# TRAZADO DE HACES DE RAYOS EN ESCENAS ESTRUCTURADAS ESPACIALMENTE MEDIANTE ÁRBOLES OCTALES*.

P. González López[1]
F. Gisbert Cantó[2]

[1] Dpto. Informática, E.U.P.A., Univ. Castilla-La Mancha, 02071 Albacete, España
[2] Dpto. LSI e IS, Fac. Informática , Univ. Politécnica Madrid, 28071 Madrid, España

**Sumario**

En este trabajo se presenta una nueva propuesta de optimización de la generación de imágenes de alta calidad (foto-realistas) basada en el trazado de rayos. En ella, se utilizan de modo conjunto las ideas aportadas por diferentes alternativas de aceleración. En primer lugar, se utilizan estructuras que permiten guiar a un rayo dentro de la escena (árboles octales), facilitando que cada rayo solo calcule su intersección con un número reducido de objetos (aquellos situados en los nodos que se encuentre en su camino). Por otra parte, en estas estructuras, se realiza el trazado de los rayos agrupados en haces. Se aporta , por tanto, una alternativa de aceleración que mejora los resultados obtenidos, respecto a otras propuestas previas, conforme se incrementan la resolución y la calidad de la imagen a generar.

## 0. Introducción

El algoritmo trazador de rayos (*Ray-Tracing)* se utiliza con frecuencia en la visualización de escenas tridimensionales, obteniendo imágenes de gran calidad (foto-realistas), aunque para ello precisen de un elevado tiempo de procesamiento. En la versión original de este algoritmo [Whi80] se propone analizar, para cada uno de los rayos, sus intersecciones con todos los objetos de la escena. Esta primera versión implica, por tanto, un elevado coste computacional y requiere nuevas propuestas que consigan su aceleración.

Arvo y Kirk [Arv89] aportan una completa descripción de diferentes alternativas de aceleración: (1) reducir el coste medio derivado del cálculo de la intersección del rayo con el entorno; (2) reducir el número total de rayos que intersectan con el entorno; y (3) sustituir los rayos individuales por entidades más generales y complejas. A estas tres alternativas se suelen unir otras que proponen reducir el tiempo de procesamiento a través del trazado simultáneo de diferentes rayos, mediante la ejecución paralela de este algoritmo.

Dichas alternativas basan su formulación en el estudio de diferentes modos de coherencia del trazado de rayos ([Oht87], [Gre89], etc.) que permiten la variación del algoritmo inicial

* Este trabajo está financiado por la CICYT, proyecto TIC97-0897-C04-02.

para conseguir una mayor eficiencia. Encontramos básicamente cuatro: coherencia de objetos, coherencia de rayos, coherencia de imagen y coherencia de fotograma.

De ellos, el más utilizado es la *coherencia de objetos* que parte de la idea de que posiciones próximas en el espacio tienden a estar ocupadas por un solo objeto de la escena o un conjunto reducido de objetos. Las técnicas que aprovechan esta característica, asocian a la escena una estructura que facilita la simulación de la trayectoria de un rayo dentro de ella. Existen dos grandes alternativas: subdivisión *guiada por el espacio* y subdivisión *guiada por los objetos*. La primera de ellas divide la escena mediante un proceso de descomposición espacial, bien sea mediante descomposición uniforme en celdas de igual tamaño ([Ama87], [Zal97], etc.) o por descomposición no uniforme en celdas de diferente tamaño. En este último caso se utilizan para ello tanto árboles binarios ([Kap85], [Jan86], etc.), como octales ([Gla84], [Sam89], [Mac90], [McN92], [End94], etc.). La segunda alternativa, asocia a la escena una jerarquía de volúmenes envolventes que contienen a los objetos ([Kay86], [Gol87], etc.).

En cualquier caso, en todas las soluciones anteriormente mencionadas, el tiempo de ejecución se ve altamente influenciado por el número de rayos a trazar. Dicho número no sólo depende de la resolución sino que se ve incrementado cuando se aportan soluciones que mitigan los problemas de *aliasing* que este algoritmo presenta, ya que la mayoría se apoyan en la idea de trazar más de un rayo por pixel. Por tanto, parece interesante aportar técnicas que no sólo aceleren el trazado de un rayo individual en la escena, sino que permitan que el tiempo necesario para trazar la escena sea lo más independiente posible del número de rayos.

Otra de las características de las escenas que tendremos en cuenta es la *coherencia de rayos,* que parte de la idea de que rayos muy próximos entre sí, es decir, aquellos cuyo origen y dirección son muy semejantes, probablemente intersectan con los mismos objetos dentro de la escena. Esta propiedad se ha utilizado para reducir el coste computacional al trazar en vez de rayos individuales, haces de rayos ([Hec84], [Spe86], etc.). De este modo, puede reducirse el problema anteriormente mencionado de la influencia del número de rayos a trazar en el tiempo total de ejecución.

