Detección de Siluetas en Modelos Volumétricos basado en un sistema de Iluminación No-Fotorrealista

Germán Arroyo, Francisco Velasco, Domingo Martín
Departamento de Lenguajes y Sistemas Informáticos
C/Periodista Daniel Saucedo Aranda s/n, E-18071, Granada
e-mail: arroyo@ugr.es, fvelasco@ugr.es, dmartin@ugr.es

Resumen

Para cualquier técnica de ilustración un paso fundamental es la extracción de siluetas y líneas de forma, las cuales ayudan al observador a discernir la forma del objeto que está visualizando, mejorando sustancialmente la visualización del mismo. En este artículo se presenta un nuevo método muy rápido para obtener la silueta y las líneas de forma en el proceso de visualización directa no-fotorrealista de volúmenes basado en trazado de rayos con un método de optimización mediante un octree. Este método consiste en una extracción de frontera basada en un buffer de imagen obtenido al mismo tiempo que se están calculando las intersecciones del rayo con el modelo, y en la combinación de dichos buffers de imagen para obtener las siluetas y líneas de forma. Una mejora substancial con otros métodos de extracción de siluetas en volúmenes es que en éste método no se requiere ninguna definición de umbral por parte del usuario, no está basado en el z-buffer y por tanto no depende de la precisión del mismo, y la posición y dirección de la cámara no afecta al rendimiento del algoritmo. Además el algoritmo es capaz de detectar siluetas para objetos que se encuentran dentro de otros objetos diferenciados tan sólo por el tono de ambos.

Palabras clave: Visualización directa de volúmenes, visualización de datos, extracción de siluetas, líneas de forma, visualización científica, ilustración de volúmenes, visualización no-fotorrealista.

1. Introducción

En la visualización científica, donde suele haber una gran cantidad de información, es imprescindible mostrar de forma efectiva la información relevante de forma que el ser humano pueda discernir claramente zonas de interés. En los datos volumétricos una parte muy importante de la visualización es la forma, usualmente más que los cambios de color o iluminación debido a que la mayoría de las imágenes obtenidas por

los métodos tradicionales de captación de este tipo de datos suelen estar en valores de grises o en colores artificiales y una visualización directa de estos datos sin ningún tipo de ajuste manual suele ser difícil de comprender para la visión humana [1, 2].

Cada vez más se está intentando desarrollar la visualización de los modelos volumétricos hacia un sistema no fotorrealista que implique algo de abstracción y que ayude al sistema de visión humano a comprender lo que está viendo. Es por ello que un paso fundamental es detectar y marcar las siluetas en los lugares de interés en este tipo de modelos, esto es un paso previo hacia cualquier técnica no fotorrealista aplicada hoy en día sobre modelos poligonales [3].

En este artículo se propone un nuevo algoritmo aplicado a la visualización de volúmenes para la detección de siluetas. Es aconsejable sin embargo, para obtener una buena clasificación de fronteras, tener un tipo de iluminación que no modifique el valor del tono de los datos en un paso previo a esta detección de siluetas.

Para comprender mejor este método de detección de siluetas y sus fundamentos primero serán revisados los trabajos previos relacionados con este nuevo método y después se pasará a describir el método en sí.

2. Trabajos previos

La mayoría de métodos no fotorrealistas son utilizados y desarrollados para modelos poligonales o sólidos donde no hay heterogeneidad interna y prácticamente no hay necesidad de tener en cuenta la transparencia de los materiales [4], sin embargo, en visualización directa de volúmenes es obligado el tener en cuenta como prioridad este concepto. No todas las técnicas no fotorrealistas son aplicables a este tipo de modelos, en especial técnicas de estética de la imagen, sin embargo muchos de los métodos basados en ilustración técnica pueden ser útiles. Todas estas técnicas y algoritmos tienen como base fundamental el tener detectadas las siluetas del modelo, lo cual hace imprescindible este paso previo a cualquier otro método que se vaya a realizar después [3, 5].

Ebert menciona algunas técnicas de realce de bordes que mejoran sustancialmente el resultado de la imagen final, sin embargo siguen sin estar marcadas las siluetas, lo que se hace es ampliar todas las fronteras existentes, lo cual da lugar a numerosos errores en la visualización o una cantidad enorme de ajuste manual [6].

