CEIG - Spanish Computer Graphics Conference (2012)
Isabel Navazo and Gustavo Patow (Editors)

A New Approach to Explore Integral Field Spectroscopy Data

C. Romo[1], J.C. Torres[1], A. León[1], E. Pérez[2]

[1]Lab. Realidad Virtual, Universidad de Granada, Spain
[2]Instituto de Astrofísica de Andalucía, CSIC, Granada, Spain

Abstract

Integral field spectroscopy (IFS) is a technique that is widely used in astronomy. It gives simultaneously the spectra of each spatial sampling element of a given field, generating a large amount of information that is also known as data cube. Visualizing and exploring these data cubes in a flexible and an intuitive way is a challenge.
Although the information collected is three-dimensional (representing the I(x, y, λ) function) and the scientific community generally recognizes the need to use 3D visualization tools, two dimensional tools are usually used to visualize the data cubes and very few 3D exploration techniques have been proposed specifically for astrophysics. This paper presents an interactive method to explore IFS data and two different GPU implementations of it.

Categories and Subject Descriptors (according to ACM CCS): I.3.6 [Computer Graphics]: Methodology and Techniques—Interaction techniques I.4.10 [Image Processing and Computer Vision]: Image Representation—Volumetric J.2 [Physical Sciences and Engineering]: Astronomy—

1. Introducción

Uno de los grandes retos en el conocimiento del universo desde el punto de vista de la investigación astrofísica es el dilucidar los procesos físicos responsables de la formación y evolución de las galaxias. Para ello se han realizado cartografiados profundos del cielo que permiten representar las propiedades de las galaxias a diferentes distancias cosmológicas, mostrando las distintas etapas en la historia de la formación estelar en el universo. Estudiando cómo cambian las propiedades morfológicas, químicas y espectroscópicas de las galaxias se han obtenido claves importantes para determinar los procesos evolutivos en la historia del universo.

La tecnología actual permite realizar cartografiados tanto en modo imagen como en modo espectroscópico. Aunque el modo imagen permite observar grandes áreas del cielo y detectar un gran número de galaxias, los cartografiados espectroscópicos son más potentes, ya que un espectro proporciona mayor información astrofísica y más detallada que una imagen. La espectroscopia de campo integral [AS06], también denominada espectroscopia 3D, es una técnica observacional que permite aunar las ventajas de las observaciones en imagen y espectroscopia, y obtener información de las poblaciones estelares en las galaxias. Un mapa completo de dichas propiedades para los diferentes tipos de galaxias permitirá tener una visión más clara de los procesos que gobiernan la evolución de éstas.

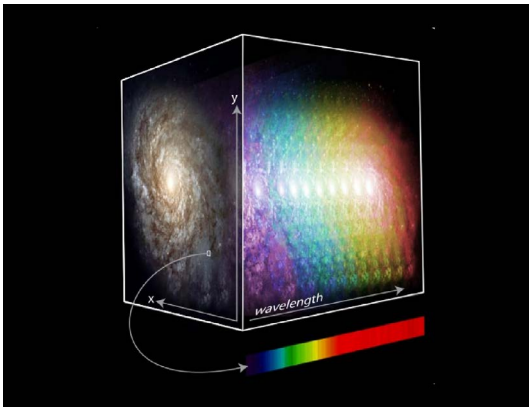

Figure 1: *Representación de un cubo de datos obtenido con la técnica de espectroscopia de campo integral. En cada píxel de la imagen se obtiene el espectro electromagnético del objeto observado [Wet10].*

La complejidad y tamaño de los conjuntos de datos que se obtienen de las observaciones astronómicas, de su análisis y de las simulaciones, está creciendo a un ritmo acelerado. Explorar estos datos, analizarlos y descubrir nuevas estructuras es cada vez más difícil utilizando solamente las

DOI: 10.2312/LocalChapterEvents/CEIG/CEIG12/001-010

herramientas tradicionales de visualización basadas en imágenes y gráficas bidimensionales. Por este motivo, es necesario disponer de herramientas de visualización interactivas adaptadas al problema.

La espectroscopia de campo integral permite obtener de forma simultánea el espectro electromagnético para cada uno de los píxeles de la imagen captada por el telescopio, obteniendo cubos de datos que contienen una gran cantidad de información (ver figura 1). Los cubos de datos, que normalmente vienen dados en formato FITS (*Flexible Image Transport System*) [WGH81], se pueden representar de forma natural como un grid 3D. Esto sugiere la utilización de técnicas de visualización de volúmenes, tal y como se ha llevado a cabo en [Joh08] y [PvdH01].

Actualmente la comunidad científica reconoce de forma generalizada la necesidad de utilizar herramientas de visualización 3D en astronomía [BGH*07, Tyl11, Kau07].

No obstante, y a pesar de la intensa actividad realizada en los últimos años, sólo se ha propuesto un número muy reducido de técnicas de visualización 3D para este propósito, que no permiten visualizar de forma suficientemente flexible los cubos de datos. Hassan ha realizado una revisión exhaustiva de las técnicas de visualización propuestas para datos astrofísicos y de los problemas no resueltos en este ámbito [HF11].

En este trabajo se propone un nuevo método de exploración que pretende adaptarse a la naturaleza de la información a representar, la cual se caracteriza por tener un gran tamaño y por la heterogeneidad y desproporción de sus dimensiones.

En la siguiente sección se realiza una breve descripción de algunas de las herramientas de análisis, visualización y exploración que se utilizan actualmente en este campo. En las secciones 3 y 4 se presenta el método exploración, basado en un algoritmo de ray casting [Lev90], y se detallan las dos implementaciones realizadas del mismo. Por último, en la sección 5, se presenta un análisis comparativo entre ellas.

2. Herramientas de visualización en astrofísica

A pesar de la naturaleza multidimensional de los datos astronómicos y astrofísicos, generalmente se utilizan herramientas de visualización basadas en imágenes 2D. Algunas herramientas diseñadas inicialmente para visualizar información bidimensional son DS9, QFitsView, Karma, GAIA o Aipsview.

