Estudio y evaluación de métodos de visualización de imágenes biológicas multidimensionales utilizando OpenGL

Autor:
Leandro A. Bugnon*
(l.bugnon@gmail.com)

Director:
Bioing. Javier Diaz Zamboni*
(javierdiaz@bioingenieria.edu.ar)

*Cátedra de Programación Avanzada, carrera Bioingeniería, Facultad de Ingeniería, Universidad Nacional de Entre Ríos.

Resumen: Las modernas tecnologías de adquisición de imágenes (TAC, RMN, microscopía Confocal, etc.) permiten el registro, no sólo de imágenes tridimensionales, sino de imágenes multidimensionales. Sin embargo, el análisis visual de tales volúmenes de información no es una tarea simple, se requiere de programas de representación gráfica, que oculten la complejidad del sistema de digitalización facilitando el análisis al experto. En este trabajo se han seleccionado para la implementación, análisis y evaluación dos técnicas de representación de imágenes multidimensionales: Proyección de intensidades máximas y una variante de la misma en la cual se atenúa la intensidad de los objetos en función de la distancia de los mismos al observador. Se las implementó utilizando OpenGL vía texturas bidimensionales y texturas tridimensionales, se evaluaron cualitativamente la calidad de las representaciones, y se cuantificaron los tiempos de graficación. Finalmente, se analizaron las ventajas y desventajas del uso de las técnicas y de las herramientas disponibles para implementarlas en un programa.

Palabras Clave: imágenes multidimensionales, proyección de intensidades máximas, texturas, OpenGL

1 Introducción

Las nuevas tecnologías y conceptos de captura de imágenes biológicas han evolucionado de tal manera, que actualmente ya no se habla de una imagen como un arreglo bidimensional (2D), sino que más bien se la trata como estructura multidimensional. Así por ejemplo, tanto la resonancia magnética nuclear como la tomografía axial computada, permiten el registro de un conjunto de datos que describen espacialmente alguna característica del tejido en estudio. Estas modernas

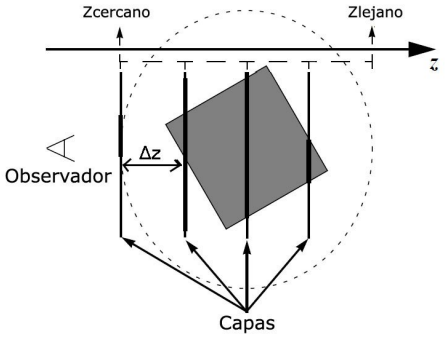
tecnologías requieren de mecanismos especializados para la representación de los datos; que oculten la complejidad de la técnica del registro digital y que muestren de una forma clara los datos, al analista que los interpreta.

Generalmente, una imagen multidimensional está compuesta por una pila de imágenes bidimensionales. De forma general, se puede definir como una función de intensidad multivariable *I=f(x,y,z,t,c)*, donde *x*, *y* y *z* son variables espaciales, *t* es el tiempo, y *c* es la componente espectral o el canal. Visualizar toda esta información en un sólo conjunto puede ser difícil si no se posee algún mecanismo que permita su reconstrucción y representación.

A partir de esta problemática general, se estudiaron dos técnicas de proyección para representar información multidimensional: proyección de intensidades máximas y un derivado de la misma en el que se agrega una ponderación de intensidad en función de la distancia al observador. Los mismos proveen una representación que tiene en cuenta toda la información de las imágenes y permiten verla de forma completa como un conjunto en una imagen proyectada de 2 dimensiones.