El método de Saito, basado en la detección de siluetas a partir de buffers geométricos (G-Buffers), el cual está aplicado a modelos poligonales, puede ser aplicado a modelos volumétricos también, sin embargo es dependiente de la resolución del buffer geométrico utilizado [7].

Nagy propone un método de calidad para la detección de siluetas en volúmenes, el cual obtiene unos resultados realmente sorprendentes, sin embargo es altamente dependiente de un umbral que hay que ajustar a mano, normalmente utilizando un método de ensayo y error [8].

En las aproximaciones para detectar siluetas, la mayoría de los algoritmos son

capaces de detectar solamente las siluetas de un objeto si existen dos de ellos de forma que uno esté dentro del otro como en el caso de la figura 1, mientras que para representar las siluetas de ambos objetos hay que pasar el algoritmo varias veces renderizando primero uno y luego otro. En el método que se propone en este artículo pueden ser detectadas varias siluetas al mismo tiempo si los objetos difieren entre sí en cuanto a una característica elegida, en nuestro caso particular, el tono.

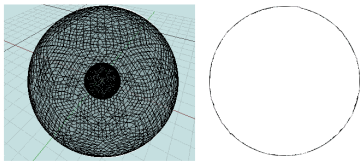

Figura 1: Un objeto está dentro de otro, pero la silueta obtenida con los métodos anteriores solamente obtienen la del objeto mayor

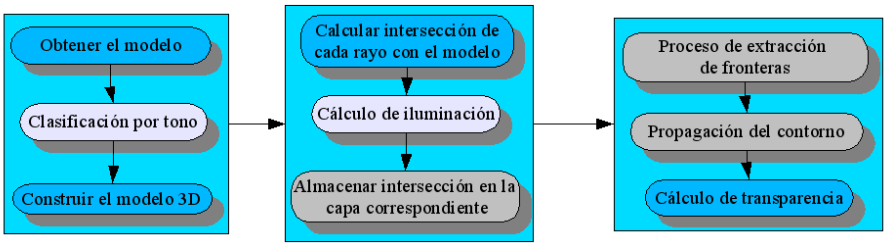

Figura 2: Cauce general del método

3. Aproximación al método

Este método altera el cauce normal en un trazado de rayos [9], incluyendo además un paso previo de clasificación. Es necesario que el modelo de iluminación empleado no cambie el tono ya que el método se basa en esta característica para diferenciar las zonas de interés. El cauce general del algoritmo se muestra en la figura 2, indicando las zonas en gris la alteración de un cauce normal en el proceso de construcción y visualización del modelo, y las de color morado pasos que, si bien modifican el cauce normal del algoritmo, no son pasos imprescindibles para el buen funcionamiento del mismo ya que son intrínsecamente dependientes de la elección de la característica del tono como elemento separador entre zonas de interés.

3.1. Clasificación y cálculo de iluminación

Debido a que la mayoría de datos que se obtienen en modelos volumétricos vienen dados como una serie continua de imágenes en escala de grises [10, 11], el método de clasificación empleado se basa en tonalidades de gris. En los cuales, a partir de ciertos umbrales, aparecen zonas de interés en el modelo.

El sistema de clasificación que se propone es utilizar un rango de tonos para cada una de las zonas de interés, de tal forma que vayan resaltadas con respecto al resto del modelo. Este proceso puede ser fácilmente implementado para que aparezca de forma casi automática dicha clasificación puesto que a partir de un umbral se puede asignar un valor de tono asociado a un color en el espacio HSV [12, 13].

El resultado de esta clasificación es un tono asociado a cada dato. Los datos que pertenecen a las mismas zonas de interés variarán su tono levemente unas respecto a otras, mientras que las que pertenecen a distintas zonas de interés tendrán tonos muy diferentes entre sí. En cuanto al método de iluminación, es de suma importancia que el modelo elegido en el proceso de visualización no modifique el parámetro de clasificación por lo que ya hemos comentado.

Una forma de solucionar este problema fácilmente es hacer independiente el cauce de visualización con respecto al de detección de siluetas, llevando un sistema de color para el proceso de iluminación y otro distinto para el proceso de clasificación, de tal forma que en los sucesivos pasos el cauce de visualización aparezca diferenciado con respecto a la detección de siluetas tal y como se muestra en la figura 3.