No obstante, en los últimos años se han realizado diversas iniciativas para aplicar software de visualización médica a modelos 3D en astronomía. El proyecto *Astronomical Medicine* [BGH*07] *(http://am.iic.harvard.edu)* se centra actualmente en la aplicación de técnicas de visualización y análisis de imágenes médicas a datos astronómicos tridimensionales. *3D Slicer* y *OsiriX* son herramientas de visualización médica que se están utilizando y adaptando para este propósito.

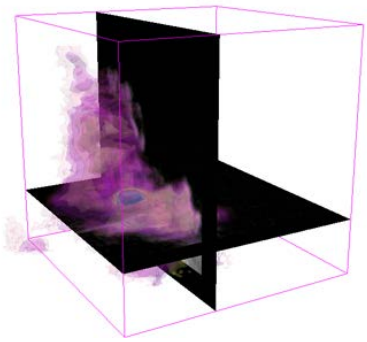

Figure 2: *Región de formación de estrellas IC348 visualizada en 3D Slicer.*

3D Slicer (http://www.slicer.org) es un visualizador médico desarrollado en el Laboratorio de Planificación Quirúrgica del *Brigham and Women's Hospital* y diseñado inicialmente como herramienta para cirujía guiada por imagen y para diagnóstico. Es capaz de visualizar isosuperficies de volúmenes y cortes de los mismos en diferentes direcciones. La figura 2 muestra la visualización con 3D Slicer de un modelo usando isosuperficies.

OsiriX (http://www.osirix-viewer.com/) es una aplicación de visualización médica desarrollada por los doctores Osman Ratib (UCLA) y Antoine Rosset (Universidad de Ginebra), creada inicialmente para visualizar y analizar conjuntos de datos radiológicos en ordenadores personales. Al igual que 3D Slicer, está implementado sobre los toolkits *ITK* y *VTK* y es de código abierto. OsiriX está implementado sólo para *Mac Os X* y actualmente no lee directamente los modelos FITS.

Hongwei propone la exploración de los modelos combinando visualización volumétrica con cortes 2D [HCWH08]. El sistema que ha desarrollado permite visualizar un submodelo, especificado mediante seis planos de corte, y ubicarlo interactivamente en un minimapa del modelo completo.

Otra aplicación de visualización de volúmenes utilizada en astrofísica es *VisIt (https://wci.llnl.gov/codes/visit/)*. Este sistema, desarrollado por el Departamento de Energía de EEUU para la visualización de resultados de simulaciones, incorpora varias herramientas de exploración que permiten visualizar secciones del modelo realizadas con diferentes elementos, tales como planos, esferas y conos.

3. Método propuesto

Como se comentó en secciones anteriores, un cubo de datos se puede representar como un grid 3D, y se puede visualizar utilizando técnicas de visualización directa. Sin embargo, los especialistas necesitan algo más que imágenes; necesitan poder explorar esta información.

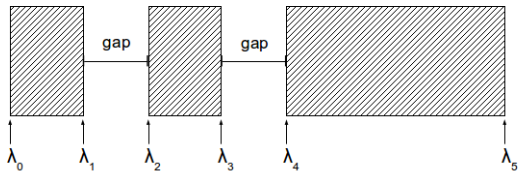

Figure 3: *Los sectores del cubo correspondientes a los intervalos de longitud de onda seleccionados se muestran a una distancia fija, gap.*

Los cubos de datos no representan objetos físicos, ya que una de sus dimensiones es espectral. Además, los modelos con los que se trabaja son muy desproporcionados, siendo la dimensión espectral entre 20 y 40 veces mayor que las dimensiones espaciales. Por tanto, la exploración debe llevarse a cabo usando técnicas de propósito especial.

En este trabajo proponemos un método de exploración basado en la visualización de un subconjunto del cubo de datos determinado por una secuencia de intervalos de longitudes de onda

$$[\lambda_0,\lambda_1],...,[\lambda_{i-1},\lambda_i],...,[\lambda_{n-1},\lambda_n] \qquad \lambda_i < \lambda_{i+1}$$

que se muestran con una distancia fija de separación (ver figura 3.

El método permite reducir el conjunto de datos visualizados en base a la longitud de onda, algo que puede ser especialmente útil debido a la estructura de los espectros de emisión y absorción de la materia. Los espectros de emisión de los elementos presentan picos característicos, es decir, emiten luz sólo a determinadas longitudes de onda. Por tanto, cuando se analiza la presencia de un determinado elemento en una zona del espacio, la observación puede centrarse únicamente en dichas longitudes de onda. Adicionalmente, la eliminación del espacio entre intervalos permite una fácil comparación entre ellos, pudiendo realizar el estudio de una manera más cómoda.

Esta forma de restringir la información se puede combinar fácilmente con la suministrada por la función de transferencia en la visualización directa, con la cual se pueden identificar y seleccionar zonas por nivel de intensidad. Esto permite focalizar la atención en conjuntos de datos más reducidos que el modelo completo, sin llegar a la simplificación de un simple corte o imagen 2D.

Por otra parte, las longitudes de onda obtenidas están modificadas por el efecto Doppler en función de la velocidad del movimiento relativo de la galaxia observada. Por tanto, lo que en principio se corresponde con una frecuencia concreta de emisión, se observará dentro de un intervalo de longitudes de onda más o menos amplio dependiendo de las diferencias de velocidad en la galaxia. Esto hace que la visualización de intervalos sea más útil que la visualización de cortes.

4. Implementación

La técnica de exploración propuesta en este trabajo se basa en la visualización directa de un modelo volumétrico generado a partir del cubo de datos, que viene dado en formato FITS. FITS es un formato estándar de archivo para almacenar conjuntos de datos astronómicos consistentes en matrices multidimensionales y tablas bidimensionales de información. [WGH81]

Para la visualización directa del modelo se ha utilizado un algoritmo de ray casting implementado en GPU.

La interacción se realiza usando un widget en el que el usuario puede especificar interactivamente los intervalos de longitudes de onda que desea visualizar. Se pueden crear, editar y borrar los intervalos de longitud de onda simplemente moviendo los límites de dichos intervalos.