Como vemos, los diferentes trabajos se apoyan en el aprovechamiento de un modo de coherencia, no siendo habitual encontrar estudios que utilicen de modo conjunto varios de ellos. En este artículo proponemos una nueva versión del algoritmo trazador de rayos que nos permite conseguir que el tiempo de ejecución se vea afectado mínimamente por la resolución de la escena, obteniendo, como veremos a continuación, una reducción significativa en el caso de escenas de alta calidad. Este se basa en la utilización, de modo conjunto, de dos ideas de aceleración: el uso de estructuras que nos permitan realizar un seguimiento del camino recorrido por un rayo en la escena (coherencia de objetos) y el trazado, en dichas escenas, de rayos agrupados en haces (coherencia de rayos).

En el siguiente apartado analizaremos en mayor detalle algunos trabajos precedentes sobre la aplicación aislada de la coherencia de rayos y de objetos. Tras esta revisión pasaremos a describir el nuevo algoritmo propuesto y analizaremos su comportamiento en varias escenas, comparando los resultados obtenidos con otros algoritmos. Finalmente recopilaremos las ideas más relevantes e indicaremos algunas de las líneas actuales de trabajo.

## 1. Utilización de la coherencia de rayos y de objetos

Nuestra alternativa de aceleración se basa por una parte en el trazado conjunto de una serie de rayos agrupados en haces y por otra en la utilización de árboles octales para dividir la escena. Como hemos visto ninguna de las ideas utilizadas son nuevas pudiendo encontrar numerosas referencias en ambos sentidos, aunque no existe ninguna de ellas que pretenda unir las ventajas de ambos métodos de aceleración.

De entre los trabajos que sustituyen el trazado de rayos por el trazado de haces podemos resaltar los de P. Heckbert y P. Hanrahan [Hec84] los cuales proponen el trazado de haces en escenas definidas a través de un conjunto de polígonos. Esta restricción, que impone que las escenas estén representadas mediante polígonos, les permite utilizar los haces para trazar tanto los rayos lanzados desde el observador como aquellos que se generan al chocar uno de ellos con un objeto de la escena, aunque en el caso de los rayos refractados los resultados obtenidos sean tan solo una aproximación del efecto óptico real.

Aunque no todas las soluciones que utilizan la coherencia de rayos son satisfactorias [Spe86], debido sobre todo a la dificultad de definir la forma del haz y determinar la pertenencia de un rayo a dicho haz, la idea general que subyace tras esta característica es muy interesante. Su relevancia es mayor a medida que aumenta el número de rayos a trazar y la proximidad entre dichos rayos. Esta situación se produce cuando deseamos obtener una imagen con una alta resolución o cuando introducimos dentro de nuestro algoritmo técnicas de sobremuestreo para resolver los problemas de "*aliasing*".

La utilización de árboles octales para acelerar el trazado de rayos fue inicialmente propuesta por A. Glassner [Gla84]. En ella se realizaba una descomposición de la escena en cubos o *voxel* de diferente tamaño apoyándose en una estructura de árbol octal (*octree*). De este modo, cada cubo tiene asociada una lista de objetos incluidos total o parcialmente en él. Cuando un rayo llega a un nodo tan solo se debe calcular la intersección con los objetos contenidos en él. Por tanto, en este tipo de técnicas debemos realizar un seguimiento del rayo de tal modo que podamos identificar los nodos que éste se encontrará en su camino.

Para realizar el seguimiento de un rayo a través de la escena tenemos que identificar dado un nodo, el siguiente en la dirección de propagación del rayo. Esta tarea se puede descomponer en dos subtareas: identificar la dirección de salida y determinar el nodo o nodos vecinos en dicha dirección. De ellas la que más tiempo consume es la obtención de

los vecinos en una determinada dirección, llegando a representar (como indican McNeill et al. [McN92]) hasta el 40% del tiempo total de procesado en el trazado de algunas escenas. Se han propuesto diferentes métodos para la búsqueda de vecinos en una dirección dada, pudiendo clasificarlos en tres grandes grupos: métodos-raíz (parten siempre desde la raíz del árbol) [Gla84]; métodos-padre (buscan el primer antecesor común dentro de la jerarquía) [Sam89]; métodos-enlaces (utilizan unos enlaces a los vecinos, previamente calculados, en las direcciones principales) [End94]. De todos ellos los más eficientes son aquellos que utilizan enlaces a los vecinos en las direcciones principales, pues el tiempo de preprocesado necesario para asociar a cada nodo terminal dichos enlaces es bajo y se ve claramente compensado cuando el número de rayos a trazar aumenta. Sin embargo, hemos de tener en cuenta que, en este caso, las necesidades de memoria aumentan al requerir almacenar por cada nodo terminal enlaces a los vecinos en las direcciones principales.