1.1 Proyección de Intensidades Máximas (PIM) *[1,2,3]*

Es uno de los principales y mas sencillos métodos para representar imágenes de uso clínico, este mecanismo de graficación tiene como entrada el volumen de datos multidimensionales y genera como resultado una imagen 2D en la pantalla, proyectando rayos (líneas rectas y paralelas, en un sistema ortogonal) a través de dicho volumen y plasmando en pantalla los puntos de máxima intensidad de cada rayo. Generalmente, se cuenta con pilas de un numero discreto de imágenes digitales representando el volumen de datos, con una determinada resolución en pixeles y separadas por una distancia de muestreo Δz, por lo que los pixeles de todo el volumen pueden ordenarse en una estructura de capas o cortes 2D alineados en un mismo eje perpendicular a las mismas, guardando una separación correspondiente a la de muestreo, para utilizarse como entrada del algoritmo de PIM.



[Figura 1]: Proceso de muestreo volumétrico, generando imágenes 2D de un volumen de prueba, tomadas sobre un eje principal, a intervalos Δz de distancia.

Los valores de intensidad de la PIM se obtienen aplicando la siguiente ecuación:

$$I(p) = \max_{0<z<D} I(z) \qquad (1)$$

Siendo *I(p)* la intensidad del píxel *p* de la imagen proyectada (correspondiente a un rayo *p′* que atraviesa *D* píxeles del volumen) e *I(z)* la intensidad del pixel *z* del volumen por el cual pasa el rayo *p′*.

A pesar de la simplicidad del cálculo, la implementacion de la *ec. (1)* resulta muy costosa para volúmenes grandes de información, ya que se deben comparar los valores de cada pixel de todo el conjunto de imágenes para hallar los máximos. Este costo puede resultar prohibitivo si, a efectos de lograr una mejor interpretación de la información, generamos proyecciones desde diferentes perspectivas permitiendo que el volumen rote o se traslade de forma dinámica, con lo cual debemos aplicar la *ec. (1)* repetidamente luego de cada transformación.

Para implementar eficientemente la PIM, resulta una buena alternativa utilizar la técnica de mezcla de colores (o más conocida por su término en inglés: *blending*) de *OpenGL[4,5]*, que permite utilizar eficientemente la capacidad de procesamiento de la tarjeta gráfica y con una fácil codificación del método. El *blending* en *OpenGL* es un estado, el cual habilitado permite operar con los valores guardados en la memoria temporal de color (en inglés: *color buffer)* combinándolos de acuerdo a la siguiente ecuación:

$$C_f = S\,C_s + D\,C_d \qquad (2)$$

Siendo C_f, C_s, C_d los valores de color final, fuente y destino, respectivamente. El valor de intensidad previamente almacenado en el *color buffer* (valor destino) es combinado linealmente con el valor nuevo que ingresa (valor fuente) para determinar el valor que finalmente quedará almacenado en el *color buffer*. Las constantes *S* y *D* determinan el factor de transmisión de cada valor. Para producir la PIM, se debe modificar la *ec. (2)* de forma que el valor final sea siempre el máximo:

$$C_f = \max(C_s, C_d) \qquad (3)$$

De la *ec. (1)*, se puede observar que la técnica PIM sólo recupera las estructuras mas brillantes del volumen, sin ninguna consideración sobre la profundidad de las mismas, por ejemplo si el máximo se encuentra delante o detrás de otras estructuras presentes en el volumen. Esto puede resultar un inconveniente según el caso particular que se presente, por lo cual surge una modificación a este mecanismo que considera el efecto de la profundidad.

1.2 PIM con atenuación de intensidad ponderada por distancia (PIMd)

Este método es una variación de la PIM, donde se agrega un factor de atenuación de intensidad dependiente de la distancia al observador. De esta forma es posible lograr una mejor sensación de profundidad, lo que contribuye a interpretar mejor los datos y corrige artefactos que producen confusión en la PIM.

Aquí de nuevo se tiene el inconveniente del procesamiento en un volumen extenso de datos, ya que se debe aplicar una ecuación de la forma:

$$I(p) = \max_{0<z<D} (d(z)\, I(z)) \qquad (4)$$

Donde *d(z)* es una función de atenuación dependiente de la distancia *z* al observador.