4.1. Visualización

El algoritmo de ray casting empleado está basado en el descrito en [Ngu07]. Las principales diferencias con respecto a este último son el cálculo de la dirección de rayo, el método de recorte con la pirámide de visión y la visualización de isosuperficies.

Cuando se lanza un rayo se necesita saber el punto por el que entra en el volumen y la dirección en la que se mueve. Una vez se tenga la dirección del rayo, se podrá recorrer para determinar el color final de cada píxel. En el algoritmo utilizado se calculan los puntos de entrada y salida del rayo y se almacenan en dos texturas 2D, una para las coordenadas de los puntos de entrada y otra para las coordenadas de los puntos de salida. Restando el valor de estas dos texturas en cada píxel se obtiene el vector director del rayo. Por otro lado, se utiliza una textura 3D para almacenar el volumen.

Este algoritmo realiza tres pasadas en GPU, cada una de las cuales ejecuta un shader específico. Las dos primeras pasadas crean las texturas 2D y en la tercera se realiza el seguimiento del rayo para cada píxel.

En la primera pasada se dibuja la caja contenedora del volumen ocultando las caras delanteras, utilizando como color las coordenadas paramétricas del volumen. El resultado que se genera es una textura 2D del tamaño del viewport que codifica como color las coordenadas de salida del rayo en cada píxel. En la segunda pasada se realiza el mismo procedimiento ocultando en este caso las caras traseras, obteniendo en la otra textura las coordenadas de entrada del rayo en el píxel.

En la tercera pasada se dibuja un rectángulo que cubre toda la ventana de dibujo para forzar al *fragment shader* a procesar cada píxel de ésta una vez. Para cada texel, el *fragment shader* resta los valores almacenados en las texturas creadas en las dos primeras pasadas, obteniendo así el vector director del rayo que pasa por el píxel.

Teniendo el punto de entrada, $\vec{p}_e$, y el vector director, $\vec{d}$, se recorre el rayo a lo largo del volumen muestreando los valores de propiedad. Para obtener las coordenadas, $\vec{p}$, de cada muestra se utiliza la ecuación paramétrica

$$\vec{p}(t) = \vec{p}_e + \vec{d}\,t \qquad t \in [0,1] \tag{1}$$

El incremento entre muestras debe ser suficientemente pequeño para que se muestren todas las celdas del volumen cruzadas por el rayo. Por otro lado, si el número de pasos es constante, la densidad de muestreo será distinta en función de la longitud del vector director, haciendo que las regiones en las que se corten porciones de volumen más pequeñas se vean más opacas. Este problema se ha resuelto realizando un número de pasos proporcional a la longitud del vector director.

Para cada valor de propiedad muestreado se toma un color de una paleta de colores que se utiliza como función de transferencia. Los colores resultantes se acumulan según las ecuaciones 2 y 3, utilizando una estrategia *early ray termination* [Lev90] para acelerar el proceso. De esta manera se acumula hasta que el rayo abandone la caja contenedora del volumen o se alcance una opacidad mayor que 0.99.

La acumulación de color y opacidad para cada nueva muestra se realiza usando las expresiones

$$C'_{rgb} = C_{rgb} + (1 - C_\alpha) \cdot M_{rgb} \cdot M_\alpha \tag{2}$$

$$C'_\alpha = C_\alpha + M_\alpha \cdot (1 - C_\alpha) \tag{3}$$

siendo C'_{rgb} y C'_α el color y opacidad resultante, C_{rgb} el color acumulado para el rayo antes del último muestreo, C_α la opacidad anterior, M_{rgb} el color y M_α la opacidad obtenidos de la paleta de colores a partir de la intensidad en el punto muestreado.

Una vez que se ha recorrido el rayo, el color obtenido se combina con el color de fondo de igual forma, usando como M_{rgb} y M_α los valores del color de fondo.

Para dibujar isosuperficies se comprueba si el intervalo de valor de propiedad definido por los dos últimos puntos muestreados en el volumen contiene al valor umbral. En estos casos se estimará el punto de intersección y se calculará la normal en dicho punto, acumulando un color obtenido aplicando un modelo de iluminación local en ese punto.

4.1.1. Clipping

Cuando la cámara se acerca al modelo, su caja contenedora puede salir de la pirámide de visión cruzando el plano de recorte delantero. En este caso no se genera información en la parte de la textura delantera correspondiente a la zona de intersección de la caja con el plano, con lo cual no se tienen las coordenadas de entrada del rayo y, por consiguiente, no se calcula correctamente el vector director del rayo. Este problema se ha resuelto dibujando el polígono de intersección de la caja contenedora con el plano de recorte delantero. Este polígono, que se calcula y se dibuja en la segunda pasada, tendrá como vértices la intersección de las aristas de dicha caja con el plano de recorte delantero. Es decir, en cada frame se dibuja en la segunda pasada las seis caras de la caja contenedora y el polígono intersección de esta con el plano de recorte delantero. Esto garantiza que se generen las calculen las coordenadas de entrada del rayo en la parte visible de la caja contenedora.

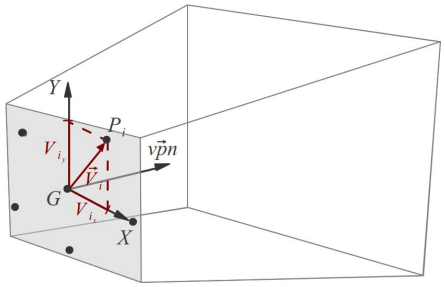

Figure 4: *Sistema de coordenadas local para el cálculo del polígono de intersección entre la caja contenedora del volumen y el frustum.*

Para obtener el polígono de intersección se calcula la ecuación del plano de recorte a partir de los parámetros de la cámara. A continuación se determinan las intersecciones de éste con las doce aristas de la caja contenedora y se ordenan con respecto a un sistema de referencia local al plano de recorte, cuyo origen es el centro de gravedad, G, del conjunto de puntos. Esta lista ordenada define el polígono convexo intersección de la caja contenedora con el plano de recorte delantero.