## 2. Trazado de haces en escenas estructuradas mediante árboles octales

Esta propuesta de aceleración pretende reducir la influencia que tiene el número de rayos a trazar dentro del tiempo total de generación de una escena. Para ello se apoya en la idea aportada por P. Heckbert y P. Hanrahan [Hec84] sobre el trazado de rayos agrupados en haces. Aunque restringimos la utilización de haces tan solo a los rayos primarios, en nuestro caso las escenas pueden contener todo tipo de objetos y no solamente aquellos definidos mediante polígonos. Del mismo modo, el algoritmo aprovecha las estructuras de árboles octales para guiar el trazado de los haces dentro de la escena, apoyándose para ello, básicamente, en los trabajos iniciales de H. Samet [Sam89] y en las mejoras propuestas por R. Endl y M. Sommer [End94]. Los algoritmos propuestos en estos trabajos también han sido implementados y servirán para comparar la eficiencia del nuestro.

Las estructuras básicas que utilizamos son los haces y los árboles octales. Un haz consta de una colección de rayos que tienen un origen común y pasan a través de un polígono plano que representa una sección del tronco de pirámide asociado al haz. Por tanto en este caso para definir un haz es suficiente con indicar su origen y una lista de vectores normalizados que determinan su dirección y están asociados a los vértices que lo delimitan.

Un árbol octal es una representación jerárquica a través de la cual el espacio es subdividido en cada paso en ocho voxels o nodos de igual tamaño, llamados octantes. Los nodos del árbol podrán ser de tres tipos: (1) nodos intermedios, es decir, aquellos que están divididos en otros de nivel inferior; (2) nodos terminales vacíos, es decir, aquellos que no contiene en su interior ningún objeto; (3) nodos terminales ocupados, es decir, aquellos que contienen un número, generalmente reducido, de objetos de la escena.

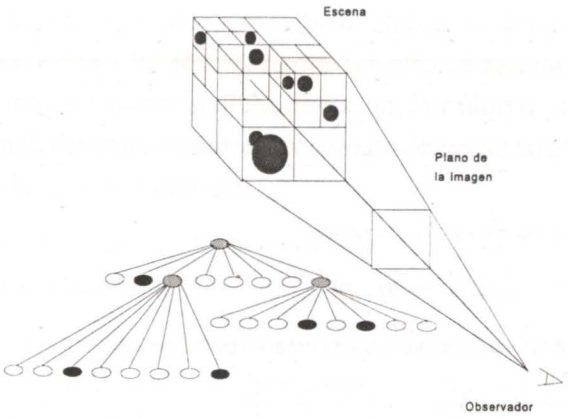


**Fig. 1**- Descomposición de una escena mediente árboles octales.

El proceso de descomposición de la escena pretende dividirla y estructurarla asociando los diferentes objetos a los voxel que componen los nodos terminales del árbol octal (fig. 1). Dicho proceso tiene en cuenta dos criterios básicos: reducir el número de objetos contenidos en un nodo y reducir el nivel de subdivisión. El primero de ellos pretende minimizar el número de intersecciones a calcular cada vez que un rayo llega a un nodo. El segundo disminuye la memoria necesaria para almacenar el árbol, a la vez que optimiza el recorrido de dicha estructura y aumenta la potencia de nuestro algoritmo al permitir que los haces contengan un mayor número de rayos.

Una vez estructurada la escena puede comenzar el proceso de trazado para lo cual proponemos el algoritmo descrito en el cuadro 1. Como se puede apreciar el tratamiento propuesto depende del tipo de nodo al que visita nuestro haz. Dentro de las tareas a realizar existen algunas comunes, aunque con algunos matices, al trazado de rayos individuales y otras propias del trazado de haces.

Como tareas comunes con el trazado de rayos individuales (marcadas en cursiva en el cuadro 1) tenemos la determinación de la dirección de salida y la búsqueda de los vecinos en una determinada dirección. En cualquier caso se debe tener en cuenta que un haz puede salir por varias direcciones a la vez y un rayo solo por una.

La determinación de las direcciones de salida se basa en el cálculo de la intersección del haz con el plano asociado a la dirección de propagación dominante. En nuestro caso, al trazar haces asociados a rayos primarios la dirección dominante de propagación es Z (asociada a la profundidad de la escena), por lo que se realiza la intersección del haz con la cara posterior del nodo en el que se encuentra. La situación de dichos puntos de intersección con respecto a la cara posterior del nodo determinará finalmente las direcciones de salida del haz original.

**Algoritmo:**

**nodo vacío**

*Determinar las direcciones de salida dado un haz y un nodo.*

Para cada dirección

Determinar el haz de salida estándar en función de la cara del nodo por la que sale.

