Reconstrucción de Superficies 3D aplicadas a imágenes médicas

Leissi Margarita Castañeda León [2] Oscar Enrique Fernández Asunción [2]
Nils Ever Murrugarra Llerena [1],[2]

[1] Sociedad de Estudiantes de Ciencia de la Computación

[2]Universidad Nacional de Trujillo

leissi.lcl@gmail.com,osciest@gmail.com,nineil.cs@gmail.com

Resumen

El presente artículo propone desarrollar un algoritmo para realizar reconstrucción de superficies en 3 dimensiones usando imágenes médicas en 2 dimensiones. Nos hemos basado en la detección de contornos utilizando FB4(Following Border 4) para obtener los contornos de las imágenes médicas respectivas y éstos serán usados para la posterior reconstrucción en 3 dimensiones para lo cual hemos empleado triangulación de superficies.

1. Introducción

En el área de la medicina cada día aparecen enfermedades más nocivas para la salud actualmente lo que esta de novedad son los tumores que aparecen en nuestro organismo y que no se ha determinado una causa dada. Ante este problema ¿qué podemos aportar computacionalmente para solucionarlo mejor dicho tratar de minimizarlo?. Sería muy bueno dar al médico una mejor vista del problema a solucionar, algo como tener un programa que nos permita reconstruir en 3 dimensiones el órgano o parte del organismo en evaluación. Una idea similar básicamente relacionado con la terapia de raciación a células cancerígenas fue lo que llevó a [Keppel, 1975] a crear un algoritmo de reconstrucción de superficies y que es en el cual nos basamos.

El presente artículo entonces tratará sobre la reconstrucción en 3 dimensiones a partir de imágenes en 2 dimensiones, obtenidas mediante tomografías o resonancias magnéticas, pero que representen a un órgano en especial (Por ejemplo: corazón, estómago, cabeza, etc.) pero que se encuentre tal que se vea como si se hubiesen hecho cortes seguidos al mismo. Por ejemplo nosotros hemos utilizado imágenes de sólo la parte de la cabeza y fueron extraídas de videos en [Center for Human Simulation, 1995].

Lo que aquí presentamos es que dado las imágenes médicas ya definidas, obtengamos de ellas los bordes o contorno de la superficie de cada corte del órgano representado. Para ello es bueno considerar un buen detector de bordes, para ello es necesario tener algunas nociones básicas acerca de Procesamiento Gráfico de Imágenes o Procesamiento Digital de Imágenes. Ya luego cada borde obtenido será representado en el espacio 3D, donde escogiendo cada ciertos puntos, se procederá al proceso de reconstrucción mediante una triangulación. Para esta parte la mayoría de métodos propuestos por diferentes autores, consideran la realización de una triangulación de Delaunay, por otro lado nosotros hemos considerado otro criterio, como ya se mencionó anteriormente, el presentado por [Keppel, 1975] basado en áreas y no en volumen, un método sencillo y muy útil que será más adelante definido.

El artículo está estructurado entonces de la siguiente manera, en la sección siguiente se presentan trabajos previos; prosiguiendo la sección tres comenta sobre el proceso de reconstrucción 2D-3D y nuestro algoritmo implementado. La sección cuatro contiene cuadros e imágenes reportando resultados. La sección cinco muestra la discusión de experimentos y por último en la sección seis nuestras conclusiones.

2. Trabajos Previos

Por los años 1994 un cadáver fue cortado por completo en pequeñas franjas consecutivamente, las cuales fueron fotografiadas y digitalizadas para crear un modelo en 3 dimensiones completo de un ser humano, el que fue usado para estudio y aplicaciones de anatomía, el presente trabajo actualmente puede ser visto en el "'National Museum of Health and Medicine in Washington, DC"' y se le denomino :"Visible Human Project"[Center for Human Simulation, 1995].

Tenemos muchos trabajos realizados en lo concerniente a Reconstrucción, tenemos a los basados en Volúmen y a los basado en Superficies, dentro de los cuales podemos mencionar al paper [Galin and Akkouche, 1998], donde presentan un eficiente método de reconstrucción de superficies dado un conjunto de contornos, y donde La superficie reconstruída se define como una superficie implícita. Crean una estratificación de los contornos poligonales en cada sección (o corte, que representa una forma poligonal sin intersecciones y que consiste de uno o varios contornos poligonales cerrados) en trapezoides para acelerar la clasificación de los puntos involucrado en el cómputo de la potencial función de campo y evitar la creación de un esqueleto geométrico.

