribir como $X' = MX\tilde{M}$.

Otro operador que resulta útil es el dilator que se define como

$$D_\lambda = e^{\frac{-\log(\lambda)\wedge E}{2}} \tag{2.33}$$

donde λ es el factor de dilatación y $E = e \wedge e_0$. Para dilatar una entidad, por ejemplo una esfera S, aplicamos el dilator de igual forma que para las rotaciones o traslaciones,

es decir, multiplicamos S por la derecha con el dilator D_λ y por su reversión $\tilde{D}_\lambda$ por la izquierda: $S' = D_\lambda S \tilde{D}_\lambda$

Para ilustrar el movimiento inducido por estos operadores, la figura 2.4 muestra la aplicación de éstos a diferentes entidades.

2.2. Flujo Vectorial Gradiente Generalizado

El *flujo vectorial gradiente generalizado* (GGVF por las siglas en inglés de *Generalized Gradient Vector Flow*) es un campo vectorial definido en el dominio de la imagen. Para obtener el GGVF, necesitamos primero construir un *mapa de bordes* $f(x, y)$ a partir de la imagen $I(x, y)$. Este mapa debe cumplir la propiedad de tener valores más grandes cerca de los bordes y se puede calcular usando el algoritmo Canny (ver apéndice B.1). Si la característica de la imagen que nos interesa es otra diferente a los bordes, el mapa puede redefinirse para tomar valores grandes en esa característica en particular.

Los mapas de bordes tienen tres características importantes. La primera es que el gradiente de $f(x, y)$, ∇f tiene vectores apuntando hacia los bordes; la segunda es que estos vectores generalmente tienen magnitudes grandes solamente cerca de los bordes; la tercera es que en regiones homogéneas donde $I(x, y)$ es constante, ∇f es prácticamente cero.

Una vez obtenido $f(x, y)$, podemos definir el flujo vectorial gradiente generalizado, GGVF, como un campo vectorial $\mathbf{v}(x, y) = [u(x, y), v(x, y)]$ derivado de la imagen que es la solución para

$$\mathbf{v} = (g(|\nabla f|)\nabla^2 \mathbf{v} - h(|\nabla f|)(\mathbf{v} - \nabla f)) \tag{2.34}$$

donde

$$g(|\nabla f|) = e^{-\frac{|\nabla f|}{\mu}} \tag{2.35}$$

$$h(|\nabla f|) = 1 - g(|\nabla f|) \tag{2.36}$$

y μ es un escalar. Las funciones de peso $g(\cdot)$ y $h(\cdot)$ se definieron de esa forma porque se desea que el campo vectorial $\mathbf{v}$ varíe lentamente (o suavemente) en localidades lejanas a los bordes, pero que se mantenga cercano a ∇f cerca de los bordes; por lo que deben ser monotonicamente no creciente y no decreciente, respectivamente.

Este campo vectorial difunde los vectores gradiente del mapa de bordes de la imagen original. Dicho campo tiene la ventaja de que amplía mucho el rango de captura; es decir, difunde el gradiente a gran parte de la imagen (a diferencia de otras propuestas como las fuerzas de potencia tradicionales [35]).

Note que en (2.34), cuando $|\nabla f|$ es pequeño, la ecuación es dominada por el primer término, el cual es llamado el término de suavidad, puesto que produce un campo vectorial que varía suavemente; por otro lado, cuando $|\nabla f|$ es grande, el segundo término domina

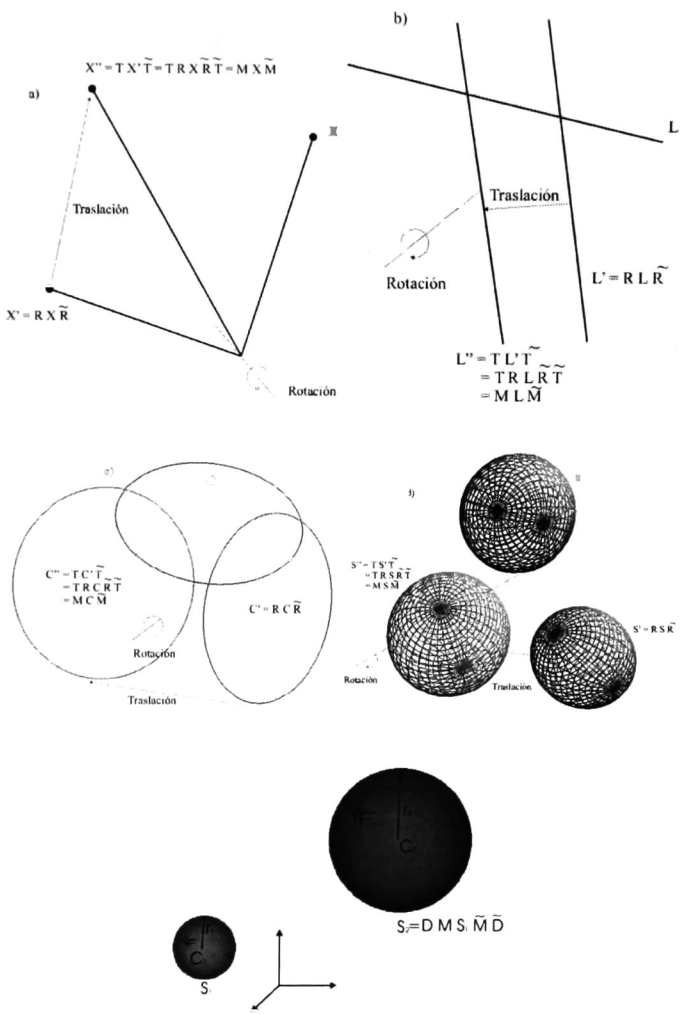


Figura 2.4: Diferentes entidades y el resultado de aplicar los versores R y T (equivalente a aplicar M). a) Punto; b) Línea; c) Círculo; d) Esfera; e) Ejemplo de transformación de una esfera usando además de rotación y traslación, un dilator.

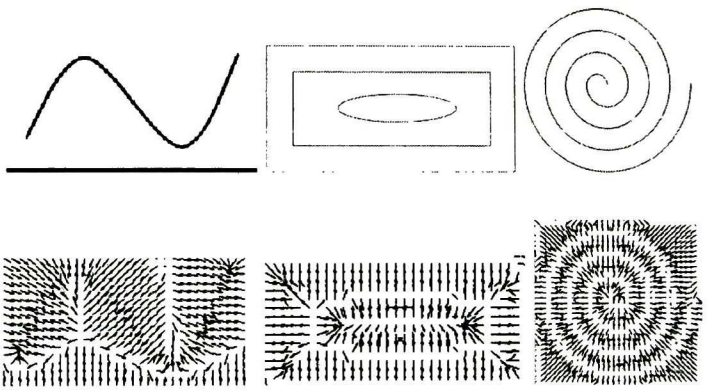


Figura 2.5: Ejemplos del flujo vectorial gradiente generalizado. Fila superior: imágenes originales; fila inferior: GGVF correspondiente

la ecuación, al cual se llama término de datos puesto que anima al campo vectorial $\mathbf{v}$ a tomar valores cercanos a ∇f. Ésto es lo que produce el efecto deseado de mantener $\mathbf{v}$ cercano al gradiente del mapa de bordes cuando éste es grande, pero forzar el campo a variar lentamente en regiones homogéneas.

2.2.1. Ejemplos

La figura 2.5 muestra algunas imágenes con el resultado de calcular el flujo vectorial gradiente generalizado. En la primer fila se encuentra la imagen original, mientras que en la segunda fila el GGVF correspondiente.

2.3. Modelos deformables

Los modelos deformables o *snakes* son curvas elásticas o superficies definidas en el dominio de la imagen, las cuales se pueden mover bajo la influencia de fuerzas internas (que vienen de la curva o superficie en sí misma) y fuerzas externas (calculadas de los datos de la imagen). Matemáticamente estos modelos se representan como curvas o superficies parametrizadas $x(u)$ donde u es el parámetro de la curva. La forma de esta curva se determina típicamente por una formulación variacional que consiste en encontrar la $x(u)$ que minimice la energía:

$$E = E_{int} + E_{ext} = \int_u E_{int}(x(u)) + E_{ext}(x(u))du \tag{2.37}$$

Esto se puede ver como la representación de la energía de la curva o superficie y la forma final de ésta corresponde al mínimo de esta energía. El primer término, E_{int}, es

conocimiento a priori sobre el modelo en sí (elasticidad, rigidez de la curva), mientras que el segundo término, E_{ext}, es una función que se deriva de los datos de la imagen y toma un valor mínimo cuando el modelo deformable yace en la característica que nos interesa (como un borde). Dada una imagen en escala de grises $I(x,y)$, las energías externas que se definen típicamente son

$$E_{ext}(x,y) = -|\nabla I(x,y)|^2 \tag{2.38}$$

$$E_{ext}(x,y) = -|\nabla(G_\sigma(x,y) * I(x,y))|^2 \tag{2.39}$$

donde $G_\sigma(x,y)$ es una función gausiana bidimensional con desviación estándar σ, y ∇ es el operador gradiente.

En la literatura se pueden encontrar básicamente dos tipos de modelos deformables o *snakes*: los modelos paramétricos y los modelos geométricos. Los primeros representan las curvas o superficies en forma paramétrica en el dominio de la imagen, permitiendo que se muevan hacia las características deseadas, como por ejemplo los bordes. Sin embargo, hay dos dificultades con los modelos deformables paramétricos. La primera es que el modelo inicial debe estar, en general, cerca del borde real del objeto de interés; de lo contrario, el modelo puede converger a un modelo erróneo. La segunda dificultad es que estos modelos tienen dificultades para adaptarse en concavidades delgadas.

Para resolver estas dificultades, se presentan en [35] los contornos activos llamados *GGVF-Snakes*, los cuales siguen la formulación de los modelos deformables pero con una nueva fuerza externa estática llamada "Flujo de Vector Gradiente Generalizado"o *Generalized Gradient Vector Flow* (sección 2.2).

2.3.1. *GGVF-Snake*

El contorno deformable *GGVF-Snake* usa el GGVF como fuerza externa y tiene varias ventajas, entre las cuales se hayan su relativa insensibilidad a la inicialización (dado el amplio rango de captura del GGVF), así como su habilidad de moverse hacia concavidades estrechas. Además no necesita conocimiento a priori sobre si el modelo se debe contraer o expandir para converger a los bordes.

La manera en que formula el modelo deformable es sustituyendo la fuerza E_{ext} en (2.37) por el campo vectorial $\mathbf{v}(x,y)$. De esta forma se logra resolver gran parte de los problemas que tienen los contornos deformables típicos, además de que esta formulación resulta menos sensible al ruido en la imagen, tal como lo muestran los resultados en [35].

2.4. Redes neuronales auto-organizadas

El desarrollo de redes organizadas topológicamente fue motivado por un intento de entender cómo las neuronas biológicas se auto-organizan para lograr tareas como el

reconocimiento de patrones en la ausencia de instrucciones que digan el objetivo de cada neurona. La red de este tipo más ampliamente usada es el modelo propuesto por Kohonen [19], llamado *self-organizing map* (SOM), el cual combina un principio de aprendizaje competitivo con una estructura topológica de los nodos de forma que los nodos vecinos tienen pesos similares. La topología es especificada en términos de relaciones de vecindad entre los nodos.

2.4.1. Algoritmo *Self-Organizing Map*

El algoritmo de aprendizaje asegura que el nodo ganador de la competencia, así como sus nodos vecinos, se moverán hacia la muestra (entrada) presentada a la red. La red es auto-organizada en el sentido de que los nodos tienden a tener pesos que capturan las características del espacio de entrada, traduciendo las relaciones de vecindad en proximidad en el espacio Euclidiano, aún si los pesos iniciales de los nodos son arbitrarios.

El algoritmo de aprendizaje SOM, es el siguiente:

Seleccionar la topología de la red; es decir, establecer las relaciones de vecindad entre los nodos para saber cuales serán adyacentes con cuales.

Inicializar los pesos con valores aleatorios pequeños.

Inicializar la distancia de vecindad actual d_0 a un entero positivo.

- Repetir los siguientes pasos hasta alcanzar el criterio de paro

  1. Seleccionar una muestra del espacio de entrada: $\mathbf{i_l}$
  2. Calcular la distancia Euclidiana cuadrada de $\mathbf{i_l}$ al vector de peso $\mathbf{w_j}$ asociado a cada nodo

     $$\sum_{k=1}^{n}(\mathbf{i_{l,k}}(t) - \mathbf{w_{j,k}}(y))^2 \tag{2.40}$$

  3. Seleccionar el nodo j con el mínimo $\sum_{k=1}^{n}(\mathbf{i_{l,k}}(t) - \mathbf{w_{j,k}}(t))^2$
  4. Actualizar los pesos de todos los nodos dentro de la distancia topológica d_t del nodo j usando la regla de actualización

     $$\mathbf{w_j}(t+1) = \mathbf{w_j}(t) + \alpha\eta(t)(\mathbf{i_l}(t) - \mathbf{w_j}(t)) \tag{2.41}$$

     donde $0 < \eta \leq \eta(t-1) \leq 1$
  5. Incrementar t

En las redes SOM, el factor de aprendizaje α decrece con el tiempo t, y la función η normalmente se define como en (2.42); además, el número de nodos se especifica desde el inicio y no hay proceso de inserción o eliminación de nodos.

$$\eta(t) = e^{\frac{D(t)}{2\sigma}} \tag{2.42}$$

Bernd Fritzke [12] propuso un modelo de red llamado *Growing Cell Structures* (GCS) en el cual el número de nodos cambia conforme las muestras se presentan a la red, lo cual resultó en un modelo más flexible capaz de aproximar algunas distribuciones de probabilidad mejor que las redes SOM. A cada j-ésimo nodo se le asoció un *contador de señal* τ_j, el cual se incrementa cada vez que el nodo resulta ganador. Además se tienen dos parámetros de aprendizaje diferentes, uno para el nodo ganador y otro para sus vecinos inmediatos. La insersión de nodos se hace después de cierto número específico de actualizaciones de nodos, y para ello se toma en cuenta el contador de señal: el nodo con el mayor valor en su contador de señal determinará dónde insertar el nuevo nodo. En una aplicación típica del modelo de Fritzke, la red crece y encoge repetidamente; la convergencia depende de la elección de los parámetros y los criterios para insertar, borrar y terminar el algoritmo.

El algoritmo llamado *neural gas* (NG) fue propuesto en [33]. En él, todos los pesos son afectados por cada muestra presentada a la red en una cantidad dependiente de la distancia entre el vector de peso y el de entrada. Para esto, se hace un ordenamiento de los nodos de acuerdo a su proximidad a la entrada presentada. En el algoritmo *neural gas* no se realiza la eliminación de nodos y tampoco se crean nuevos nodos.

2.4.2. Algoritmo *Growing Neural Gas*

En [12], Bernd Fritzke presentó una variante del modelo *neural gas* tomando conceptos de su modelo anterior GCS. Este modelo se llama *Growing Neural Gas* (GNG) y puede ser usado para encontrar estructuras topológicas que reflejen la estructura de la distribución de entrada. Este modelo asume que cada nodo k consta de lo siguiente:

- Un vector de referencia $\mathbf{w_k}$

  Una variable de error acumulado $error_k$

  Un conjunto de aristas que definen la vecindad del nodo (vecinos topológicos)

Además se tienen dos parámetros de adaptación: ϵ_w para el nodo ganador y ϵ_n para sus vecinos topológicos. El algoritmo GNG es el siguiente:

Para una entrada $\mathbf{i_l}$, localizar los dos nodos con los vectores de referencia (posiciones en el espacio) más cercanos; sean estos vectores $\mathbf{w_s}$ y $\mathbf{w_t}$, de forma que $|\mathbf{w_s} - \mathbf{i_l}|$ y $|\mathbf{w_t} - \mathbf{i_l}|$ son la menor y la segunda menor distancia, respectivamente

- La variable de error del nodo ganador s se actualiza añadiéndole la distancia cuadrada de éste a la entrada:

$$error_s = error_s + |\mathbf{w_s} - \mathbf{i_l}|^2$$

- Mover (actualizar) el nodo s y sus vecinos hacia la entrada $\mathbf{i_l}$ de la siguiente forma

$$\mathbf{w_s} = \mathbf{w_s} + \epsilon_w(\mathbf{w_s} - \mathbf{i_l})$$
$$\mathbf{w_n} = \mathbf{w_n} + \epsilon_n(\mathbf{w_n} - \mathbf{i_l})$$

- Incrementar la edad de todas las aristas del nodo s a sus vecinos

  Si los nodos s y t estan conectados por una arista, cargar el valor cero a su edad; en caso contrario, crear una arista entre ellos

- Si hay aristas con una edad mayor a a_{max}, eliminarlas. Si después de esto hay nodos aislados (sin conexiones), eliminarlos

  Después de cada cierto número λ de iteraciones, insertar un nuevo nodo. Esta inserción se realiza de la siguiente forma

  - Encontrar el nodo u con el mayor *error*
  - Entre los vecinos de u, encontrar el nodo v con mayor *error*
  - Insertar un nuevo nodo entre u y v con

$$\mathbf{w_r} = \frac{\mathbf{w_u} + \mathbf{w_v}}{2}$$

  - Crear aristas entre u y r y entre v y r, y eliminar la existente entre u y v
  - Actualizar las variables de error

$$error_u = \alpha error_u$$
$$error_v = \alpha error_v$$
$$error_r = error_u$$

- Decrementar todas las variables de error de todos los nodos j por un factor β

- Repetir los pasos anteriores si no se ha alcanzado el criterio de paro. Dicho criterio puede ser alcanzar el máximo número de nodos.

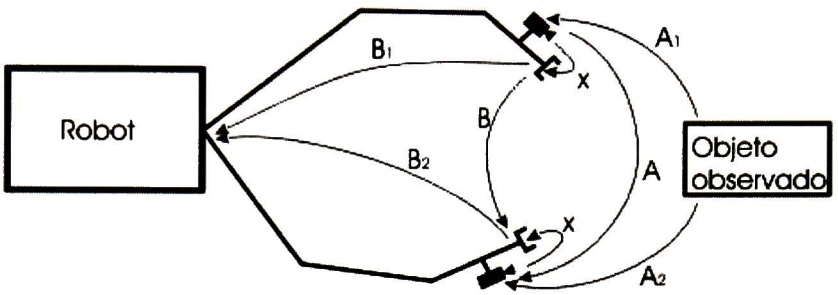


Figura 2.6: Ejemplo del problema de transformación mano-ojo

2.5. El problema de calibración mano-ojo

La calibración mano-ojo es el cálculo de la posición y orientación relativas entre una mano robótica y una cámara montada de forma rígida en ella. Usando dicha cámara, se puede determinar la posición en el sistema de coordenadas de ésta, de un objetivo a agarrar o alcanzar; sin embargo, los comandos de movimientos estan en el sistema de coordenadas de la mano robótica; por tanto conocer la transformación mano-ojo puede ser de gran utilidad en este tipo de tareas.

La manera usual de describir la calibración mano-ojo es por medio de matrices de transformación homogéneas. Denotamos por X la transformación de la cámara a la mano robótica, por A_i la transformación de la cámara al sistema de coordenadas del mundo, y por B_i la matriz de transformación de la base del robot a la mano robótica en el i-ésimo movimiento. La figura 2.6 ilustra el problema.

La transformación A_i se obtiene de las técnicas de calibración extrínseca. La transformación B_i esta dada por la cinemática directa del robot. La formulación del problema mano-ojo es bien conocida [34]

$$AX = XB \tag{2.43}$$

donde $A = A_2 A_1^{-1}$ y $B = B_2^{-1} B_1$. Para resolver este problema se requieren al menos dos movimientos con ejes de rotación no paralelos y para encontrar la solución se han propuesto varios métodos: algunos estiman primero la rotación y después la traslación [34] mientras que otros lo hacen simultáneamente [16]. Daniilidis [10] presenta una solución basada en cuaterniones duales, mientras que [3] propone el uso del álgebra de motores $G_{3,0,1}$ basada en los conceptos de álgebra geométrica.

En el presente trabajo seguimos la formulación del problema en términos de motores dentro del marco del álgebra geométrica. Sin embargo, en lugar de utilizar el álgebra de motores $G_{3,0,1}$, utilizaremos la formulación de los mismos en el álgebra geométrica conformal. De esta forma, el problema de calibración mano-ojo en (2.43) quedaría expresado en la siguiente forma usando motores

$$M_A M_X = M_X M_B \tag{2.44}$$

donde $M_A = A + A'$, $M_B = B + B'$ y $M_X = R + R'$ (ver sección 3).

En [3], se demuestra que el problema se resuelve usando las líneas definidas por los motores

$$\begin{aligned} L_A &= \mathbf{a} + \mathbf{a}' \\ &= M_X L_B \tilde{M}_X \\ &= (R + R')(\mathbf{b} + \mathbf{b}')(\widetilde{R + R'}) \\ &= R\mathbf{b}\tilde{R} + e_\infty(R\mathbf{b}\tilde{R}' + R\mathbf{b}'\tilde{R} + R'\mathbf{b}\tilde{R}) \end{aligned} \tag{2.45}$$

donde $\mathbf{a}$, $\mathbf{a}'$, $\mathbf{b}$, $\mathbf{b}'$ son bivectores (como en (2.31)). Separando la parte real y la parte multiplicada por e_∞, tenemos

$$\mathbf{a} = R\mathbf{b}\tilde{R} \tag{2.46}$$

$$\mathbf{a}' = R\mathbf{b}\tilde{R}' + R\mathbf{b}'\tilde{R} + R'\mathbf{b}\tilde{R} \tag{2.47}$$

Multiplicando por la derecha por R y usando la relación $\tilde{R}R' + \tilde{R}'R = 0$, se obtienen las relaciones siguientes

$$\mathbf{a}R - R\mathbf{b} = 0 \tag{2.48}$$

$$(\mathbf{a}'R - R\mathbf{b}') + (\mathbf{a}R' - R'\mathbf{b}) = 0 \tag{2.49}$$

Lo cual se puede expresar en forma matricial como

$$\begin{bmatrix} \mathbf{a} - \mathbf{b} & [\mathbf{a} + \mathbf{b}]_\times & \mathbf{0}_{3\times1} & \mathbf{0}_{3\times3} \\ \mathbf{a}' - \mathbf{b}' & [\mathbf{a}' + \mathbf{b}']_\times & \mathbf{a} - \mathbf{b} & [\mathbf{a} + \mathbf{b}]_\times \end{bmatrix} \begin{bmatrix} R \\ R' \end{bmatrix} = 0 \tag{2.50}$$

A esta matriz de 6×8 la llamaremos D; el vector de incógnigas $[R, R']^T$ es de 8 dimensiones; la notación $[u]_\times$ representa la matriz *skew*-simétrica formada con el vector u (véase Apéndice B.4). La matriz D esta formada solamente por bivectores (no se incluyen blades de ningún otro grado), por tanto podemos utilizar el método SVD para encontrar $[R\ R']^T$ como el kernel de D.

Considerando que se tienen $n \geq 2$ movimientos, se construye la matriz

$$C = \begin{bmatrix} D_1^T & D_2^T & D_3^T & D_4^T \end{bmatrix}^T \tag{2.51}$$

para aplicar el método SVD y encontrar la solución para $[R, R']^T$. Puesto que el rango de la matriz C es máximo 6, los últimos dos vectores singulares derechos, v_7 y v_8 corresponden a los dos valores singulares cuyo valor es cero o cercano a cero, que expanden el espacio nulo de C. Por tanto, como $[R, R']^T$ es un vector nulo de C, entonces lo podemos expresar como una combinación lineal de v_7 y v_8. Si expresamos estos vectores en términos de dos vectores de 4D $v_7 = (u_1, v_1)^T$ y $v_8 = (u_2, v_2)^T$ esta combinación lineal estaría expresada como

$$\begin{bmatrix} R \\ R' \end{bmatrix} = \alpha \begin{bmatrix} u_1 \\ v_1' \end{bmatrix} + \beta \begin{bmatrix} u_2 \\ v_2' \end{bmatrix} \tag{2.52}$$

Tomando en cuenta las restricciones

$$R\tilde{R} = 1 \quad \text{y} \quad \tilde{R}R' + \tilde{R}'R = 0 \tag{2.53}$$

obtenemos las siguientes ecuaciones cuadráticas en términos de α y β

$$\alpha^2 u_1^T u_1 + 2\alpha\beta u_1^T u_2 + \beta^2 u_2^T u_2 = 1 \tag{2.54}$$

$$\alpha^2 u_1^T v_1 + \alpha\beta(u_1^T v_2 + u_2^T v_1) + \beta^2 u_2^T v_2 = 0 \tag{2.55}$$

Para resolver estas ecuaciones, hacemos un cambio de variable sustituyendo en (2.55) $\mu = \alpha/\beta$ y obtenemos dos soluciones para μ. Regresando a (2.54) e insertando la relación $\alpha = \mu\beta$, se obtiene

$$\beta^2(\mu^2 u_1^T u_1 + \mu(2u_1^T u_2) + u_2^T u_2 = 1 \tag{2.56}$$

la cual nos lleva a dos soluciones para β.

El procedimiento se resume a continuación

1. Dados n movimientos de la mano robótica M_{B_i} y sus correspondientes movimientos en la cámara M_{A_i}, verficar si sus partes escalares son iguales.
2. Para los movimientos que cumplen con el requisito anterior, extraer las direcciones y momentos de las líneas definidas por los motores. Con ellos construir la matriz C como en (2.51).
3. Aplicar SVD a C y tomar los vectores singulares derechos v_7 y v_8 correspondientes a los dos valores singulares iguales a cero (por causa de ruido se aplica un umbral).
4. Calcular los coeficientes para (2.55) y encontrar las dos soluciones de μ.
5. Para ambos valores de μ, calcular el valor de $\mu^2 u_1^T u_1 + 2\mu u_1^T u_2 + u_2^T u_2$ y elegir el que resulte en el mayor valor para luego calcular α y después β.
6. La solución final es $\alpha v_7 + \beta v_8$

Capítulo 3

Segmentación

3.1. Nociones básicas de segmentación

Formalmente y de acuerdo a [25], un conjunto de regiones $R1, R2, ...R_n$ es la segmentación de una imagen I si

$\cup_{i=1}^{n} R_i = I$: cualquier punto en la imagen esta en una región

- $R_i \cap R_k = \emptyset, i \neq k$: las regiones no se intersectan
- R_i es conexa, $i = 1, ..., n$: las regiones son continuas, es decir que las regiones estan compuestas por pixeles contiguos
- R_i es homogénea: cada región es homogénea de acuerdo a cierto criterio (por ejemplo, nivel de gris uniforme)

De acuerdo a [25, 13], las técnicas de segmentación se pueden categorizar en tres clases generales:

1. **Umbralización:** Es el procedimiento para segmentación de imágenes más básico en el cual se aplican umbrales ya sea globalmente (umbralización estática) o localmente (umbralización dinámica). La umbralización estática es útil bajo condiciones controladas, por lo cual es una buena opción solamente para imágenes con histogramas bimodales o trimodales, pero falla en imágenes reales donde el fondo de la imagen no es constante y el contraste de objetos varía. En tales casos, la imagen puede ser subdividida y se le pueden aplicar umbrales locales; sin embargo, esto podría fallar si el histograma de la sub-imagen no es bimodal o trimodal.
2. **Basados en regiones:** Este método trata de aislar áreas de la imagen que son homogeneos de acuerdo a algún conjunto de características. Dos métodos clásicos basados en regiones son el crecimiento de regiones y dividir-y-fusionar.

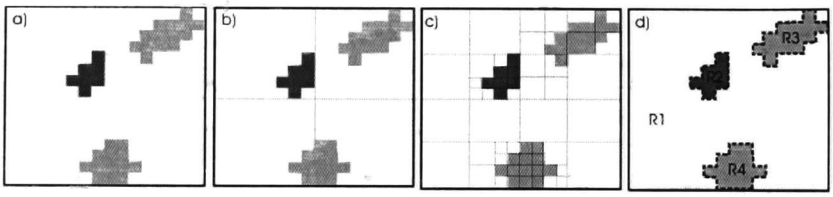


Figura 3.1: Ejemplo ilustrativo de la técnica dividir-y-fusionar: a) imagen de entrada con tres regiones irregulares sobre un fondo constante; b) división inicial; c) resultado de dividir la imagen en bloques homogeneos; d) resultado después de la etapa de fusión o unión.

- *Crecimiento de Regiones:* este algoritmo es uno de los más simples y más populares entre los algoritmos basados en regiones. Se comienza con algunos puntos semilla y luego la región crece añadiendo pixeles vecinos que son similares a las semillas de acuerdo a cierto criterio de homegeneidad (el tamaño de la región se incrementa en cada iteración, de ahí su nombre).
- *Dividir y fusionar:* esta técnica consta de dos pasos: 1. La imagen es dividida recursivamente hasta que todas las regiones cumplen con un criterio de homogeneidad. 2. Todas las regiones adyacentes se van agrupando de forma tal que la región resultante satisfacen el criterio de homogeneidad. El procedimiento típico está basado en la descomposición recursiva de regiones que no cumplen con el criterio de homogeneidad, en cuatro subregiones; luego el paso de fusión va agrupando bloques adyacentes que representan regiones homogeneas que habían sido divididas en la etapa previa. Como ejemplo véase la figura 3.1.

3. **Basadas en bordes:** Estos métodos estan basados en la detección de líneas o aristas entre las regiones. Existen dos métodos básicos: los que usan la primera derivada y los que usan la segunda derivada. En el primer caso una máscara que calcula el gradiente se convoluciona con la imagen y se obtiene un vector gradiente para cada pixel; las aristas son pixeles donde la magnitud del gradiente es un máximo local. En el segundo caso, las aristas son encontradas buscando lugares donde la segunda derivada es cero o pasa por el cero (cambia de signo). Una vez que se obtiene el mapa de aristas o bordes con cualquiera de los dos métodos, una segunta etapa agrupa los elementos del borde para formar lineas o curvas; entonces las regiones se identifican como areas cerradas en la imagen. Desafortunadamente el ruido y otros factores podrían afectar los resultados y algunas regiones podrían aparecer como no cerradas en la imagen, resultando en una mala segmentación.

Debido a las ventajas y desventajas de cada técnica, muchos métodos de segmentación estan basados en la integración de la información obtenida por dos técnicas: información de bordes y de regiones. Algunos de éstos realizan la integración en la detección de regiones, mientras que otros integran la información después de que ambos procesos han terminado.

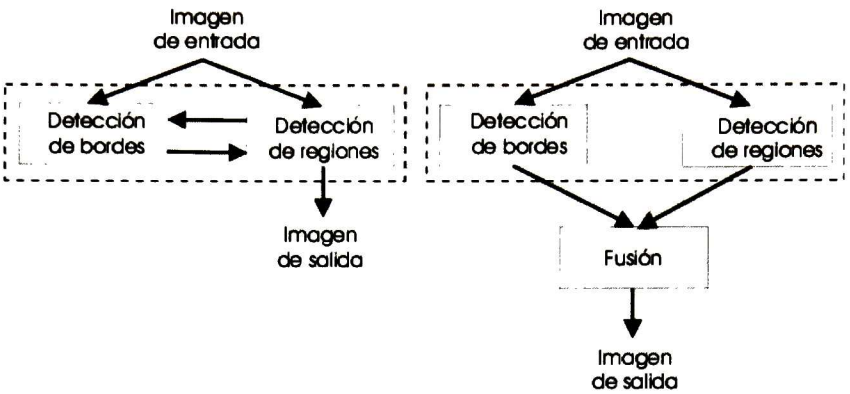


Figura 3.2: Esquemas de las dos estrategias para integrar la información (de acuerdo a [25]): a) Integración embebida;) Integración post-procesamiento

En la figura 3.2 se muestra la forma general de cada una de estas estrategias (la figura 3.2.a muestra la integración embebida y la figura 3.2.b muestra la integración post-procesamiento). La integración embebida se puede describir como la integración a través de la definición de nuevos parámetros o criterios de decisión para la segmentación. La integración post-procesamiento se realiza después de que ambas técnicas (las basadas en regiones y bordes) han sido usadas para procesar la imagen.