Para implementar la atenuación se utiliza otro estado de OpenGL, el efecto ambiental de neblina (en inglés *Fog);* OpenGL realiza *blending* entre el valor de destino y el valor fuente afectado por la función *d(z)*:

$$C_f = S\, C_s + d(z)\, C_d \qquad (5)$$

La *ec. (5)* puede modificarse de la misma manera que el caso anterior para obtener la siguiente expresión:

$$C_f = max(C_s, d(z)\, C_d) \qquad (6)$$

Así, es posible afectar la intensidad de los objetos a medida que se alejan del observador.

OpenGL provee los siguientes modos de atenuación y las respectivas ecuaciones de *d(z)*:

$$d_{lineal}(z) = \frac{(z_{fin} - z)}{(z_{fin} - z_{inicio})} \qquad (7)$$

$$d_{\exp}(z) = e^{-Dz} \qquad (8)$$

$$d_{exp2}(z) = e^{-2Dz} \qquad (9)$$

Para cada ecuación dada deben definirse los coeficientes correspondientes, que dependerán del efecto de profundidad que se desee lograr. En la *ec. (7)* z_{inicio} y z_{fin} son los valores de distancia al observador que determinan un rango donde la atenuación será lineal ($d_{lineal}(z_{inicio})=1$ y $d_{lineal}(z_{fin})=0$) mientras que en las fórmulas exponenciales *ecs. (8)* y *(9)* se aplican a partir de z=0, con una densidad *D*.

Para almacenar en memoria los datos que serán procesados con OpenGL se puede

utilizar la técnica de mapeo con texturas.

Las texturas se definen como conjuntos de *texeles*, los cuales tienen un espacio de memoria asignado donde se puede almacenar la información de los pixeles o voxeles de las imágenes (color y transparencia) a través de funciones provistas por OpenGL, que junto a otros parámetros, permiten aplicar texturas a una primitiva (puntos, lineas o polígonos) de OpenGL, de forma que al ser dibujada la escena, se encuentran íntimamente relacionadas las coordenadas espaciales de dichas primitivas y los elementos de las imágenes, es decir, las imágenes se aplican *adheridas* a las primitivas con las que se asocian. El mapeo con texturas permite por un lado reducir el costo que implica graficar píxel por píxel, y por otro aplicar de una forma mas sencilla transformaciones sobre el modelo (como son el escalamiento para producir efecto de zoom, modificaciones en la ventana, etc.) sin necesidad agregar códigos de interpolación, ya que las texturas proveen de un conjunto de herramientas eficientemente optimizadas para graficación. Cada texel guarda una relación uno a uno con los pixeles de la imagen que se quiere representar, pero se adapta a cualquier morfología de primitiva donde se aplique, utilizando en caso que sea necesario métodos de filtrado que establecen la correspondencia pixel-texel que sera graficada.

Las texturas de luminancia, compuestas por texeles que solo albergan valores de intensidad, son el caso mas sencillo, utilizadas para aplicar imágenes en tonos de grises de un solo canal (como las imágenes de TAC). Para imágenes en varios canales o componentes espectrales, puede utilizarse texturas RGB, representando en principio hasta 3 canales en una sola textura al mismo tiempo; el proceso de *blending* se aplica a cada canal *RGB* por separado y luego se combinan para mostrarse en pantalla, produciéndose suma de colores en donde se superpongan los canales. En la Figura 3 (Anexo) puede observarse la representación de una imagen de cinco dimensiones tomada de una célula S2 de Drosophila (mosca de la fruta) durante el proceso de mitosis: tres variables espaciales (resolución X-Y 171x196 pixeles y 5 planos focales que determinan 5 valores de profundidad), una variable temporal (51 instantes de tiempo) y 2 canales, representando dos proteínas (marcadas) expresadas durante el proceso.