Para calcular las coordenadas de cada uno de los puntos de intersección en el nuevo sistema de referencia, se construyen los vectores unitarios V_i a cada punto P_i desde G.

y se calcula su producto escalar con los vectores unitarios en la dirección de los ejes X e Y, tomando como X la dirección al primer punto de intersección y como Y el producto vectorial de ésta con el $\vec{vpn}$ (ver figura 4).

El conjunto se ordena dividiéndolo en dos grupos: los que tienen V_{i_y} positivo, y los que lo tienen negativo. Estos grupos corresponden a la mitad superior e inferior del polígono. La parte superior se ordena por V_{i_x} decreciente y la inferior por V_{i_x} creciente. Uniendo las dos poligonales se obtiene el polígono de recorte que se dibuja en la segunda pasada junto con la caja contenedora.

De este modo, calculamos la intersección de la caja contenedora del volumen con el plano de recorte delantero sin

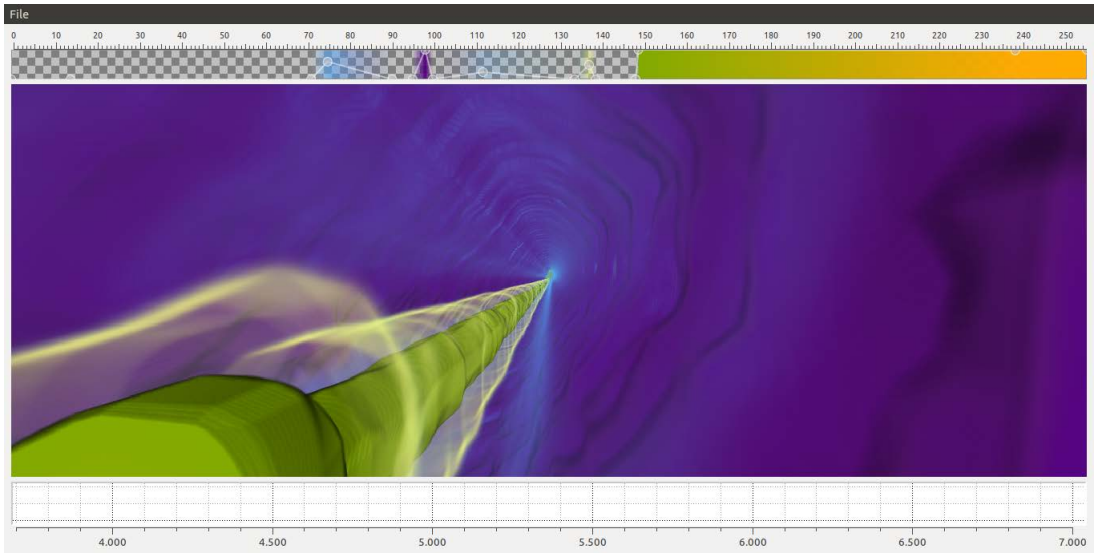

Figure 5: *Visualización del modelo cortado por la pirámide de visión.*

necesidad de utilizar información topológica de dicha caja contenedora.

La figura 5 muestra el resultado de la integración del algoritmo de recorte. Se puede observar que la cámara se encuentra en el interior del modelo.

El algoritmo descrito dibuja el modelo completo. En el método de exploración propuesto es necesario dibujar sólo los intervalos especificados por el usuario en el widget de definición de intervalos. En este trabajo hemos realizado dos implementaciones diferentes para realizar esta visualización. Las subsecciones siguientes las describen.

4.2. Algoritmo secuencial

Este algoritmo realiza la visualización de cada intervalo de longitud de onda $[\lambda_i, \lambda_{i+1}]$ como un modelo volumétrico independiente, ejecutando para cada sector del cubo el algoritmo de trazado de rayos. Esto obliga a realizar una composición de las imágenes obtenidas de cada uno de ellos. En los píxeles en los que se proyecte solamente uno de los sectores, el color debe ser el generado para dicho sector. En aquellos en los que se proyecte más de uno, el color debe ser una combinación de los colores de los sectores superpuestos, de forma que el color resultante coincida con el que se hubiese obtenido utilizando las ecuaciones 2 y 3.

El primer problema a resolver es el orden de superposición de las imágenes. Si un sector se encuentra delante de otro, debe ocultar total o parcialmente al que se encuentre tras él. Por este motivo se deben dibujar los sectores en orden inverso a su profundidad, y para ello es necesario calcular la distancia a la que se encuentran del observador.

La segunda cuestión a resolver es la combinación de los colores de los sectores que se superponen en un píxel. Esta combinación se resuelve utilizando *alpha blending*. El color resultante de la superposición de los i primeros sectores sería

$$C_a(i) = C_s(i) + C_a(i-1)\,(1 - C_s(i)_\alpha) \tag{4}$$

donde $C_a(i)$ es el valor RGBA acumulado hasta el sector i y $C_s(i)$ es el valor RGBA del sector i.

Esta expresión coincide con el cálculo del color en las expresiones 2 y 3, dado que el color de fondo tiene opacidad uno.

En este algortitmo es necesario deshabilitar el test de profundidad en la tercera pasada para permitir que se dibuje el rectángulo para todos los sectores.

La figura 6 muestra el resultado final de la utilización del método secuencial visualizando nueve sectores de un cubo de datos.