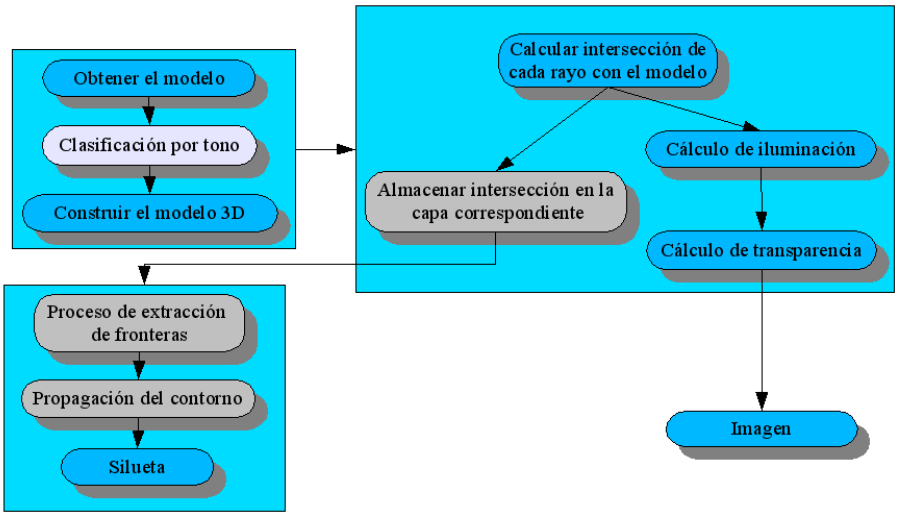

Figura 3: Cauce modificado del método para cualquier tipo de iluminación

El modelo de iluminación que se ha usado es el siguiente:

Sea I_j la fuente de luz con dirección $\vec{L_j}$ que ilumina el punto $\vec{P_i}$ con la normal asociada $\vec{N_i}$, la cámara C está orientada hacia la dirección dada por $\vec{D}$. De tal forma que los valores HSV del punto $\vec{P_i}$ vienen dados por las siguientes ecuaciones:

$$\begin{aligned} H &= TonoAsociado(\vec{P_i}) \\ S &= \vec{L_j} \cdot \vec{N_i} \\ V &= \vec{D} \cdot \vec{N_i} \end{aligned}$$

La saturación (S) sigue un modelo inspirado en el modelo Lambertiano clásico y el efecto producido es parecido, es decir, el sombreado de los objetos. Mientras que el valor (V) al depender de la dirección del observador, el efecto que produce es oscurecer las fronteras de los objetos desde ese punto de vista.

4. Ajuste de capas

El algoritmo de detección de siluetas va a operar sobre un buffer que recogerá al final del algoritmo las siluetas detectadas. Para ello se va a operar el citado buffer acumulador con otros buffers paralelos a él y extraídos del volumen (rebanadas del mismo). En esta sección mostramos cómo obtener esas rebanadas del volumen, que denominamos capas-imagen. En la siguiente mostramos como operar las capas-imagen para calcular las siluetas.

La construcción de las capas-imagen la realizamos durante el proceso de trazado de rayos, mientras se van lanzando los rayos desde el observador pasando por los píxeles vamos almacenando para cada pixel una lista de intersecciones [14].

Cada intersección viene dada por una característica c_{xyz} del dato intersectado por el rayo r_{xy} y una distancia desde el origen del rayo d_{xyz}. En nuestro caso la característica será el color con un valor de transparencia u opacidad (RGBA). Según la notación tomada x e y indican respectivamente la columna y fila del pixel tomado, mientras que z indica la posición en la lista de intersecciones asociadas al punto de coordenadas (x, y) (ver figura 4).

Cada una de estas listas de intersecciones y características está asociada, por tanto, a un pixel. Todas estas características conviene tenerlas en unos buffers independientes de tal forma que las intersecciones queden discretizadas en capas para un proceso posterior de detección de fronteras. Cada capa-imagen, por tanto, tendrá los datos del modelo tal y como si fuera hecho rebanadas siguiendo el plano de la cámara.