Se puede realizar el mapeo con texturas de dos o tres dimensiones para generar la estructura que tendrá como entrada el algoritmo de PIM y PIMd.

Si se utilizan texturas 2D, se debe generar una textura por cada capa digitalizada del volumen a representar (ver Figura 1) y cargar cada textura con los pixeles de cada capa. Aplicando las texturas a rectángulos de dimensiones proporcionales a las imágenes 2D, separados una distancia Δz y paralelos entre si, se obtiene la misma configuración espacial de la que fue tomada la imagen, siendo posible la aplicación de los métodos de proyección citados anteriormente.

La principal ventaja de utilizar texturas 2D es que comúnmente se cuenta con hardware optimizado para su procesamiento. Su desventaja es que su uso conlleva a una representación tridimensional incompleta; si se observa el modelo a un ángulo cercano a 90º del eje normal a las capas, las mismas se encuentran aproximadamente tangentes a cualquier linea de visión; dado que son capas bidimensionales, se pierde la representación de la imagen. Una alternativa posible para solucionar este problema

es reconstruir las capas de manera que estén siempre aproximadamente normales a la linea de visión. Sin embargo, esto no es aplicable si se desea una interacción dinámica del analista con los datos, ya que se deben reordenar los pixeles de todo el volumen de datos en nuevas texturas, lo que conlleva un costo computacional elevado.

Una moderna alternativa, son las texturas 3D, las cuales son inherentemente más costosas que su par bidimensional, pero permiten definir un volumen virtual donde cada texel tiene dimensión *[pixel*pixel*Δz]*, de forma que cada punto dentro del espacio está definido por un valor de intensidad. De esta forma es posible cargar todas las imágenes en un solo bloque (que a comparación de las texturas 2D, debían individualizarse por capas) para lograr una representación del volumen discreta pero sin espacios vacíos.

Es posible aplicar texturas 3D a primitivas 2D, definiendo las coordenadas para ambas, y en los puntos donde coinciden, los pixeles de las primitivas toman los valores del volumen virtual de la textura, lo cual permite generar capas en cualquier dirección con un gasto computacional mucho menor que el requerido para cargar nuevas texturas.

2 Metodología

Se trabajó en una PC con un procesador AMD Phenom 8650 de 2,3 Ghz, 2 GB de memoria RAM DDR2 y placa de video GeForce 8200 de 512 Mb.

Los algoritmos y métodos detallados en este trabajo fueron implementados en lenguaje Java, mediante la extensión (*PlugIns*) del programa de análisis y procesamiento de imágenes, *ImageJ[6,7]*. Esta aplicación facilita el manejo de diversos formatos de imagen, permitiendo concentrar el trabajo solo en la implementacion de los métodos de proyección. Los *PlugIns* fueron programados y probados en el entorno de desarrollo *Eclipse[8]*.

Para medir la eficiencia de los métodos de graficacion se utilizó una herramienta diseñada para *Eclipse*, *Eclipse Test and Performance Tools Platform (TPTP)[9]*. Debido a que estos métodos son llamados cada vez que se produce algún cambio en el modelo, modificación de la ventana o refresco del monitor, su consumo es importante si se desea que la orientación del volumen pueda modificarse dinámicamente.

Para implementar *blending*, *fog* y texturas se habilitaron los estados correspondientes utilizando la función glEnable(GLenum capability). La función de *blending* se modifica con glBlendEquation(GLenum mode), donde el parámetro mode debe ser GL_MAX para obtener la forma de la *ec. (3)*. Para definir la forma de la ecuación de Fog se tiene glFog*(GLenum pname, GLint param), siendo pname las variables a modificar y param su valor para establecer las *ec. (7),(8)* o *(9)* y sus parámetros.