4.3. Algoritmo directo

El algoritmo descrito anteriormente procesa de forma independiente cada sector, por lo que es de esperar que el tiempo de dibujo aumente con el número de sectores. Para resolver este problema se ha desarrollado un algoritmo que visualiza de forma conjunta todos los sectores del cubo de datos ejecutando una única vez el procedimiento de trazado de rayos. La principal dificultad para realizar este proceso es calcular la ecuación del rayo para cada píxel en el *fragment shader*. Una opción simple sería entregar al *fragment shader*

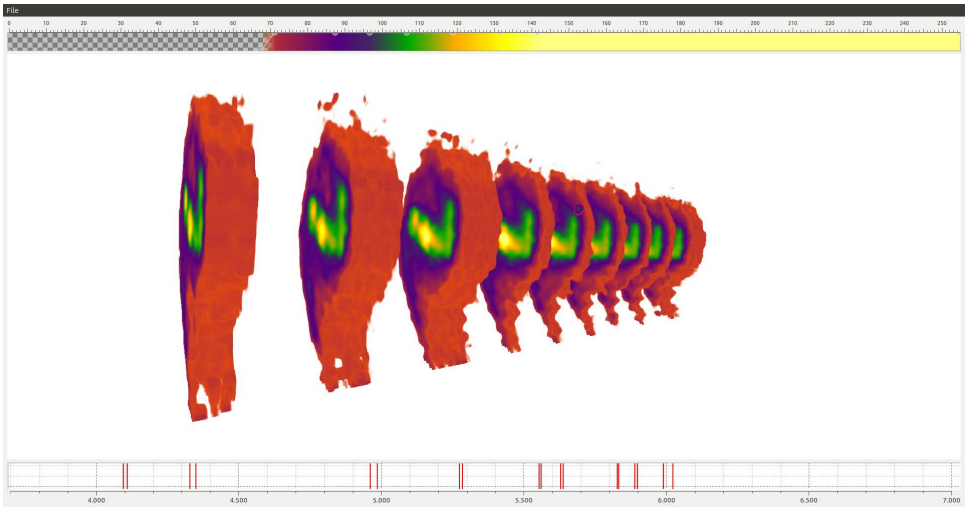

Figure 6: *Visualización utilizando el algoritmo secuencial.*

de la tercera pasada una textura con coordenadas de entrada y otra de salida para cada sector. No obstante, esto implicaría ejecutar varias veces las dos primeras pasadas y transferir a la GPU un número de texturas mayor. En esta implementatión hemos utilizado una única textura de entrada y otra de salida para todo el conjunto de sectores, de forma que el *fragment shader* de la tercera pasada se debe encargar de reconstruir la ecuación del rayo a partir de estas dos texturas.

Calcular la ecuación del rayo equivale a calcular los puntos por los que el rayo entra y sale de cada uno de los sectores del cubo 7. En cierto modo, resolver este problema implica transformar el rayo del sistema de coordenadas de visualización al sistema de coordenadas del modelo volumétrico.

4.3.1. Ecuación del rayo

Para cada píxel se genera un rayo que cruza todas las celdas del modelo que se proyectan en dicho píxel. Para obtener su color final es necesario recorrer el modelo volumétrico, en el que el trazado del rayo es una recta dentro de cada sector. Las rectas de un mismo rayo que atraviesan diferentes sectores, son distintas. No obstante, la dirección en todas ellas es la misma y coincide con el vector director del rayo.

Por tanto, para recorrer el rayo en el modelo volumétrico se deben obtener las ecuaciones de las rectas en los distintos sectores, para lo cual se debe calcular el vector director del rayo y el punto de entrada en cada uno de ellos (caso particular de la ecuación 1).

$$\vec{p}_i(t) = \vec{p_{e_i}} + \vec{d}\,t \qquad t \in [0, t_{s_i}] \tag{5}$$

donde t_{s_i} es el valor del parámetro t en el punto de salida del sector i.

El vector director se puede obtener restando las coordenadas del punto de salida y de entrada en el espacio de visualización (ver figura 7):

$$\vec{d} = \vec{p'_{s_n}} - \vec{p'_{e_1}} \tag{6}$$

donde p'_{e_1} es el punto de entrada al primer sector atravesado por el rayo y p'_{s_n} es el punto de salida del último, ambos en coordenadas de visualización. De estos puntos se dispone de sus coordenadas en el volumen, que se encuentran en las texturas delantera y trasera, por lo que es necesario convertirlas a coordenadas de visualización. Obviamente las coordenadas x e y son iguales en ambos sistemas, ya que no se realiza ninguna transformación que les afecte.

Esto no ocurre con la componente z del vector director. Se puede observar que el incremento de z en el espacio de visualización es igual al incremento en el volumen menos la longitud de los espacios no dibujados entre sectores (zonas en gris en la figura 7) más los espacios de separación entre sectores, g.

$$z'_{s_n} - z'_{e_1} = z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_k \tag{7}$$

donde n es el número de sectores atravesados y Δ_k es la diferencia entre la longitud de los espacios no dibujados entre sectores en el modelo y el gap, g, en la visualización.

$$\Delta_k = z_{e_k} - z_{s_{k-1}} - g \tag{8}$$

Así pues, la expresión del vector director del rayo es la siguiente:

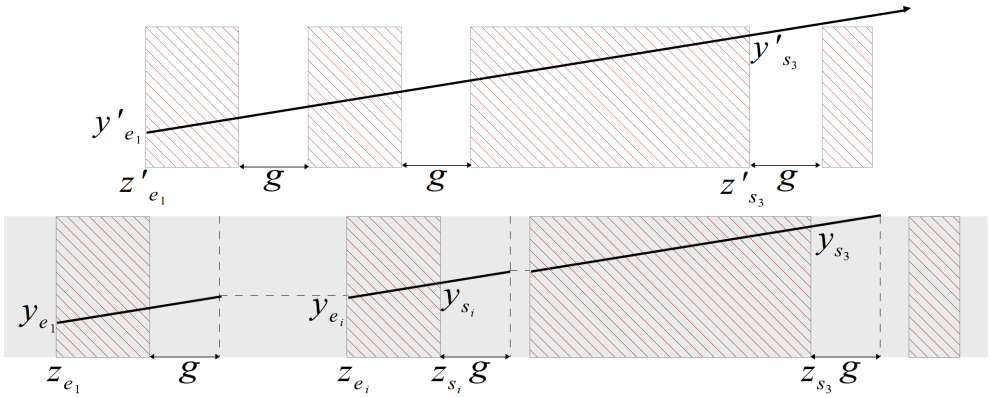

Figure 7: *Equivalencia del recorrido del rayo en el espacio de visualización (arriba) y en el volumen (abajo).*

$$\vec{d} = (x_{s_n} - x_{e_1}, y_{s_n} - y_{e_1}, z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_k) \tag{9}$$