*Buscar el nodo o nodos terminales vecinos en dicha dirección.*

Si en dicha dirección tiene vecinos.

Si tiene un solo vecino cuyo tamaño es igual o mayor que el del nodo origen.

No modificar el haz estándar.

En otro caso, (tiene varios vecinos de menor tamaño)

Dividir el haz de salida estándar en función del tamaño de sus vecinos.

Procesar el o los nuevos haces de modo recursivo.

En otro caso, (no existen vecinos en dicha dirección) el haz abandona la escena.

Finalizar su trazado.

**nodo ocupado**

Buscar los rayos contenidos en el haz.

Para cada uno de ellos calcular su intersección con los objetos contenidos en el nodo.

Si el rayo choca con uno de los objetos asociados al nodo y el punto de intersección se encuentra dentro del nodo.

Generar los diferentes rayos secundarios o de sombra y trazarlos de modo individual a través del árbol octal.

En otro caso,

Si el número de rayos que no chocan con los objetos de la escena es menor que un cierto umbral.

Trazar cada rayo primario de modo individual.

En otro caso,

Seguir trazando el haz pero asociándole sólo aquellos rayos que no han chocado previamente.

**Cuadro 1-** Descripción general del algoritmo de trazado de haces.

Tras saber por donde sale el haz debemos buscar el vecino o vecinos en cada dirección, para lo que utilizamos el método propuesto por Samet [Sam89]. Este realiza un ascenso en el árbol hasta encontrar el primer antecesor común al nodo actual y al vecino que estamos

buscando. Tras determinar dicho antecesor común realiza un descenso en el árbol buscando por las direcciones opuestas a las seguidas en el proceso de ascenso, llegando de este modo al vecino de igual o mayor tamaño en dicha dirección. Por último, si el nodo al que llegamos es un nodo no terminal, debemos seguir el proceso de descenso en el árbol hasta alcanzar los nodos terminales en la dirección deseada.

Dentro de las tareas propias del trazado de haces podemos resaltar básicamente dos: determinar los rayos asociados a un haz dado y definir el tamaño y la forma de los haces.

La determinación de los rayos contenidos en un haz se apoya en las ideas de rellenado de polígonos. Para ello, proyectamos el haz sobre el plano de la pantalla, con lo cual obtenemos un polígono que determina su silueta. Tras ello, realizamos la búsqueda de los rayos incluidos en el haz a través de una modificación del algoritmo clásico de rellenado de polígonos (algoritmo de líneas de rastreo), que nos permite determinar los puntos (rayos en nuestro caso) contenidos en el polígono que define los límites del haz.

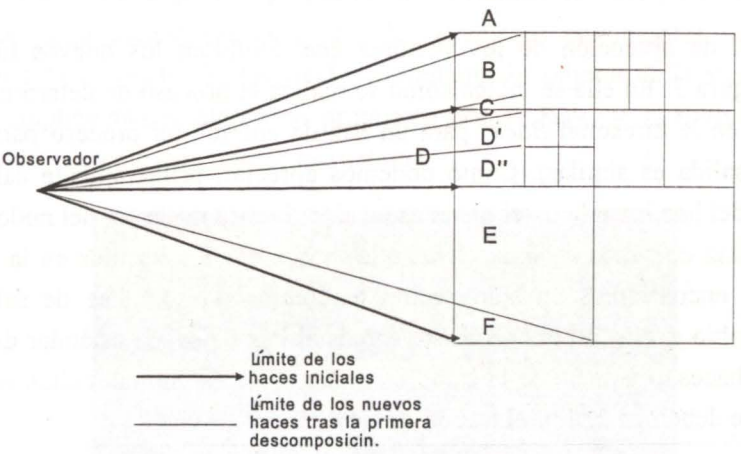


**Fig. 2**- Definición de haces en escenas 2D.

Tal vez la tarea más compleja de todas las propuestas en nuestro algoritmo, es la determinación de la forma y el tamaño de los haces de salida. Esta no solo depende del nodo que abandona el haz sino también de los nodos a los que llega (ver figura 2). Como podemos apreciar en la figura 2, si en la dirección de propagación el haz encuentra un nodo vecino de igual o mayor tamaño al actual (como sucede con los haces B, C o E) el haz de salida tan solo tiene en cuenta el haz inicial y los límites de salida del nodo en dicha dirección (a este haz lo hemos denominado haz de salida estándar). Por otra parte, si en la dirección dada encontramos varios vecinos el haz de salida estándar debe dividirse tantas veces como vecinos hayamos encontrado. Como podemos apreciar en la figura el haz de salida estándar D se divide en dos nuevos haces D' y D''.