En cuanto a la implementacion de las texturas, se asignan en la memoria de texturas los datos de imagen con el método glTexImage*(), cuyos parámetros más importantes determinan el tipo de textura (2D o 3D), tamaño de la textura (en texeles), el formato interno y el tipo de dato de la textura y de los pixeles de la imagen

(luminancia, RGB, etc.) y por ultimo un puntero a una dirección de memoria (*buffer*) de donde se encuentran almacenados los datos de las imágenes. Esta textura se aplicará a la geometría especificada a través de las coordenadas de textura glTexCoord*(), generalmente como valores flotantes en el rango [0.0 – 1.0] para las dimensiones *s*, *t* y *r* (análogas a *x*, *y*, z, respectivamente), determinando una relación geométrica entre los texeles y pixeles.

Usualmente los pixeles no coinciden exactamente con los texeles, OpenGL posee métodos de filtrado que arreglan esta correspondencia. Si se habilita el estado GL_NEAREST con la función glTexParameteri() cada pixel toma los valores del texel mas cercano; con GL_LINEAR se realiza una interpolación lineal, implicando bordes mas lisos a un costo mayor. También se puede utilizar la función glHint(GLenum target, GLenum mode) para el *blending* y *fog*, con las opciones GL_NICEST (mejor visualización) o GL_FASTEST (mas rápido).

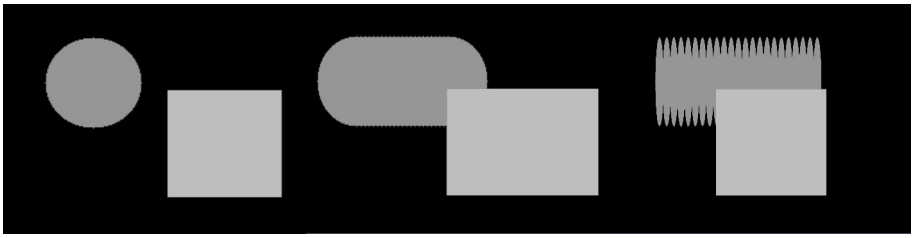
Para la evaluación de los métodos de representación y sus modos de implementacion (utilizando texturas 2D y 3D) se generaron imágenes multidimensionales de prueba, pilas de imágenes en las cuales se dibujaron con intensidades constantes dos objetos de distinto brillo (un prisma y un cilindro). Se analizó la calidad de visualización en los dos tipos de proyecciones y se evaluó la eficiencia de reconstrucción con texturas 2D y 3D, respectivamente.

La herramienta TPTP hace un seguimiento del programa, cuantificando las llamadas a la función de graficación y su consumo de tiempo promedio. Para normalizar la comparación de los métodos, se codificaron las funciones de graficación en diferentes *PlugIns* para cada cada algoritmo de proyección a evaluar, con texturas 2D y 3D. Dado que el costo computacional de la graficación es variable según la orientación del volumen, se realizó repetidamente la proyección del volumen rotándolo sobre un eje una vuelta completa, con una variación angular de un grado. De esta forma, se obtuvo el seguimiento de 360 llamadas a la función de graficación, constituyendo la muestra para el análisis.

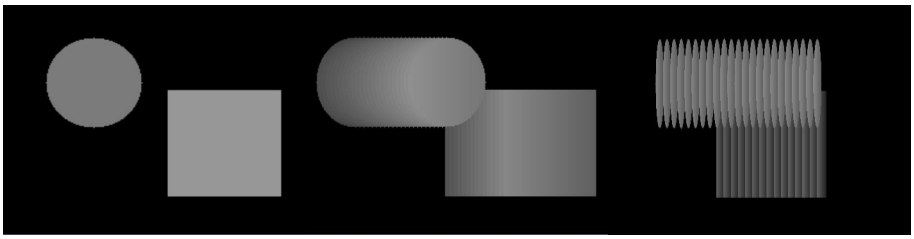
3 Resultados

Las imágenes de la Figura 2 corresponden a proyecciones del fantoma 3D generado para las pruebas. De las mismas se puede observar que la la técnica de PIMd permite un mejor análisis respecto a la ubicación de las estructuras presentes. En la Figura 4 (Anexo) puede observarse la aplicación de la PIMd a un set de imágenes de resonancia magnética del tipo T1 de la cabeza, donde se muestran las ventajas de utilizar texturas3D; debido al numero de planos tomados (139) es posible generar una reconstrucción lateral de calidad apreciable.