La inserción de una característica en un dato viene determinado por el número de capas-imagen que tengamos, éste es un parámetro definido por el usuario pero que es fácilmente calculable si el proceso de inserción se realiza después de tener cada dato en la capa-imagen en lugar de realizarlo al mismo tiempo (ya que se puede calcular el número de capas-imagen en función del número de intersecciones máximo entre todos los rayos lanzados).

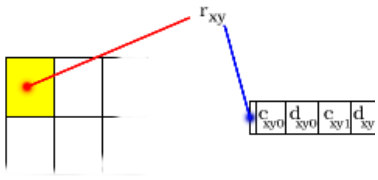

Figura 4: Para cada pixel (celda amarilla) hay un rayo asociado, que a su vez tiene asociado una serie de intersecciones dadas por características c_{xyz} y distancias d_{xyz} desde el origen del rayo

Sea r_z el número de capas-imagen elegido, sea z_{min} el mínimo de todas las distancias de todas las listas, y z_{max} el máximo de las mismas distancias. Sea $inc = \frac{(z_{max}-z_{min})}{r_z}$, definimos el operador $\div$ como el operador de división entera truncando por defecto. Sea r_x la resolución de las capas-imagen en el eje x (todas tienen la misma resolución), r_y la resolución de las capas-imagen en el eje y, s el número de listas, y n_i el número de elementos de la lista l_i. Cada una de las intersecciones en la lista l_i viene representada por tij. Como hemos comentado anteriormente, x e y indican respectivamente la columna y fila del pixel tomado.

El algoritmo para introducir los elementos es el siguiente:

- Para todo x desde 0 hasta $r_x - 1$:
  - Para todo y desde 0 hasta $r_y - 1$:
    - $i = x + y \cdot r_x$
    - Para todo j desde 0 hasta n_i:
      - $Introduce(t_{ij}, c_{xy})$

donde la función $Introduce$ es una función que asigna el valor c_{xy} a la celda (x, y, z), y donde z ha sido previamente calculada como $z = (t_{ij} \div inc) - (z_{min} \div inc)$. Obviamente i es la posición en el array de listas y j es la posición en la lista l_i.

Con este algoritmo cada intersección se ajusta a la capa-imagen más cercana. Sin embargo, varias intersecciones pueden coincidir en una misma celda de una misma capa-imagen, en ese caso se podrían tomar varias aproximaciones:

1. Tomar uno de ellos de forma indiferente.
2. Tomar el más cercano a la capa-imagen.
3. Tomar la media u otro tipo de interpolación entre los datos.

A priori podría parecer que la mejor aproximación sería la tercera opción, pero hay que tener en cuenta que una media o interpolación en los valores RGBA modifica

el tono del color resultante, siendo inútil la clasificación previa. Por ello hay que adoptar un criterio que no modifique la característica que diferencia las zonas de interés, normalmente, para un número más o menos elevado de capas-imagen, puede adoptarse la opción 1 o la 2 indistintamente sin observar ningún cambio apreciable.

Por último, aquellas celdas en las que no se introdujo ningún dato de color se fijan al color y transparencia del fondo. Hay que tener en cuenta que el algoritmo separa las capas-imagen de forma uniforme, sin embargo podría modificarse fácilmente para adaptar la separación de las mismas si se tiene un conocimiento previo de la geometría del modelo.

Una vez obtenidas las capas-imagen tenemos el modelo cortado en rebanadas pero desde la dirección y posición de la cámara, de tal forma que el cálculo de estas rebanadas se realiza al mismo tiempo que se visualiza el modelo, y el resto del algoritmo no se ve afectado en rendimiento. Todas las capas-imagen se encuentran ordenadas en profundidad, desde la más cercana a la más lejana.

5. Extracción de fronteras

Para la obtención de fronteras se define una serie de buffers auxiliares (llamados capas-acumulador) basados en el artículo de Nagy, cada una de estas capas-acumulador puede tener tres tipos de datos: *vacío*, *lleno* o *frontera*. Además el número de estas capas-acumulador es el mismo que el de las capas-imagen, y su resolución también es la misma. Cada capa-acumulador está asociada a una capa-imagen correspondiente.