Tenemos en [Keppel, 1975] y en [H Fuchs and Uselton, 1977], que utilizan la triangulación para la reconstrucción de superficies, y se basan en la teoría de grafos como será descrito más adelante.

Tambien en [OlaÑilsson, 2005], toman las rebanadas de las imágenes medicas pues a partir de ahí se plantean 3 problemas fundamentales en la reconstrucción de superficies como son tales como: 1. Problema de la correspondencia. 2. El problema del suelo de baldosas. 3. El problema de ramificación. El primer problema se centra en la correspondencia de la correcta conexión entre contornos, el segundo problema se centra en que las rebanadas triangulares deben estar acorde con la franja situada entre las dos rebanadas de triángulos adyacentes, y el tercer problema se produce cuando un contorno en un tramo puede corresponder a más de uno un contorno en el tramo adyacente para solucionar estos problemas plantean una serie de reglas que les permiten optimizar estos problemas.

Otra antecedente importante es [Chandrajit L. Bajaj, 2005], el cual tiene cuatro grandes etapas. En la primera etapa se calcula el contorno de cada entrada (imagen), los contornos también pueden ser suavizados antes de esta etapa; la segunda etapa está dado por el Cálculo de la distancia más cercana a transformar, la tercera es Estimación de las distancias a los contornos de entrada y como ultimo paso Derivar la función de la velocidad de buen polinomio que se ajusta a los valores de la distancia, utilizando métodos como Lagrange y formulación Euleriana.

En el paper [Christiansen and Sederberg, 1978] podemos notar un algoritmo para formar una superficie en 3 dimensiones a partir de cortes en 2 dimensiones continuos, algunas de los items interesantes que comenta son: que las superficies 2D tienen que ser similares en tamaño y forma, los puntos a ser tomadas deben estar en el sentido antihorario u horario y muestra una solución para cuando en una superficie 2D tiene más de un contorno en ella usando un punto intermedio para realizar la triangulación.

En [Desheng Wang and Weatherill, 2005] trata sobre un método eficiente para reconstrucción de superficies a partir de cortes seccionales, basandose en una triangulación delaunay limitada en 2 dimensiones; entre algunos de los puntos que tiene en cuenta el trabajo son: el problema de reconstrucción se divide en subproblemas llegando a pares de contornos adyacentes, solución al problema de correspondencia el cual envuelve encontrar las correctas conecciones entre los contornos de secciones adyacentes y el problema que ocurre cuando una sección corresponde a más de un contorno en la sección adyacente.

Un trabajo interesante es [Benavides, 2004], donde comentan el problema de evitar errores en el transplante de un órgano que posee algun mal en el momento de la operación. En el trabajo usan 2 videocámaras, una para capturar la vista de frente y la otra una vista de perfil. Con tales videos el proceso que realizan es el siguiente: puntos 2D que viene a ser la proyección de cada vista, vértices y polígonos en 3D, bordes y coordenadas de textura en 3D, detección de caras lateral, frente y perfil en 3D las cuales nos ayudaran a formar el objeto en 3D.

Un gran aporte es InVesalius es software libre que permite reconstruir partes del cuerpo a 3 dimensiones a partir de tomografías, este programa nos permite apreciar la parte reconstruida y además permite tener varios niveles de visión del órgano, asi podemos apreciar los músculos o huesos en caso de estar presentes.

3. Reconstrucción 2D-3D

Tenemos un conjunto de imágenes con las características de que sean imágenes de un mismo órgano (para nuestro caso lo hemos aplicado a parte específicamente la cabeza, ya que no consideramos huecos en la imagen), que sean o representen cortes seccionales (no tan separados) hechos mediante tomografías o resonancias magnéticas, con los cuales se procederá a trabajar.

Ahora para iniciar la reconstrucción en 3 dimensiones debemos primero extraer los contornos de las imágenes de 2 dimensiones. A continuación mostraremos los pasos generales para calcular las *Coordenadas del Borde*.