Para calcular el fragmento de rayo que cruza cada sector utilizando la ecuación 5, es necesario obtener también el punto de entrada. Observando las figuras anteriores se puede deducir que el rayo continúa viajando en el espacio de visualización para salvar el espacio g entre sectores. Por tanto, se puede calcular el punto del rayo a distancia g del punto de salida, y éste será el punto de entrada al siguiente sector. Es decir, las coordenadas x e y de entrada a un sector son las del rayo del sector anterior en coordenada $z = z_{s+g}$ (ver figura 7).

$$x_{e_{i+1}} = x_e + \vec{d}\, t_{s+g} y_{e_{i+1}} = y_e + \vec{d}\, t_{s+g} \tag{10}$$

Para calcular el valor del parámetro t_{s+g} se parte de la expresión de z en el sector i

$$z_i(t) = z_{e_i} + (z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_k)\, t \tag{11}$$

Despejando t, queda

$$t = \frac{z_i(t) - z_{e_i}}{z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_k} \tag{12}$$

y sustituyendo para t_{s_i+g}

$$t_{s_i+g} = \frac{z_{s_i} + g - z_{e_i}}{z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_k} \tag{13}$$

Para calcular la ecuación del rayo en cada sector usando estas expresiones es necesario conocer los sectores de entrada y salida y las coordenadas z máxima y mínima de cada sector. Para conocer los sectores de entrada y salida se ha optado por introducir el número de sector como componente α de las texturas generadas en la primera y segunda pasadas, de manera que en la tercera pasada el shader de fragmentos recibirá el número de sector de entrada y de salida para cada fragmento. Por otro lado, dado que los intervalos de longitud de onda de los sectores a dibujar son los mismos para todos los rayos, es posible pasarlos al shader como una variable. Concretamente se envía un array de ternas

$$[\lambda_{i_1}, \lambda_{f_1}, \sum_{k=1}^{1} \Delta_k], ..., [\lambda_{i_j}, \lambda_{f_j}, \sum_{k=1}^{j} \Delta_k], ..., [\lambda_{i_m}, \lambda_{f_m}, \sum_{k=1}^{m} \Delta_k] \tag{14}$$

donde λ_{i_j} es la longitud de onda inicial del sector j, λ_{f_j} su longitud de onda final y $\sum_{k=1}^{j} \Delta_k$ es la sumatoria de la expresión 8 para los j primeros sectores de todo el modelo. La utilización de la tercera componente de la terna se detalla en la sección 4.3.3.

4.3.2. Primera y segunda pasadas

El objetivo de la primera pasada es obtener una textura 2D en la que el color de cada texel se corresponda con las coordenadas de textura de las caras traseras del conjunto de sectores. Para ello se activa el test de profundidad y se dibujan todos los sectores ocultando las caras delanteras.

La función de profundidad es diferente en las dos primeras pasadas, lo que permite obtener la cara la trasera más lejana en la primera pasada y la delantera más próxima en la segunda. Así pues, en la primera pasada se dibujan todos los sectores con la función de profundidad establecida a *GL_GREATER*, almacenando el resultado en la textura 2D.

En la segunda pasada se vuelven a dibujar todos los sectores, esta vez ocultando las caras traseras y habiendo esta-

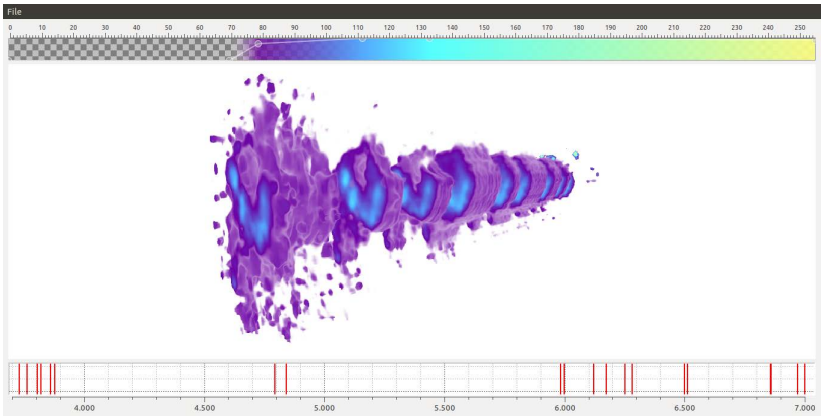

Figure 8: *Visualización usando el algoritmo directo.*

blecido previamente la función de profundidad a *GL_LESS*. Como resultado se obtiene otra textura 2D en la que el color de cada téxel se corresponde con las coordenadas de textura de las caras delanteras de todo el conjunto.

Además, en la componente α de las coordenadas de textura se codifica el número de sector que se está dibujando. De esta manera, en la tercera pasada se puede conocer el sector por el que entra y sale el rayo del conjunto, permitiendo conocer los sectores cruzados y por consiguiente, calcular el vector director.

4.3.3. Tercera pasada

En el shader de fragmentos se consulta la componente α de las texturas 2D correspondientes a las caras delanteras y traseras del volumen para obtener el número de sector por el que entra el rayo y el número de sector por el que sale, respectivamente. Con esta información se conoce qué sectores son atravesados por el rayo y se calcula el vector director de éste a partir de la ecuación 9. Para evitar calcular la sumatoria de los sectores atravesados en el shader, se utiliza la tercera componente del array 14. La sumatoria desde el primer sector atravesado hasta el último es la diferencia entre las terceras componentes de estos dos sectores.