Un punto de vista diferente para la integración de regiones e información de bordes es el uso de contornos activos, conocidos también como *snakes* (sección 2.3). Dentro de cada categoría para la integración de la información se tienen una gran variedad de métodos; algunos de ellos funcionan mejor en algunos casos, otros necesitan inicialización de parte del usuario, algunos son más sensibles al ruido, etc. El lector interesado puede consultar [25] para mayores detalles sobre diferentes métodos.

Para determinar el mejor algoritmo de segmentación, debemos tomar en cuenta las propiedades y objetivos que cada algoritmo trata de satisfacer, así como el dominio de la imagen en el que se esta trabajando [25], ya que uno que puede dar muy buenos resultados en un contexto, puede no ser tan bueno en otro.

3.1.1. Segmentación de texturas

La textura es una característica importante para la segmentación de imágenes médicas. Cuando los cirujanos inspeccionan visualmente las imágenes médicas, toman en cuenta la textura de los diferentes tejidos para interpretar lo que estan viendo. Las propiedades de textura de la imagen pueden ser extraídas usando características estadísticas, modelos espacio frecuenciales, etc.

Un operador de textura describe la textura en un área de la imágen. Así, si se usa

un operador de textura sobre toda la imagen, se obtiene una nueva imagen a la que se llama "imagen de características de textura" (o simplemente "imagen de textura"). En tal imagen se describe la textura de un vecindario alrededor de cada pixel. En la mayoría de los casos un simple operador no proporciona información suficiente sobre la textura, por lo cual se usan un conjunto de operadores, lo cual resulta en un conjunto de imágenes de características de textura que juntamente describen la textura alrededor de cada pixel. Los principales métodos para segmentación de texturas, que se pueden consultar en [7], se resumen a continuación:

Filtros de energía de textura de Laws: en el ánimo de producir métodos eficientes computacionalmente, Laws desarrolló un conjunto de máscaras de energía de textura (véanse ejemplos en la figura 3.4) a partir de filtros no recursivos unidimiensionales

$$L5 = [1 \;\; 4 \;\; 6 \;\; 4 \;\; 1]; \quad E5 = [-1 \;\; -2 \;\; 0 \;\; 2 \;\; 1]; \quad S5 = [-1 \;\; 0 \;\; 2 \;\; 0 \;\; -1];$$
$$W5 = [-1 \;\; 2 \;\; 0 \;\; -2 \;\; 1]; \quad R5 = [1 \;\; -4 \;\; 6 \;\; -4 \;\; 1] \tag{3.1}$$

Estas máscaras fueron diseñadas para actuar como filtros para comparar clases específicas de variaciones que se encuentran a menudo en las texturas, tales como aristas, ondas, puntos, etc. Con los resultados de la convolución, se pueden medir un conjunto de estadísticas para obtener información útil.

Matrices de co-ocurrencia: son escencialmente histogramas bidimensionales de la ocurrencia de pares de niveles de gris para un vector de desplazamiento dado (la posición $C_{i,j}$ de la matriz de co-ocurrencia tiene las frecuencias relativas de pares de pixels separados por una distancia d, donde uno de ellos tiene el nivel de gris i y el otro el nivel de gris j). La principal desventaja de esta técnica es su dependencia de los parámetros usados, así como el número de matrices necesarias para obtener buenas características de textura ya que puede ser potencialmente grande.

Métodos en el dominio de la frecuencia: debido a que algunas texturas exhiben una periodicidad y variación, es difícil modelarlas usando técnicas tradicionales estadísticas. Por tanto, se han propuesto características relacionadas con el espectro local. En general, los métodos usan el espectro de potencia como fuente para extraer características de textura, pero el rol del espectro de la fase no ha sido considerado suficientemente.

Características de textura perceptiva: en 1978, Tamura [15] propuso 5 características de textura que corresponden a la percepción humana: rectitud, borrocidad, aspereza, granularidad y discontinuidad. Estas máscaras son expresiones booleanas que se aplican sobre cada pixel. El valor de cada pixel está relacionado a la densidad de la característica de textura perceptiva medida.

3.2. Segmentación por combinación de información

Las técnicas simples de segmentación explicadas anteriormente no pueden ser usadas de forma aislada debido a la complejidad de las imágenes médicas del cerebro, las cuales contienen texturas de diferentes tejidos, valores similares de gris entre tejidos saludables

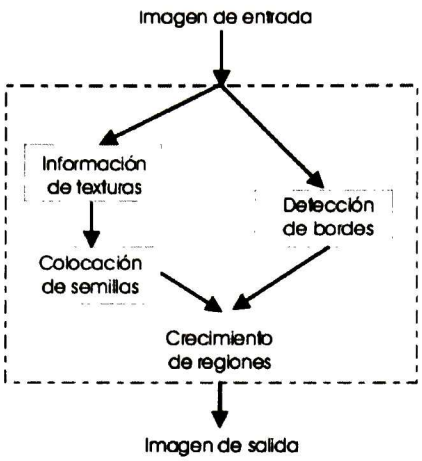


Figura 3.3: Diagrama de bloques del método propuesto.

y dañinos e incluso bordes no definidos claramente. Por esta razón, en este trabajo se decidió combinar la información de los bordes con la información obtenida de métodos de segmentación de texturas, todo esto en un esquema de crecimiento de regiones. Un diagrama de bloques del método propuesto se muestra en la figura 3.3.

Como se explicó en la sección 3.1, en las técnicas de crecimiento de regiones es necesario colocar algunos puntos iniciales, llamados semillas, a partir de los cuales la región crecerá. Para lograr hacer esto automáticamente, utilizamos la información de textura proporcionada por las máscaras de Laws y caracterizamos cada pixel con un vector k-dimensional llamado *vector de textura*, V_{xy} para un pixel en coordenadas (x, y). Luego el usuario selecciona un punto del objeto que le interesa (en nuestro caso nos interesa más la segmentación del tumor); a partir de este punto se seleccionan automáticamente también los pixeles que tienen el mismo vector de textura y utilizamos éstos pixeles como inicialización (semillas) para la estrategia de crecimiento de regiones. La información de bordes se usa para detener el crecimiento de la región.

3.2.1. Colocación de semillas

Para obtener la información de textura, usamos un conjunto de k máscaras (ver ejemplos ilustrativos en la figura 3.4)Cada una de éstas máscaras proporciona un valor para cada pixel. Nuestro objetivo es construir un vector de textura V_{xy} para cada pixel en la imagen, así que tomamos el resultado de cada máscara para cada pixel y fijamos el valor en una posición del mencionado vector a 0 ó 1, dependiendo de si el valor es cero o diferente de cero, respectivamente. Otra posición es solo para identificar a los pixeles pertenecientes al fondo de la imagen (valor fijado a 0) de los que pertenecen a la cabeza del paciente (valor fijado a 1).

a)

-1	-4	-6	-4	-1
-2	-8	-12	-8	-2
0	0	0	0	0
2	8	12	8	2
1	4	6	4	1

b)

1	-4	6	-4	1
-4	16	-24	16	-4
6	-24	36	-24	6
-4	16	-24	16	-4
1	-4	6	-4	1

c)

-1	-4	-6	-4	-1
0	0	0	0	0
2	8	12	8	2
0	0	0	0	0
-1	-4	-6	-4	-1

d)

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

Figura 3.4: Máscaras de energía de texturas usadas para caracterizar cada pixel en la imagen (estas máscaras se convolucionan con la imagen).

Llamemos V al vector de textura e identifiquemos cada uno de sus valores por $V[l], l = 1, ..., k+1$. El procedimiento es como sigue:

1. Para cada pixel p en la imagen, si $p_{ij} \in$ *background*, entonces $V[1] = 0$; en caso contrario, $V[1] = 1$

2. Realizar la convolución de la imagen con cada una de las mascaras de energía de textura para obtener las imágenes de textura $I_{t_i}, i = 1, 2, ..., k$.

3. Para cada posición (x, y) en la imagen y para cada resulado de la convolución con las máscaras, tomar el valor absoluto $val = ||I_{t_i}(x, y)||$. Si $val > 0$ entonces $V[l] = 1$; en caso contrario $V[l] = 0$, donde cada uno de los valores $l = 2, 3, ..., k+1$ corresponde a máscaras diferentes.

Como resultado de esto, encontramos que cada estructura en las imágenes médicas usadas, tiene un vector típico, como se ilustra en la figura 3.5(a) para un caso donde usamos solamente cuatro máscaras. Es importante destacar que no todos los pixeles pertenecientes al mismo objeto tienen el vector deseado debido a variaciones en los valores de los pixeles vecinos; sin embargo, un número suficiente de ellos si lo tiene. Por tanto, una vez que el usuario seleccione un punto (pixel) central del objeto en el que esta interesado, podemos usar los pixeles que tienen vectores iguales para establecerlos como puntos semilla en un esquema de crecimiento de regiones. Así, para extraer el tumor de la imagen, el siguiente paso es fijar las semillas, por lo que las colocamos en los pixeles que tienen el vector de textura típico de un tumor (ver figura 3.5(b)). Una vez establecidas estas semillas podemos comenzar la etapa de crecimiento de región.

3.2.2. Crecimiento de regiones

La técnica de crecimiento de regiones necesita no solamente los puntos semilla, sino también un criterio de crecimiento (para añadir los pixeles) y un criterio de paro. En nuestra propuesta calculamos la media μ_{seed} y la desviación estándar σ_{seed} del nivel de gris

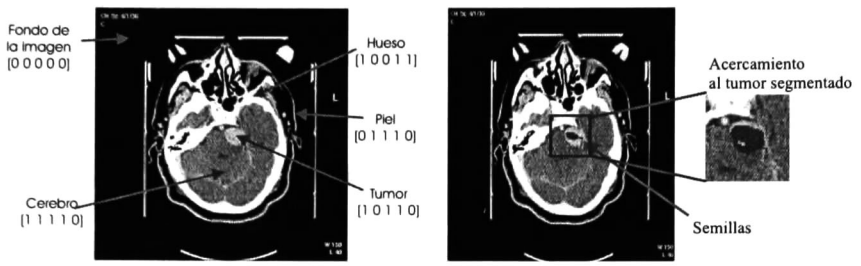


Figura 3.5: **a) Imagen izquierda**: Vectores de textura para las principales estructuras presentes en la imagen de tomografía, tomando solamente 5 máscaras. **b) Imagen derecha**: Puntos semilla fijados con los resultados de la caracterización de los pixeles del tumor.

de los pixeles establecidos como semilla; luego, para cada pixel vecino que es examinado:

$$\text{Si } \sum_{1}^{k+1}(V_{xy} \oplus V_{seed}) = 0 \quad \text{o} \quad \sum_{1}^{k+1}(V_{xy} \oplus V_{seed}) = 1 \tag{3.2}$$
$$\text{y } I(x,y) = \pm 2 * \sigma_{seeds}, \quad \text{entonces} \quad (x,y) \in R_t;$$

donde $V_{xy} \oplus V_{seed}$ es como un operador XOR ($a \oplus b = 1$ si y solo si $a \neq b$), $\sum_1^{k+1}(V_{xy} \oplus V_{seed})$ actúa como un contador del número de valores diferentes en el vector V_{xy} *vs* V_{seed} (debe diferir cuando mucho en un elemento para pertenecer a la región), R_t es la región del tumor. Ejemplo, sea $V_{xy} = [0\ 1\ 0\ 1\ 1]$ y $V_{seed} = [0\ 1\ 1\ 1\ 1]$, entonces $V_{xy} \oplus V_{seed} = [0\ 0\ 1\ 0\ 0]$, por lo que $\sum_1^{k+1}(V_{xy} \oplus V_{seed}) = 1$ y el punto (x,y) se incluye en la región. El criterio de paro toma en cuenta los bordes del objeto porque el crecimiento de la región se da en todas direcciones, pero cuando un pixel correspondiente al borde es encontrado, el crecimiento en esa dirección se detiene. La figura 3.6 muestra un ejemplo del proceso explicado anteriormente. La figura 3.6.a muestra la imagen original; 3.6.b muestra los puntos fijados como semillas (tienen el vector típico del tumor); 3.6.c muestra el resultado final después después de que todo el proceso ha terminado (el tumor segmentado); 3.6.d muestra la imagen original con el tumor resaltado. La figura 3.7 muestra varias imágenes de TC con un tumor así como el resultado con este algoritmo propuesto.

El proceso completo toma solamente unos pocos segundos por imagen y se puede utilizar para segmentar cualquier objeto, solo que en nuestro caso enfocamos la atención en la extracción del tumor.

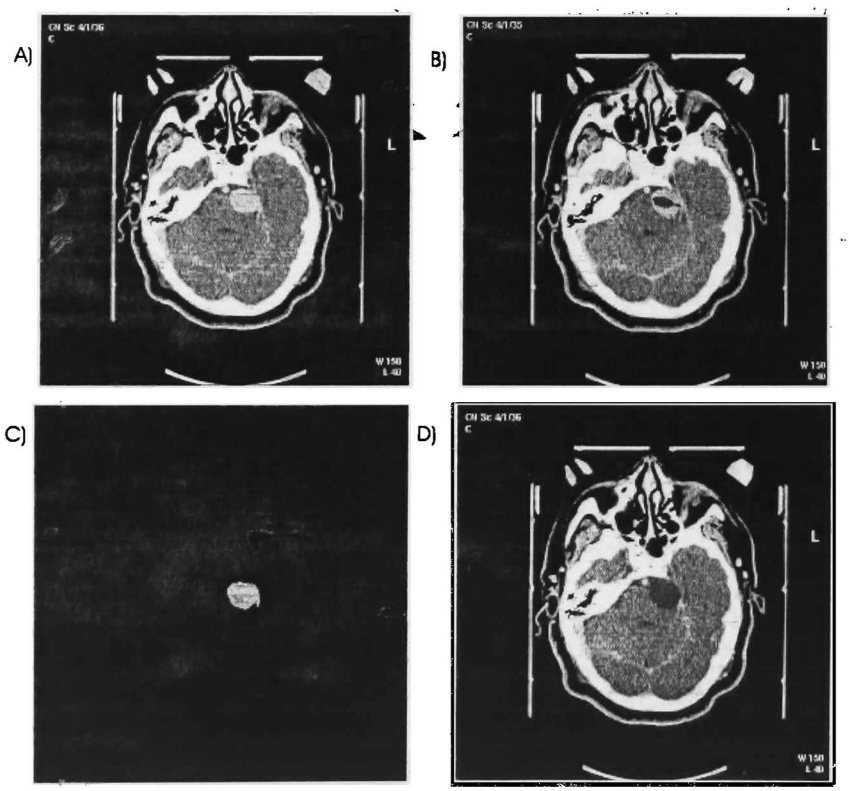


Figura 3.6: Resultados de la segmentación de un tumor en imágenes de tomografía computarizada. A) Una de las imágenes originales; B) Puntos establecidos como semilla; C) Resultado para la imagen (A) después de terminado el proceso; D) Resaltando el tumor en la imagen original.

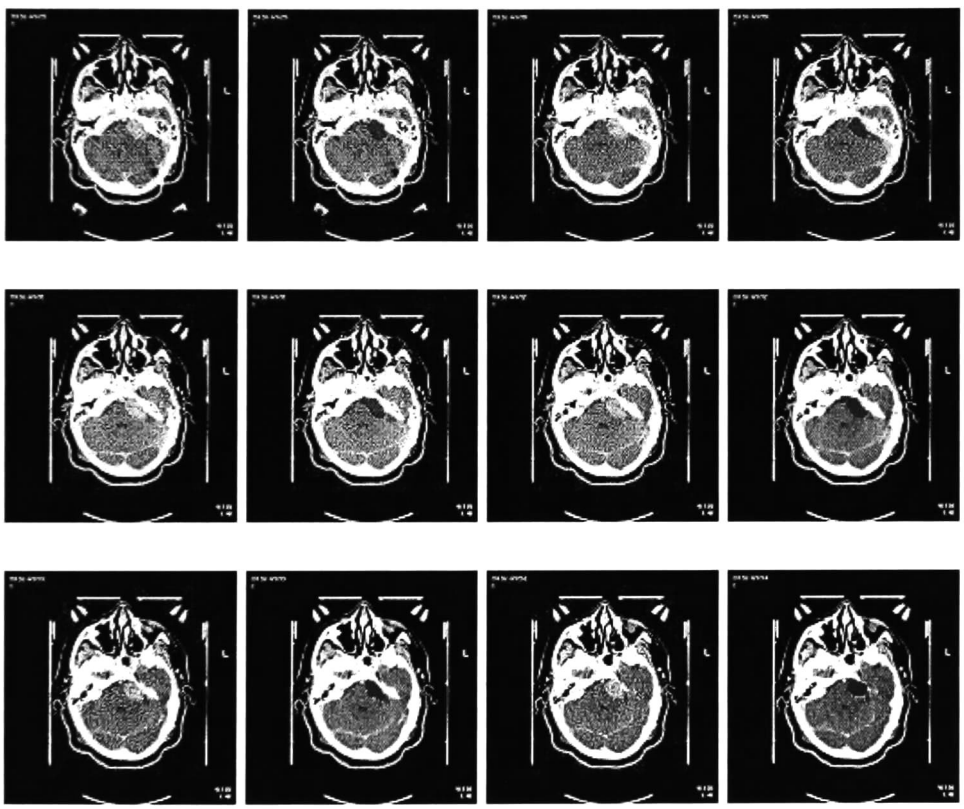


Figura 3.7: Columnas primera y tercera: las imágenes TC orignales; Columnas dos y cuatro: resultados de la segmentación

3.3. Aproximación de contornos usando redes neuronales y GGVF

En esta sección se presenta la segunda propuesta de este trabajo de tesis para segmentación de imágenes, la cual esta basada en los métodos de redes neuronales auto-organizadas (sección 2.4), pero integrando la información del *Flujo Vectorial Gradiente Generalizado* (sección 2.2). El GGVF se utiliza para guiar tanto la selección automática de los patrones de entrada como el proceso de aprendizaje de la red neuronal; al final del aprendizaje se obtienen un conjunto de transformaciones expresadas en el marco del álgebra geométrica, las cuales definen la forma del objeto en el cual estamos interesados. Se decidió usar dicho marco matemático porque las transformaciones (rígidas) de las entidades geométricas se expresan en una forma compacta como operadores llamados *versores*, los cuales son aplicados de forma multiplicativa a las entidades geométricas. Los resultados experimentales demuestran buenos resultados.

3.3.1. Determinación de la forma de un objeto

Para determinar la forma de un objeto podemos recurrir a un mapeo topográfico que utilice los puntos de interés seleccionados sobre el contorno del objeto para ajustarles un mapa con cierta topología. Este mapeo se lleva a cabo comúnmente utilizando las redes neuronales auto-organizadas o Mapas de Kohonen (SOM), o bien la llamada *Neural Gas* (NG) [18]. Sin embargo, si deseamos preservar mejor la topología, no debemos especificar *a priori* el número de neuronas en la red (como se hace para las redes SOM o NG, junto con la relación que guardan las neuronas), sino permitir que la red crezca usando un algoritmo de entrenamiento incremental [1], tal como es el caso de los modelos GCS y GNG (sección 2.4).

En este trabajo se sigue la idea de las redes crecientes y se presenta una propuesta para determinar la forma de un objeto utilizando versores en el álgebra geométrica, lo que resulta en un modelo fácil de manejar en etapas posteriores de procesamiento. Un esquema de esta propuesta se encuentra en la figura 3.8. Esta representación es compacta porque usa solamente un punto fijo como base y un conjunto de versores en el marco del álgebra geométrica (trasladores T en 2D, motores M en 3D), los cuales mueven el punto al contorno del objeto que nos interesa. Por tanto, la red tendrá versores asociados a sus neuronas y en el aprendizaje se determinarán los parámetros de éstos que mejor se ajusten a los patrones de entrada.

Además, se presenta una propuesta sobre cómo seleccionar automáticamente los patrones de entrada, para lo cual se añade una etapa de preprocesamiento en la que se analiza la imagen del objeto y se determinan las entradas haciendo uso del flujo vectorial gradiente generalizado (GGVF) y analizando las líenas de flujo (*streamlines*) seguidas por partículas sobre una malla. Una línea de flujo (streamline) podemos concebirla como el camino seguido por una particula que es colocada en la coordenada (x, y), sometida a

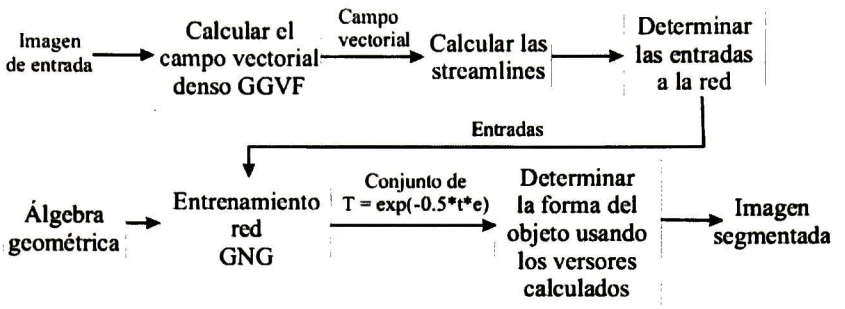


Figura 3.8: Diagrama de bloques de nuestra propuesta usando redes neuronales y GGVF

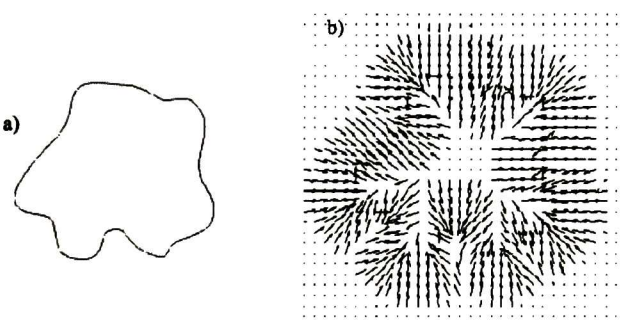


Figura 3.9: Ejemplo del campo vectorial llamado GGVF, obtenido para una imagen de prueba. a) imagen original; b) muestras del campo vectorial.

las fuerzas del campo vectorial; a esta línea de flujo la denotaremos como $S(x, y)$. La información obtenida con el GGVF también se usa en la etapa de aprendizaje como se explica más adelante.

Selección automática de muestras usando GGVF

Puesto que la meta es tener un algoritmo que necesite lo menos posible la intervención de los usuarios, la selección de los patrones de entrada debe ser tan automática como sea posible; por tanto, se necesita un método que pueda proporcionar la información necesaria para guiar al algoritmo en esta selección. Para ejemplificar el proceso se utilizará la imagen de la figura 3.9. Se recomienda haber leído al menos la sección 2.2, donde se explica cómo se obtiene el GGVF.

La selección automática de patrones de entrada se hace al analizar las líneas de flujo seguidas por puntos distribuidos en una malla sobre la imagen. Esto significa que el algoritmo sigue las líneas de flujo definidas por cada punto de la malla, las cuales

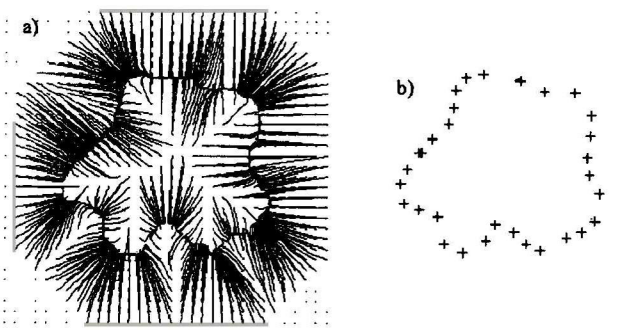


Figura 3.10: a) Ejemplo de las líneas de flujo producidas por partículas colocadas en una malla de 32x32 en el campo vectorial que se muestra en la figura 3.9; b) Puntos seleccionados como patrones de entrada a la red de acuerdo a (3.3).

guian hacia el contorno "más evidente" del objeto; luego, el algoritmo selecciona el punto donde la línea de flujo encuentra un pico en el mapa de bordes. Además de la posición $\mathbf{x} = (x, y)$ del punto, las entradas tendrán el par $\mathbf{v}_\zeta = (u_\zeta, v_\zeta)$, el valor del GGVF en esa posición. Esta información será usada en el entrenamiento junto con la posición (x, y) para determinar la topología de los datos. Resumiendo, el conjunto de entrada $\mathbf{I}$ será

$$\mathbf{I} = \{\zeta = (X_\zeta, \mathbf{v}_\zeta) | \mathbf{x} \in S(x', y') \ y \ f(x, y) = 1\} \tag{3.3}$$

donde X es la representación del punto (x, y) en $G_{4,1}$, $\mathbf{x} \in S(x', y')$ significa que $\mathbf{x} = (x, y)$ esta en el camino seguido por una partícula colocada en (x', y') (línea de flujo), y $f(x, y)$ es el valor del mapa de bordes en la posición $\mathbf{x}$ (suponiendo que es binario). Como algunas líneas de flujo pueden llevar al mismo punto o a puntos muy cercanos, podemos añadir restricciones para evitar muestras muy cercanas; una muy simple pero efectiva es que el candidato a ser incluido en el conjunto de entrada deba estar al menos a una distancia fija d_{thresh} de cualquier otra entrada.

Al realizar la selección de patrones de entrada de esta forma, se evita la necesidad de preprocesar la imagen umbralizando y resaltando contornos para realizar la selección manualmente como en [1]; ésto es porque el GGVF es muy robusto aún en la presencia de ruido y está garantizado que las líneas de flujo guiarán hacia los contornos de la imagen. La figura 3.10 muestra las líneas de flujo generadas de acuerdo al campo vectorial de la figura 3.9 y los patrones de entrada seleccionados según se ha descrito.

Aprendizaje y versores

Como se mencionó anteriormente, se usará la información proporcionada por el GGVF para guiar el aprendizaje. Los algoritmos de aprendizaje son dos; el primero se definió con base en el algoritmo SOM (sección 2.4.1) pero adaptándolo para que trabajara

en un esquema de crecimiento (insertando neuronas según se requiera); el segundo se basa en GCS y GNG. Ambos algoritmos utilizan la información del GGVF para encontrar versores en el álgebra geométrica conformal G_{41}. La red comienza con un número mínimo de unidades neuronales, y se van insertando nuevas unidades sucesivamente. La red queda especificada por

Un conjunto de unidades neuronales llamado N donde cada $\mathbf{n}_l \in N$ está especificada por

- Un versor M del álgebra geométrica conformal
- Un vector $\mathbf{v_l} = [u_l \;\; v_l]^T$ que contiene información del GGVF

Cada versor es la transformación que debe ser aplicada a un punto para colocarlo en el contorno del objeto. El conjunto de transformaciones y las relaciones entre las unidades neuronales describirá finalmente la forma del objeto.

- Un conjunto de conexiones entre las neuronas para definir su estructura topológica.

El primer algoritmo, que llamamos $GSOM$, es el siguiente:

1. Sea P_0 un punto fijo inicial sobre el cual las transformaciones se aplicarán. Tales transformaciones serán expresadas como $M = e^{-\frac{1}{2}e_\infty t}$ en el álgebra geométrica (note que en 2D usaremos motores donde $R = 1$, de forma que son en escencia trasladores). Este punto puede ser un punto aleatorio o bien el centroide definido por las entradas (Nota: $p_0 \in G_2, P_0$ es su representación en $G_{4,1}$).

2. Iniciar con dos neuronas (el número mínimo de las mismas), llamadas a y b que tendrán asociados motores aleatorios M_a and M_b:

$$M_a = e^{-\frac{1}{2}e_\infty t_a} \quad ; \quad M_b = e^{-\frac{1}{2}e_\infty t_b} \tag{3.4}$$

así como el vector $\mathbf{v_l} = [u_l \;\; v_l], l = l_a, l_b$

3. Seleccionar una entrada ζ del conjunto de entradas $\mathbf{I}$ y encontrar la neurona ganadora; esto significa encontrar la neurona $\mathbf{n_l}$ que tenga la transformación $M_l = e^{-\frac{1}{2}e_\infty t_l}$ que mueva el punto P_0 más cerca a esa entrada:

$$M_{win} = \min_{\forall M} \|(X_\zeta - (MP_0\tilde{M}))^2\| \tag{3.5}$$

4. Modificar M_{win} y todas las transformaciones M_l de las neuronas vecinas de tal forma que el M modificado representará una transformación que mueve al punto P_0 más cerca de la entrada:

$$M_{l_{new}} = e^{-\frac{1}{2}e_\infty t_l} e^{-\frac{1}{2}e_\infty \Delta t_l} \tag{3.6}$$

donde

$$\Delta t_l = \alpha\ \phi\ \eta(\mathbf{v}_\zeta, \mathbf{v}_l)\ (\langle X_\zeta - P_0 \rangle_{1_\varepsilon}) \tag{3.7}$$

donde $\langle H \rangle_{1_\varepsilon}$ deja solo los elementos de H de grado 1 que pertenecen a G_3 (ie. $H \in \langle G_3 \rangle_1$), α es un parámetro de aprendizaje, ϕ es una función que define cuánto puede aprender una neurona de acuerdo a su distancia a la neurona ganadora (normalmente definido como en (3.8)), y $\eta(\mathbf{v}_\zeta, \mathbf{v}_l)$ se define como en (3.9).

$$\phi = e^{-\frac{X_\zeta - (M P_0 \tilde{M})^2}{2\sigma}} \tag{3.8}$$

$$\eta(\mathbf{v}_\zeta, \mathbf{v}_l) = \|\mathbf{v}_\zeta - \mathbf{v}_l\|^2 \tag{3.9}$$

$\eta(\mathbf{v}_\zeta, \mathbf{v}_l)$ es una función que define una cantidad de aprendizaje dependiendo de la fuerza para enseñar de la entrada ζ y la capacidad de aprender de la neurona dadas en $\mathbf{v}_\zeta$ y $\mathbf{v}_l$, respectivamente. Finalmente, también actualizamos $\mathbf{v}_l$:

$$\mathbf{v}_l = [(u_l + \alpha\ \phi\ u_l) \quad (v_l + \alpha\ \phi\ v_l)] \tag{3.10}$$

5. Insertar nuevas neuronas de la siguiente forma:

   - Determinar las neuronas vecinas $\mathbf{n_i}$ y $\mathbf{n_j}$ y que están conectadas por una arista con longitud mayor a c_{max}
   - Crear una nueva neurona $\mathbf{n_{new}}$ entre $\mathbf{n_i}$ y $\mathbf{n_j}$ cuyo motor asociado y el vector $\mathbf{v_l}$ serán

   $$M_{n_{new}} = \frac{M_i + M_j}{2} \tag{3.11}$$

   $$\mathbf{v_{l\_new}} = \frac{\mathbf{v_i} + \mathbf{v_j}}{2} \tag{3.12}$$

   - Borrar la conexión anterior entre $\mathbf{n_i}$ y $\mathbf{n_j}$ y crear dos nuevas conexiones entre $\mathbf{n_{new}}$ con $\mathbf{n_i}$ y $\mathbf{n_j}$

6. Repetir los pasos 3 a 5 si el criterio de paro no se ha satisfecho. El criterio de paro será alcanzar el máximo número de neuronas o cuando la capacidad de aprendizaje de las neuronas $|\mathbf{v_l}|$ se aproxime a cero (menor a cierto umbral c_{min}), lo primero que suceda detendrá el proceso de aprendizaje.