- Aplicaremos el filtro de la mediana (con radio de 7, tanto en x como en y) a la imagen para la eliminación de ruido.
- Bordear la imagen (para asegurar esté dentro los contornos de la imagen deseada)
- Binarizar la imagen
- Calcular centro de masa que viene a ser nuestro punto de inicio.
- *Calcular el BF4* (Border Folowing 4) para obtener el contorno de la imagen dado el punto de inicio.

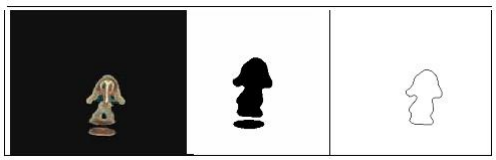

Figura 1: Una Imagen Médica, su binarización y su Contorno

Debemos aclarar que en nuestra implementación no hemos considerado la existencia de huecos, y tampoco imágenes con más de 2 objetos. Por ejemplo ver Figura 1, donde se puede observar la presencia se dos contornos, sin embargo dado el punto de inicio, sólo se obtuvo uno de ellos, que por conveniencia tiene que ser el de mayor tamaño (que es calculado mediante el centro de masa de la imagen binarizada).

Continuando, tenemos que el Algoritmo FB4 el cual nos permite encontrar el contorno de cierto objeto dado el punto de inicio hallado, para ello calculamos el punto de borde el cual es el punto de más a la izquierda del punto de inicio y que pertence al objeto y a la vez calculamos el punto blanco en esa misma dirección, el tener éstos dos puntos nos permitirá calcular otros 2 adyacentes a ellos y siguientes en el contorno, este paso se realizará hasta llegar a éstos dos puntos tomados como iniciales. Para información más a detalle ver [Chang and Saavedra, 2001].En el Algoritmo 1 se detalla este proceso.

Algoritmo 1 Calcular el BF4(I,x,y)

Requiere: La matriz I de la imagen, el punto (x,y) parte del interior del objeto.
Asegurar: Dos vectores: bordeX, bordeY, que contendrán los puntos del borde
$(px, py) \leftarrow$ Buscar el punto más a la izquierda desde el punto central y que colinda con un punto blanco.
b $\leftarrow$ 1
$P_0 \leftarrow$ p
$Q_0 \leftarrow$ punto de la izquierda de P_0
Añadir a los arreglos del borde (bordeX, bordeY) el punto P_0.
Mientras $b == 1$ **hacer**
  [vx,vy,qx,qy] $\leftarrow$ *Determinar-Borde-Contorno(px,py,qx,qy,I)*
  Si ($p_0x == vx$ y $p_0y = vy$) ó ($p_0x == vx$ y $p_0y = vy - 1$) ó ($p_0x == vx$ y $p_0y = vy + 1$) **entonces**
    b $\leftarrow$ 0
  Si no
    añadir vx a bordeX
    añadir vy a bordeY
    px $\leftarrow$ vx
    py $\leftarrow$ vy
  Fin Si
Fin Mientras
Devolver(bordeX,bordeY)

Continuando veamos el punto de triangulación para lo que debemos tener en cuenta, dados dos contornos $P_i = (p_1, p_2, ..., p_n) y Q_i = (q_1, q_2, ..., q_n)$ se desea generar una triángulación de

Algoritmo 2 Determinar-Borde-Contorno(cx, cy, qx, qy, I)

Requiere: La matriz I de la imagen,punto central c:(cx,cy) y q:(qx, qy) para empezar a bordear.
Asegurar: punto borde (vx, vy) y q:(qx, qy) actualizado.

```
b ← 1
i ← 1
v ← Determinar Nº vecino(cx, cy, qx, qy)
sv ← Determinar la secuencia de Vecinos de v
(vx,vy) ← Determinar vecino dado cx, cy, sv(i)(secuencia).
Mientras b == 1 e i <= 7 hacer
  Si I(vx, vy) == Blanco entonces
    i ← i+1
    (vx,vy) ← Determinar vecino dado cx, cy, sv(i)(secuencia).
  Si no
    M ← vecindad del punto (cx,cy)
    Si es4Conexo(M,cx,cy,vx,vy) entonces
      b ← 0
    Si no
      i ← i+1
      (vx,vy) ← Determinar vecino dado cx, cy, sv(i)(secuencia).
    Fin Si
  Fin Si
Fin Mientras
Si i == 8 entonces
  qx ← -1
  qy ← -1
  cx ← -1
  cy ← -1
Si no
  (vx,vy) ← Determinar vecino dado cx, cy, sv(i)(secuencia).
Fin Si
Devolver(vx,vy,qx,qy)
```