Para cada sector se calcula el punto de entrada del rayo. En el caso de ser el primero, las coordenadas coinciden con las de la textura delantera; en caso contrario, las coordenadas x e y se pueden obtener utilizando las expresiones

No obstante, como se está iterando sobre el rayo, se conocen las coordenadas de salida en el sector anterior, por lo que en el shader se utilizan las siguientes expresiones que son equivalentes:

$$x_{e_{i+1}} = x_{s_i} + \vec{d}t_g y_{e_{i+1}} = y_{s_i} + \vec{d}t_g \tag{15}$$

$$t_g = \frac{g}{z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_{k_i}} \tag{16}$$

El motivo de usar estas ecuaciones es que el cálculo es más simple y sobrecarga menos el shader. Las coordenadas de salida en el sector anterior, x_{s_i} e y_{s_i}, ya se conocen y t_g se puede precalcular por ser el mismo para cada sector. En cambio, t_{s+g} sí varía y habría que calcularlo. La coordenada z del punto de entrada en el siguiente sector, z_{e_i+1}, se obtiene directamente de la primera componente del array de ternas que se pasó al shader de fragmentos (ecuación 14).

Finalmente, con el vector director y el punto de entrada en el sector, se calcula la ecuación 5 para recorrer el rayo y muestrear los valores de intensidad.

El procesamiento en el sector concluye cuando el color tiene opacidad mayor que 0.99 o cuando el rayo lo abandona. El valor del parámetro t en el punto de salida del rayo en el sector i se puede obtener operando en la expresión 12:

$$t_{s_i} = \frac{z_{s_i} - z_{e_i}}{z_{s_n} - z_{e_1} - \sum_{k=2}^{n} \Delta_k} \tag{17}$$

Cuando el sector es atravesado cruzando todos sus valores de longitud de onda, las coordenadas z de entrada y de salida se obtienen directamente del array 14. Esto puede no ocurrir con el primero y el último de los sectores atravesados por el rayo (ver figura 7). Para calcular t_s en el primer sector se sustituye z_{e_i} por la coordenada z del punto de entrada del rayo obtenida de la textura 2D correspondiente a las caras delanteras. Para el último, se sustituye z_{s_i} por la coordenada z del punto de salida del rayo obtenida de la textura 2D correspondiente a las caras traseras.

En la figura 8 se muestra el resultado final de la utilización del algoritmo directo visualizando diez sectores de un cubo de datos.

5. Evaluación

Se ha realizado un estudio experimental del comportamiento de los algoritmos. Para ello se han comparado los dos métodos propuestos, entre sí y con el algoritmo de ray casting original. Las medidas del tiempo de dibujo se han tomado con cada uno de los métodos, de un mismo modelo, con la misma posición de la cámara y la misma función de transferencia. Para medir el tiempo se ha forzado a que el programa redibuje constantemente y se han contado los frames dibujados en un segundo.

Las pruebas se han llevado a cabo en un ordenador personal con un procesador Intel® Core™ i3-530 a 2.93 GHz, una tarjeta gráfica NVIDIA GeForce 9800 GT, 4 GB de memoria RAM y una distribución Ubuntu 11.10 del sistema operativo Linux, con kernel 3.0. La ventana de visualización utilizada tiene una resolución de 1342x490 píxeles.

En primer lugar se han comparado el algoritmo original de ray casting con el algoritmo secuencial y el directo, visualizando un cubo de datos completo en doce configuraciones diferentes. La cámara se ha situado en tres orientaciones distintas (frontal, lateral y oblicua), cada una de ellas a dos distancias del modelo (cercana y lejana). Se han usado dos funciones de transferencia, una asigna valores de opacidad altos y otra bajos.

La tabla 1 muestra la velocidad de dibujo (en frames por segundo) para los tres métodos en los doce casos. Puede observarse que el rendimiento disminuye con el área cubierta por el modelo en la imagen. Así, en los casos en los que el modelo se encuentra más cerca el rendimiento es menor. Esto se debe a que el shader de la tercera pasada tiene una mayor carga de trabajo en los píxeles en los que se proyecta el modelo. También se aprecia una disminución general de la velocidad en los casos con función de transferencia que asigna colores con transparencia alta, debido a que el rayo recorre más celdas antes de alcanzarse el valor de opacidad uno, que es una de las condiciones de parada del rayo. Aunque hay ligeras diferencias, no se aprecia una tendencia clara en la comparación entre los métodos.

A continuación se comparó el algoritmo secuencial y el directo para distinto número de sectores, usando las configuraciones descritas previamente. A modo de ejemplo, las tablas 2 y 3 muestran el comportamiento de los métodos secuencial y directo para distinto número de sectores en un modelo visualizado con la configuración Frente Cercano Transparente y Oblicuo Lejano Opaco, respectivamente. El tiempo de dibujo del algoritmo secuencial aumenta linealmente con el número de sectores, mientras que el del directo es casi independiente del número de sectores.

Las figuras 9 y 10 muestran el comportamiento de los dos algoritmos en dos de estas configuraciones. Las pruebas han puesto de manifiesto que:

- El comportamiento relativo entre el algoritmo directo y el secuencial depende del número de sectores. Para un sector funciona ligeramente mejor el método secuencial, pero con más sectores se comporta mejor el método directo. La diferencia entre ambos aumenta con el número de sectores; ésta es mayor cuando hay mayor solapamiento entre sectores en pantalla, debido a que en estos casos el método secuencial tiene que procesar cada uno de los píxeles varias veces. La diferencia también aumenta con la opacidad de la paleta. Cuando ésta es alta, hay mayor diferencia entre los algoritmos secuencial y directo.
- La diferencia en velocidad de refresco entre ambos puede llegar a ser de hasta cuatro veces, a favor del algoritmo directo para una visualización de hasta veinte sectores. La penalización del algoritmo directo con un único sector oscila entre el 8 % y el 15 % en las pruebas realizadas.

Caso	RC	Sec.	Dir.
Oblicuo Cercano Transparente	8	8	8
Oblicuo Cercano Opaco	8	8	8
Lateral Cercano Transparente	10	10	13
Lateral Cercano Opaco	12	12	14
Frontal Cercano Transparente	11	10	10
Frontal Cercano Opaco	19	19	16
Oblicuo Lejano Transparente	11	11	12
Oblicuo Lejano Opaco	15	14	17
Lateral Lejano Transpanrente	12	12	16
Lateral Lejano Opaco	14	14	19
Frontal Lejano Transparente	18	18	19
Frontal Lejano Opaco	23	23	24

Table 1: *Velocidad de dibujo en frames por segundo para los tres métodos. RC: Ray casting sin sectores; Sec: Algoritmo secuencial; Dir: Algoritmo directo.*

N. sectores	Secuencial	Directo
1	35	32
2	22	23
5	15	23
10	11	22
20	7	19

Table 2: *Velocidad de dibujo en frames por segundo en el caso Frente Cercano Transparente para distinto número de sectores.*

6. Conclusiones

En este trabajo se ha abordado la visualización y exploración de conjuntos de datos de espectroscopia de campo integral.