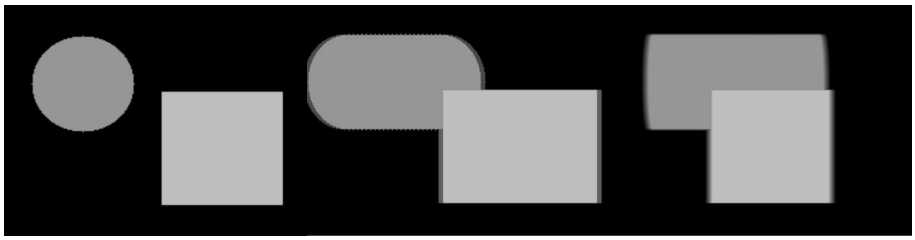
En la Tabla 1 se volcaron los costos de tiempo en mili-segundos de la función de graficacion para diferentes tamaños de imagen, y para las dos técnicas de proyección estudiadas. En la Tabla 2 se detallan los valores de tiempo comparando los diferentes tipos de implementacion con texturas.



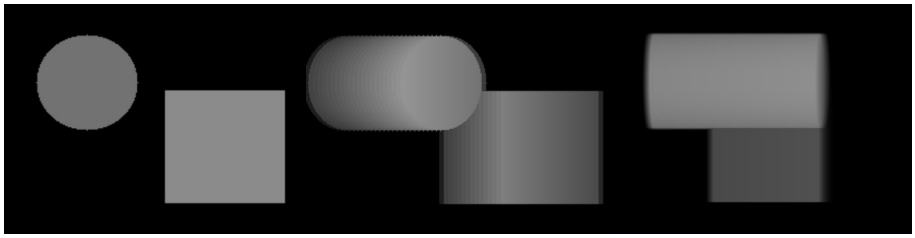
(a)



(b)



(c)



(d)

[Figura 2]: Representaciones comparativas de las imágenes de prueba de dimensiones e intensidades conocidas (el prisma tiene mayor intensidad que el cilindro). De izquierda a derecha rotación sobre el eje vertical en sentido antihorario. (a): PIM con texturas 2D; (b): PIMd con texturas 2D; (c): PIM con texturas 3D; (d): PIMd con textura 3D.

[Tabla 1]: Costo de la función de graficación de ambos algoritmos para diferentes tamaños de imagen implementando texturas 2D.

Tamaño *[pixel]*	PIM *[mS]*	PIMd *[mS]*
400x400x27	16	16
256x256x128	15,8	15,9
256x256x256	19,7	22,3
512x512x128	36,5	40,2
512x512x256	63	71,3

[Tabla 2]: Costo de la función de graficación aplicando PIMd con texturas 2D y texturas 3D. Se detalla en la segunda columna la implementación de textura 3D sin realizar cambios de orientación de capas y en la tercer columna los valores correspondientes a un método que corrige los defectos de proyección mencionados anteriormente cambiando la orientación de las placas a partir de un angulo +/- 55º del de proyeccion original.

Tamaño *[pixel]*	T2D *[mS]*	T3D *[mS]*	T3D Mod. *[mS]*
400x400x27	16	16,2	18,9
256x256x128	15,9	16,3	21,5
256x256x256	22,3	24,6	35,2
512x512x128	40,2	49,6	68,9
512x512x256	71,3	84,4	122,3

4 Conclusiones

Los métodos de proyección presentados han demostrado ser de utilidad para reconstruir imágenes multidimensionales provenientes de diferentes sistemas de adquisición (ver Figuras 3 y 4, Anexo). Pueden reconstruirse conjuntos de imágenes de alta resolución con una computadora de prestaciones medias, ya que se hace uso de los recursos que brinda OpenGL. Sin embargo, se requiere de una tarjeta gráfica con acelerador para descongestionar el uso del CPU.

