Técnicas para Calcular la Entropía Dependiente de la Vista de una Escena 3D

Pascual Castelló
Departamento de Lenguajes y sistemas Informáticos
Universitat Jaime I
Campus Riu Sec
12080 Castellón de la Plana
castellp@uji.es

Miguel Chover
Departamento de Lenguajes y sistemas Informáticos
Universitat Jaime I
Campus Riu Sec
12080 Castellón de la Plana
chover@uji.es

Mateu Sbert
Institut d'Informàtica i Aplicacions
Universitat de Girona
Campus Montilivi
17071 Girona
mateu@ima.udg.es

Miquel Feixas
Institut d'Informàtica i Aplicacions
Universitat de Girona
Campus Montilivi
17071 Girona
feixas@ima.udg.es

Resumen

La entropía dependiente de la vista es una métrica que permite evaluar la bondad de la visibilidad de una escena desde una posición de cámara. En este trabajo se realiza un estudio de las diferentes técnicas que permiten calcular esta entropía. El objetivo de este estudio es identificar la técnica que permite un cálculo en tiempo real para escenas 3D de gran complejidad.

1. Introducción

Recientemente se han desarrollado varios métodos, que tienen como denominador común el uso del concepto de complejidad dependiente del punto de vista [2, 1, 7, 3, 5, 6]. La noción de complejidad dependiente del punto de vista se usa en varias áreas de informática gráfica tales como *scene understanding* y exploración de mundos virtuales, radiosidad e iluminación global, modelado y *rendering* basado en imágenes, etc.

Entre las medidas que se han introducido para el cálculo de la complejidad una de las más fructíferas hasta el momento ha sido la entropía [3, 4]. Sin embargo, el cálculo de la función de entropía puede ser costoso, especialmente cuando se tiene una escena muy compleja y se deben evaluar múltiples puntos de vista. En este artículo evaluaremos distintas alternativas para su cálculo utilizando distintos modelos con creciente complejidad.

En la siguiente sección daremos una definición de la entropía dependiente de la vista. En la sección 3 presentaremos las distintas técnicas para su cálculo y en la sección 4 se evaluarán esas técnicas con distintos modelos. En la sección 5 presentamos nuestras conclusiones y trabajo futuro.

2. Entropía Dependiente de la Vista

La entropía de un punto de vista, basada en la definición de la entropía de Shannon, fue introducida por Vázquez et al. [3, 5] y es una medida de la información proporcionada por un punto de vista.

La entropía de Shannon de una variable aleatoria discreta X, que toma valores en el conjunto $\{a_1, a_2, ..., a_n\}$, se define como

$$H(X) = -\sum_{i=1}^{n} p_i \log p_i,$$

donde $p_i=Pr[X=a_i]$, los logaritmos están expresados en base 2 y 0 log 0 = 0 por razones de continuidad. Ya que -log p_i representa la información asociada con el resultado a_i, la entropía nos da la información media (o incertidumbre) proporcionada por una variable aleatoria.

Para definir la entropía de un punto de vista, utilizamos la distribución construida a partir del área relativa de las caras (polígonos) proyectada sobre la esfera de direcciones centrada en el punto de vista. Así, dada una escena S y un punto de vista p, la entropía de p es definida como

$$I(S, p) = -\sum_{i=0}^{N_f} \frac{A_i}{A_t} \log \frac{A_i}{A_t},$$

donde N_f es el número de polígonos de la escena, A_i es el área proyectada por el polígono i sobre la esfera, A_0 representa el área proyectada por el *background* en escenas abiertas, y A_t es el área total de la esfera. En una escena cerrada, la esfera está completamente cubierta por los polígonos proyectados y consecuentemente $A_0=0$.

La máxima entropía se conseguirá cuando un cierto punto de vista vea todos los polígonos con

la misma área relativa. Así, en una escena abierta, la máxima entropía viene dada por log (N_f + 1) y, en una escena cerrada, por log N_f. Entre todos los posibles puntos de vista, se considera que el mejor es el que tiene máxima entropía. Eso es, el que proporciona máxima información.