Para el segundo algoritmo se tienen dos parámetros de aprendizaje: e_w y e_n, para la neurona ganadora y para sus vecinos directos, respectivamente. Estos parámetros no se decrementan en el tiempo como el parámetro α en el algoritmo anterior (GSOM), sino que se mantienen constantes durante todo el proceso. Además, cada neurona $\mathbf{n_l}$ se compone

de $M_{\mathbf{n_l}}$ y dos atributos: sc_l y rsf_l. El primero, sc_l, se incrementa cada que la neurona $\mathbf{n_l}$ es la neurona ganadora, mientras que rsf_l se define como

$$rsf_l = \frac{sc_l}{\sum_{\forall \mathbf{n_j}} sc_j} \tag{3.13}$$

Este parámetro actúa como un indicador de dónde insertar nuevas unidades neuronales. El algoritmo es el siguiente:

1. Sea P_0 un punto fijo inicial sobre el cual las transformaciones $M = e^{-\frac{1}{2}e_\infty t}$ se aplicarán. Este punto puede ser un punto aleatorio o bien el centroide definido por las entradas (Nota: $p_0 \in G_2, P_0 \in G_{4,1}$).

2. Iniciar con el mínimo número de neuronas y asociarles motores aleatorios; supongamos que iniciamos con 2 neuronas a y b, con M_a y M_b:

   $$M_a = e^{-\frac{1}{2}e_\infty t_a} \quad ; \quad M_b = e^{-\frac{1}{2}e_\infty t_b} \tag{3.14}$$

   así como el vector $\mathbf{v_l} = [u_l \;\; v_l], l = l_a, l_b$

3. Seleccionar una muestra ζ del conjunto de entradas $\mathbf{I}$ y encontrar la neurona ganadora; es decir, la neurona n_l que tenga el motor $M_l = e^{-\frac{1}{2}e_\infty t_l}$ que mueva el punto P_0 más cerca a esa entrada:

   $$M_{win} = \min_{\forall M} \|(X_\zeta - (MP_0\tilde{M}))^2\| \tag{3.15}$$

   A esta neurona incrementarle su sc_{win} y recalcular rsf_{win}

4. Modificar M_{win} y todas las transformaciones M_l de las neuronas vecinas de tal forma que el M modificado representará una transformación que mueve al punto P_0 más cerca de la entrada:

   $$M_{l_{new}} = e^{-\frac{1}{2}e_\infty t_l} e^{-\frac{1}{2}e_\infty \Delta t_l} \tag{3.16}$$

   donde

   $$\Delta t_{win} = e_w \, \eta(\mathbf{v}_\zeta, \mathbf{v_{win}}) \, (\langle X_\zeta - P_0' \rangle_{1\mathcal{E}}) \tag{3.17}$$

   para la neurona ganadora, mientras que para sus vecinas directas usamos

   $$\Delta t_n = e_n \, \eta(\mathbf{v}_\zeta, \mathbf{v_n}) \, (\langle X_\zeta - P_0' \rangle_{1\mathcal{E}}) \tag{3.18}$$

   donde $\langle H \rangle_{1_\mathcal{E}}$ deja solo los elementos de H de grado 1 que pertenecen a G_3 ($H \in \langle G_3 \rangle_1$) y P_0' es la posición actual $P_0' = MP_o\tilde{M}$. ϕ se define como en (3.19), y $\eta(\mathbf{v}_\zeta, \mathbf{v_l})$ se define como en (3.20).

   $$\phi = e^{-\frac{X_\zeta - (MP_0\tilde{M})^2}{2\sigma}} \tag{3.19}$$

$$\eta(\mathbf{v}_\varsigma, \mathbf{v}_\mathbf{l}) = \|\mathbf{v}_\varsigma - \mathbf{v}_\mathbf{l}\|^2 \tag{3.20}$$

$\eta(\mathbf{v}_\varsigma, \mathbf{v}_\mathbf{l})$ es una función que influye en la cantidad de aprendizaje y con la cual estamos tomando en cuenta la información brindada por el GGVF. Finalmente, también actualizamos $\mathbf{v}_\mathbf{l}$:

$$\mathbf{v}_{\mathbf{win}}^{\mathbf{new}} = [u_{win}^{new} \; v_{win}^{new}]^T \tag{3.21}$$

$$\mathbf{v}_{\mathbf{n}}^{\mathbf{new}} = [u_{n}^{new} \; v_{n}^{new}]^T \tag{3.22}$$

donde

$$u_{win}^{new} = (u_{win} + e_w \; u_{win}) \qquad v_{win}^{new} = (v_{win} + e_w \; v_{win}) \tag{3.23}$$

$$u_{n}^{new} = (u_{n} + e_n \; u_{n}) \qquad v_{n}^{new} = (v_{n} + e_n \; v_{n}) \tag{3.24}$$

5. Cada cierto número λ de iteraciones:

   - Determinar la neurona $\mathbf{n_i}$ con el valor más grande en rsf_l. Si alguna de sus vecinas directas, digamos n_j, esta conectada por una arista con longitud mayor a c_{max}, entonces crear una nueva neurona $\mathbf{n_{new}}$ entre $\mathbf{n_i}$ y $\mathbf{n_j}$ cuyo motor asociado y el vector $\mathbf{v_l}$ serán

   $$M_{n_{new}} = \frac{M_i + M_j}{2} \tag{3.25}$$

   $$\mathbf{v_{l\_new}} = \frac{\mathbf{v_i} + \mathbf{v_j}}{2} \tag{3.26}$$

   Esta nueva neurona tendrá $sc_{new} = 0$ and $rsf_{new} = 0$.

   - Borrar la conexión anterior entre $\mathbf{n_i}$ y $\mathbf{n_j}$ y crear dos nuevas conexiones entre $\mathbf{n_{new}}$ con $\mathbf{n_i}$ y $\mathbf{n_j}$

6. Repetir los pasos 3 a 5 si el criterio de paro no se ha satisfecho. El criterio de paro será alcanzar el máximo número de neuronas o cuando la capacidad de aprendizaje de las neuronas $|\mathbf{v_l}|$ se aproxime a cero (menor a cierto umbral c_{min}), lo primero que suceda detendrá el proceso de aprendizaje.

Entrenando la red encontramos el conjunto de motores M (trasladores en este caso, pero veremos más adelante el uso de motores $M = TR$) que definen posiciones en una trayectoria; tales posiciones minimizan el error

$$\chi = \sum_{\forall \varsigma} (((M_\varsigma P_0 \tilde{M}_\varsigma) - X_\varsigma)^2) \tag{3.27}$$

donde M_ς es el motor que mueve P_0 más cerca de la entrada ς.

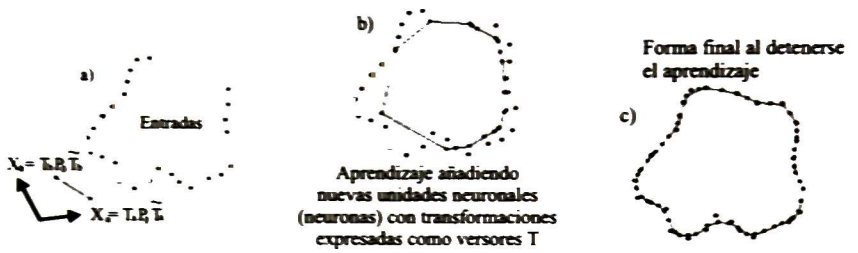


Figura 3.11: a) Entradas a la red y la inicialización de acuerdo a los dos trasladores M_a, M_b; b) Forma después de solo una iteración; c) Forma final definida con los 50 trasladores estimados.

3.3.2. Experimentos

Dado que los dos algoritmos que proponemos comienzan de la misma forma (cómputo del GGVF, líneas de flujo, determinación de entradas) y lo único que los diferencia es el algoritmo de aprendizaje, ilustramos el proceso general con una imagen artificial, la cual se muestra en la figura 3.9.a. Las entradas a la red fueron seleccionadas de acuerdo al proceso descrito en la sección 3.3.1; es decir, calculando el campo vectorial GGVF (ver figura 3.9.b) y las líneas de flujo (figura 3.10.a), resultando en el conjunto de entrada mostrado en la figura 3.10.b. La figura 3.11 muestra diferentes etapas en el desarrollo del algoritmo de aprendizaje. el cual en este caso corresponde al primero de los descritos (GSOM$_1$: la figura 3.11.a muestra las dos posiciones iniciales resultantes al aplicar M_a y M_b (los trasladores asociados con las dos neuronas iniciales) a P_0; la figura 3.11.b muestra el resultado después de algunas iteraciones del algoritmo (17 trasladores diferentes. correspondientes a 17 unidades neuronales en la red); la figura 3.11.c muestra la forma final cuando la red se ha detenido (50 trasladores diferentes, correspondientes a 50 unidades neuronales en la red).

Los dos algoritmos de aprendizaje descritos fueron aplicados a conjuntos de imágenes médicas tanto incluyendo información del GGVF como sin dicha información. Los resultados muestran que ambos algoritmos se ven beneficiados con el uso de la información del GGVF; sin embargo. el segundo algoritmo hace una mejor aproximación a los contornos de los objetos. La figura 3.12.a muestra los errores promedio para algunas imágenes cuando el algoritmo GSOM alcanza el criterio de paro. Note que usando la información del GGVF se obtiene un error menor. lo cual confirma la afirmación de que esta información ayudaría a guiar el aprendizaje para obtener una mejor aproximación a la forma del objeto. La figura 3.12.b muestra los errores promedio usando el segundo algoritmo. con y sin la información del GGVF. Nuevamente los resultados son mejores incorporando dicha información. Sin embargo, si comparamos los resultados con ambos algoritmos, podemos

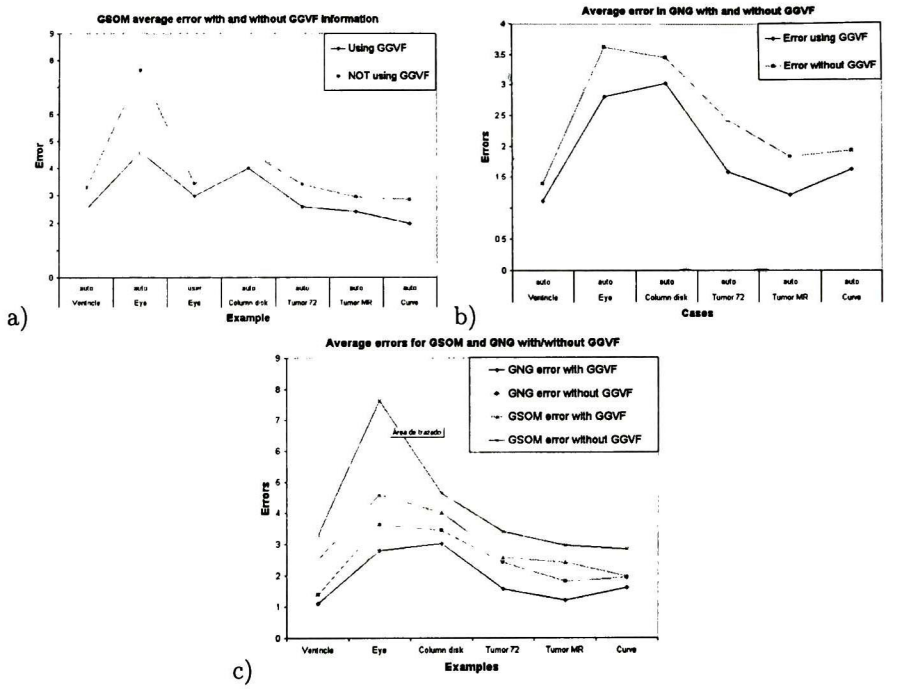


Figura 3.12: a) Errores promedio para diferentes casos usando el algoritmo GSOM con información del GGVF y sin ella; b) Errores promedio obtenidos usando el algoritmo basado en GCS y GNG con información del GGVF y sin ella; c) Comparación entre los errores obtenidos con ambos algoritmos usando y sin usar la información del GGVF.

observar en la figura 3.12.c que el segundo algoritmo obtiene valores de error menores.

Después de hacer algunas pruebas en imágenes, se observó en ambos algoritmos que la incorporación de la información gradiente contribuía a aproximar mejor la forma de los objetos. Observe que las redes tradicionales GCS y GNG normalmente se adaptan mejor a la topología del espacio de entrada que la red SOM; en nuestro caso la incorporación del GGVF hace que la red basada en SOM se aproxime al desempeño de la red basada en GCS y GNG, sin incorporar en esta última el GGVF. Sin embargo, la incorporación del GGVF en la segunda red aproxima mejor la forma de los objetos. Por tanto, se hicieron más pruebas con el segundo método (red basada en GCS y GNG) para comparar numérica y visualmente los resultados que se obtenían incorporando y sin incorporar la información del GGVF.

Una muestra de uno de los conjuntos de imágenes de prueba se muestra en la figura 3.13; la figura 3.14 muestra los campos vectoriales obtenidos para las regiones de interés (ROI) de la figura 3.13, mientras que en 3.15 se muestran las líneas de flujo de las imágenes en la figura 3.14. La figura 3.16 muestra los resultados obtenidos sin la incorporación del

Tabla 3.1: Errores calculados con (3.27) para las imágenes de las figuras 3.16 y 3.17. ϵ_{Ng}: error sin la información del GGVF (fig. 3.16); ϵ_{Sg}: error usando GGVF (fig. 3.17). La tercera y última columnas indican si la topología es correcta o incorrecta; el primer SI/NO en cada paréntesis corresponde al caso sin información GGVF, mientras que el segundo SI/NO es para el caso con información de GGVF.

Img. No.	ϵ_{Ng}	ϵ_{Sg}	Correcto	Img. No.	ϵ_{Ng}	ϵ_{Sg}	Correcto
1	1.72	1.22	(NO,SI)	17	1.8	0.21	(NO,SI)
2	1.135	0.28	(NO,SI)	18	0.75	0.24	(NO,SI)
3	1.63	1.13	(SI,SI)	19	1.01	0.23	(SI,SI)
4	0.27	2.17	(NO,SI)	20	0.23	0.22	(NO,SI)
5	1.24	0.87	(SI,SI)	21	2.54	0.26	(NO,SI)
6	1.34	0.3	(NO,SI)	22	0.23	0.203	(SI,SI)
7	0.54	1.06	(SI,NO)	23	0.23	0.24	(NO,SI)
8	3.08	1.82	(NO,SI)	24	1.91	0.23	(SI,SI)
9	0.85	0.27	(NO,SI)	25	0.26	0.93	(NO,SI)
10	0.84	0.61	(SI,SI)	26	0.98	0.99	(SI,SI)
11	0.93	0.91	(NO,SI)	27	2.004	1.04	(NO,SI)
12	1.08	0.9	(NO,SI)	28	1.04	1.76	(SI,SI)
13	2.37	0.24	(SI,SI)	29	3.03	1.31	(SI,SI)
14	1.04	0.29	(SI,SI)	30	1.38	0.26	(SI,SI)
15	3.42	0.96	(NO,SI)	31	0.27	0.23	(NO,SI)
16	2.54	1.04	(NO,SI)	32	1.17	0.26	(SI,SI)

GGVF, mientras que 3.17 muestra los resultados incorporando dicha información. La tabla 3.1 muestra los errores obtenidos: ϵ_{Ng} es el error sin la información del GGVF y ϵ_{Sg} es el error usando dicha información. La columna *"Img No."* de la tabla es el número de la imagen, contando de izquierda a derecha y de arriba hacia abajo las imágenes de la figura 3.13; la tabla contiene el error para 32 imágenes, aunque en la figura solo se muestran algunas de ellas.

Si se observa la tabla 3.1, se notarán casos en los que ϵ_{Ng} es menor a ϵ_{Sg}, lo cual significa que la red ha creado y posicionado sus neuronas cerca de las muestras de entrada; sin embargo, si se observa la imagen del resultado correspondiente, se notará que el uso de GGVF ayuda a ajustar mejor la red a la topología de los datos de entrada, comparado con el caso que no usa esa información. La columna *"Correcto"* de la tabla 3.1 indica si se obtuvo la forma correcta del objeto o no (si se alcanzó la topología esperada del objeto). Lo mismo se observó en diferentes conjuntos de imágenes; la figura 3.18 muestra el porcentaje de imagenes en diferentes conjuntos de prueba, en las cuales el algoritmo obtuvo una topología correcta al alcanzar el criterio de paro, tanto usando la información del gradiente como sin usarla.

Las tablas 3.2 y 3.3 muestran algunos resultados para otros casos. La tabla 3.2 muestra los errores obtenidos en otras imágenes de prueba, comparando cuando se usa la

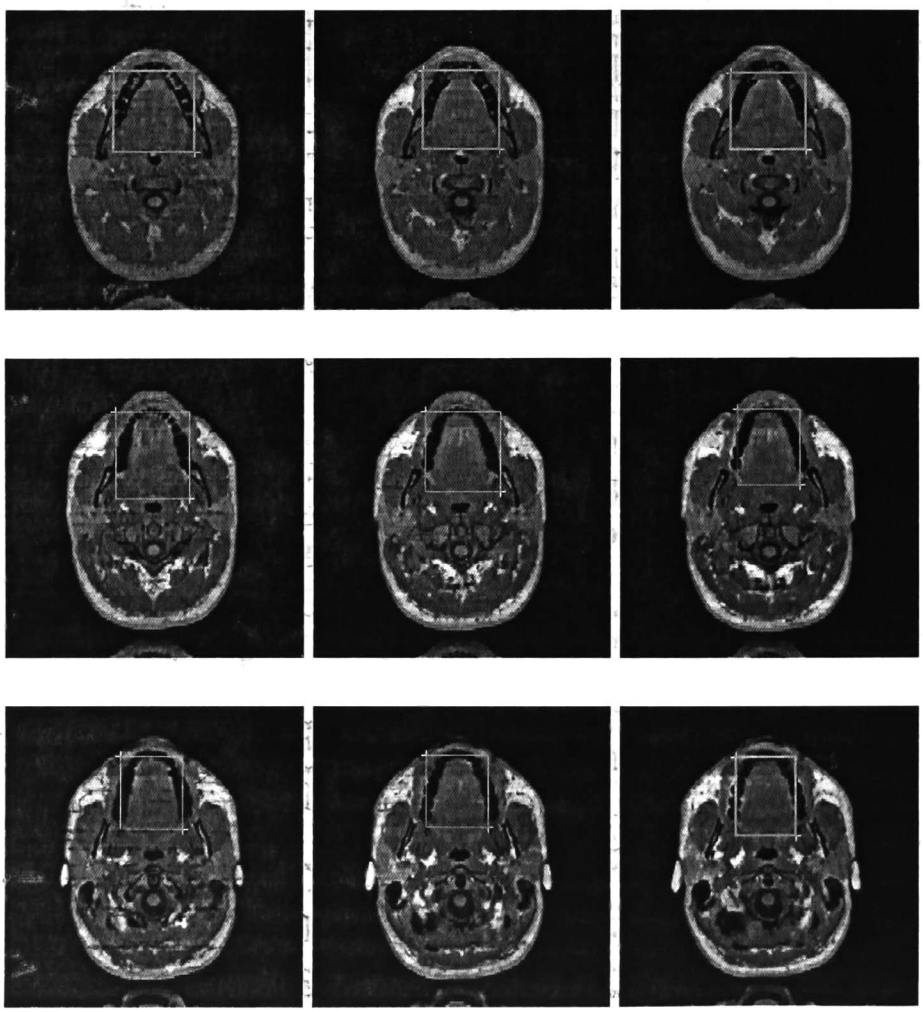


Figura 3.13: Imágenes originales y la región de interés de uno de los conjuntos de imágenes de prueba.

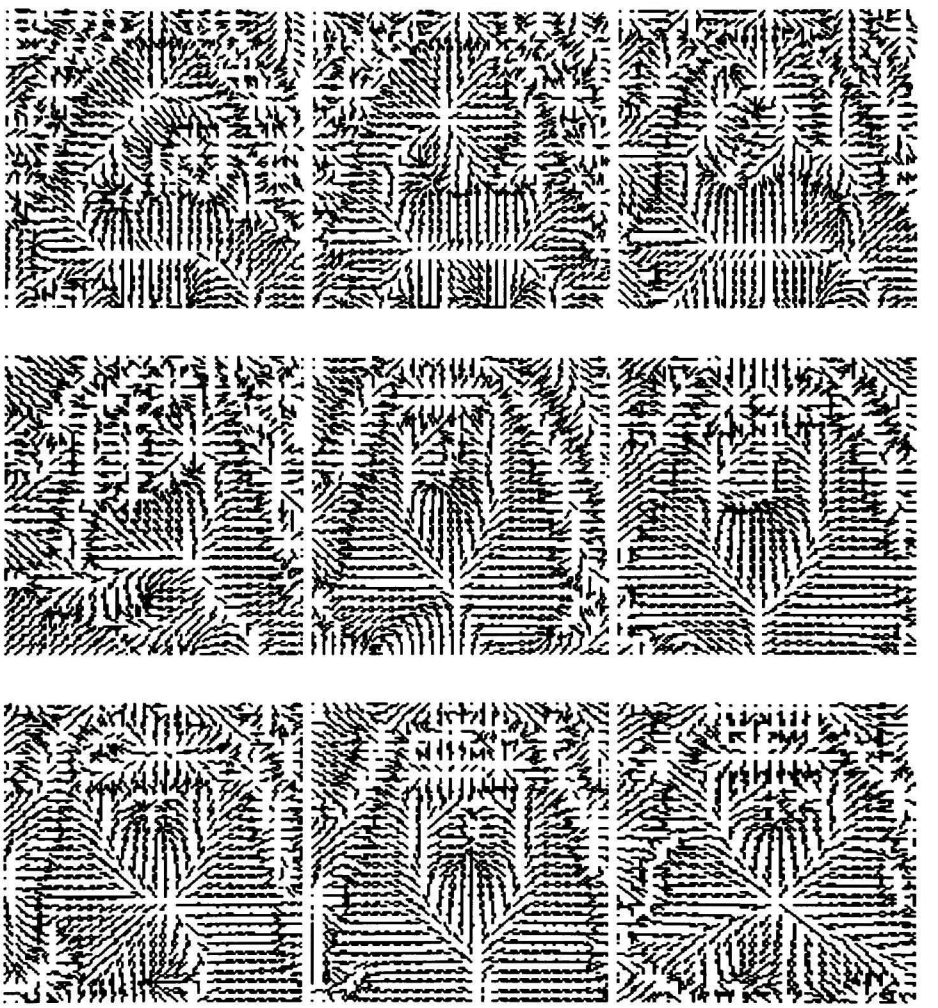


Figura 3.14: GGVF de la region de interés de cada imagen de la figura 3.13.

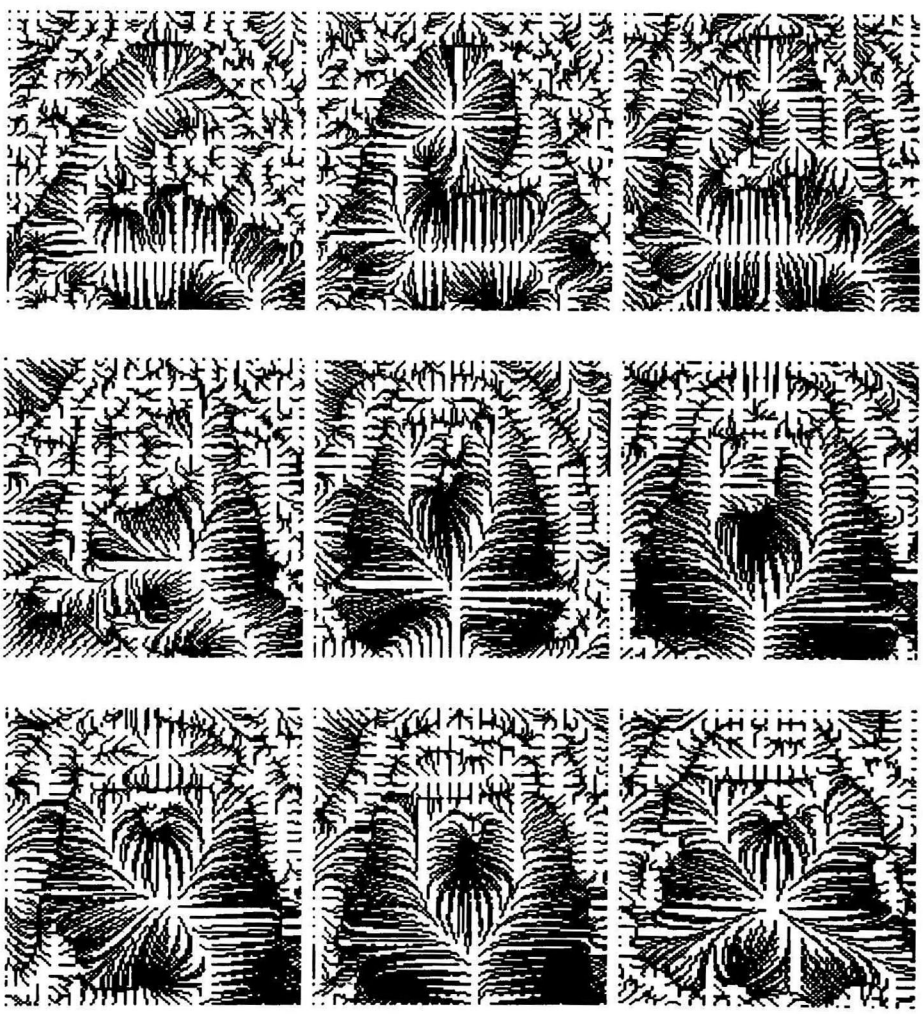


Figura 3.15: Líneas de flujo definidas por los campos vectoriales de la figura 3.14.

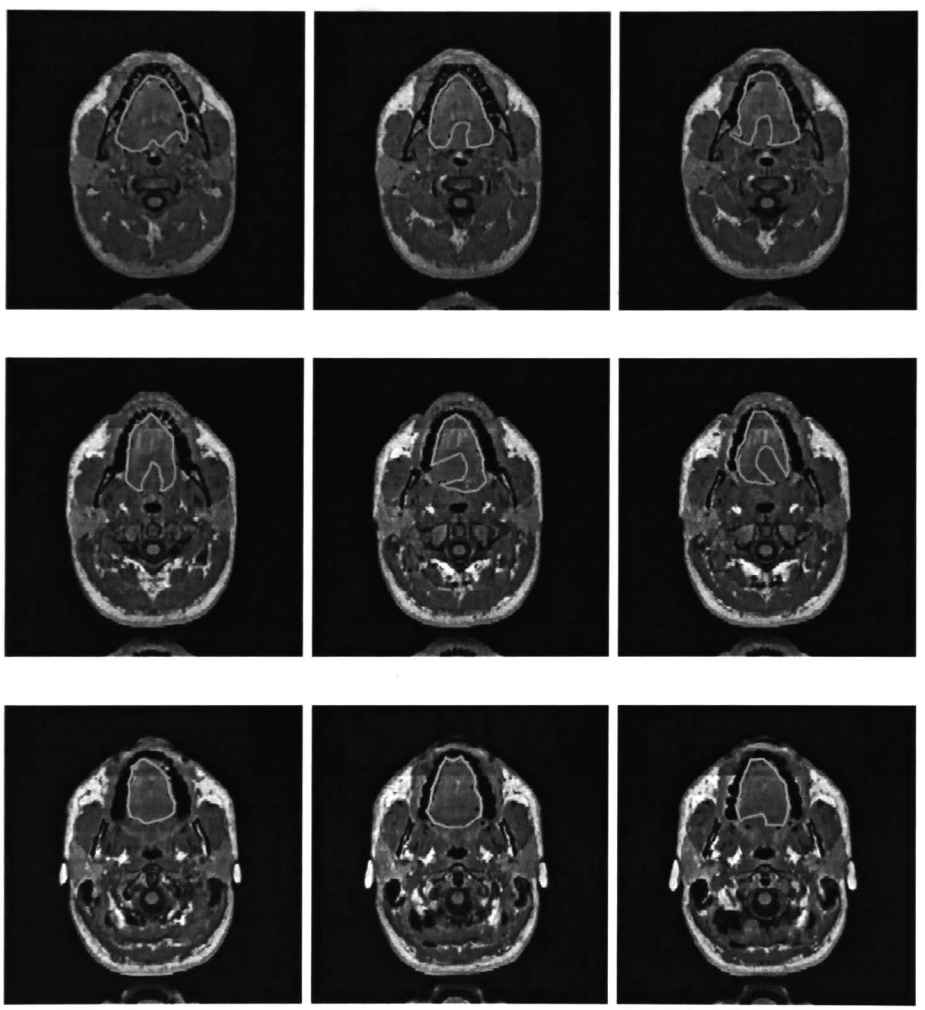


Figura 3.16: Resultados sin incorporar la información del GGVF.

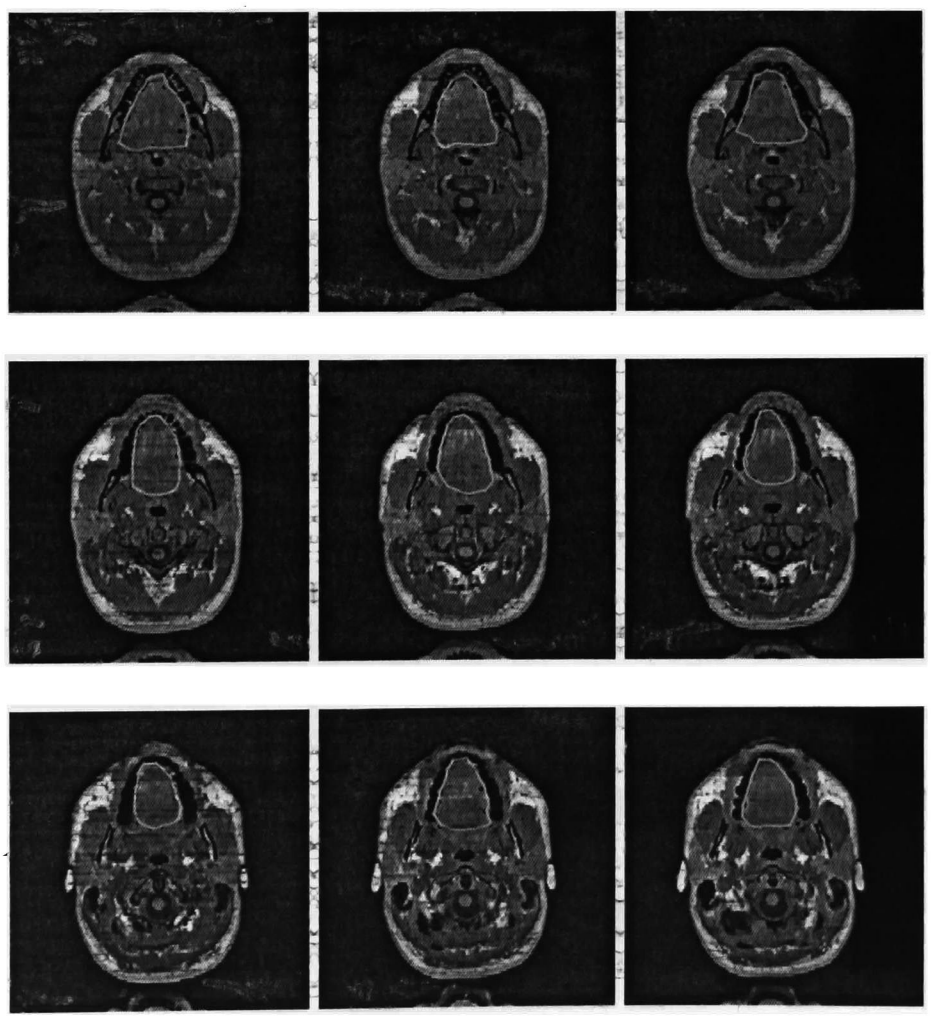


Figura 3.17: Resultados incorporando la información del GGVF.