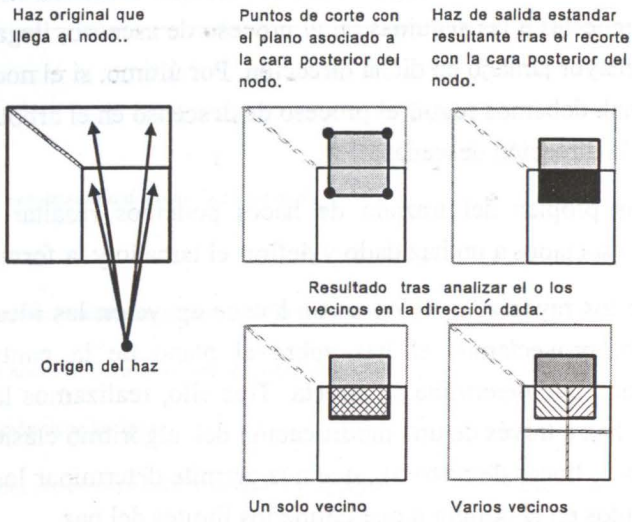


**Fig. 3-** Proceso de obtención de los vectores que delimitan los nuevos haces.

El proceso final de obtención de los vectores que delimitan los nuevos haces aparece descrito en la figura 3. En ella se indica como se realiza el proceso de determinación de los haces de salida en la dirección *Back* para un haz de entrada (el proceso para el resto de direcciones de salida es similar). Como podemos apreciar, primeramente calculamos los puntos de corte del haz inicial con el plano asociado a la cara posterior del nodo, obteniendo el denominado haz de salida estándar. Tras ello, buscamos los vecinos en la dirección de propagación. Si encontramos un solo vecino tendremos un solo haz de salida igual al estándar. En cambio si encontramos varios vecinos el haz de salida estándar debe dividirse para ajustar los haces al tamaño de la cara de entrada de cada uno de dichos vecinos (en la figura vemos que debemos dividir el haz de salida estándar en dos).

Tras calcular los puntos de corte con el plano dado, el problema de determinación de los nuevos límites del haz se reduce a realizar el recorte de un polígono respecto a una ventana rectangular (cara del voxel). Para ello utilizamos el algoritmo de recorte clásico propuesto por Sutherland y Hodgman [Sut74].

## 3. Análisis de resultados

Para demostrar la potencia del algoritmo que proponemos vamos a procesar varias escenas y a comparar los resultados, con los obtenidos por otras alternativas de aceleración. En primer lugar lo compararemos con otros algoritmos que realizan un trazado de rayos individuales en escenas estructuradas mediante árboles octales. Para ello hemos seleccionado el algoritmo inicial propuesto por H. Samet [Sam89] (denominado Samet-Estándar) y una optimización a dicho algoritmo propuesta por R. Endl y M. Sommer [End94] (denominado Samet-Net). La diferencia básica entre ambas propuestas se encuentra en el modo de buscar un vecino en una dirección dada. El algoritmo *Samet-Estándar* propone recorrer el árbol

ascendiendo primero hasta un nodo común para después descender hasta alcanzar el vecino de igual o mayor tamaño. Sin embargo, el algoritmo *Samet-Net* utiliza unos enlaces, previamente calculados, que apuntan a cada vecino en las direcciones principales. Como veremos en los resultados obtenidos por ambos algoritmos las ventajas del segundo aumentan al hacerlo el número de rayos a trazar en la escena y el nivel de descomposición del árbol.

Por otra parte, hemos seleccionado un algoritmo que propone otra estrategia de descomposición de la escena, en este caso basada en la obtención de una jerarquía de volúmenes envolventes. Su utilización nos permitirá ver como nuestro algoritmo se comporta bastante bien aun en aquellas escenas en las que los algoritmos basados en la descomposición de la escena mediante árboles octales obtienen peores resultados que aquellos que utilizan jerarquías de volúmenes. Dentro de las diferentes alternativas existentes basadas en jerarquías de volúmenes envolventes nosotros vamos a utilizar las implementadas dentro las librerías desarrolladas por N. Wilt [Wil94], denominadas OORT. En ellas se implementa el algoritmo propuesto por J. Goldsmith y J. Salmon [Gol87] que nos permite asociar a la escena una jerarquía de volúmenes envolventes (en este caso el volumen es un cubo alineado con los ejes) que contiene los objetos de la misma. Del mismo modo OORT implementa el algoritmo propuesto por T. Kay y J. Kajiya [Kay86] para recorrer de modo eficiente este tipo de estructuras.