3. Técnicas para Calcular la Entropía

Para poder calcular la entropía es necesario como paso previo obtener el área en píxeles de cada triángulo[1] visible desde una posición de cámara determinada. En los apartados siguientes se analiza las diferentes técnicas que permiten calcular estas áreas.

3.1. Histograma de OpenGL

El histograma de OpenGL fue utilizado por primera vez para el cálculo de la entropía en [8]. Básicamente, cuenta las apariciones de un valor de color de una determinada componente. Sin embargo, podemos utilizarlo también para calcular el área de los triángulos que son visibles desde un punto de vista, sin necesidad de leer del buffer. El histograma de OpenGL está acelerado por hardware, aunque hay pocas tarjetas que realmente lo implementen (3DLabs WildCat). Normalmente suele ser responsabilidad del controlador del fabricante, como ocurre generalmente en las extensiones OpenGL y generalmente se implementa por software.

Para obtener el área de cada triángulo visible necesitamos asignar a cada triángulo un color diferente. Una limitación importante es que los histogramas tienen un tamaño fijo, normalmente de 256 valores diferentes. Éste suele ser el valor más frecuente en muchas tarjetas gráficas. El comando *glGetHistogram* devuelve una tabla que cuenta cada valor de color separado por canales. Si utilizamos los cuatro canales de color RGBA, nos devolverá una tabla de 256 elementos de 4 valores enteros. Siendo cada entero el numero de píxeles que tienen esa componente de valor. Por lo tanto, si queremos detectar un triángulo éste debe de estar codificado utilizando un solo canal. Esto nos da un total de 1020 valores diferentes. Es decir: para el canal R (1,0,0,0) hasta (255,0,0,0), para el canal G (0,1,0,0) hasta (0,255,0,0), para el canal B (0,0,1,0) hasta (0,0,255,0) y para el canal A (0,0,0,1) hasta (0,0,0,255). El valor (0,0,0,0) se reserva para el *background*.
Evidentemente el principal inconveniente de esta técnica, es que para objetos de más de 1020 triángulos es necesario realizar varias pasadas de *rendering*. En cada pasada obtendríamos el área de 1020 triángulos diferentes del objeto.
Utilizando histogramas de un número mayor de elementos, y realizando un *rendering* fuera de pantalla, nos permitirían lógicamente aumentar el número de colores, y por lo tanto realizar menos pasadas de *rendering*. Sin embargo, esta posibilidad esta fuera de la especificación de OpenGL y es dependiente del hardware. En varias tarjetas analizadas no ha sido posible emplear un histograma de mayor tamaño.

3.2. Histograma por Software y Hardware

Actualmente, con la aparición sobre todo de nuevos buses simétricos como el PCI Express, la operación de lectura del buffer no es ya tan costosa como antiguamente. Así pues, es posible obtener un histograma evitando realizar varias pasadas de *rendering*. La forma de obtenerlo es muy sencilla. Simplemente se asigna un color diferente a cada triángulo, se renderiza el objeto completo, posteriormente se realiza una lectura del buffer y se analiza este buffer píxel a píxel obteniendo datos sobre su color. Utilizando una codificación de color de RGBA con un valor para cada canal de un byte se puede realizar con una única pasada de *rendering* hasta 256*256*256*256 triángulos.

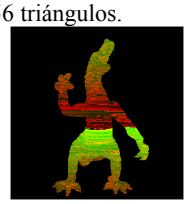

Figura 1. Entropía (I = 3,140178) para Dragon utilizando el histograma por SW+HW.

3.3. Occlusion Query

Esta extensión de OpenGL se utiliza normalmente para conocer qué objetos de la escena están ocultados por otros, y que por lo tanto no deberían

[1] Sin pérdida de generalidad, utilizamos de aquí en adelante triángulos.

enviarse a renderizar. Sin embargo, también puede emplearse para calcular el área de los triángulos que son visibles desde una posición de cámara determinada.