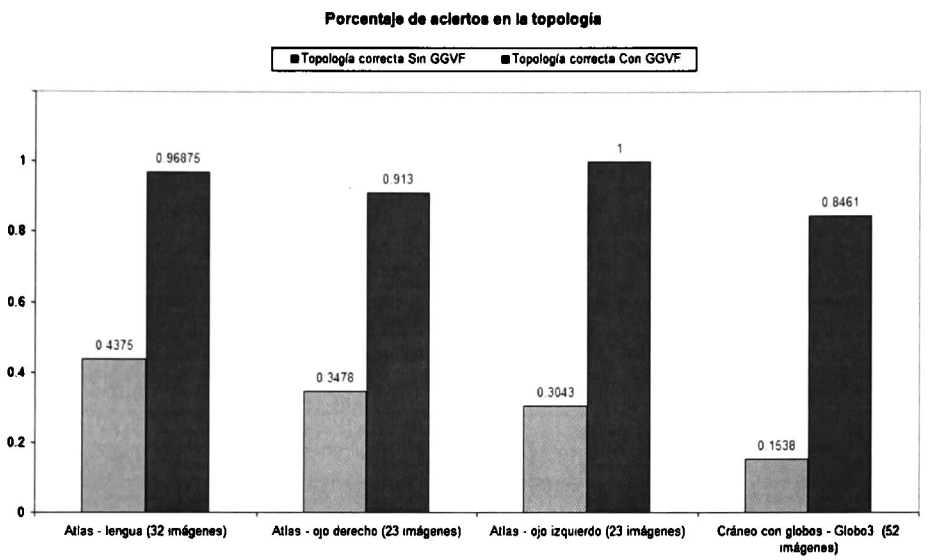


Figura 3.18: Porcentaje de imágenes en cada conjunto de prueba para las cuales se obtuvo una topología correcta al alcanzar la red el criterio de paro.

Tabla 3.2: Errores obtenidos. ϵ_{Ng}: error sin GGVF; ϵ_{Sg}: error con GGVF.

Ejemplo	ϵ_{Ng}	ϵ_{Sg}	Ejemplo	ϵ_{Ng}	ϵ_{Sg}
Ventriculo 1	3.29	2.51	Ojo 1	7.63	6.8
Ojo 2	3.43	2.98	Vértebra 1	4.65	4.1
Tumor 1	3.41	2.85	Tumor 2	2.95	2.41
Curva	2.84	1.97	Vértebra 2	2.9	2.5

información del GGVF y cuando no se usa; se puede observar nuevamente que la información del gradiente reduce el error en la aproximación del contorno. La tabla 3.3 muestra los errores obtenidos usando (3.27) deteniendo el algoritmo con diferentes números de neuronas usando la información del GGVF.

La figura 3.19 muestra el resultado cuando el algoritmo se aplica a una imagen de resonancia magnética (RM) para obtener la forma del ventrículo. La figura 3.19.a muestra la imagen del cerebro original y la región de interés (ROI); la figura 3.19.b muestra una parte amplificada del campo vectorial en la región de interés; la figura 3.19.c las líneas de flujo en la región de interés definidas por partículas colocadas sobre los vértices de una malla de 32×32 distribuida sobre la imagen; la figura 3.19.d muestra la inicialización con los dos trasladores M_a, M_b; la figura 3.19.e muestra la forma final obtenida, y finalmente la figura 3.19.f la imagen original con el objeto segmentado.

El ejemplo de la figura 3.20 presenta una imagen de tomografía computarizada

Tabla 3.3: Errores obtenidos por el algoritmo usando la información del GGVF y deteniendo el aprendizaje al alcanzar diferentes números de neuronas N_{max}; ϵ_{Sg} es el error calculado.

Ejemplo	N_{max}	ϵ_{Sg}	N_{max}	ϵ_{Sg}
Tumor 1	5	11.1	10	4.4
Tumor 1	15	3.6	20	2.4
Ventriculo 1	10	16.5	20	9.3
Ventriculo 1	30	6.9	40	4.7
Ventriculo 1	50	3.2	60	2.5
Ojo 1	5	13.4	10	7.7
Ojo 1	15	5.9	20	5.2
Curva	20	7.4	30	5.0
Curva	40	3.9	50	3.1
Curva	60	2.8	70	1.9

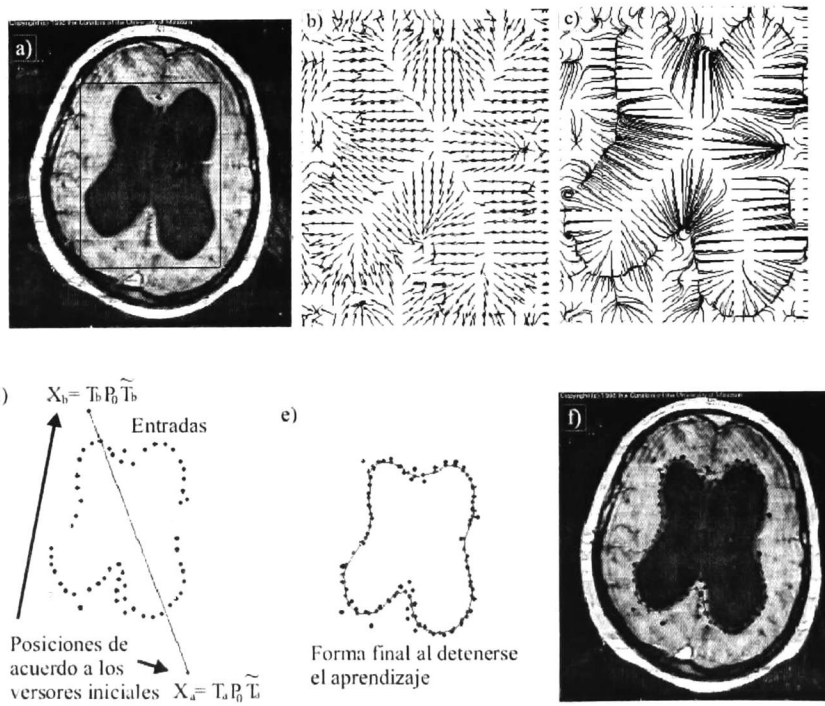


Figura 3.19: a) Imagen original y la región de interésn (ROI); b) Acercamiento del campo vectorial denso de la región de interés; c) Acercamiento de las líneas de flujo en la región de interés; d) Entradas e inicialización; e) Forma final de acuerdo con 54 trasladores estimados (54 neuronas); f) Imagen segmentada de acuerdo a los resultados.

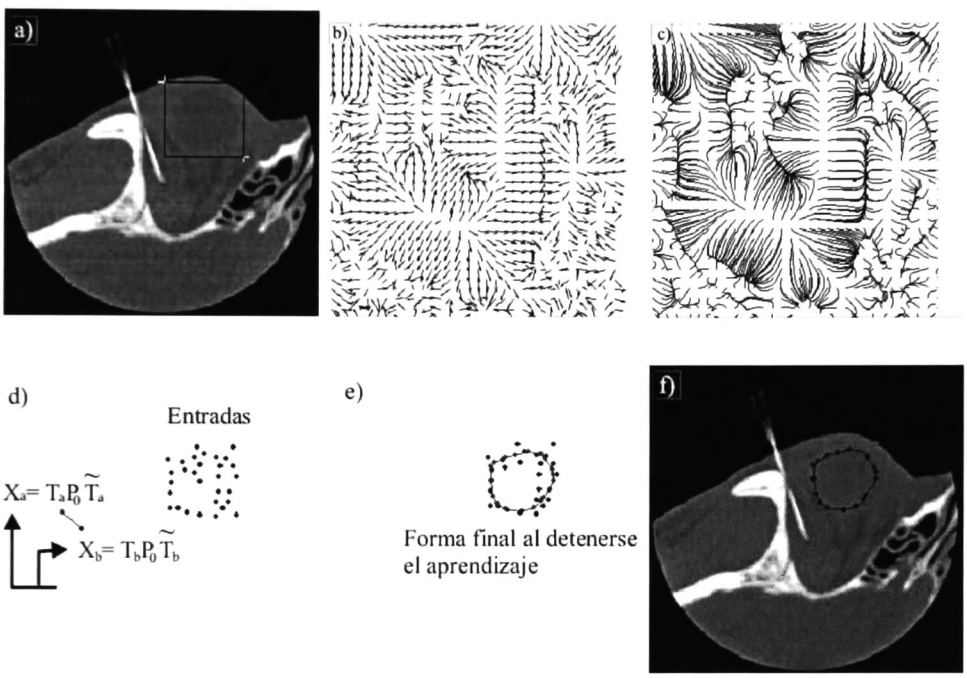


Figura 3.20: a) Imagen original del cerebro con la región de interés marcada; b) Acercamiento del campo vectorial en la región de interés; c) Acercamiento de las líneas de flujo en la región de interés; d) Entradas e inicialización de M_a and M_b; e) Forma final obtenida con 25 trasladores; f) Imagen original con el objeto segmentado.

que contenía un objeto con contorno borroso. Se seleccionó esta imagen como ejemplo para mostrar que aún en imágenes muy ruidosas o borrosas, el algoritmo obtiene buenos resultados. Es de notar que si se usa la *GGVF-Snake* [35], ésta no tiene éxito en encontrar la forma del objeto sin importar si la inicialización de la *snake* se da dentro, fuera o sobre el contorno (borroso) del objeto en cuestión, como se muestra en la figura 3.21; mientras que usando GGVF con la red neuronal, la forma resultante es más cercana a la forma esperada. La figura 3.20 muestra el proceso para la imagen CT; note que solamente trabajamos en la región de interés. La figura 3.20.a muestra la imagen original y la región de interés; la figura 3.20.b el campo vectorial calculado para la región de interés; la figura 3.20.c las líneas de flujo definidas por partículas colocadas en una rejilla uniforme de 32×32; la figura 3.20.d muestra las entradas seleccionadas y la inicialización con los trasladores M_a, M_b; la figura 3.20.e muestra la forma final obtenida; finalmente la figura 3.20.f muestra el resultado sobrelapado con la imagen original, mostrando que el algoritmo obtiene muy buenos resultados para la tarea de segmentación.

Es importante destacar que aunque los métodos de las figuras 3.20 y 3.21 usan la

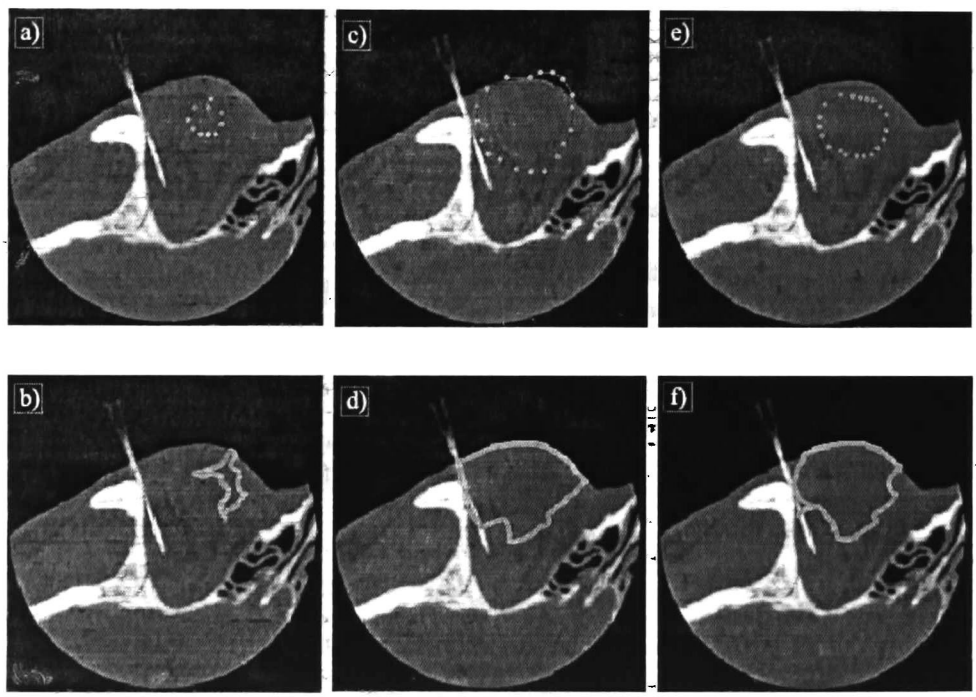


Figura 3.21: Resultados obtenidos usando contornos activos (GGVF-snake) para segmentar el mismo objeto que en la figura 3.20. Note que aunque dicha propuesta usa la información del GGVF, no tiene éxito al segmentar el objeto, sin importar si la inicialización del contorno activo se hizo dentro, fuera o sobre el contorno del objeto. a) Inicialización del *snake* dentro del objeto; b) Resultado final para la inicialización dada en (a); c) Inicialización del *snake* fuera del objeto; d) Resultado final para la inicialización dada en (c); e) Inicialización del *snake* sobre del contorno del objeto; f) Resultado final para la inicialización dada en (e)

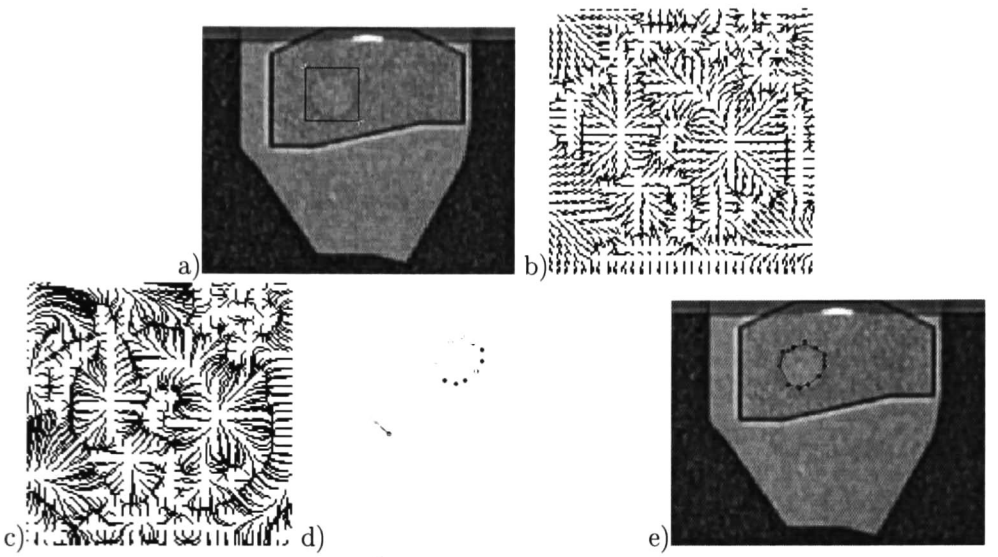


Figura 3.22: Aplicación en tareas de inspección visual: a) Imagen original y la región de interés (ROI); b) Acercamiento del campo vectorial GGVF de la ROI; c) Acercamiento de las líneas de flujo en la ROI; d) Entradas y forma inicial de acuerdo a las dos transformaciones iniciales M_a and M_b; e) Forma final definida por los 15 motores estimados.

información del GGVF para encontrar la forma de un objeto, la forma final estimada es mejor usando el método con redes neuronales que el que usa contornos activos; éste último (ver figura 3.21) falla al segmentar el objeto sin importar si la inicialización del contorno se dió dentro, fuera o sobre el contorno del objeto de interés. Además, el hecho de expresar la forma como un conjunto de trasladores independientes de estar referidos a un sistema de coordenadas fijo, que mueven una entidad a posiciones sobre el contorno, nos permite tener un modelo que puede ser utilizado en futuras aplicaciones que puedan requerir la deformación del modelo inicial, en especial si el modelo no esta basado en puntos, sino en alguna otra entidad geométrica dentro del marco del AG, ya que no necesitamos cambiar de versores, sino que aplicamos los mismos y en la misma forma.

Otro ejemplo similar al anterior, pero en otro entorno de trabajo es el que se muestra en las figuras 3.22 y 3.23. La figura 3.22 muestra la aplicacion del algoritmo propuesto para tareas de inspección visual en la industria. La imagen corresponde a un cabezal de lectura de un disco duro. La figura 3.22.a muestra la imagen original y la región de interés; la figura 3.22.b el campo vectorial de la ROI; la figura 3.22.c los stream lines definidos por partículas en una rejilla de 32x32; la figura 3.22.d muestra las entradas seleccionadas y la inicialización de la red M_a, M_b; la figura 3.22.e muestra la forma final obtenida. La figura 3.23 muestra la aplicación del algoritmo *GGVF-snake* al mismo problema. Al igual que en el caso de las figuras 3.20 y 3.21, la forma final es mejor estimada con el algoritmo propuesto que con los contornos activos, ambos usando la información del GGVF.

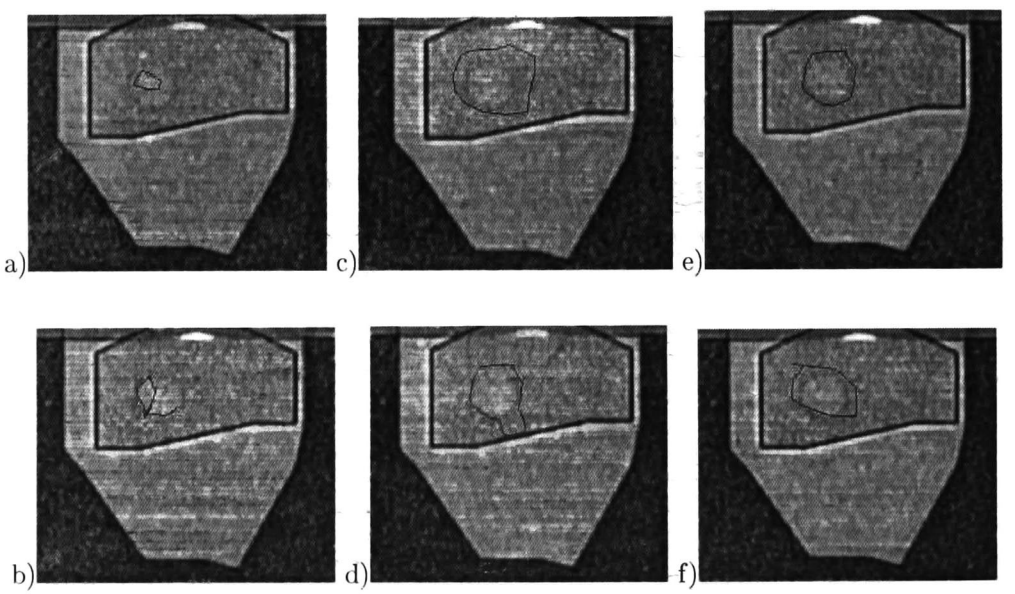


Figura 3.23: Resultados obtenidos usando el método de contornos activos *GGVF-snake* para segmentar el objeto en la imagen de la figura 3.22. a) Inicialización del *snake* dentro del objeto; b) Resultado obtenido con la inicialización mostrada en (a); c) Inicialización del *snake* fuera del objeto; d) Resultado obtenido con la inicialización (c); e) Inicialización del *snake* sobre el contorno del objeto; f) Resultado obtenido con la inicialización mostrada en (e)

La figura 3.24 es un conjunto de imágenes CT de un paciente con un tumor cerebral, mientras que la figura 3.25 muestra la aplicación del algoritmo a imágenes RM. La columna izquierda de cada figura muestra la imagen original y la región de interés, mientras que la columna derecha muestra el resultado del método propuesto.

Es importante mencionar que el tiempo computacional requerido para todas las imágenes mostradas en este trabajo tomó solamente unos pocos segundos. El cálculo del GGVF es la tarea que más consume tiempo en el algoritmo, pero para imágenes de 64x64 esto toma aproximadamente 3 segundos, mientras que toma 20 segundos para imágenes de 256x256 y 110 segundos para imágenes de 512x512. Esta es la razón por la que se decidió no calcularlo para la imagen completa, sino solamente para regiones de interés.

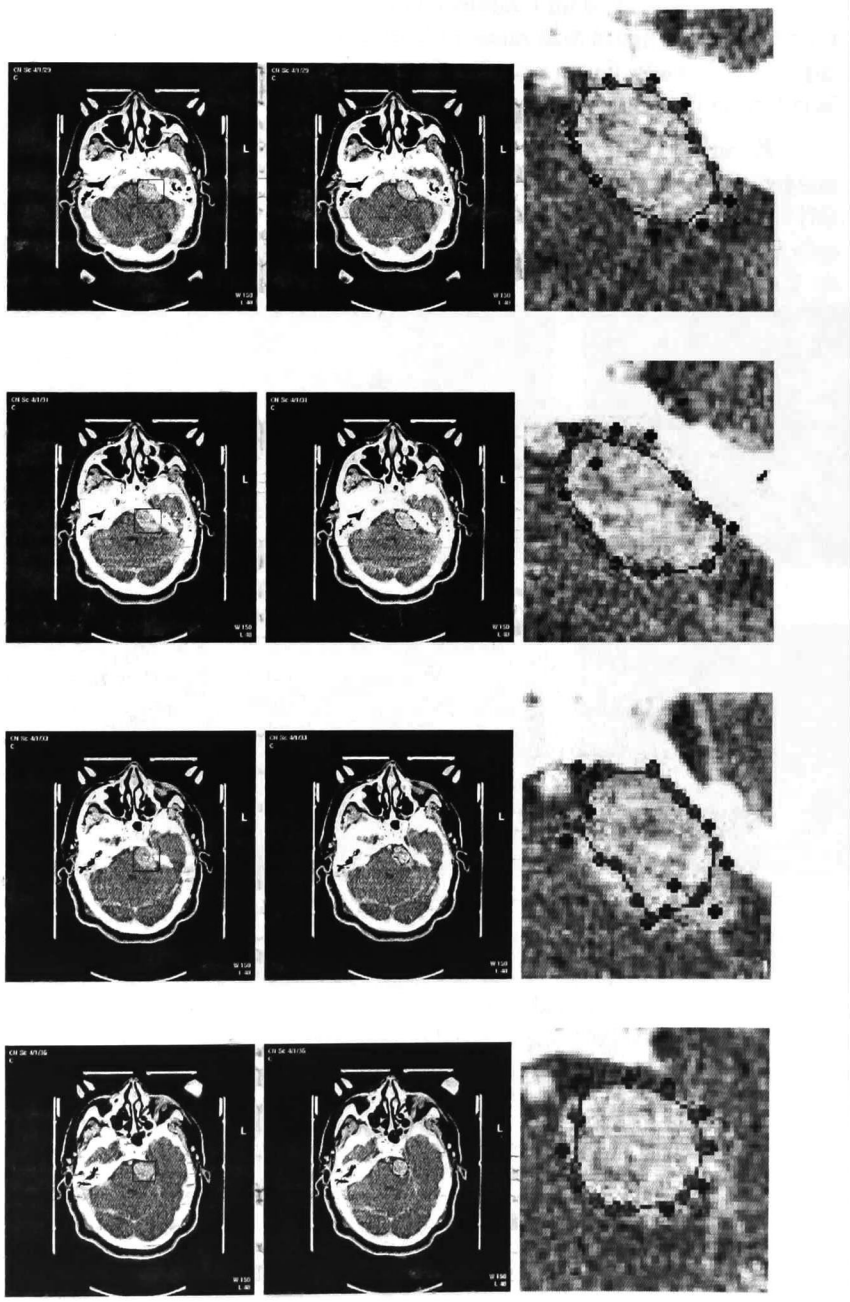


Figura 3.24: **Columna izquierda:** imagen original y la región de interés. **Columna central:** resultado de la segmentación. **Columna derecha:** Acercamiento del resultado en la columna central

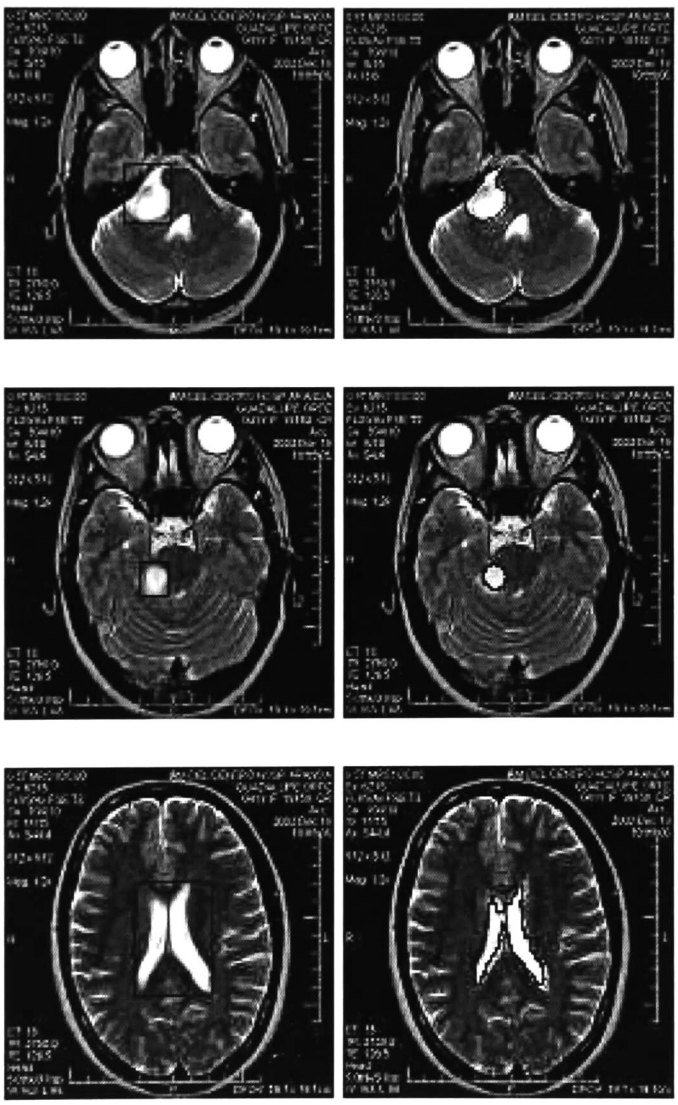


Figura 3.25: **Columna izquierda:** imagen original y la región de interés. **Columna derecha:** resultado de la segmentación.

Capítulo 4

Modelado tridimensional

En el análisis de imágenes médicas, la disponibilidad de modelos tridimensionales es de gran interés para los médicos porque les permiete tener una mejor comprensión de la situación a la que se enfrentarán y tales modelos son realativamente fáciles de construir. Ejemplos de tales modelos son algunos de los que se construyeron para este trabajo de tesis y que se muestran en las figuras 4.1 a 4.3.

4.1. Modelado de superficies usando redes neuronales y GGVF

La aproximación de contornos de objetos 2D usando una red neuronal basada en GCS y GNG se trató en el capítulo 3 (sección 3.3), incorporando la información del GGVF y trabajando en el marco del álgebra geométrica conformal. En esta sección se presenta la versión 3D del algoritmo de aprendizaje. En este caso incorporamos el uso de rotores para formar los motores $M = TR$ asociados a las neuronas. El objetivo es aproximar las

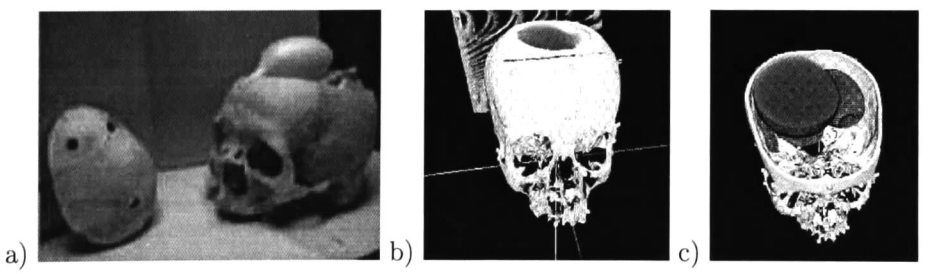


Figura 4.1: Ejemplo de modelo tridimensional: a) cráneo humano con globos en su interior rellenos con sulfato de cobre para simular el cerebro humano en las imágenes de tomografía; b) y c) Modelo virtual 3D

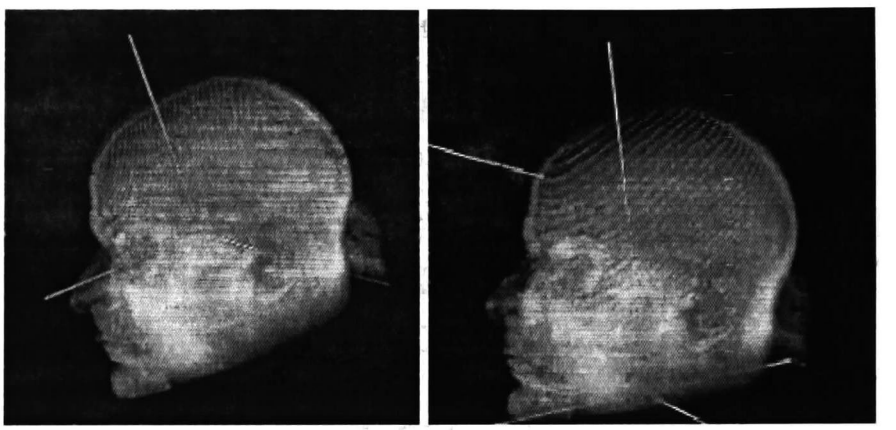


Figura 4.2: Ejemplo de modelo tridimensional: cabeza humana (imágenes de paciente real)

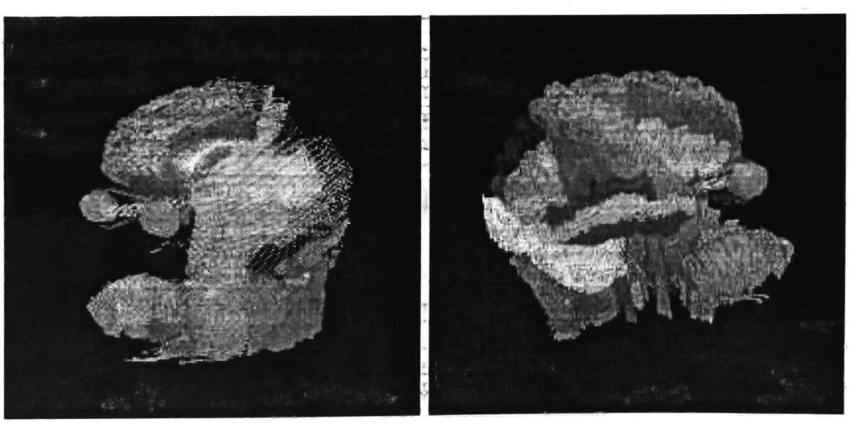


Figura 4.3: Ejemplo de modelo tridimensional basado en un atlas de la cabeza humana

superficies de los objetos 3D. El algoritmo es como sigue (recuerde que e_w y e_n son los parámetros de aprendizaje para la neurona ganadora y sus vecinas, respectivamente):

1. Sea P_0 un punto inicial al cual se aplicarán las transformaciones encontradas. Estas transformaciones están expresadas en 3D como $M = T\ R = e^{-\frac{t}{2}e_\infty} e^{\frac{\theta}{2}\mathbf{b}}$.