El valor vacío es marcado cuando el valor alfa en una celda de la capa-acumulador es cero, y es lleno en cualquier otro caso. De esta forma todo lo que es semi-opaco es marcado a lleno.

Con esta aproximación el resultado obtenido es similar al de la figura 5. Mientras que en el artículo de Nagy las fronteras eran calculadas con la condición siguiente: *Un valor lleno es modificado a frontera si no todos los ocho vecinos son llenos*. En el método que se propone aquí el cambio a frontera se produce con la siguiente condición: *Un valor lleno es modificado si alguno de los ocho vecinos es vacío o el tono del valor lleno difiere con el tono de alguno de sus vecinos llenos en un error* δ. Este error δ suele coincidir con la diferencia entre los tonos realizados en la primera clasificación para las distintas zonas de interés, o bien, puede ser definido manualmente. De esta forma todos los valores llenos que se encuentren limitando un tono o vacío son marcados como frontera. Hay que tener en cuenta que este proceso de extracción se efectúa para cada capa-acumulador de forma independiente.

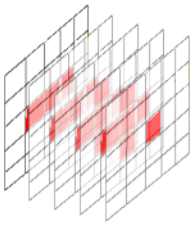

Figura 5: Después de aplicar el test del canal de transparencia, el resultado queda marcado en lleno (color rojo) para todos los datos que eran semitransparentes, y vacíos (blancos) para todo lo que tenía un nivel de transparencia máximo

6. Extracción de siluetas

Para obtener la silueta Nagy define un acumulador y un operador, en este artículo presentamos dos acumuladores y dos tipos de operadores ya que se desea obtener silueta, no sólo del objeto que tenga frontera con una zona vacía, sino también entre elementos que tengan un tono distinto. De esta forma uno de los acumuladores (lo llamaremos A_0) tendrá almacenado cuatro tipos de valores: *vacío*, *lleno*, *frontera* o *silueta*. Mientras que el otro acumulador (lo llamaremos A_1) tendrá almacenado un valor entre 0 y 360 correspondiente al tono de un color en el espacio HSV.

Partimos con los acumuladores con un valor por defecto y vamos actualizándolos al ir operándolos con las capas (tanto las capas-imagen como las capas-acumulador) desde la más cercana a la más lejana. Los operadores propuestos en este artículo ($\oplus$ y $\otimes$) se muestran en la figura 6 y 7 respectivamente. Y la justificación del resultado de las operaciones es la siguiente:

En el operador $\oplus$ tenemos los casos triviales de operación indicados con color negro, en los que si en una celda del acumulador A_0 tiene el valor vacío, se le asigna lo que haya en la capa-acumulador que se está procesando; cuando está lleno se mantiene el valor que había (salvo en el Caso A que explicaremos más adelante). Para el caso en el que lo que hay en la celda del acumulador A_0 sea frontera, si en la capa-acumulador que se está procesado hay un vacío significa que siguiendo el rayo visual del observador se ha pasado de frontera a vacío, lo cual significa que ese punto debe ser silueta. Cuando una celda del acumulador A_0 se ha definido como silueta simplemente se mantiene.

Los casos especiales A y B están situados en la tabla cuando se pasa de frontera a lleno o de lleno a frontera, en cuyo caso hay que examinar la característica asociada (en nuestro caso el tono) al acumulador y a la capa-imagen en ese lugar, de modo que se pueda discernir si está habiendo un cambio de tono, en cuyo caso estamos ante un cambio de objeto y debe considerarse una frontera o una silueta; o por el contrario si estamos dentro de la misma tonalidad, estamos ante el mismo objeto y por tanto el

operador dará como resultado la etiqueta *lleno*. Así los casos se definen de la siguiente forma:

- Caso A: Si $Tono(A_0) = Tono(capa - imagen)$ entonces marcar el acumulador como *lleno*, en otro caso marcarlo como *frontera*.
- Caso B: Si $Tono(A_0) = Tono(capa - imagen)$ entonces marcar el acumulador como *lleno*, en otro caso marcarlo como *silueta*.