Siguiendo las últimas tendencias en visualización científica aplicadas a astrofísica, hemos utilizado visualización de volúmenes como base, procesando los cubos de datos obtenidos como un grid 3D que representa un volumen. Por este

N. sectores	Secuencial	Directo
1	41	38
2	27	31
5	16	25
10	10	20
20	5	10

Table 3: *Velocidad de dibujo en frames por segundo en el caso Oblicuo Cercano Opaco para distinto número de sectores.*

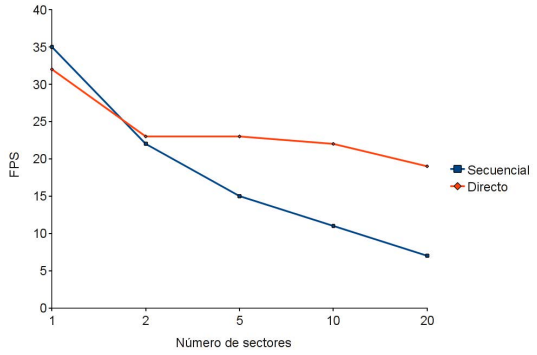

Figure 9: *Gráfico de la velocidad de dibujo en frames por segundo en el caso Frente Cercano Transparente.*

motivo se ha adaptado e implementado un algoritmo de ray casting de volúmenes en GPU con visualización de isosuperficies.

Se ha analizado el problema de la visualización interactiva de esta información y se ha propuesto un método de exploración basado en la visualización de sectores del modelo en intervalos de longitud de onda. Este método permite acotar la zona visualizada del cubo de datos a intervalos relevantes de longitud de onda, lo que permite focalizar la atención en zonas específicas y favorece la comparación del comportamiento en distintas zonas del espectro.

Como trabajo futuro pretendemos adaptar el método de visualización para utilizarlo en sistemas inmersivos.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Consejería de Innovación Ciencia y Empresa de la Junta de Andalucía a través del proyecto de excelencia PE09-TIC-5276.

References

[AS06] ALLINGTON-SMITH J.: Basic principles of integral field spectroscopy. *New Astronomy Reviews 50* (2006), 244–251. 1

[BGH*07] BORKIN M., GOODMAN A., HALLE M., ALAN D., J. K.: Application of Medical Imaging Software to the 3D Visualization of Astronomical Data. In *AAS/AAPT Joint Meeting, American Astronomical Society Meeting 209* (2007). URL: http://am.iic.harvard.edu/. 2

[HCWH08] HONGWEI L., CHI-WING F., HANSON A.: Visualizing multiwavelength astrophysical data. *Visualization and Computer Graphics, IEEE Transactions on 14*, 6 (nov.-dec. 2008), 1555 –1562. doi:10.1109/TVCG.2008.182. 2

[HF11] HASSAN A., FLUKE C. J.: Scientific visualization in astronomy: Towards the petascale astronomy era. *Publications of the Astronomical Society of Australia 28* (2011), 150–170. 2

[Joh08] JOHANNESSEN R. A.: Interactive Visual Analysis and Exploration of Astrophysical Data. In *Seminar in Visualization* (2008), Viola I., Hauser H., (Eds.), The Eurographics Association. URL: http://www.ii.uib.no/vis/teaching/vis-seminar/2008-fall/johannessen/johannessen_paper.pdf. 2

[Kau07] KAUFFMANN J.: The Need for 3D Visualization in Astronomy. ADASS Tutorial: 3D Visualization in Astronomy. In *Astronomical Data Analysis Software & Systems XVII* (2007). 2

[Lev90] LEVOY M.: Efficient ray tracing of volume data. *ACM Transactions on Graphics 9 (3)* (1990), 245–261. 2, 4

[Ngu07] NGUYEN H.: Volume rendering. In *GPU Gems 3* (2007), Nguyen H., (Ed.), Addison-Wesley, pp. 665–667. URL: http://developer.nvidia.com/node/187. 3

[PvdH01] POLDER G., VAN DER HEIJDEN G. W.: Visualization of Spectral Images. In *Proceedings of SPIE Vol. 4553* (2001), Yair Censor M. D., (Ed.), pp. 132–137. 2

[Tyl11] TYLER R.: The Importance of 3D Visualization in Astronomy, 2011. URL: people.cs.uct.ac.za/~scfinniss/project/files/robin_synth.pdf. 2

[Wet10] WETHEREL T.: The astronomy opportunity of the century. *ScienceWise Magazine 7 (4)* (2010). 1

[WGH81] WELLS D. C., GREISEN E. W., HARTEN R. H.: Fits: A flexible image transport system. *Astronomy & Astrophysics Supplement Series* (1981), 363–370. URL: http://heasarc.gsfc.nasa.gov/docs/heasarc/fits_overview.html. 2, 3

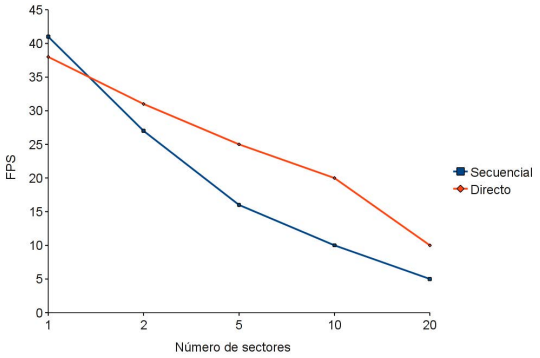

Figure 10: *Gráfico de la velocidad de dibujo en frames por segundo en el caso Oblicuo Cercano Opaco.*