2. Iniciar con un número mínimo de neuronas con sus motores M y sus vectores $\mathbf{v_l} = [u_l,\ v_l,\ w_l]$ asociados.

3. Tomar una muestra ζ del conjunto de entradas $\mathbf{I}$ y encontrar la neurona ganadora; es decir, la neurona que tenga el motor M_l que mueva el punto P_0 más cerca de la muestra tomada

$$M_{win} = \min_{\forall M} \sqrt{(X_\zeta - M P_0 \tilde{M})^2} \tag{4.1}$$

   A esta neurona incrementarle su sc_{win} y recalcular rsf_{win}

4. Modificar M_{win} y los M_l de las neuronas vecinas de forma que los M_l modificados representen una transformación que mueve P_0 más cerca de la entrada.

$$M_{l_{new}} = \Delta M_l M_{l_{old}} \tag{4.2}$$

   Donde cada motor $\Delta M = \Delta T \Delta R$. La rotación ΔR se calcula como en (2.26), donde θ es el ángulo entre la posición actual $P_0' = M P_0 \tilde{M}$ y la posición P_0' trasladada por el vector $\mathbf{v}$, esto es: $P_0'' = T_\mathbf{v} P_0' \tilde{T}_\mathbf{v}$ donde $T_\mathbf{v} = e^{\frac{1}{2}e_\infty \mathbf{v}}$; el bivector $\mathbf{b}$ dual al eje de rotación se calcula como en (2.16):

$$\theta = \arccos\left(\frac{P_0' \cdot P_0'' + \frac{1}{2}|P_0'|^2 + \frac{1}{2}|P_0''|^2}{|P_0'||P_0''|}\right) \tag{4.3}$$

(4.4)

   Así, para la neurona ganadora se calcula

$$\Delta R_{win} = e^{\frac{\eta(\mathbf{v}_\zeta, \mathbf{v_{win}})\ \Delta\theta_{win}}{2}\mathbf{b}} \tag{4.5}$$

$$\Delta T_{win} = e^{-\frac{\Delta t_{win}}{2} e_\infty} \tag{4.6}$$

   donde $\eta(\mathbf{v}_\zeta, \mathbf{v_{win}})\ \Delta\theta_{win}$ resulta en una fracción del ángulo $\Delta\theta_{win}$

$$y\Delta\theta_{win} = e_w\ \theta \tag{4.7}$$

$$\Delta t_{win} = e_w\ \eta(\mathbf{v}_\zeta, \mathbf{v_{win}})\ (\langle X_\zeta - P_0' \rangle_{1\varepsilon}) \tag{4.8}$$

   Mientras que para las neuronas vecinas a la ganadora

$$\Delta R_n = e^{\frac{\eta(\mathbf{v}_\zeta, \mathbf{v_n})}{2}\mathbf{b}\Delta\theta_\mathbf{n}} \tag{4.9}$$

$$\Delta T_n = e^{-\frac{\Delta t_n}{2} e_\infty} \tag{4.10}$$

donde

$$\Delta\theta_n = e_n\,\theta \tag{4.11}$$

$$\Delta t_n = e_n\,\eta(\mathbf{v}_\zeta, \mathbf{v_n})\,(\langle X_\zeta - P'_0\rangle_{1\varepsilon}) \tag{4.12}$$

donde $\langle H\rangle_{1_\varepsilon}$ deja solo los elementos de H de grado 1 que pertenecen a G_3 ($H \in \langle G_3\rangle_1$) y P'_0 es la posición actual $P'_0 = MP_o\tilde{M}$. ϕ es una función que define la cantidad que una neurona aprende de acuerdo a su distancia a la neurona ganadora (definida como en (4.13)), y $\eta(\mathbf{v}_\zeta, \mathbf{v_l})$ es definida como en (4.14).

$$\phi = e^{-\frac{(M_{win}P_0\tilde{M}_{win} - M_lP_0\tilde{M}_l)^2}{2\sigma}} \tag{4.13}$$

$$\eta(\mathbf{v}_\zeta, \mathbf{v_l}) = \|\mathbf{v}_\zeta - \mathbf{v_l}\|^2 \tag{4.14}$$

Así se obtiene $\Delta M = \Delta T\Delta R$, y el nuevo motor $M_{l_{new}}$. Finalmente, actualizar $\mathbf{v_l}$:

$$\mathbf{v_{l_{new}}} = [u_{l_{new}} \;\; v_{l_{new}} \;\; w_{l_{new}}]^T \tag{4.15}$$

donde

$$u_{l_{new}} = (u_l + \alpha\,\phi\,u_l) \qquad v_{l_{new}} = (v_l + \alpha\,\phi\,v_l) \qquad w_{l_{new}} = (w_l + \alpha\,\phi\,w_l) \tag{4.16}$$

5. Cada cierto número λ de iteraciones:

   Determinar la neurona $\mathbf{n_i}$ con el valor más grande en rsf_l. Si alguna de sus vecinas directas, digamos $\mathbf{n_j}$, esta conectada por una arista con longitud mayor a c_{max}, entonces crear una nueva neurona $\mathbf{n_{new}}$ entre $\mathbf{n_i}$ y $\mathbf{n_j}$ cuyo motor asociado y el vector $\mathbf{v_l}$ serán

   $$M_{n_{new}} = \frac{M_i + M_j}{2} \tag{4.17}$$

   $$\mathbf{v_{l\_new}} = \frac{\mathbf{v_i} + \mathbf{v_j}}{2} \tag{4.18}$$

   Esta nueva neurona tendrá $sc_{new} = 0$ y $rsf_{new} = 0$.

   Borrar la conexión anterior entre $\mathbf{n_i}$ y $\mathbf{n_j}$ y crear dos nuevas conexiones entre $\mathbf{n_{new}}$ con $\mathbf{n_i}$ y $\mathbf{n_j}$

6. Repetir los pasos 3 a 5 hasta alcanzar el criterio de paro, el cual puede ser alcanzar el máximo número de neuronas.

Al igual que en el caso 2D, conforme evoluciona la red se va minimizando el error

$$\chi = \sum_{\forall\zeta}(((M_\zeta P_0\tilde{M}_\zeta) - X_\zeta)^2) \tag{4.19}$$

donde M_ζ es el motor que mueve P_0 más cerca de la entrada ζ.

La figura 4.4.a muestra la cabeza de un paciente con un tumor cuya superficie queremos aproximar. La figura 4.4.b muestra los vectores del GGVF en un arreglo tridimensional de tamaño 32x32x16; la figura 4.4.c muestra las entradas determinadas, mientras que la figura 4.4.d muestra la inicialización de la red. Las figuras 4.4.e-g muestran algunas etapas del proceso de adaptación conforme la red va determinando el conjunto de transformaciones M. La figura 4.4.g es la forma final estimada con un total de 170 motores M (los asociados a 170 neuronas). Las figuras 4.5 a 4.7 muestran los resultados obtenidos

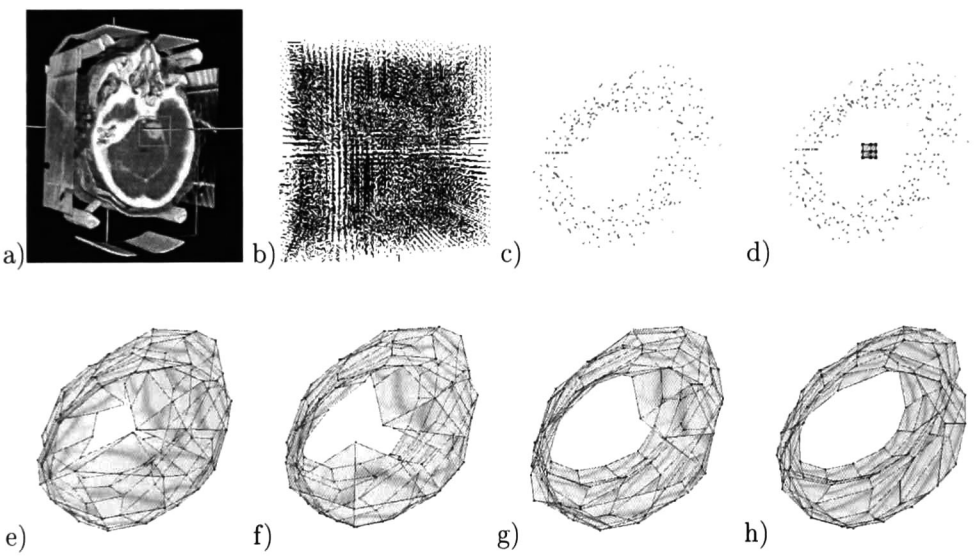


Figura 4.4: El algoritmo en la estimación de superficies 3D. a) Modelo tridimensional de la cabeza de un paciente con un tumor en la región marcada; b) Vectores del GGVF en un arreglo 3D de 32x32x16; c) Entradas determinadas por el GGVF y el mapa de bordes; d) Entradas y la inicialización de la red; e-g) Etapas durante el proceso de adaptación; h) Forma estimada al finalizar el aprendizaje con un total de 170 unidades neuronales.

con otros dos ejemplos usando datos volumétricos; el objeto de la figura 4.7 no es de datos médicos, sino que corresponde a una pera. Cada figura muestra las entradas a la red en a), las entradas y la inicialización de la red en b), mientras que en c) se muestran los resultados cuando la red alcanza el máximo número de neuronas. Finalmente, en d) se muestra cómo deciende el error calculado con (4.19).

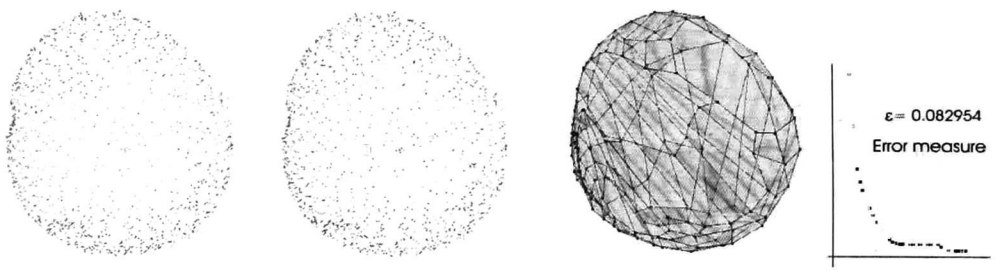


Figura 4.5: Ejemplo de la estimación de superficies 3D. **a)** Entradas a la red seleccionadas usando el GGVF y las líneas de flujo; **b)** Entradas y la inicialización de la red con nueve unidades neuronales; **c)** Resultado después de alcanzar el máximo número de neuronas (150); **d)** Medida del error usando (4.19)

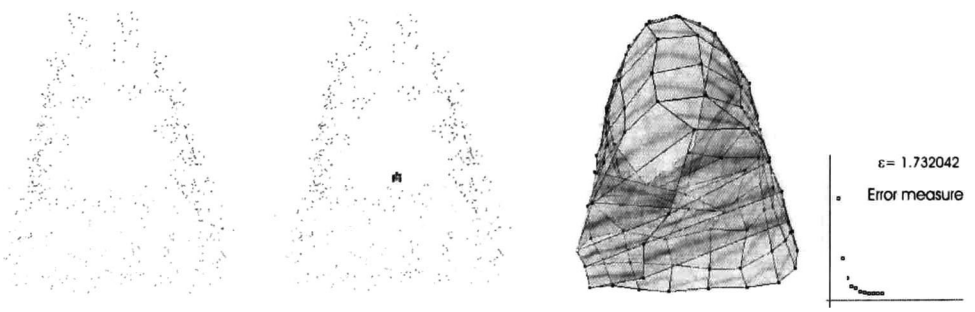


Figura 4.6: **a)** Entradas a la red seleccionadas usando el GGVF y las líneas de flujo; **b)** Entradas y la inicialización de la red con nueve unidades neuronales; **c)** Resultado después de alcanzar el máximo número de neuronas (130); **d)** Medida del error usando (4.19)

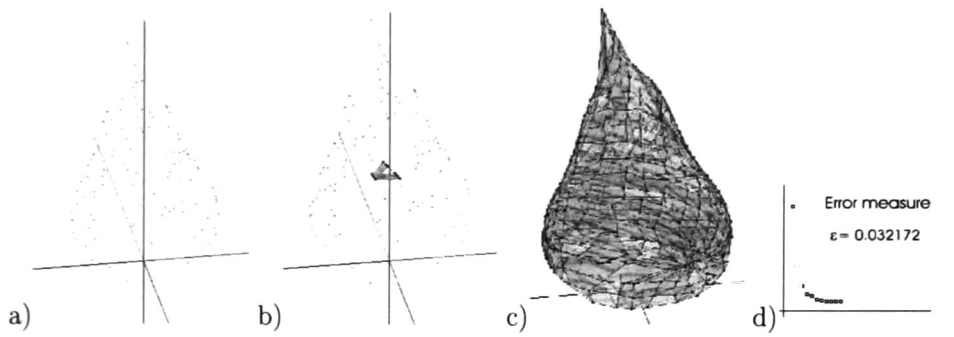


Figura 4.7: Definición de la superficie de una pera. **a)** Entradas a la red seleccionadas usando el GGVF y las líneas de flujo; **b)** Entradas y la inicialización de la red con nueve unidades neuronales; **c)** Resultado después de alcanzar el máximo número de neuronas (300); **d)** Medida del error usando (4.19)

4.2. Modelado 3D usando esferas

En la literatura sobre modelado de datos volumétricos podemos encontrar el algoritmo llamado Unión de Esferas (UoS por sus siglas en inglés), el cual usa las esferas para construir modelos tridimensionales de objetos. Sin embargo, nosotros usaremos las ideas básicas del algoritmo *marching cubes* para desarrollar un método alternativo para construir tales modelos, el cual tiene la ventaja de que reduce el número de esferas necesario.

4.2.1. Unión de esferas

El algoritmo Unión de Esferas [27] se puede resumir en los siguientes pasos:

- Dado un conjunto de puntos del borde del objeto 3D, calcular la tetrahedrización de Delaunay.
- Calcular la esfera que circunscribe a cada tetrahedro.
- Verificar cuáles esferas de los datos originales están dentro del objeto definido.
- Simplificar la representación densa de esferas agrupando y eliminando esferas no significativas o redundantes.

Este algoritmo tiene una complejidad en el peor de los casos de $O(n^2)$ tanto en tiempo como en número de esferas, donde n es el número de puntos que pertenecen al borde del objeto.

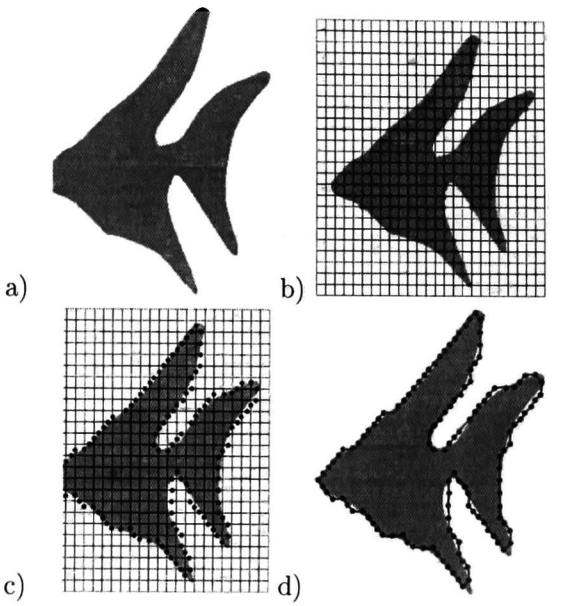


Figura 4.8: El algoritmo *marching cubes* en caso 2D. a) Objeto cuya silueta queremos representar; b) Determinación de vértices dentro del objeto; c) Nuevos vértices insertados; d) Aproximación de la superficie uniendo los nuevos vértices.

4.2.2. *Marching cubes*

Para explicar el algoritmo, veamos primero su equivalente en 2D. Supongamos que deseamos aproximar una silueta 2D como la que se muestra en la figura 4.8.a. Para ello colocamos una rejilla o malla (los cuadros juegan el papel de los cubos del espacio 3D). Luego se determina cuáles de los vértices de cada cuadro de la malla están dentro de la silueta que deseamos aproximar (véase figura 4.8.b). Posteriormente se insertan nuevos vértices colocados en la posición media de la distancia entre cada vértice dentro del objeto y su vecino fuera del objeto (vecino es el vértice que está conectado por una arista de la malla) como se muestra en la figura 4.8.c. Por último, se unen los vértices insertados usando lineas y se obtiene una aproximación de la superficie original (véase figura 4.8.d. Obviamente, entre más fina sea la malla, mejor será la aproximación de la silueta del objeto.

En el caso 3D el algoritmo es similar. La clave es subdividir el espacio en una serie de pequeños cubos; luego, el algoritmo se mueve o marcha a través de cada cubo para determinar si los vértices de éstos pertenecen a la superficie o no. Supongamos que tenemos un conjunto de m imágenes que contienen información sobre la superficie del objeto que

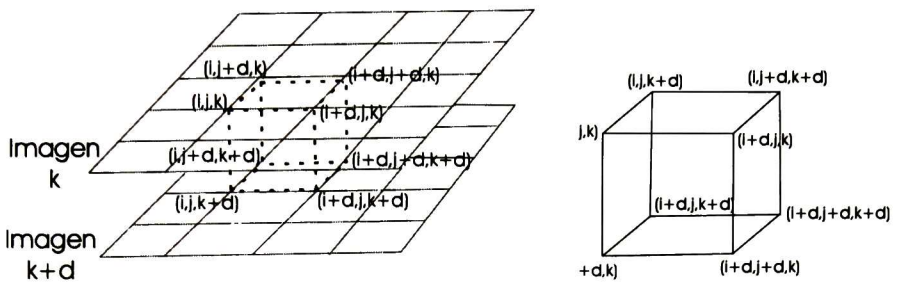


Figura 4.9: **a) Imagen izquierda**: El algoritmo *marching cubes* básico. **b) Imagen derecha**: Orden de los vértices para los cubos. Este orden se toma en cuenta para definir los centros y radios de los 15 casos básicos.

deseamos aproximar. Lo primero que hay que hacer es crear un conjunto de *cubos lógicos*, donde cada cubo tiene 4 vértices de la k-ésima imágen y 4 vértices de la $(k+1)$-ésima imágen. Entonces el algoritmo determina la forma en que el cubo interseca a la superficie; es decir, cuáles vértices del cubo están dentro y cuáles fuera de la superficie. Luego, el algoritmo se mueve (o marcha) al siguiente cubo. En cada cubo asignamos un valor de 1 a los vértices que están adentro de la superficie y un valor de 0 a los que están fuera (véase figura 4.9(a)).

Dado que hay 8 vértices en cada cubo y dos estados para cada uno (1 ó 0), tenemos $2^8 = 256$ formas en las que la superficie puede intersectar al cubo. En [20] se mostró cómo estos 256 casos pueden ser reducidos a solamente 15 básicos; las permutaciones de estos 15 casos usando complementariedad y simetría rotacional producen los 256 casos. Con este número de casos básicos es fácil crear conjuntos de polígonos predefinidos para hacer una aproximación apropiada de la superficie [20, 11].

4.2.3. *Marching cubes* usando esferas

Así como en el algoritmo *marching cubes*, veamos primero el caso 2D para modelar usando círculos. Supongamos queremos aproximar la forma del objeto mostrado en la figura 4.8.a. El proceso es similar al caso equivalente en 2D del algoritmo *marching cubes*:

- Colocar la malla o rejilla y determinar los puntos dentro y fuera de la superficie (igual que en las figuras 4.8.a y 4.8.b).
- Dibujar círculos con centro en cada vértice dentro de la superficie que se encuentre conectado con otro vértice fuera de la superficie.
- Por último, dibujar círculos centrados en los vértices exteriores que se encuentran conectados con dos vértices interiores.

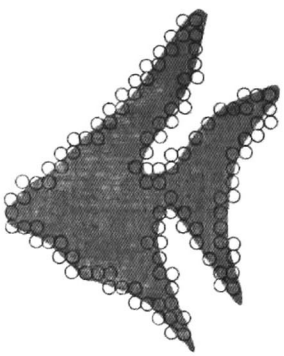


Figura 4.10: Equivalente 2D para obtener una representación del objeto basada en círculos.

- Como resultado se tiene una aproximación a la superficie original (véase figura 4.10). Nuevamente, entre más fina sea la malla, mejor será la aproximación de la superficie.

Para el caso 3D, el principio es el mismo que en el algoritmo *marching cubes*: dado un conjunto de m imágenes, dividir el espacio en cubos lógicos (cada cubo con 8 vértices, 4 de la imagen k y 4 de la imagen $k+1$); luego determinar cuáles de los vértices de cada cubo están dentro o fuera de la superficie. Posteriormente se define el número de esferas para cada cubo de acuerdo a lo que se muestra en la figura 4.11 y en (4.20)

$$\begin{aligned} S^j_{p_i} &= c_{p_i} + \frac{1}{2}(c^2_{p_i} - \rho^2_{p_i})e + e_0 \\ S^j_{m_i} &= c_{m_i} + \frac{1}{2}(c^2_{m_i} - \rho^2_{m_i})e + e_0 \\ S^j_{g_i} &= c_{g_i} + \frac{1}{2}(c^2_{g_i} - \rho^2_{g_i})e + e_0 \end{aligned} \tag{4.20}$$

donde el subíndice i indica la i-ésima esfera del caso j-ésimo tomando los índices de los vértices del cubo según la figura 4.9(b). Nótese que usamos los mismos 15 casos básicos del algoritmo *marching cubes* porque el total de 256 casos puede obtenerse con esta base. También nótese que en lugar de triángulos, definimos esferas y nuestra meta no es tener un algoritmo de "renderización", sino una representación de los datos volumétricos basada en esferas la cual pueda ser útil en el proceso de registro.

Resultados experimentales

Como ejemplos, se presentan dos conjuntos de tomografía computarizada como sigue:

- Un conjunto de 26 imágenes de un cráneo humano al cual se le colocaron algunos globos en el interior rellenos con agua y sulfato de cobre para simular el cerebro humano

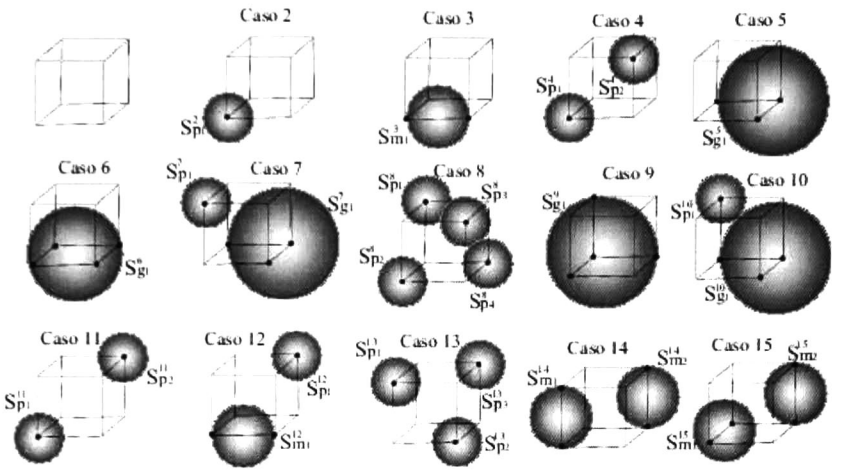


Figura 4.11: Los 15 casos básicos de la superficie intersectando los cubos. Las esferas se numeran comenzando en la esquina superior izquierda.

- Un conjunto de 36 imágenes de tomografía computarizada pertenecientes a un paciente con un tumor visible en 16 de las imágenes.

El primer paso es segmentar el cerebro (o el tumor) del resto de las estructuras. Una vez hecho ésto, las imágenes se binarizan para hacer más fácil la determinación de pixeles dentro y fuera de la superficie. El detector de bordes Canny (Apéndice B.1) se usa para obtener los puntos del contorno del objeto en cada imagen.

Las figuras 4.12 y 4.13 muestran una imagen del cráneo con el cerebro simulado, el objeto segmentado y la aproximación de la forma de éste usando círculos. La figura 4.14 muestra lo mismo pero usando las imágenes que corresponden a un paciente real y un tumor segmentado; mientras que la figura 4.15 muestra el caso de la extracción de tejido cerebral. La figura 4.16 muestra los resultados para el caso 3D del algoritmo presentado; la figura 4.16.a es el modelo para los globos en el cráneo, mietras que 4.16.b es el modelo del tumor segmentado en las imágenes del paciente. La tabla 4.1 es una comparación entre los resultados obtenidos con el algoritmo Unión de Esferas y nuestra propuesta para el caso de los globos dentro del cráneo. La primera fila muestra el peor caso con ambas propuestas; la segunda muestra el número de esferas con mejoras en el algoritmo de unión de esferas (lo cual se hace agrupando varias esferas en una sola que las contenga) y haciedo un poco menos finos los cubos lógicos ($d = 3$). El número de puntos correspondientes a los bordes obtenidos con el filtro canny es de $n = 3370$ para ambos casos. Nótese la reducción en el número de primitivas necesarias al usar la propuesta presentada: 13480 contra 11866 y 8642 contra 2602. La tercera y cuarta columna pertenecen a otros ejemplos con imágenes artificiales donde $n = 10329$ y $n = 2641$ respectivamente, y también en tales ejemplos es evidente la reducción en el número de entidades.

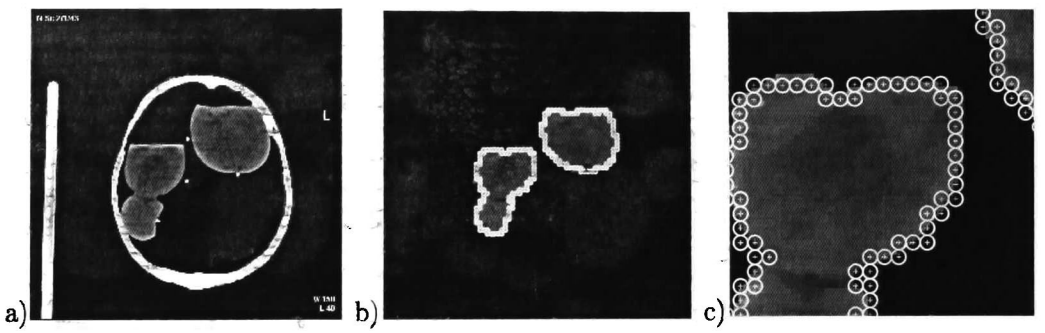


Figura 4.12: Cerebro simulado: a) Una de las imágenes de tomografía originales; b) Objeto segmentado y su aproximación usando circulos; c) Acercamiento (*zoom*) de (b).

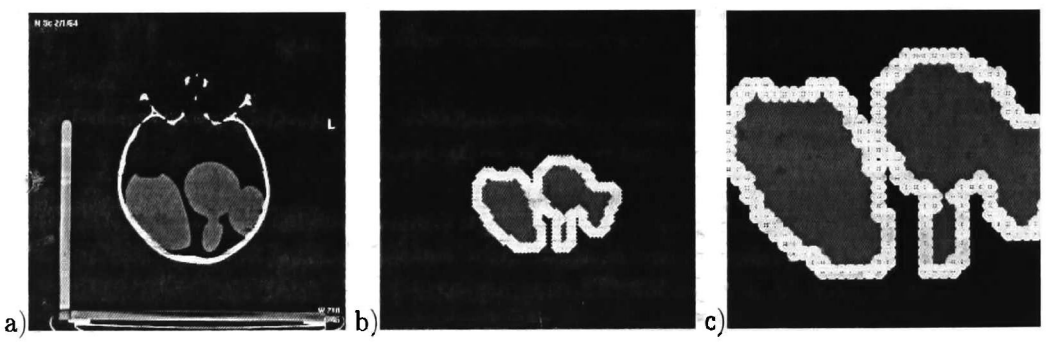


Figura 4.13: Cerebro simulado: a) Una de las imágenes de tomografía originales; b) Objeto segmentado y su aproximación usando círculos; c) Acercamiento (*zoom*) de (b).

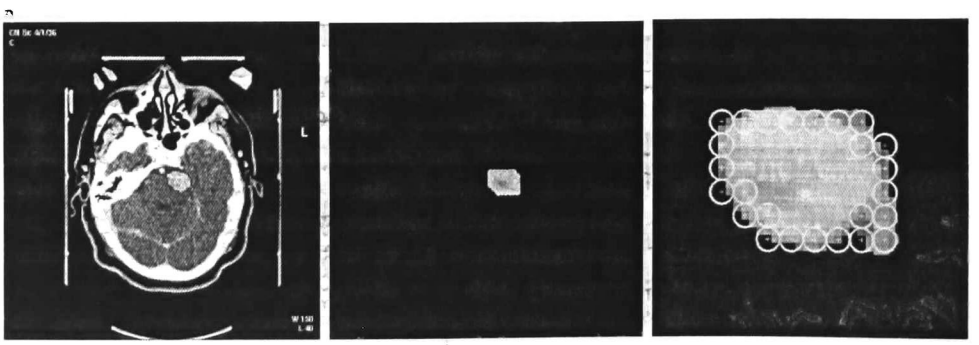


Figura 4.14: Caso real de un paciente: a) Imagen de tomografía original; b) Objeto segmentado (el tumor) y su aproximación usando círculos; c) Acercamiento (*zoom*) de (b).

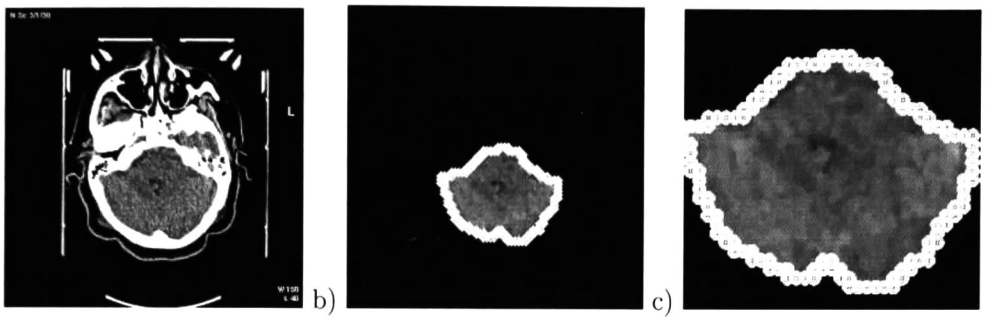


Figura 4.15: Caso real de un paciente: a) Imagen de tomografía original; b) Objeto segmentado (tejido cerebral) y su aproximación usando círculos; c) Acercamiento (*zoom*) de (b).

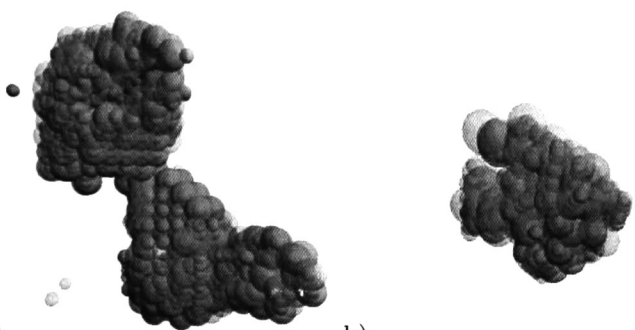


Figura 4.16: Aproximación de la forma de objetos tridimensionales con la propuesta presentada; a) aproximación de la estructura del cerebro simulado usando globos (datos sintéticos); b) aproximación de la forma del tumor extraído de las imágenes reales de un paciente.