Algoritmo 3 Es 4 Conexo(I,p_ix, p_iy, p_fx, p_fy)

Requiere: Imagen I de 3 x 3, puntos central p_i:(p_ix,p_iy), punto p_f:(p_fx, p_fy) punto vecino de p_i.

Asegurar: valor entero igual a 1 si es 4 conexo, caso contrario 0.

```
b ← 0
Si |(p_f x − p_i x) + (p_f y − p_i y)| == 1 entonces
  b ← 1
  Si (p_f x − p_i x) == −1 y (p_f y − p_i y) == −1 entonces
    Si I(1,2) == 0 y I(2,1) == 0 entonces
      b ← 1
    Fin Si
  Fin Si
  Si (p_f x − p_i x) == −1 y (p_f y − p_i y) == 1 entonces
    Si I(1,2) == 0 y I(2,3) == 0 entonces
      b ← 1
    Fin Si
  Fin Si
  Si (p_f x − p_i x) == 1 y (p_f y − p_i y) == 1 entonces
    Si I(3,2) == 0 y I(2,3) == 0 entonces
      b ← 1
    Fin Si
  Fin Si
  Si (p_f x − p_i x) == 1 y (p_f y − p_i y) == −1 entonces
    Si I(3,2) == 0 y I(2,1) == 0 entonces
      b ← 1
    Fin Si
  Fin Si
Fin Si
Devolver(b)
```

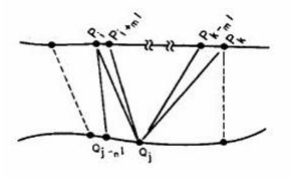

Figura 2: Formación de Triángulos entre 2 contornos

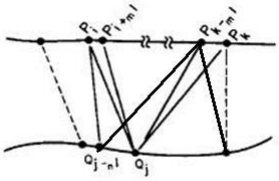

Figura 3: Triángulos Superpuestos

la forma P_i, P_{i+1}, Q_j ó Q_i, Q_{i+1}, P_j entre todos los puntos de los contornos P y Q (ver Figura 2), la principal restricción es que los triángulos que se generen no se superpongan entre ellos[H Fuchs and Uselton, 1977] por ejemplo ver Figura 3.

Para evitar esta situación se debe crear un grafo dirigido G-$< V, A >$ (ver Figura 4), en el cuál los vértices corresponden a un conjunto de posibles arcos entre los puntos de P y Q, los arcos corresponden a un conjunto de posibles triángulos. Es decir: V=v_{ij}/i=0,1,...,m-1; j=0,1, ... n-1 y v_{ij} corresponde al arco P_iQ_j.

Entonces dado que es definido mediante un problema de grafos, nosotros para obtener una buena triangulación necesitamos obtener el camino de menor costo, donde el costo es respecto a cada triángulo(el costo esta dado por el valor del área de dicho triangulo, donde el triángulo en el grafo viene a ser arcos). Entonces dado los 2 contornos, nosotros podemos formar un grafo con un costo mínimo con respecto a las áreas de los triángulos generados en el grafo (ver Figura 5).

Otro punto importante en la triangulación es que el punto P_0 y Q_0 tengan la menor distancia entre ellos para hacer un buen recorrido en la triangulación. El grafo anterior puede expresarse de la siguiente manera en la Figura 6. Por último en el algoritmo 4 se plasma nuestra idea.