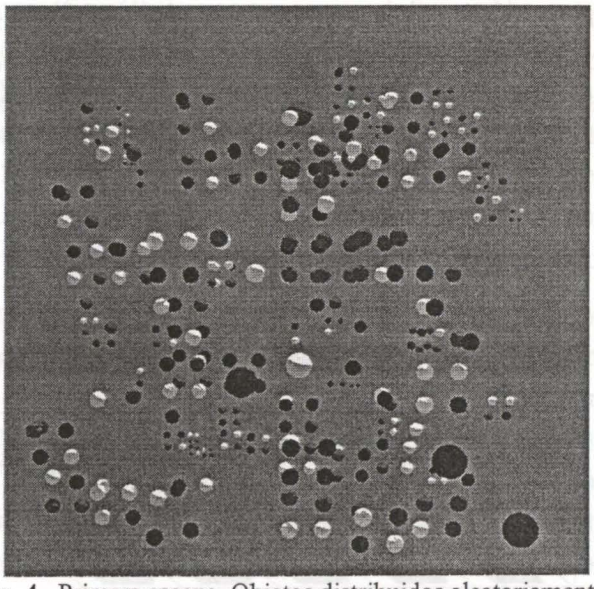

**Fig. 4**– Primera escena. Objetos distribuidos aleatoriamente.

De las diferentes escenas utilizadas para probar nuestro algoritmo, hemos seleccionado dos. La primera de ellas (fig.4) es una escena que contiene, distribuidos aleatoriamente, unos 400 objetos y cuyo árbol binario contiene a su vez 1.250 nodos. En el segundo ejemplo (fig. 5) utilizaremos un menor número de objetos fuertemente concentrados, especialmente en el centro de la escena. El pequeño tamaño de los objetos y el criterio de descomposición (un

nodo no puede contener más de dos objetos) ha implicado que el árbol tenga un elevado nivel de descomposición, y por tanto un gran número de nodos (450), en proporción con el número de objetos de la escena (60).

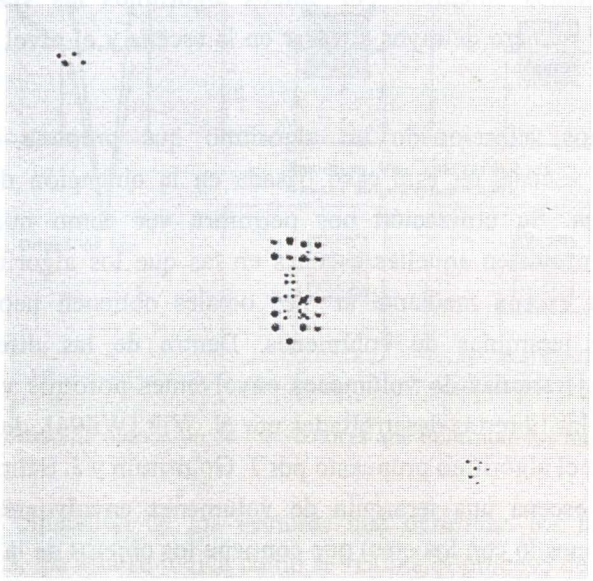

**Fig. 5**– Segunda escena. Objetos fuertemente concentrados en el centro de la escena.

Las pruebas se han hecho aumentando progresivamente la resolución de la imagen (pasando de 320x200 a 1280x1024) y por tanto los rayos a trazar. De este modo podemos analizar el efecto que el aumento del número de rayos tiene en el tiempo de procesamiento. En las figuras 6 y 7 podemos observar los resultados obtenidos para la primera y segunda escena respectivamente.