Por su naturaleza, la PIM lleva aparejada un cierto efecto de confusión al no poder distinguir separación entre estructuras brillantes, o incluso pueden ocluirse entre ellas. El hecho de permitir movilidad al volumen graficado reduce en gran medida esta

confusión; se forma una imagen conceptual mucho más clara si dinámicamente se efectúan rotaciones en torno a un eje central, de aquí la importancia de que la función de graficación pueda ejecutarse a una velocidad que permita una reconstrucción en tiempo real.

En casos donde existan estructuras de importancia en todo el volumen, puede ser deseable la evaluación sobre el estado de alguna de ellas en particular. Por lo dicho anteriormente y remitiéndonos a la *Figura 2*, el uso de PIM crea artefactos que inducen a confusión fácilmente, prestando el caso a suponer erróneamente que una estructura mas brillante podría estar más cercana al observador. En estos casos la PIMd permite sortear estos inconvenientes, no sólo separando las estructuras conflictivas, sino también marcando de forma mas nítida los bordes, sin mayor costo agregado. Modificando correctamente las *ecs. (7)*, *(8)* y *(9)* pueden fijarse la tasa de atenuación en función de la distancia y sus umbrales, lo que posibilita tanto observar la información completa como una parte de ella, distinguiéndola fuertemente de las demás. Ademas puede seccionarse el volumen de datos en el sentido que se desee de forma muy rapida, permitiendo separar diferentes elementos.

Las texturas 3D son una componente relativamente nueva de OpenGL. Han dado muy buenos resultados, a pesar de su costo inherentemente mayor, permite reconstruir un volumen de datos en su totalidad eligiendo convenientemente en qué condiciones cambiar la orientación de las placas para la proyección. Es aquí donde puede ensayarse algún método de mejora de eficiencia, evaluando la relación costo/beneficio de la operación. Una posibilidad es encontrar una imagen que represente a varias secciones*[1]* tomando las intensidades máximas del conjunto, de manera de reducir el número total de capas graficadas.

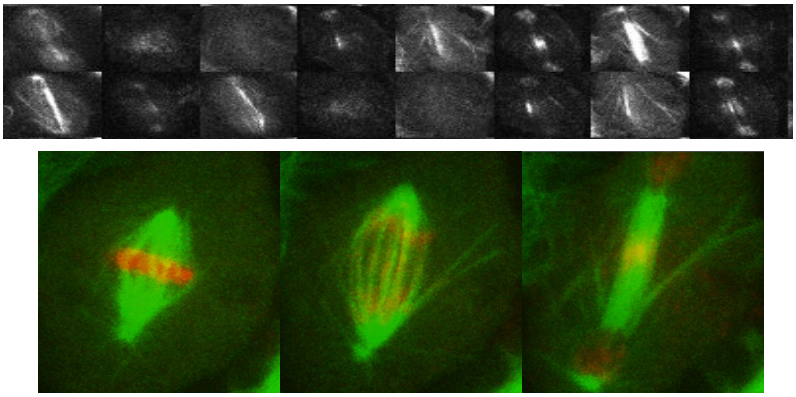
Se puede comentar también que el uso de los elementos citados permite una sencilla implementación; particularmente se pueden aprovechar las facilidades de la maquina virtual Java y el estándar OpenGL para proveer una alta compatibilidad en cualquier plataforma.

Como continuación de este trabajo se pueden estudiar e implementar métodos mas modernos de visualización*[2],* para conseguir representaciones más intuitivas y robustas, que permitan tanto estudios cualitativos como también cuantitativos para diferentes aplicaciones clínicas.