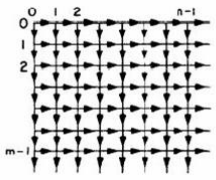

Figura 4: Representación de triangulación mediante un grafo

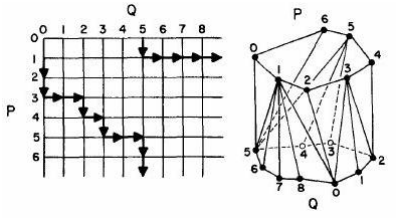

Figura 5: Camino de menor costo

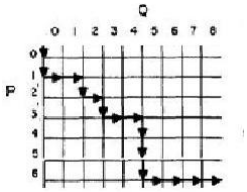

Figura 6: Camino de menor costo mejorado

Algoritmo 4 triangular

Requiere: v1 y v2 representan a los contornos P y Q respectivamente
Asegurar: Vector de Triangulos

```
p ← 1
q ← 1
crear vectorT
Mientras p < tamanho(P) y q < tamanho(Q) hacer
  t1 ← Formar Triangulo P_p P_{p+1} Q_q
  t2 ← Formar Triangulo Q_q Q_{q+1} P_p
  a1 ← area(t1)
  a2 ← area(t2)
  Si a1 < a2 entonces
    agregar a vectorT t1
    p ← p+1
  Si no
    agregar a vectorT t2
    q ← q+1
  Fin Si
Fin Mientras
Devolver(vectorT)
```

Nº de imágenes	Tiempo(seg)	Nº de triángulos
109	35.46	12634
66	14.80	9039
47	9.04	6416
34	5.49	4503

Cuadro1: np=10 y nz =10

Nº de imágenes	Tiempo(seg)	Nº de triángulos
109	12.51	4005
66	6.03	2984
47	3.64	2169
34	1.89	1525

Cuadro 3: np=30 y nz =10

Nº de imágenes	Tiempo(seg)	Nº de triángulos
109	18.40	6294
66	8.05	4546
47	5.04	3232
34	3.42	2268

Cuadro 2: np=20 y nz =10

Nº de imágenes	Tiempo(seg)	Nº de triángulos
109	34.83	12634
66	14.26	9039
47	8.50	6416
34	5.13	4503

Cuadro 4: np=10 y nz =20

Nº de imágenes	Tiempo(seg)	Nº de triángulos
109	35.28	12634
66	14.53	9039
47	8.30	6416
34	5.35	4503

Cuadro 5: np=10 y nz =30

4. Experimentos y Resultados

Para la realización de experimentos de nuestro algoritmo implementado hemos tomado como muestra un total de 256 fotos de 320 * 240 píxeles considerando 4 objetos en total. Cada 1º, 2º ,3º y 4º objeto estan representados por 109,66,47 y 34 imágenes respectivamente.

Realizamos experimentos variando nuestros 3 parámetros de entrada: nº de imágenes, distancia en z(distancia entre cada corte), y np(significa cada cuántos puntos seleccionaremos uno para la triangulación). Para medir nuestros resultados calculamos el tiempo de reconstrucción del objeto a 3D dado la variación de estos parámetros, los resultados se muestran en los cuadros del 1 al 5, a la vez también puede observar el número de triángulos generados para cada caso. Se debe además considerar que los experimentos fueron ejecutados en un Intel Pentium 4, CPU de 3.20 GHz, y 512MB de RAM.

Tambien veamos un ejemplo de la aplicación , dado los cortes de la Figura 7, obtendremos el resultado de la Figura 8.

5. Discusión de los Experimentos

De los experimentos realizados podemos apreciar que mientras aumente nuestro parámetro de entrada np, el tiempo de reconstrucción 3D disminuye. Por otro lado notamos que el parámetro dz, no es influyente en el tiempo de reconstrucción ya que en los resultados los tiempos fueron muy parecidos, pero genera una imagen un poco más alargada mientras sea mayor.

Por ultimo dado un mayor número de imagenes el tiempo de reconstrucción es mayor pero la calidad de detalles de la imagen es más apreciable.