⊕	Vacío	Lleno	Frontera	Silueta
Vacío	Vacío	Lleno	Silueta	Silueta
Lleno	Lleno	Lleno	Caso B	Silueta
Frontera	Frontera	Caso A	Frontera	Silueta

Figura 6: Operador $\oplus$, el resultado indica el estado del acumulador A_0 tras operarlo con una capa-acumulador. Los posibles valores del operando capa-acumulador están representados en la columna de la izquierda, mientras que los del operando acumulador A_0 (antes de operarlo) están mostrados en la parte superior

Debido a estos dos casos especiales en los que hay que hacer una comparación entre tonos, el acumulador A_1 y el operador $\otimes$ se hacen necesarios ya que se necesita almacenar los tonos de los últimos valores que modificaron el acumulador A_0. El operador $\otimes$ se define según se muestra en la figura 7 con dicho objetivo de que cada celda almacene el tono que provocó un cambio en su correspondiente celda del acumulador A_0. De igual modo, también tenemos dos casos especiales A y B que definimos a continuación:

- Caso A: Si $Tono(A_0) = Tono(capa - imagen)$ entonces no tocar A_1, en otro caso igualarlo al tono de la capa-imagen.
- Caso B: Si $Tono(A_0) = Tono(capa - imagen)$ entonces igualar A_1 al tono de la capa-imagen, en otro caso no tocar A_1.

El operador booleano de igualdad que se usa en los casos A y B de los operadores $\oplus$ y $\otimes$ está sujeta al mismo error δ que se mencionaba en la extracción de fronteras.

6.1. Ajuste del parámetro δ

El problema que se origina con el parámetro δ es que para el sistema de visión humano hay una clara distinción entre un tono azul y uno rojo, sin embargo el concepto

⊗	Vacío	Lleno	Frontera	Silueta
Vacío	Color acum.	Color acum.	Color acum.	Color acum.
Lleno	Color capa	Color capa	Caso B	Color acum.
Frontera	Color capa	Caso A	Color capa	Color acum.

Figura 7: Operador $\otimes$, el resultado indica el tono a almacenar en el acumulador A_1 tras operar el acumulador A_0 con una capa-acumulador. Los posibles valores del operando capa-acumulador están representados en la columna de la izquierda, mientras que los del operando acumulador A_0 (antes de operarlo) están mostrados en la parte superior

de asignar un número o un intervalo es bastante más abstracto y difícil de discernir. De esta forma podemos diferenciar en el círculo cromático de la figura 8 ciertos intervalos de color que el ojo humano diferencia mejor que otros, definiendo una serie de "colores" para cada zona. Si $\delta = 1$ cada cambio de tono será marcado como límite entre dos zonas de interés y provocaría siluetas en el interior de los objetos debido a esos matices de tonalidad. Es por ello que cada color se introduce en un grupo y se define δ de acuerdo con estos grupos. Además el sistema de visión humano aprecia mejor la diferencia entre los tonos azules y rojos que los naranjas y rojos, por ello δ debe quedar afectada por la distancia de estos grupos en el círculo cromático.

Sea $\delta_2 = d \cdot \delta$, donde d es la menor distancia entre los grupos de los dos tonos que se están comparando y δ_2 el nuevo error tomado para la comparación de tonos. Por tanto, δ_2 sustituye a δ en todas las partes del algoritmo mostrado hasta este punto. Este proceso es prácticamente automático ya que sólo hay que definir una paleta de tonos en el círculo cromático.

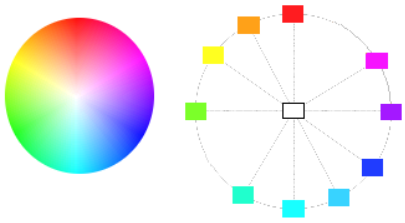

Figura 8: Círculo cromático y clasificación de intervalos de tonos más o menos distinguibles por el ser humano

7. Resultados

El método propuesto ha sido probado sobre un trazado de rayos optimizado con un octree, para el cálculo de normales ha sido implementado una convolución con tres máscaras de Sobel, una para cada coordenada 3D [15]. Todos los datos tienen asociados un valor de transparencia que se ajusta manualmente en el mismo proceso de clasificación. Se ha utilizado un procesador AMD Athlon 800 MHz, con 640 MB de memoria RAM, con una tarjeta gráfica GForce2 FX y un sistema operativo Linux (Debian) con kernel versión 2.4.