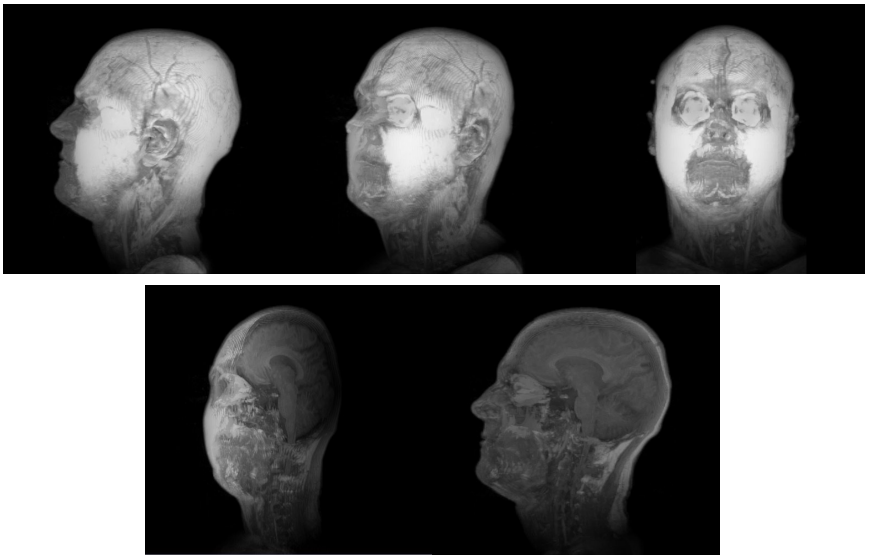
Referencias

1. Shareef, N., Crawfis, R.: A view-dependent approach to MIP for very large data, Dept. of CIS, Ohio State University, Columbus, OH (2007)
2. Fishman, E.K., Ney D.R., Heath D.G., et al: Volume rendering versus maximum intensity projection in CT angiography: what works best, when, and why. *Radiographics(*2006) 26(3):905-22
3. Mroz, L., Hauser, H., Gröller, E.: Interactive High-Quality Maximum Intensity Projection. *Computer Graphics Forum(*2000);19(3):341-350.
4. Wright, R. S. Jr., Lipchak, B., Haemel, N.: OpenGL SuperBible, Comprehensive Tutorial and Reference. Addison-Wesley, 4th Edition (2007).
5. Neider, J., Davis, T.: OpenGL Programming Guide alias The Red Book, Addison-Wesley, 2nd Edition (1997).
6. Burger, W., Burge, M. J.: Digital Image Processing, An algorithmic introduction using Java, Springer (2007)
7. ImageJ, Image Analisis and Prosesing in Java, http://rsbweb.nih.gov/ij
8. Eclipse Galileo, http://www.eclipse.org
9. TPTP, Eclipse Test and Performance Tools Platform, http://www.eclipse.org/tptp

Anexos



[Figura 3]: PIM de una pila de imágenes tomadas durante la mitosis celular. Arriba se observa algunos elementos de la pila, mostrando la complejidad que se puede obtener para realizar una evaluacion cualitativa. Abajo, se muestra la célula en los tiempos 3, 30 y 38, donde se identifican claramente el material genético (rojo) y el citoesqueleto celular (verde). Cortesía de Eric Griffins (imagenes libres en el website de ImageJ*[7]*)



[Figura 4]: Proyecciones de resonancia magnética nuclear tipo T1 de la cabeza humana, en cortes sagitales, cortesía de Jeff Orchard (http://www.cs.uwaterloo.ca/~jorchard/mri/). Arriba: Se observa la reconstrucción con PIMd y texturas3D, para los angulos 0º,45º y 90º respectivamente. Abajo: La misma técnica tomando sólo la mitad derecha del modelo. Se observan claramente estructuras internas, antes ocluidas debido al alto brillo de los tejidos adiposos de la piel.