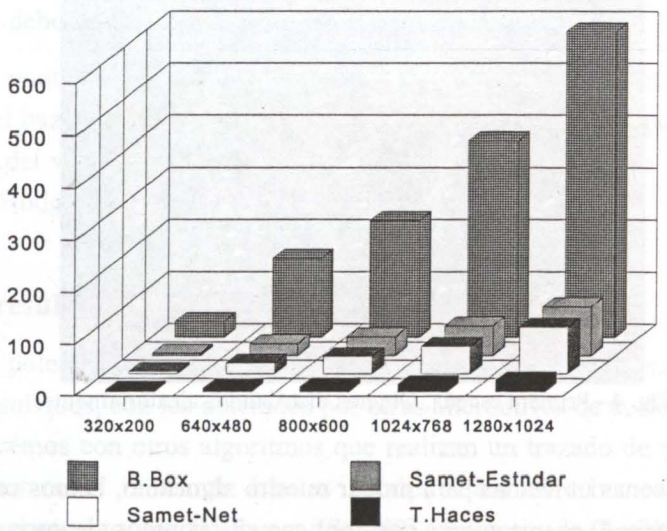


**Fig. 6 [a]**- Resultados completos de la primera escena.

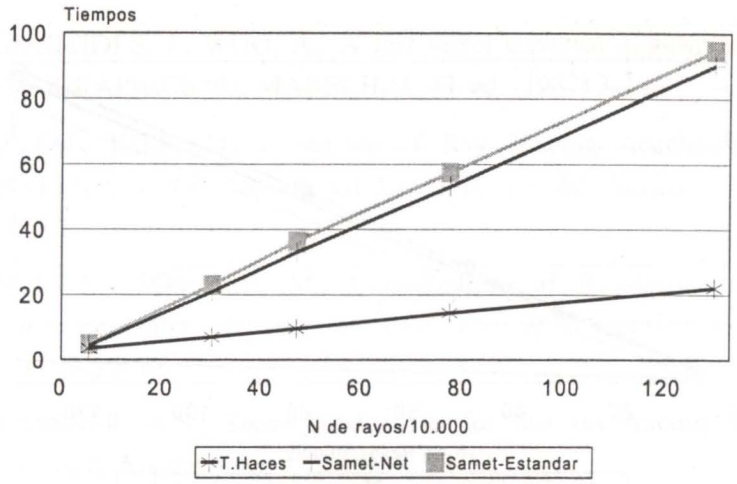


**Fig. 6 [b]-** Resultados de los métodos basados en árboles octales para la primera escena.

Si analizamos los resultados obtenidos por los distintos algoritmos vemos como en el caso de la primera escena (ver fig. 6 [a]) cualquiera de los algoritmos que utilizan árboles octales se comporta mejor que aquellos que utilizan jerarquías de volúmenes (B.Box). Si eliminados los resultados obtenidos por el B.Box y analizamos en detalle el comportamiento del resto de algoritmos (fig. 6[b]), vemos como el Samet-Net mejorar sus resultados respecto al Samet-Estándar conforme aumenta la resolución. Lo cual es debido a que al aumentar el número de rayos que deben recorrer el árbol se rentabiliza el tiempo utilizado para calcular los enlaces que apuntan a los vecinos de un nodo. En cualquier caso si algo llama la atención es el comportamiento del algoritmo que nosotros proponemos (T. Haces), en el que la pendiente de la recta de tiempos es mucho menor, lo que demuestra que la influencia del número de rayos a trazar sobre el tiempo de procesamiento es mucho menor.

El comportamiento de la segunda de las escenas a analizar (fig. 7) es similar si comparamos los diferentes algoritmos que utilizan los árboles octales, pero, sin embargo, vemos que el que implementa las jerarquías de volúmenes (B.Box) se comporta mejor que el Samet-Estándar o el Samet-Net. Este resultado se ha producido por la excesiva subdivisión del árbol octal asociado a dicha escena, efecto semejante al que se produciría si tuviéramos un pequeño objeto dentro de una gran escena (p.e. si tenemos una tetera dentro de un campo de futbol). En este caso, al tener un reducido número de objetos fuertemente concentrados en el centro de la escena los volúmenes envolventes producen estructuras de descomposición aparentemente más eficientes. Aún en este tipo de escenas, en las que los árboles octales no parecen la mejor alternativa de descomposición, hay que resaltar el excelente comportamiento de nuestro algoritmo. Esto se debe a que aunque el nivel de subdivisión es elevado, existen grandes zonas de la escena que no contienen objetos y, por tanto, en ellas los rayos viajan siempre incluidos dentro de los haces, no siendo necesario determinar realmente los rayos contenidos hasta que el haz finaliza su recorrido y abandona la escena.

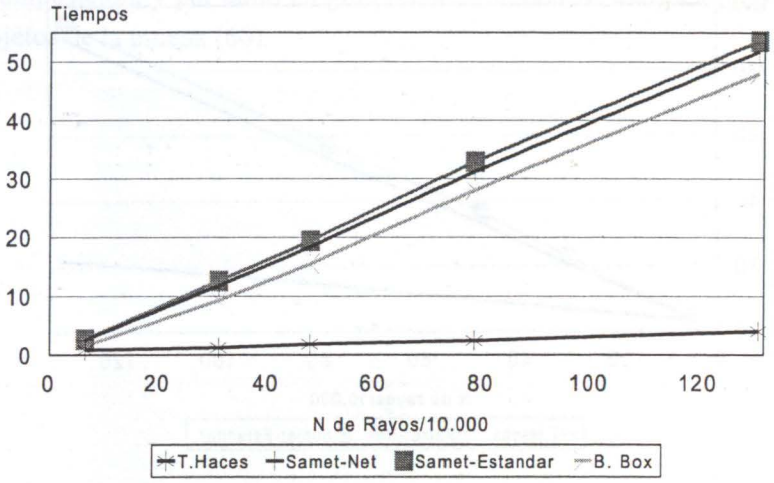


**Fig. 7**- Resultados obtenidos por los distintos algoritmos para la segunda escena.

Por tanto, tras analizar los resultados obtenidos en las diferentes pruebas realizadas, podemos concluir que nuestro algoritmo tiene un comportamiento mejor que otras alternativas de trazado de rayos individuales e incluso en aquellas escenas en las que los árboles octales tal vez no se ajustan muy bien y producen estructuras muy pesadas de recorrer, nuestro algoritmo obtiene resultados excelentes.