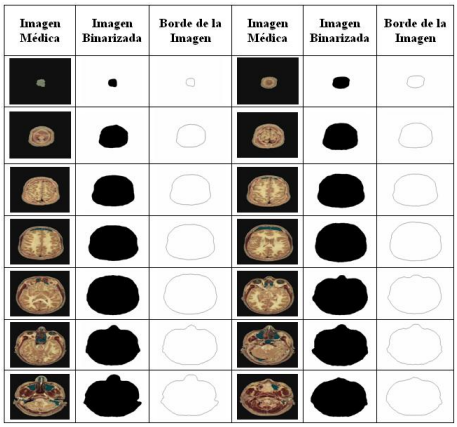

Figura 7: Cortes desde arriba hacia abajo en la cabeza

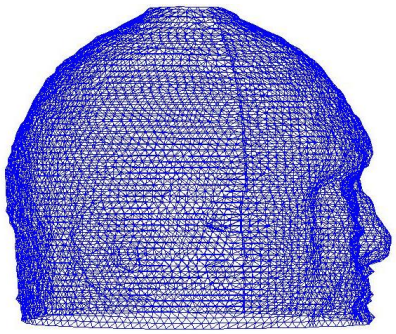

Figura 8: Resultado obtenido dado las entradas de Figura 7

6. Conclusiones

Como conclusión obtenida del trabajo realizado podemos aseverar que el método mientras más imagenes reciba proporcionará objetos en 3 dimensiones con más detalle.

El algoritmo de triangulación usado es simple y facil de implementar y con una complejidad computacional lineal en comparación con la triangulación de Delaunay en 3D que nos da una complejidad computacional de O(n$\log$ n), además brinda resultados buenos, sin embargo Delaunay nos daría muchos mejores resultados respecto a la visualización.

Una de las desventajas que presenta el método es el coste computacional, ya que dado un mayor nº de imagenes demorara más tiempo en crear el objeto 3D, pero esto es compensado por los detalles obtenidos en el objeto. Se llegó a ésto dado los resultados mostrados en los cuadros

Otra desventaja es que no tenemos niveles de visión en el objeto, solo podemos apreciar la parte más externa.

Como cuestión futura se considera implementar un algoritmo que nos permita tener varios niveles de visión como el presentado por Invesalius.

Referencias

[Benavides, 2004] Benavides, J. (2004). Reconstruction of medical images (organs) of 2d to 3d during the transplant or organ surgeries with a certain type of cancer, since a medical movie using filters of wavelet and neural networks and its visualisation with opengl. *Universidad Nacional de San Agustín*.

[Center for Human Simulation, 1995] Center for Human Simulation (1995). Center for human simulation. http://www.uchsc.edu/sm/chs/.

[Chandrajit L. Bajaj, 2005] Chandrajit L. Bajaj, Edward J. Coyle, K.-N. L. (2005). Surface reconstruction via contour metamorphosis: An eulerian approach with lagrangian particle tracking. *LinkAoping University*.

[Chang and Saavedra, 2001] Chang, V. and Saavedra, J. (2001). Métodos alternativos para el mejoramiento automático del contraste de imágenes. *Escuela Académico Profesional de Informática, Universidad Nacional de Trujillo*.

[Christiansen and Sederberg, 1978] Christiansen, H.Ñ. and Sederberg, T. W. (1978). Conversion of complex contour line definitions into polygonal element mosaics. *Brigham Young University Provo, Utah*.

[Desheng Wang and Weatherill, 2005] Desheng Wang, Oubay Hassan, K. M. and Weatherill, N. (2005). Efficient surface reconstruction from contours based on two-dimensional delaunay triangulation. *Civil and Computational Engineering Centre, School of Engineering, University of Wales Swansea, Singleton Park, Swansea SA2 8PP, U.K.*

[Galin and Akkouche, 1998] Galin, E. and Akkouche, S. (1998). Fast surface reconstruction from contours using implicit surfaces. *Laboratoire d'Informatique Graphique Image et Modélisation, Ecole Centrale de Lyon*.

[H Fuchs and Uselton, 1977] H Fuchs, M. K. and Uselton, P. (1977). Optimal surface reconstruction from planar contours. *University of Texas,Dallas*.

[Keppel, 1975] Keppel, E. (1975). Approximating complex surfaces by triangulation of contour lines. *IBM J. RES. Develop.*

[OlaÑilsson, 2005] OlaÑilsson, David Breen, K. M. (2005). Arbitrary topology shape reconstruction from planar cross sections. *Department of Computer Science, Purdue University*.