Techniques for Computing Viewpoint Entropy of a 3D Scene

Pascual Castelló[1], Mateu Sbert[2], Miguel Chover[1], Miquel Feixas[2]

[1] Departamento de Lenguajes y sistemas Informáticos, Universitat Jaime I,
Campus Riu Sec, 12080 Castellón de la Plana, Spain
{castellp, chover}@lsi.uji.es
http://graficos.uji.es
[2] Institut d'Informàtica i Aplicacions, Universitat de Girona,
Campus Montilivi, 17071 Girona, Spain
{Mateu.Sbert, Miquel.Feixas}@ ima.udg.es

Abstract. Viewpoint entropy is a metric that allows measuring the visibility goodness of a scene from a camera position. In this work, we analyze different software and hardware assisted techniques to compute the viewpoint entropy. The main objective of this study is to identify which of these techniques can be used in real time for 3D scenes of high complexity. Our results show that interactivity can be obtained with occlusion query technique and that for real time we need a hybrid software and hardware technique.

1 Introduction

Recently, several methods have been developed to compute the goodness of a viewpoint. These methods have in common the use of the viewpoint complexity concept [2, 3, 15, 7, 8, 9, 13]. The notion of viewpoint complexity is used in several areas of Computer Graphics such as scene understanding and virtual world exploration, radiosity and global illumination, image-based modelling and rendering, etc.

In scene understanding and virtual world exploration, viewpoint entropy is used to automatically calculate suitable positions and trajectories for a camera exploring a virtual world [2, 3, 6, 1, 12].

In Monte Carlo radiosity and global illumination, viewpoint complexity is used to improve the scene subdivision in polygons and the adaptive ray casting [4, 5, 14].

In image-based modelling, viewpoint entropy is used to compute a minimum optimized set of camera positions.

Among the metrics that have been introduced for the complexity calculation, the viewpoint entropy has been the most fruitful metric up to date [10]. Recently, it has been embedded to the field of volume visualization to compute the best n-views of a volumetric object. However, the viewpoint entropy computation can be very expensive, especially when a very complex scene and multiple viewpoints have to be evaluated.

In this paper, we will assess different alternatives for its calculation on several geometric models with increasing complexity. For our calculation, we will make use of the facilities of modern hardware cards such as OpenGL histogram and Occlusion query as well as the new symmetric bus PCI Express [16, 17].

2 Viewpoint Entropy

The viewpoint entropy, based on the Shannon entropy definition, was introduced in [7, 9] and is a measure of the information provided by a point of view.

The Shannon entropy of discrete random variable X, with values in the set $\{a_1, a_2, ..., a_n\}$, is defined as

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

where $p_i=Pr[X=a_i]$, the logarithms are taken in base 2 and 0 log 0 = 0 for continuity. As -log p_i represents the information associated with the result a_i, the entropy gives the average information (or the uncertainty) of a random variable. The unit of information is called a bit.

To define viewpoint entropy we use as probability distribution the relative area of the projected faces (polygons) over a sphere of directions centered in the viewpoint. Thus, given a scene S and a viewpoint p, the entropy of p is defined as

$$I(S, p) = -\sum_{i=0}^{N_f} \frac{A_i}{A_t} \log \frac{A_i}{A_t}, \tag{2}$$

where N_f is the number of polygons of the scene, A_i is an approximation on a plane of the projected area of polygon i over the sphere, A_0 represents the projected area of background in open scenes, and A_t is the total area of the sphere. In a closed scene, the whole sphere is covered by the projected polygons, and thus $A_0=0$.

The maximum entropy is obtained when a certain point can see all the polygons with the same relative area. So, in an open scene, the maximum entropy is log (N_f + 1) and, in a closed scene it is equal to log N_f. Among all the possible viewpoints, the best is the one that has maximum entropy, i.e. maximum information captured.

3 Techniques for Computing the Entropy

In order to compute the viewpoint entropy, we need the number of pixels covered for each visible triangle from a particular camera position. This number will give us the projected area. Next we analyze several techniques that allow us to compute those areas.

3.1 OpenGL Histogram

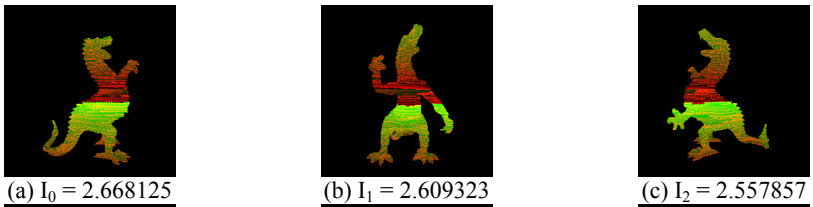
The OpenGL histogram was first used to compute the entropy in [11]. The OpenGL histogram let us analyze the colour information of an image. Basically, it counts the appearances of a colour value of a particular component. However, we can also use it to calculate the area of triangles that are visible from a viewpoint, without reading the buffer. Since version 1.2, OpenGL includes an extension called *glHistogram*. This extension is part of the image processing utilities. The OpenGL histogram is hardware-accelerated, although there are just a few graphics cards that actually support it (for instance, 3DLabs WildCat) and often is implemented in software.