Tabla 4.1: Comparación entre el número de esferas usando la unión de esferas y nuestra propuesta; n es el número de puntos de borde; d es la distancia entre vértices en los cubos lógicos.

n/d	Num de esferas en cada método	
	UoS	Propuesto
3370 / 1	13480	11866
3370 / 3	8642	2602
10329 / 2	25072	8362
2641 / 1	6412	5008

Capítulo 5

Registro de modelos

La *registración* o *proceso de registro* consiste en establecer un marco de referencia geométrico común entre dos o más conjuntos de datos. Éstos pueden ser datos tomados de diferentes modalidades, o en diferentes tiempos. El registro se hace se hace con el propósito de tener más información preoperativa e intraoperativa para diagnóstico y navegación. Es decir, al realizar el registro se puede hacer seguimiento óptico de la posición de instrumentos quirúrgicos y mostrarlos en el modelo virtual creado con imágenes prequirúrgicas, comparar información obtenida antes y durante la cirugía, etc.

5.1. Registro del paciente e instrumental con el modelo virtual

Si nos situamos en el quirófano, hay múltiples sistemas de coordenadas locales que debemos relacionar para ofrecerle al cirujano un modelo virtual de lo que sucede en la realidad. La figura 5.1 ilustra un escenario de este tipo. En nuestro caso, contamos con el sistema de seguimiento óptico cuyo nombre de fábricca es *Enhanced Hybrid Polaris System*, el cual se muestra en la figura 5.2 y al cual nos referiremos en adelante solo con el nombre de *Polaris*. Este sistema emite luz infraroja, la cual es reflejada por marcadores que se colocan al objeto que nos interesa rastrear (figura 5.2.c); al ser reflejada y detectada por el sistema Polaris, se estima la posición 3D de los diferentes marcadores. Lo primero que hay que hacer es alinear al paciente con el modelo virtual. Para ésto se le colocan al paciente marcadores fiduciales en la cabeza antes de tomar las imágenes con las cuales se construye el modelo. Estos marcadores quedan visibles en las imágenes CT o MR (ver figura 5.3). Una vez construído el modelo y teniendo al paciente en el quirófano, se forman los conjuntos de puntos definidos por esos fiduciales; el primero referido al sistema de coordenadas del modelo y el segundo al sistema de coordenadas del sistema de seguimiento Polaris. Con estos conjuntos de puntos se estima la transformación que relaciona ambos sistemas de coordenadas utilizando algún método como el RPM [14] para

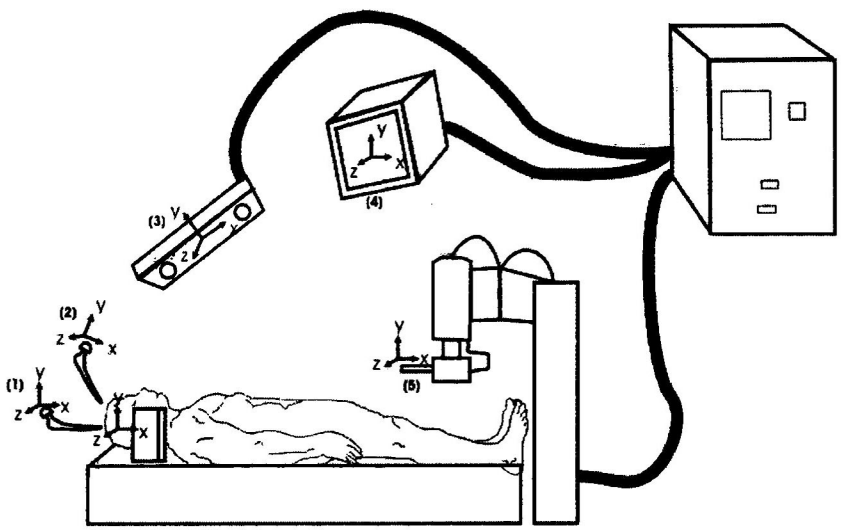


Figura 5.1: Algunos de los diferentes sistemas de coordenadas presentes en la sala de operaciones: instrumental quirúrgico (1,2), sistema de seguimiento óptico (3), modelo virtual (4), endoscopio (5).

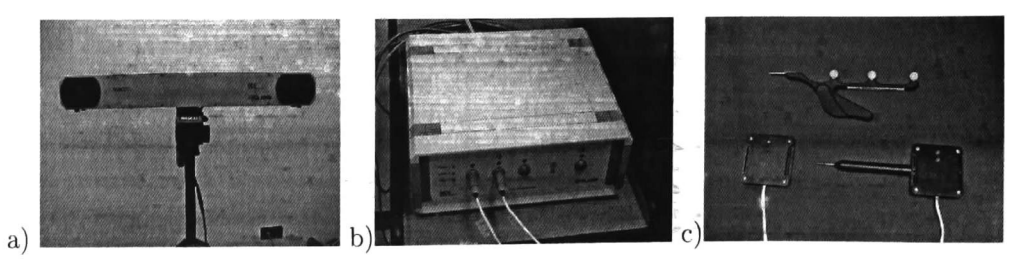


Figura 5.2: Sistema de seguimiento óptico *Polaris*.

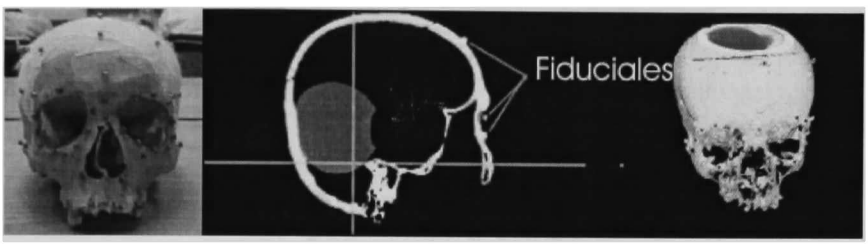


Figura 5.3: Puntos utilizados para relacionar al paciente con el modelo virtual construído.

descubrir primero las correspondencias.

La transformación obtenida relacionará todo lo que suceda en quirófano con el modelo virtual ya que es la transformación del sistema de coordenadas de la Polaris al sistema de coordenadas del modelo virtual.

Cuando se utiliza además información proveniente de endoscopía o de neuro sonografía, el siguiente paso es relacionar lo que ve la cámara del endoscopio (o el ultrasonido) con los otros sistemas de coordenadas. Para resolver este problema en específico, se realizó junto con la estudiante de doctorado Silena Herold García, una adaptación del algoritmo de calibración mano-ojo (*hand-eye calibration*, sección 2.5), con el fin de calcular la transformación entre el sistema de coordenadas de la cámara endoscópica y el sistema de coordenadas de los marcadores colocados al endoscopio. Llamaremos ahora a este problema la *calibración endoscopio-Polaris* [17]. El escenario se muestra en la figura 5.4. Como se observa, tenemos además una transformación entre la rejilla de calibración y el sistema de seguimiento Polaris, expresada con $\mathbf{M_{B_g}}$, la cual utilizaremos para validar los resultados obtenidos. como se describe más adelante. Las transformaciones, expresadas en motores $M = TR$ del AGC, asi como los sitemas de coordenadas que relacionan son:

- $\mathbf{O_p}$: Origen del sistema de seguimiento Polaris.
- $\mathbf{O_{m_e}}$: Origen de los marcadores fijados al endoscopio.

  $\mathbf{O_{c_e}}$: Origen de la cámara del endoscopio.

  $\mathbf{O_{g_c}}$: Origen de objeto de calibración (rejillao *grid* de calibración).
- $\mathbf{M_{A_i}}$: Transformación de $\mathbf{O_{g_c}}$ a $\mathbf{O_{c_e}}$ en el i-ésimo movimiento de la cámara Ésta es dada por los parámetros extrínsecos de la cámara (ver Apéndice C).
- $\mathbf{M_{B_i}}$: Transformación de $\mathbf{O_{m_e}}$ a $\mathbf{O_p}$. Se obtiene del sistema de seguimiento óptico Polaris.
- $\mathbf{M_X}$: Transformación de $\mathbf{O_{c_e}}$ a $\mathbf{O_{m_e}}$. Se obtiene con el método de calibración mano ojo (sección 2.5).

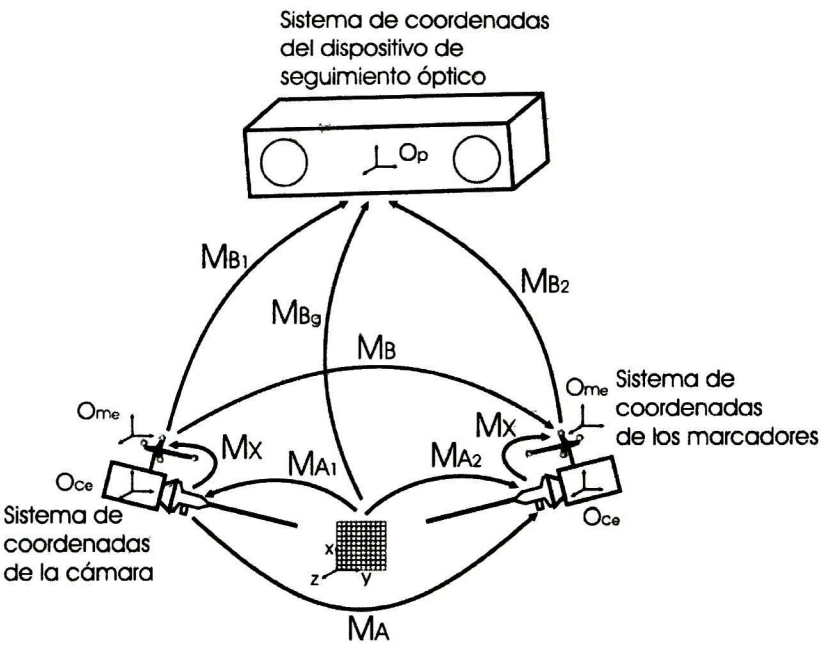


Figura 5.4: El problema de calibración entre la cámara endoscópica y el sistema de seguimiento Polaris.

- $\mathbf{M_{B_g}}$: Transformación de $\mathbf{O_{g_c}}$ a $\mathbf{O_p}$. Se obtiene directamente del sistema de seguimiento óptico Polaris al colocarle marcadores a la rejilla de calibración, los cuales puedan ser seguidos por el sistema.

Para validar los resultados obtenidos en la calibración endoscopio-Polaris, utilizamos la misma rejilla con la que se realizó la calibración de la cámara endoscópica[1], la cual es una especie de tablero de ajedrez de 12×15 cuadros, el cual se muestra en la figura 5.5.a. Sea X_g el conjunto de puntos correspondientes a las esquinas de los cuadros de la rejilla de calibración, con respecto a O_{g_c}. Estas coordenadas están expresadas en milímetros según las medidas de los cuadros de calibración. En la rejilla de calibración usada (figura 5.5.a), cada cuadro tiene 1.25 mm por lado.

1. Partiendo de los puntos X_g, llevarlos a la cámara por el camino definido por $\mathbf{M_{A_i}}$. Sea X_{A_i} el conjunto de puntos resultante.

2. Proyectar los puntos X_{A_i} a la imagen.

   $$x_{A_i} = K \ [R_{M_A} \ \ t_{M_A}] \ X_{A_i} \tag{5.1}$$

   Estos puntos deben coincidir con los puntos de las esquinas de los cuadros de la rejilla de calibración en la imagen (figura 5.5).

[1]Puede ver el procedimiento de calibración de cámaras en apéndice C

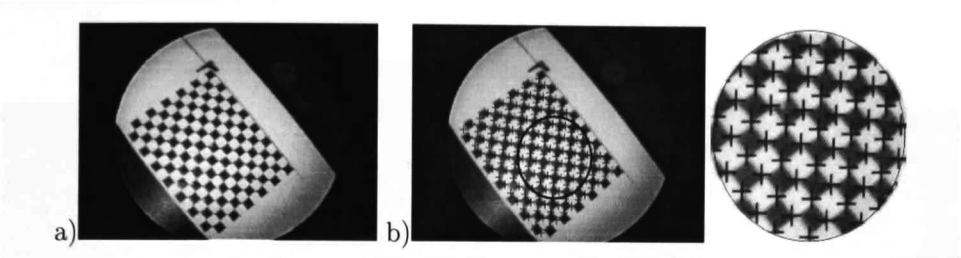


Figura 5.5: a) Imagen original; b) Resultado de la proyección con (5.1); se incluye un acercamiento de la zona marcada con el círculo para visualizar mejor.

3. Partiendo nuevamente de X_g, llevarlos a la cámara pero esta vez por el camino definido por las transformaciones $\mathbf{M_{B_g}}$, $\mathbf{M_{B_i}}$ y $\mathbf{M_X}$. Sea $X_{M_{B_iX}}$ el conjunto de puntos resultante.

4. Proyectar los puntos $X_{M_{B_iX}}$ a la imagen.

   (5.2) $$x_{B_i} = K \ [R_{M_{B_iX}} \ \ t_{M_{B_iX}}] \ \ X_{M_{B_iX}}$$

   En condiciones ideales, sin ruido, éstos deberían coincidir con las proyecciones x_{A_i}, pero debido a condiciones de iluminación y errores numéricos, puede haber un pequeño error (ver figura 5.6.a).

5. Medir el error ϵ entre la proyección de estos puntos

   (5.3) $$\epsilon = \frac{\sum_{i=1}^{n} (x_{A_i} - x_{B_i})}{n}$$

6. Como resultado de ruido introducido en las lecturas del sistema Polaris y la estimación de las transformaciones, puede haber un ligero corrimiento lineal entre la rejilla resultante x_{A_i} y x_{B_i}; para corregir este pequeño desplazamiento se calcula el centroide de cada uno de estos conjuntos de puntos, c_{A_i} y c_{B_i}, y luego se desplazan los puntos x_{B_i} para hacer coincidir sus centroides.

   (5.4) $$x'_{B_i} + (c_{A_i} - c_{B_i})$$

   Con esta corrección volvemos a medir el error promedio mediante

   (5.5) $$\epsilon' = \frac{\sum_{i=1}^{n} (x_{A_i} - x'_{B_i})}{n}$$

   En la figura 5.6.b se muestra el resultado de hacer esta corrección al resultado de la figura 5.6.a.

La figura 5.7 muestra un ejemplo de los resultados utilizando la cámara endoscópica sin el telescopio; mientras que la figura 5.8 resultados con el telescopio colocado a la

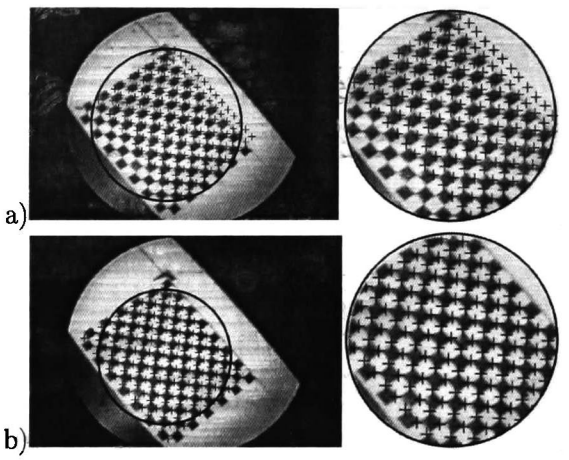


Figura 5.6: a) Resultado de la proyección con (5.2); b) Resultado aplicando (5.4)

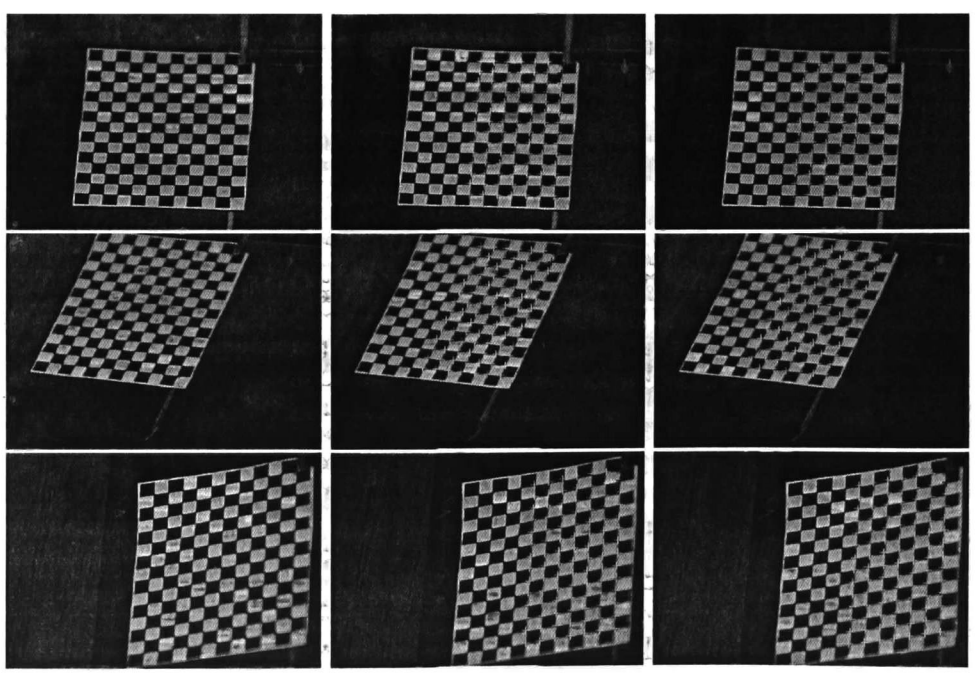


Figura 5.7: Validación de la calibración endoscopio-Polaris sin colocar el telescopio a la cámara. Columna izquierda: imagenes originales; Columna central: resultado de proyectar según (5.1); Columna derecha: resultado de proyectar con (5.2)

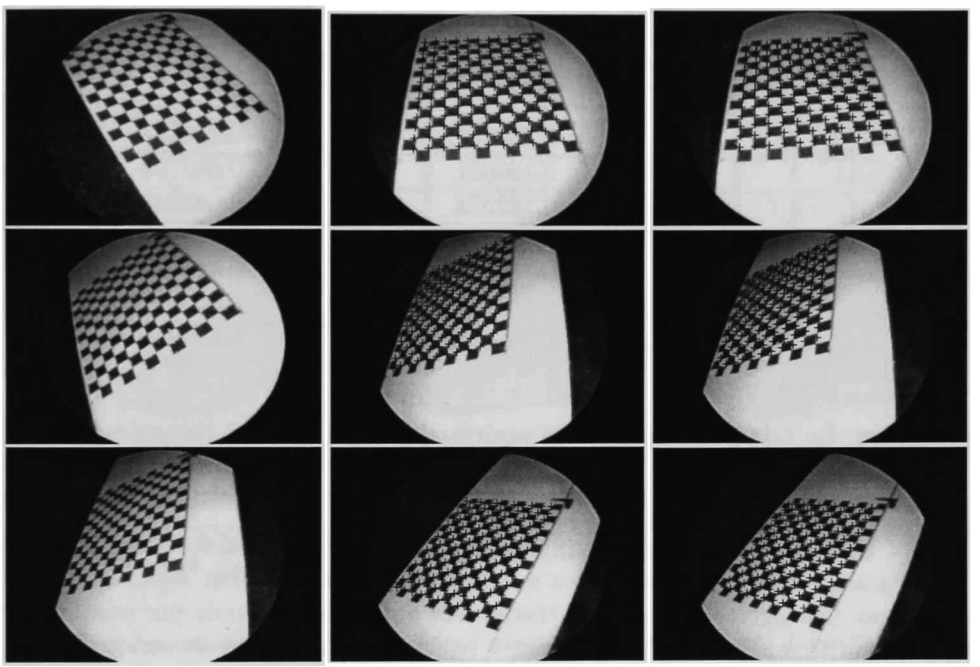


Figura 5.8: Validación de la calibración endoscopio-Polaris colocando el telescopio a la cámara. Columna izquierda: imagenes originales; Columna central: resultado de proyectar según 5.1; Columna derecha: resultado de proyectar con (5.2)

Tabla 5.1: Errores promedio de acuerdo a (5.5) obtenidos en milímetros al proyectar en la imagen los puntos X_g según (5.1) y (5.2) sin colocar el telescopio a la cámara y corrigiendo según (5.4).

Imagen	ϵ con (5.3)	ϵ con (5.5)	Imagen	ϵ con (5.3)	ϵ con (5.5)
1	0.029167	0.246190	6	0.026618	0.285241
2	0.029030	0.291577	7	0.036032	0.242242
3	0.029826	0.327040	8	0.030865	0.290385
4	0.038039	0.254144	9	0.030271	0.263296
5	0.054647	0.273854	10	0.042959	0.247765

Tabla 5.2: Errores promedio de acuerdo a (5.5) obtenidos en milímetros al proyectar en la imagen los puntos X_g según (5.1) y (5.2) colocando el telescopio a la cámara y corrigiendo según (5.4).

Imagen	ϵ con (5.3)	ϵ con (5.5)	ϵ con (5.3)	ϵ con (5.5)	
1	0.068417	0.111318	9	0.039312	0.041918
2	0.067803	0.222595	10	0.244908	0.258944
3	0.048298	0.150661	11	0.063193	0.462141
4	0.076784	0.162378	12	0.048048	0.142006
5	0.125940	0.149677	13	0.048910	0.051733
6	0.057448	0.146358	14	0.031168	0.103236
7	0.163959	0.204298	15	0.057036	0.082447
8	0.075360	0.090288	16	0.033915	0.102294

cámara. La tabla 5.1 muestra los errores obtenidos para 10 imágenes diferentes sin telescopio, mientras que la tabla 5.2 muestra los errores con 16 imágenes con el telescopio colocado a la cámara.

Una vez alineados los instrumentos del quirófano y el paciente con el modelo virtual, el siguiente reto es actualizar los modelos cuando éstos sufren alguna deformación en el tiempo. Las siguientes secciones tratan este punto, comenzando por una breve aplicación del algoritmo expuesto en la sección 4.1 para transformación de superficies (sección 5.2) y culminando con una propuesta para utilizar una versión modificada del algoritmo TPS-RPM para alineamiento de modelos basados en esferas (sección 5.3).

5.2. Transformación de modelos usando redes neuronales

Otra aplicación donde resulta útil el algoritmo expuesto en la sección 4.1, es la transformación de un modelo que se obtiene en un tiempo t_1 en otro obtenido en t_2. El primer modelo será el modelo a deformar, mientras que el segundo actuará como el conjunto esperado. Sean X_{1_i} las posiciones definidas por los versores de las neuronas del primer modelo, y X_{2_j} las posiciones definidas por los versores de las neuronas del segundo modelo. Las posiciones X_{2_j} actuarán como las entradas a la red, a las cuales se tratará de ajustar la topología de las neuronas del segundo conjunto. El algoritmo es básicamente el mismo que se expuso en la sección 4.1, excepto la inicialización y que en este caso no se realiza inserción de unidades neuronales:

1. Sean P_{0_1} el centroide del primer modelo y X_{1_i} las posiciones definidas por los motores de sus unidades neuronales $X_{1_i} = M_{1_i}\ P_{0_1}\ \tilde{M}_{1_i}$.

2. Las posiciones $X_{2_j} = M_{2_j}\ P_{0_2}\ \tilde{M}_{2_j}$ serán las entradas a la red.

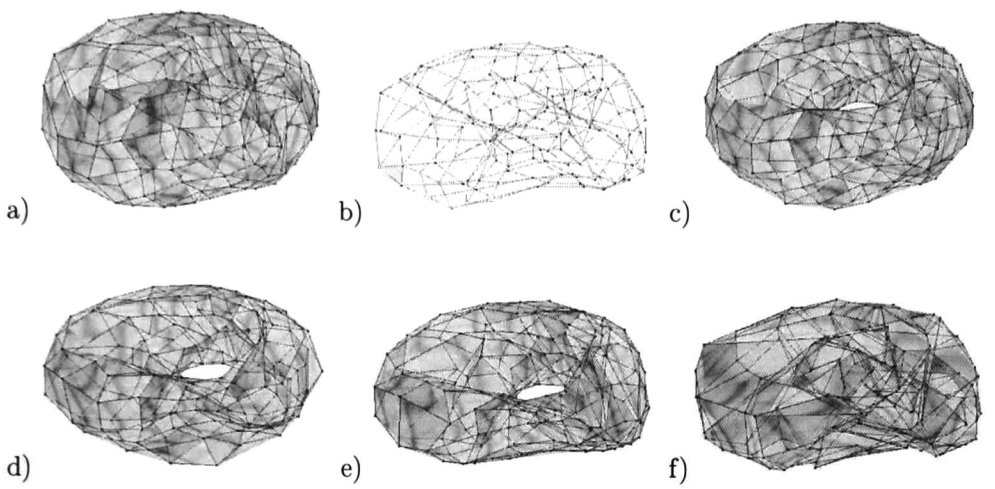


Figura 5.9: La superficie 3D mostrada en (a) se transforma en la mostrada en (b). Los incisos (c) a (f) muestran diferentes etapas durante la adaptación de la red; la forma final obtenida se muestra en (f).

3. Repetir los pasos 3 y 4 del algoritmo expuesto en la sección 4.1 hasta alcanzar el criterio de paro. No se realiza inserción de neuronas.

Las figuras 5.9 a 5.11 muestran algunas etapas al transformar una superficie en otra. La figura 5.9.a muestra la forma inicial del objeto, el cual es deformado para tomar la forma de la figura 5.9.b; las figuras 5.9.c-f muesran algunas etapas del proceso. Note que la figura 5.9.f luce como 5.9.b, tal como se espera.

La figura 5.10.a muestra la forma inicial, la cual debe ser transformada en la que se muestra en la figura 5.10.b; la figura 5.10.c muestra una etapa durante el proceso, mientras que la figura 5.10.d muestra el resultado obtenido.

En el caso mostrado en la figura 5.11 se tiene un modelo 3D con forma irregular y éste debe ser transformado para tomar la forma de una especie de pera. La figura 5.11.a muestra la forma inicial; la forma esperada esta en la figura 5.11.b; varias etapas del proceso se muestran en 5.11.c-f, siendo 5.11.f el resultado obtenido.

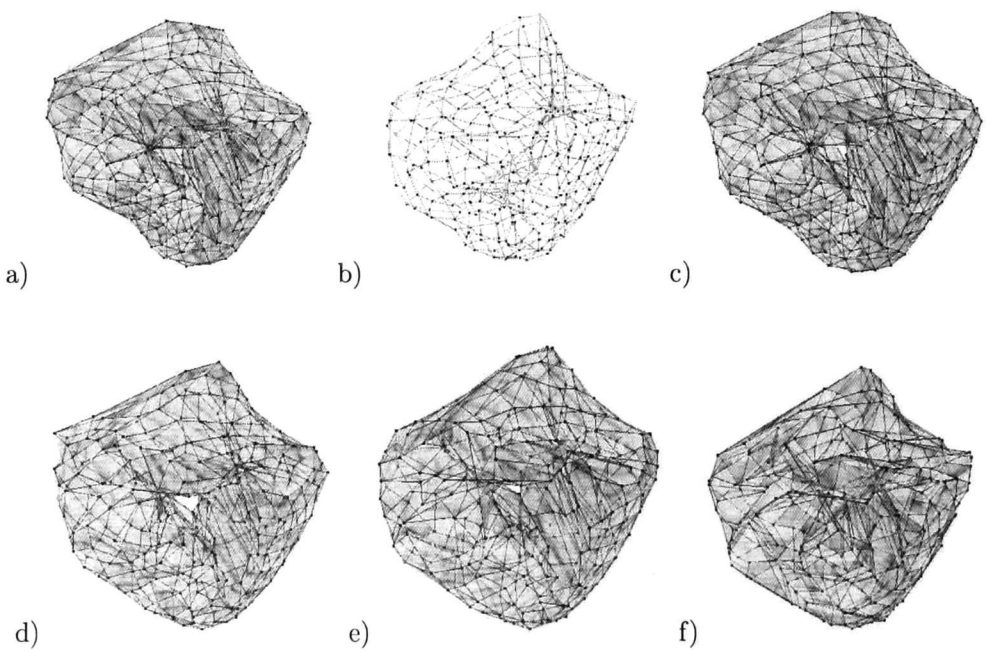


Figura 5.10: La superficie 3D mostrada en (a) se transforma en la mostrada en (b). El inciso (c) muestra una etapa durante la adaptación de la red; la forma final obtenida se muestra en (d).

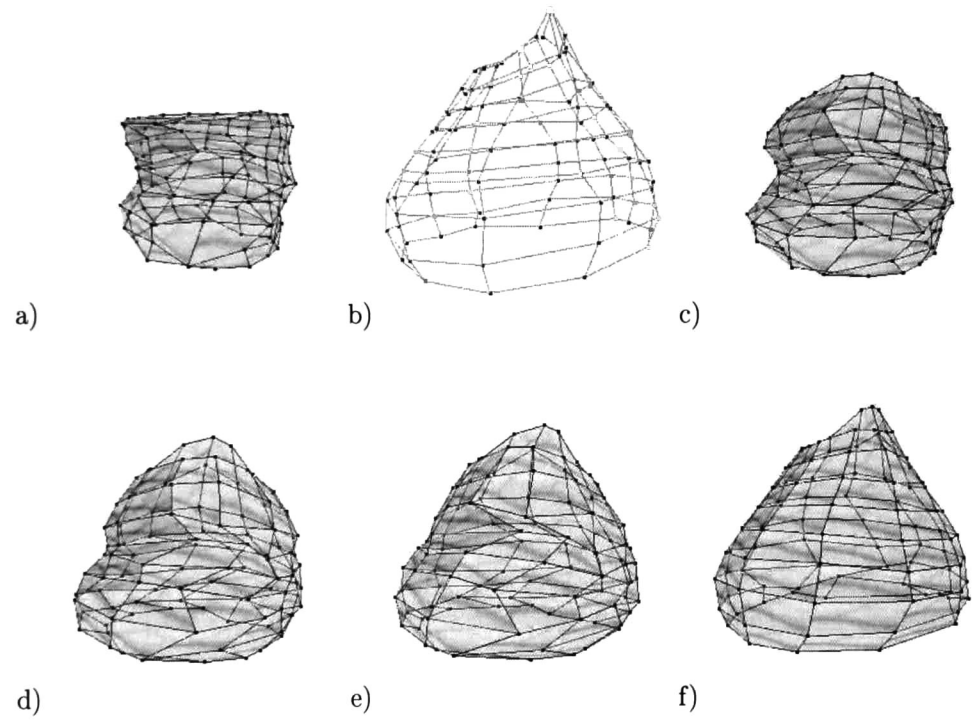


Figura 5.11: La superficie 3D mostrada en (a) se transforma en la mostrada en (b). Los incisos (c) a (f) muestran diferentes etapas durante la adaptación de la red; la forma final obtenida se muestra en (f).

5.3. Alineamiento de modelos basados en esferas

El problema de registro es encontrado frecuentemente en visión computacional y procesamiento de imágenes médicas. Supongamos que tenemos dos conjuntos de puntos y uno de ellos es el resultado de la transformación del otro pero no conocemos tal transformación ni la correspondencia que existe entre los puntos. En tal situación, necesitamos un algoritmo que sea capaz de estimar ambas incógnitas lo mejor posible. Si además la transformación es no rígida, la complejidad se incrementa. En la variedad de algoritmos de registro existentes, encontramos algunos que asumen el conocimiento de una de las incógnitas anteriores y resuelven para la otra, pero también hay algunos que tratan con ambas; entre éstos, dos destacados son el *Iterated Closest Point (ICP)* y el *Thin Plate Spline - Robust Point Matching* [14, 30]. Dado que se ha mostrado [5] la superioridad de TPS-RPM sobre ICP, se utilizará éste para el alineamiento de esferas, adaptándolo para trabajar con estas entidades. Es importante notar que los trabajos anteriores mostraban la aplicación de TPS-RPM con puntos en el espacio 2D y 3D, pero ahora tenemos esferas como puntos en el espacio 5D del AGC.