La extensión de OpenGL ARB_occlusion_query devuelve el número de píxeles que son visibles. Para poder calcular el área de cada triángulo visible de un objeto utilizando esta técnica se procedería de la siguiente manera: primero se renderiza el objeto completamente, con esto se inicializa el buffer de profundidad, posteriormente se renderiza cada triángulo independientemente. Mediante este procedimiento es necesario realizar n + 1 pasadas de *rendering* siendo n el número de triángulos del objeto. Cabe mencionar que solamente en la primera pasada se renderiza completamente la geometría, en las sucesivas se renderiza únicamente un triángulo individualmente.

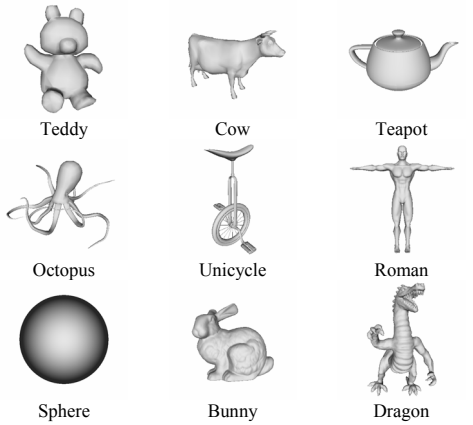

Figura 2. Modelos utilizados en los experimentos.

4. Resultados

Hemos calculado la entropía desde 6 posiciones de cámara distribuidas regularmente sobre una esfera que envuelve al objeto utilizando las distintas técnicas anteriormente descritas. Los valores de entropía son los mismos independientemente de la técnica empleada. Para comparar las distintas técnicas, hemos medido el tiempo necesario para calcular la entropía de las diferentes cámaras. Como casos de prueba hemos utilizado los objetos de diferente complejidad que aparecen en la Figura 2. Estos modelos han sido visualizados a una resolución de 256x256 píxeles utilizando vertex arrays de OpenGL.

Los resultados han sido obtenidos con un PC Xeon 2,4Ghz 1GB RAM y con dos diferentes GPUs PCI Express NVIDIA Quadro NVS 280 64 MB y una ATI X800 XT de 256 MB. Cabe destacar que de las dos GPUs analizadas, sólo el modelo de NVIDIA soporta el histograma de OpenGL.

Modelo	Vértices	Triángulos	Pasadas de Render	Histograma OpenGL (ms)
Teddy	1.598	3.192	4	2.317,252
Cow	2.904	5.804	6	3.497,404
Teapot	3.644	6.320	7	4.082,005
Octopus	4.242	8.468	9	5.261,136
Unicycle	6.973	13.810	14	8.243,972
Roman	10.473	20.904	21	12.496,437
Sphere	15.314	30.624	31	18.771,404
Bunny	34.834	69.451	69	44.372,069
Dragon	54.296	108.588	107	72.488,374

Tabla 1. Tiempos de cálculo de la entropía utilizando el histograma de OpenGL.

Modelo	Vértices	Triángulos	Pasadas de Render	Histograma SW+HW (ms)
Teddy	1.598	3.192	1	17,893
Cow	2.904	5.804	1	19,107
Teapot	3.644	6.320	1	19,374
Octopus	4.242	8.468	1	20,696
Unicycle	6.973	13.810	1	23,241
Roman	10.473	20.904	1	29,965
Sphere	15.314	30.624	1	36,752
Bunny	34.834	69.451	1	74,936
Dragon	54.296	108.588	1	84,438

Tabla 2. Tiempos de cálculo de la entropía utilizando el histograma por SW+HW.

Modelo	Vértices	Triángulos	Pasadas de Render	Occlusion Query (ms)
Teddy	1.598	3.192	3.193	376,403
Cow	2.904	5.804	5.805	678,946
Teapot	3.644	6.320	6.321	739,788
Octopus	4.242	8.468	8.469	990,219
Unicycle	6.973	13.810	13.811	1.602,748
Roman	10.473	20.904	20.905	2.460,382
Sphere	15.314	30.624	30.625	3.616,807
Bunny	34.834	69.451	69.452	8.122,324
Dragon	54.296	108.588	108.589	12.588,589

Tabla 3. Tiempos de cálculo de la entropía utilizando Occlusion Query.