Todas las imágenes de esta sección tienen dos zonas de interés, los huesos marcados en verde y los demás tejidos en naranjas, la distancia en el círculo cromático entre los verdes y naranjas es de entre 30 y 40 tonos. Las imágenes se muestran dobles, las siluetas por si mismas en blanco sobre negro y superpuestas con una transparencia del 40 % sobre el volumen coloreado según el modelo de color HSV. La resolución del modelo es de 128x128x128.

El número de capas (acumulador o imagen) utilizadas (r_z) puede parecer una limitación a priori, sin embargo, al depender del número de intersecciones del rayo con el modelo puede ser fácilmente calculable, aun así, suele ser una buena aproximación el tomar la mitad de la resolución del modelo (ver figura 9). Las imágenes, de izquierda a derecha, han sido obtenidas con los valores $r_z = 100$, $r_z = 50$, y $r_z = 10$ respectivamente. La tolerancia de tonalidad es $\delta = 30$.

Puede observarse como si bien un número muy pequeño de capas hace que se pierda información y se detecten pocas siluetas, un número excesivamente alto tampoco aporta una gran mejora, obteniéndose resultados aceptables con un número intermedio de capas.

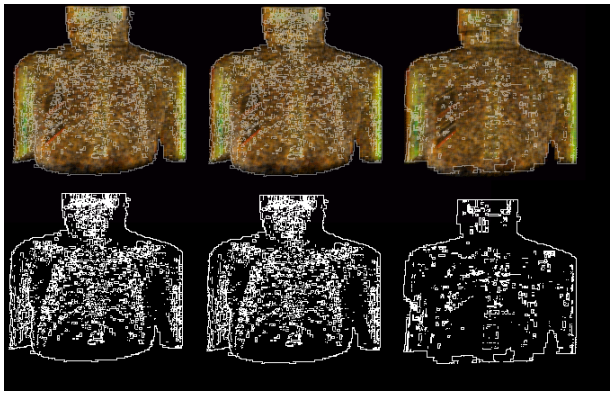

Figura 9: Imágenes obtenidas variado el número de capas que se obtiene del volumen

El parámetro δ tampoco es una limitación ya que puede calcularse de forma au-

tomática (ver figura 10) a partir de la clasificación inicial de tonos, aunque puede ajustarse a mano para tener un mayor control sobre las siluetas detectadas (ver figura 11).

En la figura 10 se ha calculado δ automáticamente resultando el valor 35 como media de los valores 30 y 40 que, como hemos comentado, es la distancia en el círculo cromático entre los tonos verdes y naranjas que usamos. Se observa como se obtiene un resultado aceptable.

Figura 10: Imagen obtenida asignándole δ un valor calculado automáticamente. r_z también a sido calculado automáticamente según el número máximo de intersecciones resultando $r_z = 56$.

En la figura 11 se observa como influye δ en los resultados, de izquierda a derecha tiene los valores 10, 50 y 100 respectivamente. Obviamente, ya que define el margen de tolerancia entre lo que supone estar en la misma tonalidad y tener un cambio de tonalidad, los valores pequeños hacen que aparezca mucho ruido y los valores muy altos hace que se queden siluetas sin encontrar.

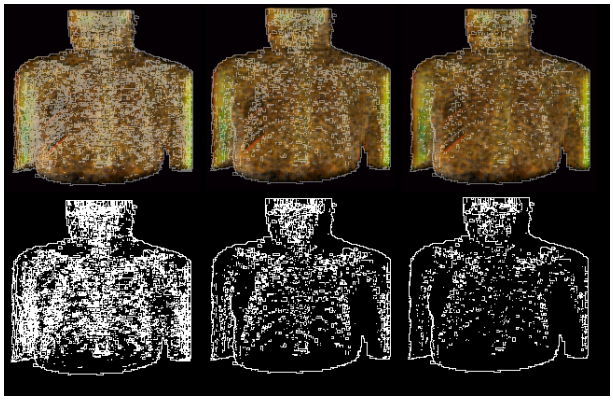

Figura 11: Imágenes obtenidas con diferentes valores de *delta*

8. Conclusiones y trabajo futuro

Hemos presentado un nuevo método de detección de siluetas específicamente diseñado para volúmenes que contienen zonas de interés en el interior de los mismos. Se basa en un proceso previo de clasificación de la información volumétrica y en un sistema de iluminación no fotorrealista.