La notación es como sigue: $\mathbf{V} = \{S_j^I\}, j = 1, 2, ..., k$, es el conjunto inicial del modelo basado en esferas, o conjunto de esferas en tiempo t_1; $\mathbf{X} = \{S_i^F\}, i = 1, 2, ..., n$, es el conjunto esperado, es decir, el conjunto del modelo basado en esferas deformado o en tiempo t_2; $\mathbf{U} = \{S_i^E\}, i = 1, 2, ..., n$, es el conjunto estimado o resultante de la aplicación del algoritmo - el superíndice indica si la esfera S está en el conjunto inicial (I), el esperado (F) o el estimado (E).

Resolver el problema de correspondencia significa encontrar la matriz de correspondencias Z (ecuación (5.6)) que indica cuál esfera en el conjunto $\mathbf{V}$ se transforma en cuál esfera del conjunto $\mathbf{X}$ por medio de una transformación f; es decir que aplicando esta transformación f a cada esfera en el conjunto inicial $\mathbf{V}$ se debería obtener el conjunto estimado $\mathbf{U}$, el cual debería ser igual al conjunto esperado $\mathbf{X}$. Para tomar en cuenta los *outliers*, la matriz Z de correspondencias se extiende como en (5.6). La parte interna de Z (de tamaño $k \times n$) define las correspondencias (si S_j^I corresponde con S_i^F entonces $z_{ji} = 1$, en caso contrario $z_{ji} = 0$). La fila y la columna extra fueron añadidas para tratar con los *outliers*, pero como se menciona en [14], resolver el problema de correspondencias binarias y tratar con los *outliers* puede resultar un problema extremadamente engorroso.

$$
(5.6) \qquad Z = \begin{array}{c|ccccccc}
z_{ji} & S_1^F & S_2^F & S_3^F & S_4^F & \dots & S_n^F & outlier_{n+1}^F \\
S_1^I & 1 & 0 & 0 & 0 & \dots & 0 & 0 \\
S_2^I & 0 & 1 & 0 & 0 & \dots & 0 & \dots \\
S_3^I & 0 & 0 & 1 & 0 & \dots & 0 & \dots \\
\dots & \dots & \dots & \dots & \dots & \dots & \dots & \dots \\
S_k^I & 0 & 0 & 0 & 0 & \dots & 1 & 0 \\
outlier_{k+1}^I & 0 & 0 & 0 & 1 & \dots & 0 &
\end{array}
$$

El algoritmo TPS-RPM usa dos técnicas para resolver el problema de corresponden-

cia:

Asignación suave: la idea básica es permitir a la matriz de correspondencias M tomar valores continuos en el intervalo [0,1]; esta "difusidad" de correspondencias mejora gradualmente durante el proceso de optimización sin dar saltos en el espacio de permutaciones de matrices binarias.

Templado determinista: técnica utilizada para controlar la "difusidad" de las correspondencias en la función de costo introduciendo un parámetro T de temperatura, el cual se reduce en cada etapa del proceso de optimización empezando con un valor T_0 y reduciendo por un factor r hasta alcanzar alguna temperatura final preestablecida.

Usando estas dos técnicas, el problema es minimizar la función de costo:

$$\begin{aligned} E(M,f) &= \sum_{i=1}^{n}\sum_{j=1}^{k} m_{ji}\left|S_i^F - f(S_j^I)\right|^2 + \lambda\left|Lf\right|^2 \\ &\quad + T\sum_{i=1}^{n}\sum_{j=1}^{k} m_{ji}\log m_{ji} - \varsigma\sum_{i=1}^{n}\sum_{j=1}^{k} m_{ji} \end{aligned} \tag{5.7}$$

donde m_{ji} debe satisfacer que $\Sigma_{i=1}^{n+1} m_{ji} = 1$ para $j \in \{1,2,...,k+1\}$; $\Sigma_{j=1}^{k+1} m_{ji} = 1$ para $i \in \{1,2,...,n+1\}$ y $m_{ji} \in [0,1]$. El parámetro λ también se reduce en un esquema de templado $\lambda_i = \lambda_{init} T$. La idea básica en esta heurística es que las transformaciones más globales y rígidas deben ser favorecidas primero con valores grandes de λ.

Para resolver, se siguen los siguientes dos pasos:

Actualización de la correspondencia: para las esferas S_j^I, $j = 1,2,...,k$ y $S_i^F, i = 1,2,...,n$ modificar m_{ji}:

$$m_{ji} = \frac{1}{T} e^{-\frac{(S_i^F - f(S_j^I))^2}{T}} \tag{5.8}$$

para los *outliers* $j = k+1$ e $i = 1,2,...,n$:

$$m_{k+1,i} = \frac{1}{T_0} e^{-\frac{(S_i^F - f(S_{k+1}^I))^2}{T_0}} \tag{5.9}$$

y para los *outliers* $j = 1,2,...,k$ e $i = n+1$:

$$m_{j,n+1} = \frac{1}{T_0} e^{-\frac{(S_{n+1}^F - f(S_j^I))^2}{T_0}} \tag{5.10}$$

Actualización de la transformación: Aquí se necesita actualizar la posición de las esferas (cambiando sus centros) y sus radios. Así que para actualizar la transformación sufrida por sus posiciones usamos

$$\begin{aligned} E_{tps}(D,W) &= \|Y_c - V_c D - \Phi_c W\|^2 + \lambda_1 (W^T \Phi_c W) \\ &\quad + \lambda_2 [D - I]^T [D - I], \end{aligned} \tag{5.11}$$

donde V_c es la matriz formada usando (2.18) en AG como $V_c = \{(S_j^I \wedge E) \cdot E\}, j = 1, 2, ..., k$, Y_c es interpretado como la nueva posición estimada (actual) de los centros de las esferas, Φ_c se forma con $\phi_b = (S_b^I \wedge E)\ E - (S_a^I \wedge E)\ E$; por tanto, Φ_c captura información acerca de la estructura de los conjuntos, y D, W representan las deformaciones rígidas y no rígidas que afectan las posiciones de las esferas, respectivamente, y se obtienen al resolver

$$f((S_a^I \wedge E) \cdot E | D, W) = ((S_a^I \wedge E) \cdot E)D + \phi((S_a^I \wedge E) \cdot E)W. \tag{5.12}$$

Los radios son actualizados con un dilator aplicado a cada esfera, el cual se forma de acuerdo a

$$D_\lambda = e^{\frac{-\log(\lambda) \wedge E}{2}} = e^{-\frac{\log(\rho_{j_{max}}/\rho_{i_{max}}) \wedge E}{2}} \tag{5.13}$$

donde $\rho_{j_{max}}$ y $\rho_{i_{max}}$ son los radios del par de esferas que "*más corresponden*" de acuerdo a cada fila de la matriz M; es decir, los índices i y j de los conjuntos inicial y esperado, respectivamente, donde la correspondiente i-esima fila de M tiene el valor máx(m_{ij}). Los dilatores se aplican a las esferas

$$S'^I_j = D_\lambda S_j^I D_\lambda. \tag{5.14}$$

El proceso de templado ayuda a controlar el proceso alternado de actualización. El parámetro T se inicializa en T_0 y es reducido gradualmente $T_i = T_0 r$, donde r es la razón de templado, hasta alcanzar una temperatura final. Los parámetros λ_1 y λ_2 siguen también un esquema de templado.

Note que con la información codificada en una sola entidad, cada iteración del algoritmo modifica no solamente la posición de cada esfera, sino también su radio y, por tanto, la forma general del objeto modelado con esferas. La siguiente sección presenta algunos ejemplos ilustrativos. Algunos de ellos se diseñaron solo para presentar la silueta de un objeto cuya forma ha sufrido una transformación no rígida, mientras que otros ejemplos muestran volúmenes modelados con esferas de los cuales el último es un tumor extraído de las imágenes CT.

5.3.1. Resultados experimentales

Las figuras 5.12 a 5.16 muestran algunos ejemplos de la aplicación del algoritmo para alinear conjuntos de esferas. La figura 5.12 es el caso más simple que se muestra pero es con fines ilustrativos; consta solamente de dos conjuntos diferentes de esferas relacionadas por una transformación no rígida (no es un modelo de algún objeto volumétrico concreto); en dicha figura se puede observar que el algoritmo obtiene la forma general definida por los conjuntos (de acuerdo a lo esperado). La figura 5.13 es un ejemplo simple pero útil; muestra solamente el contorno de la figura de un pez modelado con esferas con el fin de

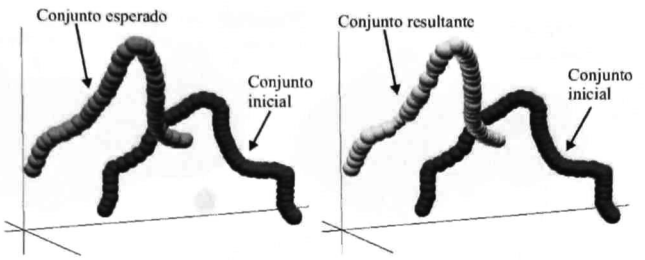


Figura 5.12: Dos conjuntos diferentes de esferas relacionados por una deformación no rígida que deben ser alineados. Note que la forma general es alcanzada.

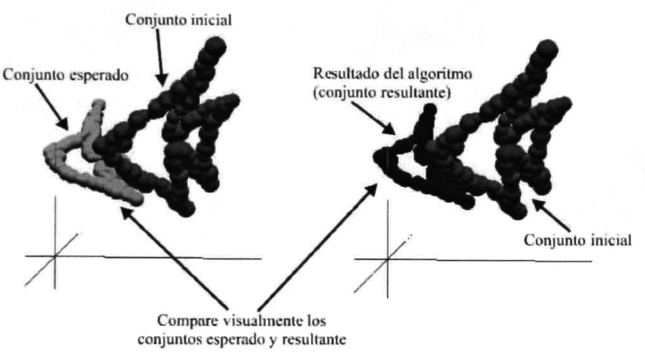


Figura 5.13: La silueta de un pez modelada con esferas para mostrar visualmente los resultados del algoritmo

que el lector pueda visualizar mejor la transformación y los resultados. Por otro lado, las figuras 5.14 a 5.16 muestran ejemplos con volúmenes completos modelados con esferas.

En la figura 5.13 se fijaron dos radios diferentes para el conjunto inicial de 90 esferas, y se colocaron las diferentes esferas alternando una con radio ρ_1 y otra con radio ρ_2; el conjunto esperado (o conjunto deformado) es una deformación no rígida de la silueta del pez y también se le han intercambiado los tamaños de los radios a las esferas. Por tanto, para este ejemplo el algoritmo debe ajustar la forma de la silueta y también los radios de las esferas. Los resultados se pueden observar y comparar observando la figura 5.13.

El ejemplo de la figura 5.14 muestra un modelo 3D basado en 500 esferas; en esta figura se muestra el modelo original (conjunto inicial) y el modelo deformado (conjunto esperado u objetivo). También se muestra el resultado del algoritmo y se ha incluido una capa de esferas de cada uno de los conjuntos para que se pueda visualizar mejor la deformación.

Las figuras 5.15 y 5.16 muestran la aplicación del algoritmo en presencia de *outliers*. La figura 5.15 tiene un modelo inicial con 159 esferas y un modelo esperado con 143, mientras que la figura 5.16 muestra un conjunto inicial con 309 esferas y uno esperado

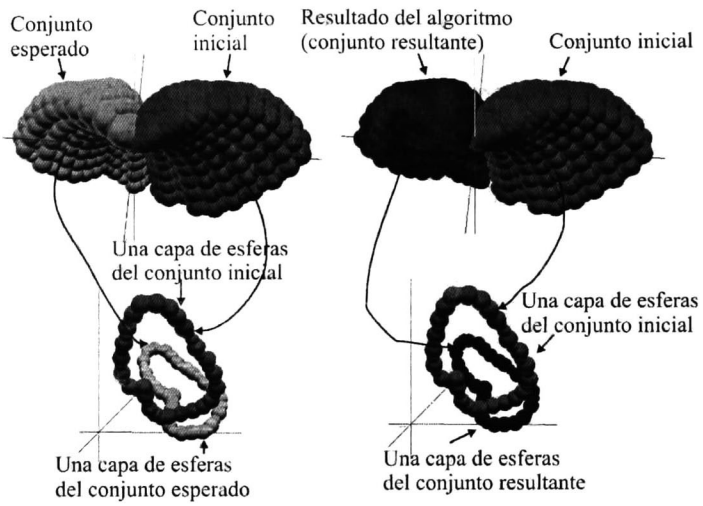


Figura 5.14: Un objeto 3D modelado con esferas que muestra que el algoritmo ajusta tanto la posición de las esferas como sus radios, resultando en un ajuste general del volumen.

con 280. Estas figuras también contienen una gráfica del error medido como la distancia promedio entre las posiciones del conjunto esperado y el resultante de cada iteración del proceso de optimización (este error se calcula de acuerdo a (5.15)). Las imágenes de los conjuntos inicial, esperado y resultante se presentan separadas porque no es posible visualizarlas juntas ya que se sobrelapan; sin embargo, el lector puede usar los ejes de coordenadas para relacionar las imágenes.

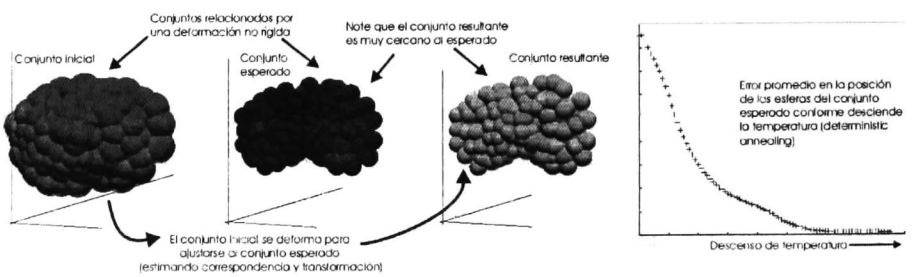


Figura 5.15: Un ejemplo con 159 esferas en el conjunto inicial y 143 en el conjunto esperado (se tienen *outliers*).

Cuando detectamos un tumor cerebral (como el que se muestra en la figura 5.17.a), el primer paso es segmentarlo (ver figura 5.17.b) y luego construir un modelo del mismo; este modelo puede ser hecho con un conjunto de esferas (como en la figura 5.17.c). La figura 5.18 muestra la forma original del tumor en el tiempo t_1 y el modelo bajo la deformación no rígida (inducida artificialmente). Finalmente es posible ver en la figura el resultado del alineamiento o actualización del modelo original. Note que aunque la forma cambia y se

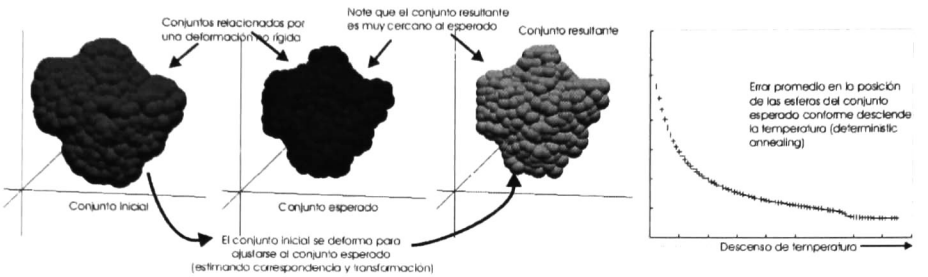


Figura 5.16: Un ejemplo con 309 esferas en el conjunto inicial y 280 en el conjunto esperado (tenemos *outliers*).

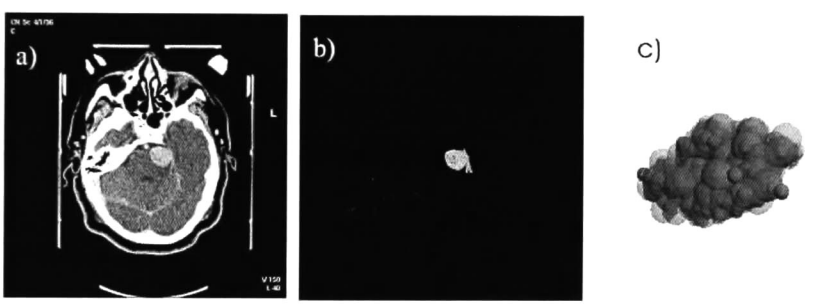


Figura 5.17: Extracción del tumor de las imágenes TC y el modelo 3D basado en esferas. a)Imagen TC del cerebro con un tumor; b) Tumor segmentado de la imagen TC; c) Acercamiento del modelo 3D basado en esferas del tumor

colocó intencionalmente lejos del original, el algoritmo obtiene buenos resultados.

La Tabla 5.3 muestra el error promedio en la posición de los centros de las esferas tomados entre las esferas de los conjuntos resultante y esperado. Esta distancia promedio se calcula en voxels y tomando en cuenta solo las esferas en correspondencia

$$\epsilon = \frac{\sum_{i,j} \sqrt{c(i,j)(S_i^F - S_i^E)^2}}{N} \tag{5.15}$$

donde N es el número de pares de esferas en correspondencia y $c(i,j) = 1$, si S_i^F corresponde a S_i^E, o bien $c(i,j) = 0$ si estas esferas no son correspondientes.

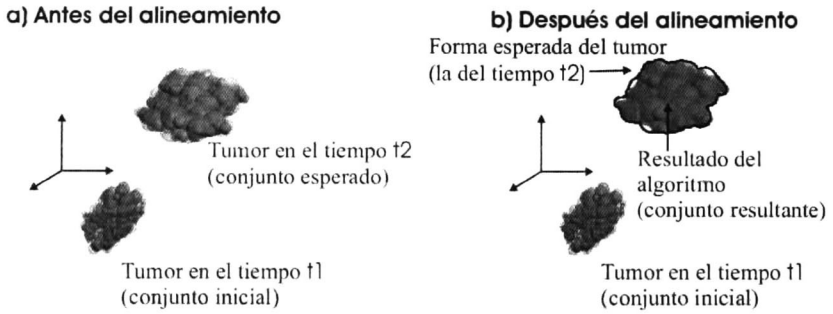


Figura 5.18: El modelo de un tumor basado en esferas; este tumor se deforma en el tiempo (deformación simulada para el experimento) y el modelo original utilizado en la planeación de la cirugía es actualizado con el algoritmo expuesto.

Tabla 5.3: Errores promedio en la posición de los centros de las esferas de los conjuntos esperado y resultante, calculado según (5.15)

Ejemplo de la Fig. No.	Núm. de esferas en el conj. inicial	Núm. de esferas en el conj. esperado	Error promedio en voxels calculado con (5.15)
Figura 5.12	60	60	12.86
Figura 5.13	98	98	0.72
Figura 5.14	500	500	2.80
Figura 5.15	159	143	18.04
Figura 5.16	309	280	15.63
Figura 5.18	340	340	4.65

Capítulo 6

Conclusiones y trabajo a futuro

En este trabajo de tesis se han tratado los problemas de segmentación de imágenes, aproximación de superficies tridimensionales, modelado tridimensional basado en esferas y el proceso de registro o *registración* de equipo médico y de modelos basados en esferas. En las siguientes secciones presentamos los principales resultados, conclusiones y trabajo a futuro de éstos problemas.

6.1. Segmentación

6.1.1. Principales resultados

En el capítulo 3 se han expuesto dos propuestas para la segmentación de imágenes médicas, con especial atención a la segmentación de tumores.

- En la primera se propone utilizar un conjunto de *descriptores de textura* para crear un conjunto de "imágenes de textura" con las cuales caracterizar cada pixel de la imagen por medio de un vector k-dimensional; es decir, con los descriptores de textura se construye un vector de valores de textura para cada pixel de la imagen. Una vez hecho esto, el usuario selecciona el objeto que le interesa segmentar, se toma el vector con los valores de textura (que hemos llamado "vector característico") de ese pixel y se buscan todos los que tengan ese mismo vector o uno similiar con un elemento diferente como máximo. Entonces se toman los pixeles resultantes como semillas en un esquema de crecimiento de regiones que integra la información de las texturas y de los contornos.

- La segunda propuesta utiliza redes neuronales auto-organizadas integrando la información del campo vectorial llamado *Generalized Gradient Vector Flow* (GGVF). Esta información se utliza para seleccionar las entradas a la red neuronal y como parámetro que guía en el proceso de aprendizaje de la red. Durante el aprendizaje

de la red, se determina un conjunto de transformaciones expresadas en el marco del álgebra geométrica conformal, las cuales mueven un punto a posiciones sobre el contorno del objeto de interés, lo cual junto con las relaciones de vecindad entre neuronas definen la forma del objeto (topología de la red). Los experimentos hechos con varias imágenes médicas se presentaron para mostrar que el algoritmo es útil para las tareas de segmentación, lo cual resulta especialmente visible en el ejemplo de las figuras 3.20 y 3.22, donde el objeto a ser segmentado no tiene contornos bien definidos, pero los resultados obtenidos con el método propuesto son muy buenos. Así mismo se realizaron comparaciones entre dos algoritmos, integrando y sin integrar la información del gradiente, cuyos resultados se resumen en tablas de comparación cuantitativas en el capítulo 3.

De estas dos estrategias, la segunda es más robusta que la primera en situaciones donde las texturas de los diferentes tejidos son parecidas, lo cual lleva a que la primera estrategia segmente como parte de un objeto algo que no pertenece a él; sin embargo, la primera estrategia es más global ya que se aplica sobre toda la imagen, mientras que la segunda se aplica a una región de interés para que la extracción de las entradas a la red pueda ser automática con la información brindada por el GGVF.

6.1.2. Trabajo a futuro

Para poder aplicar globalmente en la imgen la propuesta basada en redes neuronales, podría modificarse ésta de forma que utilice la información gradiente para crear tantas redes como sea necesario al ir evolucionando; es decir, que se creen diferentes redes para los diferentes objetos presentes en la imagen, donde cada una de ellas adaptará su topología para aproximar el contorno de éstos.

6.2. Modelado

6.2.1. Principales resultados

El capítulo 4 estuvo dedicado a la representación volumétrica de datos y se mostró como obtener representaciones usando una versión en 3D de la propuesta hecha para segmentación de imágenes integrando información del gradiente de la imagen en un esquema de redes neuronales auto-organizadas. Así mismo se presentó una propuesta para crear modelos basados en esferas. El método presentado esta basado en las ideas del algoritmo *marching cubes* pero utiliza esferas y no esta diseñado para propósitos de renderizado, sino para reducir el número de esferas con respecto al algoritmo Unión de Esferas y reducir el tiempo en el registro de modelos de este tipo.

6.2.2. Trabajo a futuro

Aunque en [27] se afirma que los modelos basados en esferas pueden ayudar en el tratamiento de ciertas afecciones al determinar la cantidad de radiación a aplicar en una determinada zona, basándose en el tamaño de la esfera que cubre esa región, en este trabajo no se incluye una análisis de la parte médica sobre ese punto. Para ello sería necesario simular un tratamiento con radiación en estos modelos, así como la forma en que afectan al tejido circundante; simular las características fisiológicas de los tejidos, etc.

6.3. Registro de modelos

6.3.1. Principales resultados

El registro (alineamiento) de modelos 3D se trató el capítulo 5, destacando que la propuesta basada en redes neuronales presentada en el capítulo 4 también se utiliza con éxito en este caso. Además se puso atención especial al registro no rígido de modelos basados en esferas sin conocimiento a priori sobre las correspondencias o transformaciones y tratando con outliers. Para esto se siguió el mismo método que el algoritmo TPS-RPM pero se modificó para trabajar en el marco del AGC. Las esferas son representadas en AGC mediante un vector de 5 dimensiones, así evitamos el uso de ecuaciones como (6.1) o (6.2) y el uso de ecuaciones cuadráticas o la búsqueda de los parámetros adecuados para el alineamiento de los modelos.

$$(x - x_0)^2 + (y - y_0)^2 + (z - z_0)^2 = r^2 \tag{6.1}$$

$$x = x_0 + r\cos\theta\sin\phi, \quad y = y_0 + r\sin\theta\sin\phi, \quad z = z_0 + r\cos\phi \tag{6.2}$$

6.3.2. Trabajo a futuro

Aunque se mostró la manera de registrar los diferentes equipos y modelos involucrados en la cirugía (sección 5.1), no se desarrolló un algoritmo para la fusión de la información brindada por los diferentes medios como endoscopía o neurosonografía con el modelo virtual. Toda esta información tiene que presentarse al médico de manera natural, es decir, que no requiera entrenamiento especial para su interpretación, sino que se adapte a lo que se conoce como *ergonomía de los sistemas médicos* [28].

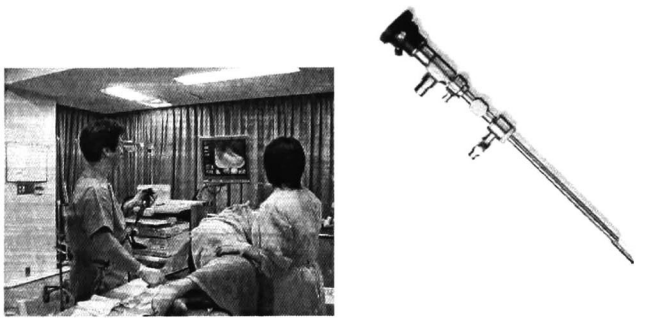


Figura 6.1: Equipo de endoscopía utilizado en procedimientos de mínima invasión.

6.4. Uso de Endoscopía y Neurosonografía

En los procedimientos quirúrgicos conocidos como *mínimamente invasivos*, los endoscopios (ver figura 6.1) se introducen al paciente a través de orificios naturales o pequeñas incisiones que no superan los 2cm. El endoscopio permite accesar a la región de interés y transmite las imágenes de video a un monitor para que el médico realice la cirugía. Estos procedimientos reducen la morbilidad respecto a los procedimientos tradicionales debido a que con estas pequeñas insiciones reducen el daño a tejidos sanos y reducen el dolor y el tiempo de de convalecencia del paciente.

Como parte fundamental del proyecto Salud-Conacyt 49, debe incluirse el uso de endoscopía y neurosonografía. Para ello habrá que fusionar la información de ambos medios con el modelo virtual. El primer paso en la fusión de endoscopía ya está dado, como se explicó en la sección 5.1. Falta registrar la imagen con el modelo, combinando las ventajas en información que brinda cada uno; es decir, aunque la imagen endoscópica muestra exactamente lo que ocurre dentro del paciente (el endoscopio actúa como los ojos del cirujano), tiene la limitación de no poder ver más allá del tejido que esta frente al lente del endoscopio, pero el modelo virtual, al estar registrado, puede mostrar lo que hay detrás del tejido opaco que muestra la imagen de video.

La integración de sonografía en neurocirugía es un campo naciente que esta siendo impulsado por una pequeña parte de la comunidad científica médica, quien cree que es de gran ayuda, especialmente por las nuevas tecnologías que permiten crear modelos tridimensionales con esta técnica en tiempo real. Por tanto, la integración de la neurosonografía al proyecto es un reto con mucho potencial para presentar nuevos algoritmos y realizar publicaciones internacionales.

Apéndice A

Imágenes Médicas

A.1. Rayos X

Los rayos X fueron descubiertos en 1895 por Wilhelm Rontgen y actualmente son los más comúnmente utilizados y de importancia tecnológica de entre las técnicas de radiación o evaluación no destructiva. Ésto es debido a la naturaleza penetrante de los rayos X y su fácil interpretación. Los métodos de radiación tienen la ventaja de que su interpretación es intuitiva en alguna forma, aunque el analizar e interpretar las imágenes requiere habilidad y experiencia. Sin embargo, el usuario casual de los servicios de imágenes de radiación puede hasta cierto punto reconocer fallas y defectos, ciertamente sin lograr una interpretación de experto. Un ejemplo de una imagen de rayos X se puede ver en la figura A.1

La desventaja de éstos métodos es que puede ser peligroso aplicarlos porque usan radiación y, a menudo, altos voltajes, lo cual crea preocupación por la seguridad personal, obliga a medidas reguladoras de seguridad, como son el uso de materiales pesados como

Figura A.1: Ejemplo de la imagen obtenida con Rayos X

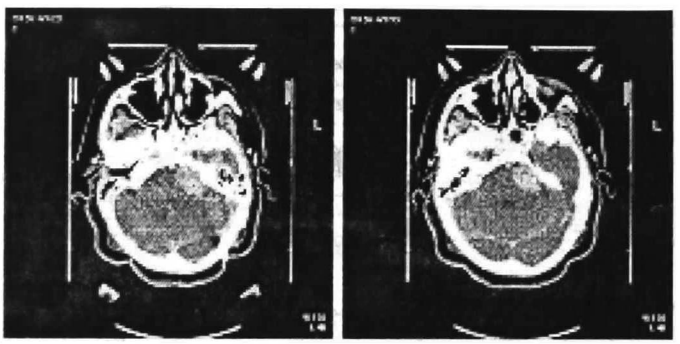


Figura A.2: Ejemplo de la imagen de Tomografía Computarizada

escudo y el requerimiento de operadores capacitados.

A.2. Tomografía Computarizada

A principios de los 70's se desarrolló la tomografía computarizada (TC) como una herramienta para el diagnóstico médico por medio de imágenes. Sus inventores, Cormack y Hounsfield fueron galardonados con el Premio Nobel en medicina en 1979. Al principio los sistemas de TC estaban basados en arreglos de detectores en línea y eran de uso limitado, pero a mitad de los 80's se expandió su uso para propósitos industriales y científicos, por lo que los industriales comenzaron a utilizar detectores en arreglos por área.

La tomografía computarizada médica opera bajo ciertas restricciones, incluyendo la cantidad de radiación o dosis por escaneo, y en ocasiones es difícil mantener sin movimiento al paciente o a los órganos. Estas limitaciones en la cantidad de radiación en la TC llevan a limitaciones en el desempeño de la imagen y en su resolución espacial. Para obtener imágenes de mejor calidad en tejidos duros o huesos, se requieren mayores dosis de energía. Alternativamente, para obtener mayor resolución espacial se necesita más alineamiento, detectores más pequeños o más adquisición de datos (más escaneos). Todas estas circunstancias resultan en mayores dosis de rayos X que se aplican al paciente o mayor incomodidad para él. Un ejemplo de imágenes tomográficas se puede ver en la figura A.2.

A.3. Resonancia Magnética

Desde su introducción en aplicaciones clínicas a principios de los 80's, la técnica de resonancia magnética (RM) o tomografía por resonancia magnética (TRM) se ha desarrollado como la modalidad preferida en muchas situaciones de diagnóstico clínico debido a su contraste sin paralelos de tejidos suaves, combinado con su alta resolución espacial y su

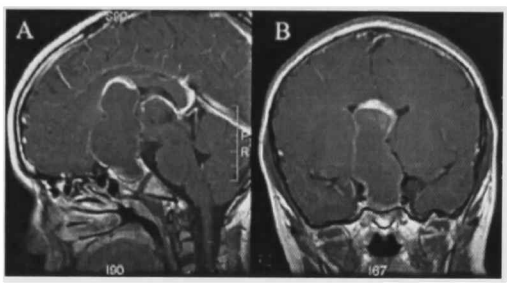


Figura A.3: Ejemplo de la imagen obtenida con Resonancia Magnética

capacidad para generar imágenes en orientaciones arbitrarias o incluso de volúmenes enteros. Además se empieza a explotar su potencial para desplegar vasos sanguíneos, mapear funciones del cerebro y analizar el metabolismo.