In order to obtain the area of each visible triangle, we need to assign a different colour to each triangle. An important limitation is that histograms have a fixed size, normally of 256 different values. This is the most common value in many graphics cards. The *glGetHistogram* command returns a table that counts each colour value separated into channels. If we use the 4 RGBA colour channels, a 256 item table of 4 integer values will be returned, where each integer is the number of pixels this component has. Thus, if we want to detect a triangle, this should be codified using one single channel. This gives us a total of 1020 different values. That is to say, for channel R (1,0,0,0) up to (255,0,0,0), for channel G (0,1,0,0) up to (0,255,0,0), for channel B((0,0,1,0) up to (0,0,255,0) and finally for channel A (0,0,0,1) up to (0,0,0,255). The value (0,0,0,0) is reserved for the background.

Obviously the main drawback of this technique is that we need several rendering passes for objects with more than 1020 triangles. In each pass, we will obtain the area of 1020 different triangles. Using histograms with a higher number of items and making a rendering off-screen, will increase the number of colours and therefore making necessary less rendering passes. However, this possibility is outside the OpenGL specification and is hardware dependent. It was not possible for us to use a larger size histogram in the several graphics cards tested.

3.2 Hybrid Software and Hardware Histogram

The OpenGL histogram allows us to obtain the area of each visible triangle. However, as we said in the previous section, several rendering passes are needed for objects with more than 1020 triangles. Currently, new symmetric buses have appeared such as the PCI Express. In this new bus the buffer read operation is not as expensive as before. Therefore, it is possible to obtain a histogram avoiding making several rendering passes. The way to get it is very simple. A different colour is assigned to each triangle and the whole object is sent for rendering. Next, a buffer read operation is done, and we analyze this buffer pixel by pixel retrieving data about its colour. Using a RGBA colour codification with a byte value for each channel, up to 256*256*256*256 triangles can be calculated with only one single rendering pass. In Figure 1 we show an example of the entropy calculation using this method.



(a) $I_0 = 2.668125$ (b) $I_1 = 2.609323$ (c) $I_2 = 2.557857$

(d) $I_3 = 2.387822$ (e) $I_4 = 1.964738$ (f) $I_5 = 1.224885$

Fig. 1. Entropies from 6 different viewpoints for the Dragon model obtained with the Hybrid Software and Hardware Histogram. The maximal entropy viewpoint corresponds to (a).

3.3 Occlusion Query

This OpenGL extension is normally used to identify which scene objects are hidden by others, and therefore we shouldn't send them to render. In fact, what we do is just to render the bounding box of an object and, if it is not visible, the object is not sent for rendering. However, it can also be used to compute the area of the triangles that are visible from a particular camera position.

The OpenGL ARB_occlusion_query extension returns the number of visible pixels. In order to compute the area of each visible triangle from an object with this technique we will proceed as follows. First, the whole object is sent for rendering and the depth buffer is initialized. Second, we independently send each triangle for rendering. With this procedure it is necessary to make n + 1 rendering passes, n being the number of triangles in an object. We must mention that only in the first pass the whole geometry is rendered. In the following passes, one single triangle is rendered. However, a high number of renderings can significantly penalize this technique. In order to improve the results, this extension can be used asynchronously in contrast to its predecessor HP_occlusion_query. That is to say, it does not use a "stop-and-wait" execution model for multiple queries. This allows us to issue many occlusion queries before asking for the result of any one. But we must be careful with this feature because, as we mentioned above, this extension was not designed to deal with thousands of multiple queries. Thus, we can have some limitations depending on the graphics card.

3 Comparison

We calculated the viewpoint entropy from 6 camera positions, regularly distributed over a sphere that covers the object, using the different techniques we described

above. In order to compare them, we measured the time needed to compute the entropy from those cameras. As test models, we used several models of different complexities (see Figure 2). All models were rendered in a 256x256 pixels resolution using OpenGL vertex arrays. We used two different PCs: a Xeon 2.4 GHz 1GB RAM with an ATI X800XT 256MB and a Pentium IV 3.0 GHz 2 1GB RAM with an NVIDIA GeForce 6800GT 256MB. We must emphasize that between the two analyzed GPUs, only the NVIDIA card supports the OpenGL histogram.