Hay que tener en cuenta que las siluetas en un modelo volumétrico no pueden aparecer de la misma forma que en un modelo sólido, debido a que las partes de interés en un volumen pueden encontrarse en cualquier zona del interior del objeto mientras que en los objetos sólidos tan solo pueden aparecer en la superficie del mismo. De esta forma las siluetas en un modelo volumétrico deben estar tanto dentro del objeto como fuera del mismo.

Queremos destacar que el método no requiere de la definición previa de un umbral y que por tanto puede obtener simultáneamente las siluetas de objetos diferentes en cuanto al material que lo constituyen. Tan solo hay que definir en el proceso de clasificación las tonalidades que se le asignan a las diferentes zonas de interés. Estas tonalidades se eligen teniendo en cuenta la característica del ojo humano de distinguir mejor unos tonos que otros.

Hemos apreciado que al ejecutar el algoritmo se produce un solapado de las fronteras en las partes internas del modelo debido a las numerosas concavidades y convexidades de los modelos médicos reales. Como trabajo futuro nos hemos propuesto el mejorar las siluetas, de manera que no aparezcan de forma independiente para cada elemento interno del volumen visualizado, sino que todas las zonas de interés sean un todo y la silueta sea única para dicha zona.

9. Agradecimientos

Este trabajo ha sido parcialmente subvencionado por el Ministerio de Ciencia y Tecnología y con los fondos FEDER a través del proyecto TIC2001-2099-C03-02.

Referencias

[1] P. Rheingans, "Expressive Volume Rendering". Journal of WSCG, Vo. 12, No. 1-3, WSCG'2004, February 2-6, Czech Republic, 2004.

[2] K. Brodlie, J. Word, "Recent Advances in Volume Visualization", Computer Graphics Forum, Vo. 20, No. 2, pp. 125-148, 2001.

[3] B. Gooch, P. Sloan, A. Gooch, P. Shirley, R. Riesenfeld, "Interactive Technical Illustration", Proceedings of the 1999 symposium on Interactive 3D graphics, ACM Digital Library, pp. 31-38, 1999.

[4] J. Diepstraten, "Transparency in Interactive Technical Illustrations", Computer Graphics 2002, pp. 317-325, 2002.

[5] A. Gooch, B. Gooch, P.Shirley, E. Cohen, "A Non-Photorealistic Lighting Model For Automatic Technical Illustration", Proceedings of the 25th annual conference on Computer graphics and interactive techniques, ACM Digital Library, p. 447-452, 1998

[6] D. Ebert, P. Rheingans, "Volume Illustration: Non-Photorealistic Rendering of Volume Models", Proceedings of IEEE Visualization '00, pp. 195-202, IEEE Computer Society Press, 2000.

[7] Saito T, Takahashi T, "Comprehensible Rendering of 3-D Shapes," In SIGGRAPH 1990 Conference Proceedings, August 1990.

[8] Z. Nagy, R. Klein, "High-Quality Silhouette Illustration for Texture-Based Volume Rendering", Journal of WSCG, 12, 2004.

[9] M. Levoy, "Efficient Ray-Tracing of Volume Data". ACM Transactions on Graphics, Vo.9, N. 3, pp. 245-261, 1990.

[10] F. Velasco. "Representación y Visualización de Datos Volumétricos", Tesis doctoral, Dpto. de Lenguajes y Sistemas Informáticos, ETS Ingeniería Informática, Universidad de Granada, 2003.

[11] I. Boada. "Towards Multiresolution Integrated Surface Volume Data Representations", Doctoral Thesis, Technical University of Catalonia, Spain, 2001.

[12] J. Foley, Van Dam, S. Feiner, J. Hughes, "Introduction to Computer Graphics", Addison-Wesley, 1996.

[13] D. Seegmiller, "Digital Character Design and Painting", Graphics Series, 2003.

[14] M. Levoy, "Volume Rendering - Display of Surfaces from Volume Data". IEEE Computer Graphics and Applications, Vo. 8, N. 3, p. 29-37, 1988.

[15] A. Jain, "Fundamentals of Digital Image Processing", Prentice Hall, 1989.