La resonancia magnética es el fenómeno de acuerdo al cual las partículas con un momento angular y magnético particular responen ante un campo magnético, ya sea absorbiendo o emitiendo energía electromagnética. Este efecto se llama Resonancia Electrón-Spin (RES) o Resonancia Paramagnética de Electrones (RPE) para electrones en átomos, moléculas y cristales, mientras que es llamado Resonancia Magnética Nuclear (RMN) para nucleidos. El efecto RES fue descubierto en 1944 por el científico ruso Zavoisky. La RMN fue observada independientemente en 1945 por Bloch et al. en la Universidad de Stanford en California, y por Purcell et al. en Cambridge, MA. y el Premio Nobel de física se otorgó en 1952 a estos dos grupos. Véase ejemplo de imagenes de resonancia magnética en la figura A.3

Actualmente, la máquina con la que se realiza la resonancia magnética es grande y cilíndrica (con forma de tubo) y crea un fuerte campo magnético alrededor del paciente. El campo magnético, junto con una radiofrecuencia, altera el alineamiento natural de los átomos de hidrógeno en el organismo. Luego, se utiliza una computadora para formar una imagen bidimensional (2D) de una estructura u órgano del cuerpo en función de la actividad de los átomos de hidrógeno. La resonancia magnética no utiliza radiación, a diferencia de las radiografías o la tomografía computarizada. Se crea un campo magnético y se envían pulsos de ondas de radio desde un escáner. Las ondas de radio golpean el núcleo de los átomos del cuerpo, desplazándolos fuera de su posición normal. Mientras los núcleos se vuelven a alinear en la posición correcta, envían señales de radio. Estas señales son recibidas por una computadora que las analiza y las convierte en una imagen de la parte del cuerpo que está siendo examinada. Algunos estudios de RMN usan algún medio de contraste oral o inyectado para obtener imágenes más claras.

Para detalles técnicos sobre las técnicas de imagenología presentadas en este apéndice, véase [32].

Apéndice B

Algoritmos y definiciones

B.1. Detector de bordes Canny

Existen diversos algoritmos para detección de bordes en las imágenes, pero el algoritmo Canny obtiene generalmente muy buenos resultados. Cuando hablamos de "borde" en una imagen, normalmente nos referimos a un lugar donde hay un "pico" distintivo o significativo en el gradiente de la imagen. Claro que la noción de "pico" en una imagen dependerá de la resolución de ésta y del tamaño de la ventana que se analiza, ya que lo que pudiera parecer algo suave en una pequeña región de una imagen de alta resolución, pudiera parecer una discontinuidad en una región más grande de una imagen submuestreada. Por tanto, aquí diremos que un pixel está en el borde solamente si la norma del gradiente $||\nabla I||$ tiene un máximo local comparado con sus pixeles vecinos.

El algoritmo del detector de bordes Canny tiene los siguientes pasos para determinar qué pixeles de una imagen $I(x, y)$ estan en un borde:

- Fijar un umbral $\tau > 0$ y una desviación estándar $\sigma > 0$ para la función gausiana g_σ usadas para derivar el filtro.
- Calcular el vector gradiente $\nabla I(x, y) = [I_x \;\; I_y]^T$ para cada pixel de la imagen
- Si $||\nabla I(x, y)||^2 = \nabla I^T \nabla I$ es un máximo local y sobrepasa un determinado umbral τ, entonces el pixel (x, y) se marca como un pixel de borde

Conceptualmente, el gradiente de la imagen $\nabla I(x, y) = [I_x(x, y) \;\; I_y(x, y)]^T \in \Re^2$ se define como el vector cuyos componentes individuales estan dados por las dos derivadas parciales

$$I_x(x, y) = \frac{\partial I}{\partial x}(x, y) \tag{B.1}$$

$$I_y(x, y) = \frac{\partial I}{\partial y}(x, y) \tag{B.2}$$

Normalmente solo se escribe $\nabla I = [I_x \;\; I_y]^T$ para simplificar la notación. Las derivadas parciales normalmente se implementan por diferencias finitas.

B.2. Operaciones básicas en AG

En esta sección se introducen algunos conceptos básicos que son útiles en algunas secciones del documento, sobre todo aquellas donde se explica cómo realizar transformaciones (rotaciones, traslaciones) en álgebras geométricas. Se pueden consultar mayores detalles en [4].

Inversión: se dice que un elemento M en $G_{p,q,r}$ es invertible si existe un elemento N en $G_{p,q,r}$ tal que $MN = NM = 1$. El elemento N es único en caso de existir y es llamado el *inverso* de M.

r-vector: es la combinación lineal de r-blades (recuerde que un r-blade, también llamado blade de grado r, es el producto exterior (wedge) de r vectores diferentes y es denotado por $\langle M \rangle_r$).

Reversión: se define como

$$\langle M^\dagger \rangle_i = (-1)^{\frac{i(i-1)}{2}} \langle M \rangle_i \tag{B.3}$$

es decir, la reversión es un anti-automorfismo (un anti-automorfismo es un mapeo lineal que invierte el orden de los productos geométricos y no abandona el espacio de partida).

Involución de grado: se define como

$$\langle M^\star \rangle_i = (-1)^i \langle M \rangle_i \tag{B.4}$$

Multivector par: un multivector M es par (o tiene paridad par) si $M^\star = M$

Multivector impar: un multivector M es impar (o tiene paridad impar) si $M^\star = -M$

Magnitud de un multivector: se define como

$$|M| = \sqrt{\sum_{i=0}^{n} |\langle M \rangle_i|^2} \tag{B.5}$$

donde

$$|\langle M \rangle_i| = \sqrt{|\langle M \rangle_i \cdot \langle M \rangle_i|} \tag{B.6}$$

(para realizar el producto punto entre dos blades se utiliza (B.7)).

El producto de un r-blade con un s-blade se define recursivamente mediante

$$(B.7)\quad (a_1 \wedge ... \wedge a_r) \cdot (b_1 \wedge ... \wedge b_s) = \begin{cases} ((a_1 \wedge ... \wedge a_r) \cdot b_1) \cdot (b_2 \wedge ... \wedge b_s) & si\ r \geq s \\ (a_1 \wedge ... \wedge a_{r-1}) \cdot (a_r \cdot (b_1 \wedge ... \wedge b_s)) & si\ r < s \end{cases}$$

y

$$(B.8)\quad (a_1 \wedge ... \wedge a_r) \cdot b_1 = \sum_{i=1}^{r} (-1)^{r-i} a_1 \wedge ... \wedge a_{i-1} \wedge (a_i \cdot b_1) \wedge a_{i+1} \wedge ... \wedge a_r$$

$$(B.9)\quad a_1 \cdot (b_1 \wedge ... \wedge b_s) = \sum_{i=1}^{s} (-1)^{i-1} b_1 \wedge ... \wedge b_{i-1} \wedge (a_r \cdot b_i) \wedge b_{i+1} \wedge ... \wedge b_s$$

B.3. Los espacios OPNS e IPNS

En el álgebra geométrica, los blades tienen una interpretación geométrica que se basa en su interpretación como subespacios lineales. Por ejemplo, dado un vector $a \in \mathbb{R}^n$, podemos definir una función $\mathcal{O}_a$ como

$$\mathcal{O}_a : x \in \mathbb{R}^n \rightarrow x \wedge a \in G_n$$

donde G_n es el álgebra de Cliford sobre el espacio $\mathbb{R}^n$ El *kernel* de esta función es el conjunto de vectores en $\mathbb{R}^n$ tales que $\mathcal{O}_a$ mapea a cero. Este *kernel* se llama *Outer Product Null Space (OPNS)* o Espacio Nulo del Producto Exterior de a y se denota por $\mathcal{NO}(a)$. De la definición del producto wedge, sabemos que $x \wedge a = 0$ cuando x y a son linealmente dependientes, por tanto, $\mathcal{NO}(a)$ se puede expresar en términos de a como

$$\mathcal{NO}(a) = \{\alpha a : \alpha \in \mathbb{R}\}$$

De manera general, el OPNS de algún k-blade $\langle A \rangle_k \in G_n$ es un subespacio lineal k-dimensional de $\mathbb{R}^n$:

$$(B.10)\quad \mathcal{NO}(\langle A \rangle_k) := \{x \in \mathbb{R}^n : x \wedge \langle A \rangle_k = 0\}$$

El *Inner Product Null Space (IPNS)* o Espacio Nulo del Producto Interior de un balde $\langle A \rangle_k \in G_n$, denotado por $\mathcal{NI}(\langle A \rangle_k)$, es el *kernel* de la función $\mathcal{I}_{\langle A \rangle_k}$ definida como

$$(B.11)\quad \mathcal{I}_{\langle A \rangle_k} : x \in \mathbb{R}^n \rightarrow x \cdot \mathcal{I}_{\langle A \rangle_k} \in G_n$$

así, el *kernel* es

$$(B.12)\quad \mathcal{NI}(\langle A \rangle_k) := \{x \in \mathbb{R}^n : \mathcal{I}_{\langle A \rangle_k}(x) = 0 \in G_n\}$$

Por ejemplo, consideremos un vector $a \in \mathbb{R}^3$, entonces $\mathcal{NI}(a)$ esta dado por

$$\mathcal{NI}(a) := \{x \in \mathbb{R}^3 : x \cdot a = 0\}$$

es decir, todos los vectores que son perpendiculares al vector a, pertenecen a su IPNS. La representación de una entidad expresada en IPNS puede ser obtenida desde su expresión en OPNS al multiplicar ésta última por el pseudoescalar del álgebra en la que se trabaja; por tal motivo en ocasiones se hace referencia a la representación OPNS como representación dual (ver tabla 2.1).

B.4. Representación con matriz del producto cruz

Dados dos vectores u y v, su producto cruz es un vector w perpendicular a ambos con coordenadas dadas por

$$u \times v = \begin{bmatrix} u_2v_3 - u_3v_2 \\ u_3v_1 - u_1v_3 \\ u_1v_2 - u_2v_1 \end{bmatrix} \tag{B.13}$$

Si dejamos u fijo, el producto cruz se puede representar por un mapeo de $\mathbb{R}^3 \rightarrow \mathbb{R}^3$: $u \rightarrow u \times v$, el cual es lineal en v y por tanto se puede representar por una matriz. Esta matriz esta dada por

$$\hat{u} = \begin{bmatrix} 0 & -u_3 & u_2 \\ u_3 & 0 & -u_1 \\ -u_2 & u_1 & 0 \end{bmatrix} \tag{B.14}$$

Por tanto, podemos escribir $u \times v = \hat{u}v$. Algunos autores denotan la matriz $\hat{u}$ como $u_\times$ o $[u]_\times$. Note que $\hat{u}$ es una matriz 3×3 de las llamadas *skew symetric*, es decir que cumple con la propiedad $\hat{u}^T = -\hat{u}$.

Apéndice C

Formación de la imagen y calibración de cámaras

C.1. Formación de la imagen

El proceso de formación de la imagen y la corrección de la distorsión existente en ella se describe brevemente a continuación, aunque pueden consultarse amplios detalles en [36].

Suponiendo que tenemos una cámara perspectiva ideal, un punto del espacio 3D, $P = [X \;\; Y \;\; Z]^T$, en el sistema de coordenadas de la cámara se proyecta en el plano de la imagen en el punto $p = [x \;\; y]^T$

$$p = \begin{bmatrix} x \\ y \end{bmatrix} = \frac{f}{Z} \begin{bmatrix} X \\ Y \end{bmatrix} \tag{C.1}$$

donde f es la distancia focal (ver figura C.1). En coordenadas homogéneas, esta relación se puede escribir como

$$Z \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{C.2}$$

Puesto que la coordenada Z (profundidad del punto p) normalmente es desconocida, podemos escribirla simplemente como un escalar positivo $\lambda \in \mathcal{R}$

$$\lambda p = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} P \tag{C.3}$$

donde $P = [X \;\; Y \;\; Z \;\; 1]^T$ y $p = [x \;\; y \;\; 1]^T$ están ahora en su representación homogénea.

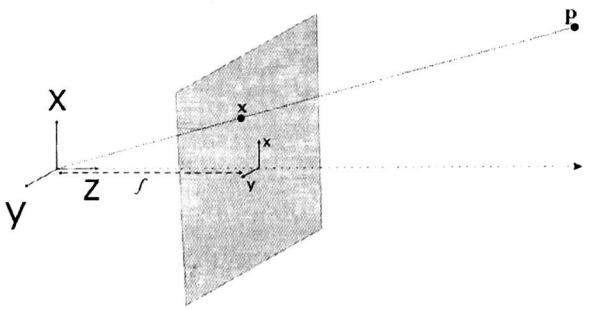


Figura C.1: Un punto P del espacio 3D es proyectado en la imagen en el punto p

Si el punto que consideramos está en el sistema de coordenadas del mundo, $P_0 = [X_0 \ Y_0 \ Z_0 \ 1]^T$, necesitamos hacer una transformación rígida para llevarlo al sistema de coordenadas de la cámara

$$\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{C.4}$$

De esta forma, el modelo geométrico completo de la formación de la imagen en una cámara ideal esta dado por

$$\lambda \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{C.5}$$

C.1.1. Cámara con parámetros intrínsecos

En la práctica, cuando se obtienen imágenes con una cámara, las medidas se obtienen en términos de pixeles (i, j), con el origen del sistema de coordenadas de la imagen en la esquina superior izquierda de la misma. Para poder utilizar el modelo dado en (C.5), se debe especificar la relación entre el sistema de coordenadas del plano de la retina y el arreglo de pixeles. El primer paso es especificar las unidades de los ejes x e y: si (x, y) están especificados en términos de unidades y (x_s, y_s) son versiones escaladas que corresponden a las coordenadas de los pixels, entonces la transformación la podemos escribir

$$\begin{bmatrix} x_s \\ y_s \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix} \tag{C.6}$$

Cuando $s_x = s_y$, el pixel es cuadrado. Ahora bien, x_s y y_s están dados con respecto al *punto principal* (donde el eje Z interseca al plano de la imagen). Dado el hecho de que

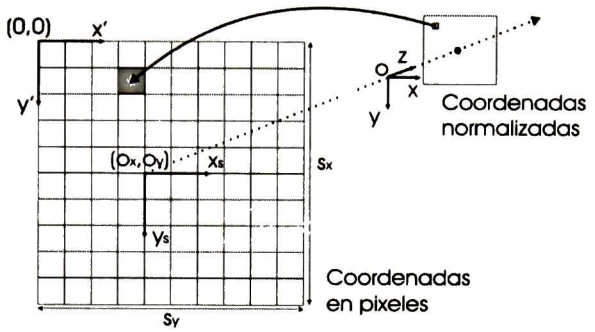


Figura C.2: Transformación de coordenadas normalizadas a coordenadas en pixels.

las coordenadas en pixeles están con respecto a la esquina superior izquierda, se debe necesariamente trasladar el origen a esta esquina (véase figura C.2)

$$x' = x_s + o_x \tag{C.7}$$

$$y' = y_s + o_y \tag{C.8}$$

donde (o_x, o_y) son las coordenadas (en pixeles) del punto principal con respecto al sistema de coordenadas de la imagen. Lo anterior lo podemos representar en coordenadas homogéneas como

$$p' = \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{C.9}$$

En caso de que los pixeles no sean rectangulares, se puede añadir otro factor s_θ llamado *skew factor* proporcional a $\cot(\theta)$, donde θ es el ángulo entre los ejes x_s y y_s.

Al combinar el modelo de la sección anterior con estas transformaciones, se obtiene un modelo más realista de la formación de la imagen

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & s_\theta & o_x \\ 0 & s_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} \tag{C.10}$$

si el punto $P = [X \;\; Y \;\; Z \;\; 1]^T$ esta referido al sistema de coordenadas de la cámara; o bien

$$\lambda \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} fs_x & fs_\theta & o_x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \begin{bmatrix} X_0 \\ Y_0 \\ Z_0 \\ 1 \end{bmatrix} \tag{C.11}$$

cuando el punto $P_0 = [X_0 \quad Y_0 \quad Z_0 \quad 1]^T$ esta en el sistema de coordenadas del mundo. Normalmente a la matriz

$$K = \begin{bmatrix} fs_x & fs_\theta & x \\ 0 & fs_y & o_y \\ 0 & 0 & 1 \end{bmatrix} \tag{C.12}$$

se le conoce como *matriz de parámetros intrínsecos*, mientras que a la matriz

$$\Pi_0 = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \tag{C.13}$$

se le llama *matriz de proyección ideal*, y la matriz

$$G = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix} \tag{C.14}$$

se conoce normalmente como *matriz de parámetros extrínsecos.*

C.1.2. Distorsión Radial

Además de la distorsión dada por los parámetros de la matriz K (parámetros intrínsecos), se pueden observar distorsiones significativas en direcciones radiales. El modelo más simple y efectivo de tal distorsión es

$$x = x_d(1 + a_1 r^2 + a_2 r^4) \tag{C.15}$$

$$y = y_d(1 + a_1 r^2 + a_2 r^4) \tag{C.16}$$

donde (x_d, y_d) son las coordenadas de los puntos distorsionados, $r^2 = x_d^2 + y_d^2$, y a_1, a_2 son parámetros que modelan la cantidad de distorsión.

Otro modelo más general esta dado por

$$\mathbf{x} = c + f(r)(\mathbf{x_d} - c) \tag{C.17}$$

$$f(r) = 1 + a_1 r + a_2 r^2 + a_3 r^3 + a_4 r^4 \tag{C.18}$$

donde $\mathbf{x_d} = (x_d, y_d)$ son las coordenadas de la imagen distorsionada, $r^2 = ||\mathbf{x_d} - c||^2$, $c = (c_x, c_y)$ es el centro de distorsión (no necesariamente coincidente con el centro de la imagen), y $f(r)$ es el factor de corrección de la distorsión.

C.2. Calibración de cámaras

Para encontrar la matriz de parámetros intrínsecos K de la cámara, Zhang [37] propuso una técnica que requiere solamente que la cámara observe un patrón planar en

al menos dos orientaciones diferentes (una especie de tablero de ajedrez). A continuación presentamos una breve descripción de este método.

Sin pérdida de generalidad se asume que el plano esta en $Z = 0$ y siendo K la matriz de parámetros intrínsecos, podemos representar la proyección del punto $[X \;\; Y \;\; 0 \;\; 1]^T$ a la imagen como

$$\lambda \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = K \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} \tag{C.19}$$

$$= K \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix} \tag{C.20}$$

Por tanto, el punto $P = [X \;\; Y \;\; 1]^T$ (representado así porque Z siempre es cero) y su imagen $p = [u \;\; v \;\; 1]^T$ están relacionados por una homografía H

$$\lambda p = HP \quad \text{con} \quad H = K[r_1 \;\; r_2 \;\; t] \tag{C.21}$$

Dado que r_1 y r_2 son ortonormales, tenemos

$$h_1^T K^{-T} K^{-1} h_2 = 0 \tag{C.22}$$

$$h_1^T K^{-T} K^{-1} h_1 = h_2^T K^{-T} K^{-1} h_2 \tag{C.23}$$

Sea $B = K^{-T}K^{-1}$ Se puede verificar que B es simétrica y puede ser definida por el vector

$$b = [B_{11} \;\; B_{12} \;\; B_{22} \;\; B_{13} \;\; B_{23} \;\; B_{33}]^T \tag{C.24}$$

Sea h_i la i-ésima columna de H: $h_i = [h_{1i} \;\; h_{2i} \;\; h_{3i}]^T$; entonces

$$h_i^T B h_j = v_{ij}^T b \tag{C.25}$$

donde

$$v_{ij} = [h_{1i}h_{1j}, \;\; h_{1i}h_{2j} + h_{2i}h_{1j}, \;\; h_{2i}h_{2j}, \;\; h_{3i}h_{1j} + h_{1i}h_{3j}, \;\; h_{3i}h_{2j} + h_{2i}h_{3j}, \;\; h_{3i}h_{3j}]^T$$

Por tanto, las ecuaciones (C.22) y (C.23) se pueden escribir

$$\begin{bmatrix} v_{12}^T \\ (v_{11} - v_{22})^T \end{bmatrix} b = 0 \tag{C.26}$$

Si se hace lo mismo para n imágenes y se apilan las ecuaciones, obtenemos

$$Vb = 0 \tag{C.27}$$

donde V es una matriz $2n \times 6$. La solución dada en [37] es encontrar el vector propio de V^TV asociado al valor propio más pequeño, o bien, el vector singular derecho de V asociado con el valor singular más pequeño. Una vez que b es estimado, se puede calcular los valores de la matriz de parámetros intrínsecos K de la siguiente forma:

$$o_y = (B_{12}B_{13} - B_{11}B_{23})/(B_{11}B_{22} - B_{12}^2) \tag{C.28}$$

$$\lambda = B_{33} - [B_{13}^2 + o_y(B_{12}B_{13} - B_{11}B_{23})]/B_{11} \tag{C.29}$$

$$\alpha = fs_x = \sqrt{\lambda/B_{11}} \tag{C.30}$$

$$\beta = fs_y = \sqrt{\lambda B_{11}/(B_{11}B_{22} - B_{12}^2)} \tag{C.31}$$

$$\gamma = fs_\theta = -B_{12}\alpha^2\beta/\lambda \tag{C.32}$$

$$o_x = \gamma o_y/\beta - B_{13}\alpha^2/\lambda \tag{C.33}$$

Una vez calculada la matriz K, los parámetros extrínsecos de cada imagen se obtienen de la siguiente forma

$$r_1 = \lambda K^{-1}h_1 \tag{C.34}$$

$$r_2 = \lambda K^{-1}h_2 \tag{C.35}$$

$$r_3 = r_1 \times r_2 \tag{C.36}$$

$$t = \lambda K^{-1}h_3 \tag{C.37}$$

con $\lambda = 1/|K^{-1}h_1| = 1/|K^{-1}h_2|$

Bibliografía

[1] J. García Rodríguez A. Angelopoulou, A. Psarrou and K. Revett. Automatic landmarking of 2d medical shapes using the growing neural gas network. In *Proceedings of the workshop CVBIA helded in the International Conference on Computer Vision, ICCV 2005*, pages 210–219, October 2005.

[2] M.C. Andrade. An interactive algorithm for image smoothing and segmentation. *Electronic Letters on Computer Vision and Image Analysis*, 4(1):32–48, 2004.

[3] E. Bayro-Corrochano. Motor algebra for 3d kinematics: The case of the hand-eye calibration. *Journal of Mathematical Imaging and Vision*, 13:79–100, 2000.

[4] E. Bayro-Corrochano. Robot perception and action using conformal geometry. In E. Bayro-Corrochano, editor, *Handbook of Geometric Computing. Applications in Pattern Recognition, Computer Vision, Neurocomputing and Robotics*, pages 405–458 (chap. 13). Springer Verlag, Heidelberg, 2005.

[5] E. Bayro-Corrochano and J. Rivera-Rovelo. Non-rigid registration and geometric approach for tracking in neurosurgery. In *Proceedings of the International Conference on Pattern Recognition*, pages 717–720, Cambridge, UK, August 2004.

[6] J. Burguet and I. Bloch. Homotopic labeling of elements in a tetrahedral mesh for the head modeling. In *Proceedings of the CIARP 2004*, LNCS 3287, pages 566–573, 2004.

[7] M.J. Chantler. *The effect of variation in illuminant direction on texture classification.* PhD thesis, Dept. Computing and Electrical Engineering, Heriot-Watt University, August 1994. Ph.D. thesis.

[8] Mayo Clinic. Analyze software.

[9] H. Li D. Hestenes and A. Rockwood. New algebraic tools for classical geometry. In G. Sommer (Ed.), editor, *Geometric Computing with Clifford Algebras*. Springer-Verlag, Heidelberg, (2001).

[10] Kostantinos Daniilidis. Hand-eye calibration using dual quaternions. *The International Journal of Robotics Research*, 18(3):286–298, March 1999.

[11] P. Meseure F. Triquet and C. Chaillou. Fast polygonization of implicit surfaces. In *International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision*, pages 283–290, February 2001.

[12] B. Fritzke. A growing neural gas network learns topologies. *Advances in Neural Information Processing Systems*, 7, 1995.

[13] K.S. Fu and J.K. Mui. A survey on image segmentation. *Pattern Recognition*, 12:395–403, 1980.

[14] A. Rangarajan H. Chui. A new point matching algorithm for non-rigid registration. In *Proceedings of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 2, pages 44–51, 2000.

[15] S. Mori H. Tamura and T. Yamawaki. Textural features corresponding to visual perception. *IEEE Transactions on Systems Man and Cybiernetics*, 8(6):460–473, 1980.

[16] R. Horaud and F. Dornaika. Hand-eye calibration. *International Journal on Robotics Research*, 14(3):195–210, June 1995.

[17] Eduardo Bayro-Corrochano Jorge Rivera-Rovelo, Silena Herold-García. Hand-eye calibraton and geometric algebra for endoscopic surgery. In *submited to elsewhere*, 2007.

[18] C. Mohan K. Mehrotra and S. Ranka. *Elements of Artificial Neural Networks*, chapter Unsupervised learning, pages 157–213.

[19] T. Kohonen. Self-organization and associative memory. *Springer Verlag*, 1988.

[20] W. Lorensen and H. Cline. Marching cubes: a high resolution 3d surface construction algorithm. *Computer Graphics*, 21-4:163–169, July 1987.

[21] E. Bullit M. Prastawa and G. Gerig. Robust estimation for brain tumor segmentation. In *Medical Image Computing and Computer Assisted Intervention*, volume 2, pages 530–537, 2003.

[22] S. Ho M. Prastawa, E. Bullit and G. Gerig. A brain tumor segmentation framework based on outlier detection. *Medical Image Analysis Journal*, 8(3):275–283, September 2004.

[23] K. van Leemput N. Moon, E. Bullit and G. Gerig. Automatic brain and tumor segmentation. In *Proceedings of the fifth international conference on medical image computing and computer assisted intervention 2002*, pages 372–379, September 2002.

[24] K. van Leemput N. Moon, E. Bullit and G. Gerig. Model based brain and tumor segmentation. In *Proceedings of the 16th International Conference on Pattern Recognition*, pages 528– 531, August 2002.

[25] X. Mu noz. *Image segmentation integrating color, texture and boundary information.* PhD thesis, Ph.D. Thesis in Computer engineering, Girona, Italy, December 2002.

[26] C. Perwass and D. Hildenbrand. Aspects of geometric algebra in euclidean, projective and conformal space. Technical Report Technical Report No. 0310, Christian-Albrechts-University of Kiel, 2003.

[27] V. Ranjan and A. Fournier. Union of spheres (uos) model for volumetric data. In *Proceedings of the Eleventh Annual Symposium on Computational Geometry*, Vancouver, Canada, 402-403 1995.

[28] Klaus D.M. Resch. *Transendoscopic Ultrasound for Neurosurgery.* Springer Verlag, 2005.

[29] B. Rosenhahn and G. Sommer. Pose estimation in conformal geometric algebra. Technical Report Technical Report No. 0206, Christian-Albrechts-University of Kiel, 2000.

[30] S. Pappu S. Gold, A. Rangarajan. New algorithms for 2d and 3d point matching. *Pattern recognition*, 31(8):1019–1031, 1998.

[31] G. Gerig S. Ho, E. Bullitt. Level-set evolution with region competition: automatic 3-d segmentation of brain tumors. In *Proceedings of 16th International Conference on Pattern Recognition*, volume 1, pages 532–535, 2002.

[32] S. Stergiopoulos. *Advanced Signal Processing Handbook: Theory and Implementation for Radar, Sonar, and Medical Imaging Real-Time Systems*, pages 15–1 to 15–43 and 17–1 to 17–28. CRC Press LLC, 2001.

[33] S.G. Berkovich T.M. Martinetz and K.J. Schulten. Neural gas network for vector quatization and it application to time series prediction. *IEEE Transactions on Neural Networks*, 4:558–569, July 1993.

[34] R.Y. Tsai and R.K. Lenz. A new technique for fully autonomous and efficient 3d roboticcs hand/eye calibration. *IEEE Transactions on Robotics and Automation*, 5(3):345–358, June 1989.

[35] Ch. Xu. *Deformable models with applications to human cerebral cortex reconstruction from magnetic resonance images.* PhD thesis, Johns Hopkins University, 1999.

[36] Jana Kosecka Shankar Sastry Yi Ma, Stefano Soatto. *An Invitation to 3D Vision: From Images to Geometric Models.* Springer, 2004.

[37] Zhengyou Zhang. A flexible new technique for camera calibraton. *Microsoft Research*, pages 1–21, 1999.

Publicaciones

Artículos de Journal

- "Medical image segmentation, volume representation and registration using spheres in the geometric algebra framework", *Pattern Recognition*, Journal of the Pattern Recognition Society, 40, pp. 171-188, 2006.
- "Surface approximation using growing self-organizing nets and gradient information", submited to *Applied Bionics and Biomechanics Journal*, Woodhead Publishing Limited, Cambridge, England

Capítulo en libro

- "Geometric Neural Computing for 2D contour and 3D surface reconstruction", invitado a publicarse en el libro *Pattern Recognition Research Horizons*, Nova Science Publishers Inc.

Artículos en Conferencias Internacionales

- "Medical image segmentation using self-organizing neural network and Clifford geometric algebra", in *Proceedings of the International Joint Conference on Neural Networks (IJCNN)* 2006, Vancouver, Canada, July 15-21, 2006
- "Non-rigid alignment and real time tracking using the geometric algebra framework", in *Proceedings of the International Conference on Pattern Recognition (ICPR)* 2006, Hong Kong, China, August 20-24, 2006

  "Object Segmentation Using Growing Neural Gas and Generalized Gradient Vector Flow in the Geometric Algebra Framework", in Proceedings of the *Conferencia Ibero-Americana de Reconocimiento de Patrones (CIARP) 2006*, Cancun, México, November 14-27, 2006
- "Medical image segmentation and the use of geometric algebras in medical applications", in Proceedings of the Conferencia Ibero-Americana de Reconocimiento de Patrones (CIARP) 2005, Havana, Cuba, Noviembre 2005, pp. 729-740
- "Segmentation and volume representation based on spheres for non-rigid registration," in Proceedings of the Computer Vision for Biological Image Application, a workshop of the ICCV 2005; Beijing, China, Oct. 21, 2005, pp. 449-458

  "Non-Rigid registration and geometric approach for tracking in neurosurgery", in Proceedings of the International Conference on Pattern Recognition (ICPR) 2004, Cambridge, UK, pp. 717-720

CENTRO DE INVESTIGACIÓN Y DE ESTUDIOS AVANZADOS DEL I.P.N.
UNIDAD GUADALAJARA

El Jurado designado por la Unidad Guadalajara del Centro de Investigación y de Estudios Avanzados del Instituto Politécnico Nacional aprobó la tesis

Segmentación, Reconstrucción 3D y Registro para Neurocirugía

del (la) C.

Jorge RIVERA ROVELO

el día 20 de Marzo de 2007.

Dr. Eduardo José Bayro Corrochano
Investigador CINVESTAV 3D
CINVESTAV Unidad Guadalajara

Dr. Juan Luis Del Valle Padilla
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. Alexander Georgievich Loukianov
Investigador CINVESTAV 3C
CINVESTAV Unidad Guadalajara

Dr. Rodrigo Ramos Zúniga
Profesor Investigador Titular C
Centro Universitario de Ciencias de la Salud de la Universidad de Guadalajara

Dr. Ricardo Vilalta López
Investigador CINVESTAV
CINVESTAV