En la Tabla 1 se puede observar los resultados obtenidos para el cálculo de la entropía mediante el histograma de OpenGL. Estos tiempos son elevadamente altos para permitir un cálculo interactivo, incluso para objetos de baja complejidad. Esto es debido fundamentalmente a

que se realizan varias pasadas de *rendering* del objeto completamente para objetos de varios miles de triángulos y al coste de la propia operación del histograma de OpenGL. Estos resultados han sido obtenidos mediante el modelo de NVIDIA descrito anteriormente.

En la Tabla 2 se muestra los resultados obtenidos para el cálculo de la entropía mediante el histograma por software y hardware. En esta tabla podemos apreciar que los tiempos se mantienen bastante bajos incluso a medida que se aumenta la complejidad. Principalmente porque se realiza siempre una única pasada de *rendering* y a que la operación de lectura del buffer tiene un coste muy bajo.

En la Tabla 3 se puede ver los resultados obtenidos utilizando la técnica de *occlusion query*. En esta tabla podemos ver claramente que los tiempos obtenidos aumentan proporcionalmente en relación a la complejidad del modelo de prueba analizado. De la misma manera que en la técnica anterior, aquí se mantiene la proporción porque el número de pasadas de *rendering* es directamente proporcional al número de triángulos. Aunque sólo se realiza un *rendering* completo del objeto en la primera pasada. Los resultados obtenidos en la Tabla 2 y 3 han sido obtenidos mediante el modelo de ATI descrito anteriormente.

5. Conclusiones y Trabajo Futuro

La entropía es una métrica que ha sido utilizada principalmente para determinar el mejor punto de vista de un objeto 3D. En este trabajo hemos realizado un estudio para determinar qué técnica permite calcular la entropía dependiente de la vista de forma más eficiente. De las diferentes técnicas analizadas la que resulta mejor es el cálculo de la entropía mediante el histograma por software y hardware. Mediante esta técnica podemos conseguir el cálculo de la entropía prácticamente en tiempo real incluso para objetos complejos.

En el futuro, estudiaremos el cálculo de la entropía utilizando su descomposición jerárquica o recursiva, con el fin de utilizarla en un entorno multirresolución. Esto nos va a permitir, por un lado, distribuir el cálculo entre varios procesadores en caso de escenas altamente complejas y, por el otro, parar el cálculo en el nivel de información que nos interese.

Agradecimientos

Este trabajo ha sido financiado por el gobierno español (TIN2004-07451-C03-03 y FIT-350101-2004-15) y la Unión Europea (IST-2-004363 y fondos FEDER).

Referencias

[1] P. Barral, G. Dorme, D. Plemenos. Scene understanding techniques using a virtual camera. Eurographics 2000, Interlagen (Switzerland), August 20-25, 2000, Short papers proceedings.

[2] P. Barral, G. Dorme, D. Plemenos. Visual understanding of a scene by automatic movement of a camera. International Conference GraphiCon'99, Moscow (Russia), August 26 – September 3, 1999.

[3] P. P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint Selection Using Viewpoint Entropy. Vision, Modeling, and Visualization 2001 (Stuttgart, Germany), pp. 273-280, 2001.

[4] Pere Pau Vázquez, PhD thesis, On the Selection of Good Views and its Application to Computer Graphics. Technical University of Catalonia, 2003.

[5] P. P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Automatic View Selection Using Viewpoint Entropy and its Application to Image-Based Modeling. Computer Graphics Forum, desember-2003.

[6] J. Rigau, M. Feixas, and M. Sbert. Information Theory Point Measures in a Scene. IIiA-00-08-RR, Institut d'Informàtica i Aplicacions, Universitat de Girona (Girona, Spain), 2000.

[7] M. Sbert, M. Feixas, J. Rigau, F. Castro, and P. P. Vázquez. Applications of Information Theory to Computer Graphics. Proceedings of 5th International Conference on Computer Graphics and Artificial Intelligence, 3IA'02 (Limoges, France), pp. 21-36, 2002.

[8] P. P. Vázquez, M. Sbert. On the fly best view detection using graphics hardware 4th IASTED International Conference on Visualization, Imaging, and Image Processing (VIIP 2004).