## 4. Conclusiones

En este trabajo hemos descrito un nuevo algoritmo de trazado de rayos uniendo las ideas aportadas por diferentes alternativas de aceleración: utilizar estructuras que permitan dirigir el recorrido de un rayo (árboles octales) y trazar, en dichas estructuras, un conjunto de rayos agrupados en un haz. Con esta propuesta se ofrece una alternativa de aceleración que mejora los resultados respecto al resto, al aumentar la resolución y la calidad de la imagen a obtener.

A su vez, el modo en que se han definido y trazado los haces permite que estos sean independientes y por tanto puedan trazarse de modo paralelo, abriendo de este modo nuevas expectativas de aceleración al algoritmo propuesto.

Este mismo proceso, además, puede extenderse y aplicarse a los rayos de sombra, al tener estos también un origen común (en este caso la fuente de luz) que nos permite a partir de la estructura del árbol, definir haces de sombra. Esta idea está siendo actualmente implementada y comparada con otras alternativas de aceleración como la que hemos utilizado hasta ahora en este algoritmo (*shadow caching* [Wil94]), que consiste en asociar a cada luz un enlace al último objeto opaco que encontró el anterior rayo de sombra que se dirigió a dicha fuente de luz.

## Referencias


[Ama87] AMANTIDES, J.; WOO, A.; A fast voxel traversal algorithm for ray tracing. EUROGRAPHICS '87, MARECHAL, G. ed. (1987) 3-10.

[Arv89] ARVO,J.; KIRK, D.; A survey of Ray Tracing Acceleration Techniques. Introduction to Ray Tracing, GLASSNER A.S. ed. Academic Press, 1989, pp. 201-262.

[End94] ENDL, R.; SOMMER, M.; Classification of Ray-Generators in Uniform Subdivisions and Octrees for Ray Tracing. Computer Graphics Forum, 13,1(1994) 3-19.

[Gla84] GLASSNER, A.S.; Space subdivision for fast ray tracing. IEEE Computer Graphics & Applications, 4,10 (1984) 15-22.

[Gla89] GLASSNER A.S.; Introduction to Ray Tracing. Academic Press, 1989.

[Gol87] GOLDSMITH, J.; SALMON, J.; Automatic creation of object hierarchies for ray tracing. IEEE Computer Graphics & Applications,7, 5, (1987) 14-20.

[Gre89] GREEN, S.A.; PADDON, D.J.; Exploiting Coherence for Multiprocessor Ray Tracing. IEEE Computer Graphics & Applications, 4,10, (1989) 12-26.

[Hec84] HECKBERT, P.; HANRAHAN, P.; Beam Tracing Polygonal Objects. Computer Graphics, 18, 3, (1984) 119-127.

[Jan86] JANSEN, F.W.; Data structures for ray tracing. Data structures for Raster Graphics, KESSENER, L. et al. (ed.),Springer Verlag, (1986) 57-73.

[Kap85] KAPLAN, M.R.; Space tracing a constant time ray tracer. State of the Art in Image Synthesis. SIGGRAPH '85, 11, (1985).

[Kay86] KAY, T.; KAJIYA, J.; Raytracing complex scenes. Computer Graphics, 20, 4, (1986) 269-278.

[Mac90] MACDONALD, J.D.; BOOTH, K.S.; Heuristics for ray tracing using space subdivision. The Visual Computer, 6, 3, (1990) 153-166.

[McN92] McNEILL,M.D. et al.; Performance of Space Subdivision Techniques in Ray Tracing. Computer Graphics Forum, 11, 4, (1992) 213-220.

[Oht87] OHTA, M.; MAEKAWA, M.; Ray Coherence Theorem and Constant Time Ray Tracing Algoritm. Computer Graphics, (1887) 303-314.

[Sam89] SAMET, H.; Implementing Ray Tracing with Octrees and Neighbor Finding. Computer & Graphics, 13, 4, (1989) 445-460.

[Sut74] SUTHERLAND, I.E.; HODGMAN, G.W.; Reentrant Polygon Clipping. Comm of the ACM, 17, 1, (1974) 32-42.

[Spe86] SPEER, L.R. et al.; A Theoretical and Empirical Analysis of Coherent Ray-Tracing. Computer-Generated Images (Proc. Of Graphics Interface '85), (1986) 11-25.

[Whi80] WHITTED, T.; An Improved Illumination Model for Shaded Display. Comm of the ACM, 23, 6, (1980) 343-349.

[Wil94] WILT, N.; Object-Oriented Ray Tracing in C++. John Wiley & Sons, Inc. 1994.

[Zal97] ZALIK,B.; CLAPWORTHY, G.; OBLONSEK, C.; An Efficient Code-Based Voxel-Traversing Algorithm. Computer Graphics Forum, 16, 2, (1997) 119-128.