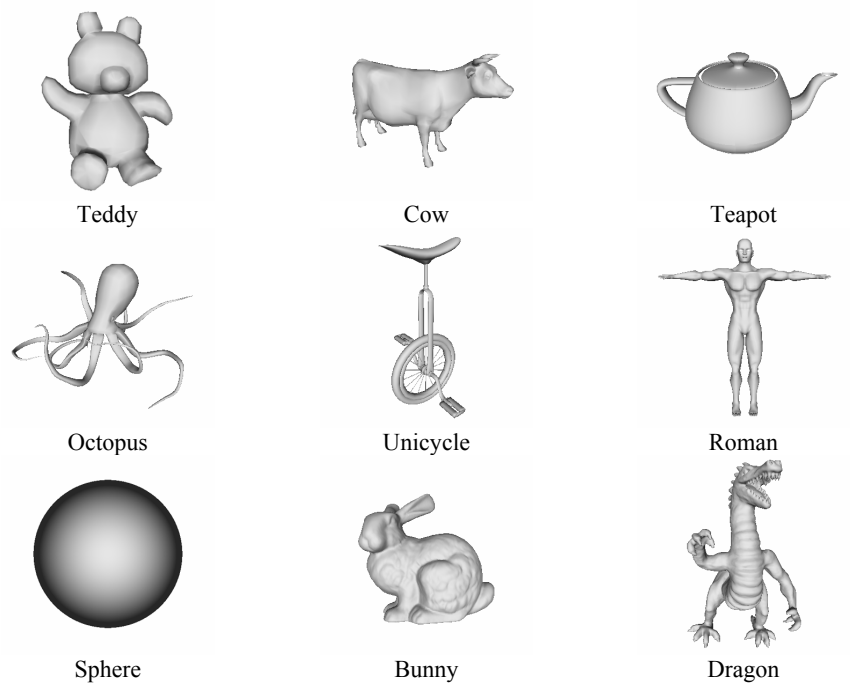


Fig. 2. Models used in our experiments.

Table 1. Results obtained for the viewpoint entropy calculation with the OpenGL Histogram

Model	**Vertices**	**Triangles**	**Rendering passes**	**OpenGL Histogram(ms)**	
				GeForce 6800 GT	Radeon X800 XT
Teddy	1,598	3,192	4	2,811.45	-
Cow	2,904	5,804	6	4,227.28	-
Teapot	3,644	6,320	7	4,927.65	-
Octopus	4,242	8,468	9	6,339.67	-
Unicycle	6,973	13,810	14	9,886.66	-
Roman	10,473	20,904	21	14,888.38	-
Sphere	15,314	30,624	31	22,136.10	-
Bunny	34,834	69,451	69	50,445.86	-
Dragon	54,296	108,588	107	80,029.94	-

Table 2. Results obtained for the viewpoint entropy calculation with the Hybrid Software and Hardware Histogram

Model	Vertices	Triangles	Rendering passes	SW+HW Histogram(ms)	
				GeForce 6800 GT	Radeon X800 XT
Teddy	1,598	3,192	1	11.66	16.62
Cow	2,904	5,804	1	13.36	19.10
Teapot	3,644	6,320	1	14.84	19.37
Octopus	4,242	8,468	1	17.28	20.69
Unicycle	6,973	13,810	1	18.53	23.24
Roman	10,473	20,904	1	24.12	29.85
Sphere	15,314	30,624	1	36.65	38.09
Bunny	34,834	69,451	1	57.91	67.04
Dragon	54,296	108,588	1	79.35	88.75

Table 3. Results obtained for the viewpoint entropy calculation with the Occlusion Query

Model	Vertices	Triangles	Rendering passes	Occlusion Query(ms)	
				GeForce 6800 GT	Radeon X800 XT
Teddy	1,598	3,192	3,193	26.88	25.19
Cow	2,904	5,804	5,805	47.49	44.41
Teapot	3,644	6,320	6,321	50.88	48.31
Octopus	4,242	8,468	8,469	67.17	64.48
Unicycle	6,973	13,810	13,811	109.88	100.78
Roman	10,473	20,904	20,905	162.75	151.31
Sphere	15,314	30,624	30,625	238.09	221.42
Bunny	34,834	69,451	69,452	553.36	460.74
Dragon	54,296	108,588	108,589	829.75	665.33

In Table 1, we show the results obtained with the OpenGL histogram. These times are too high to allow an interactive calculation, even for objects with a low complexity. This is fundamentally due to the several rendering passes of the whole object that we make when we use objects of several thousand triangles. The main cost component is the OpenGL histogram operation.

Table 2 shows the results with the hybrid software and hardware histogram. In this table we can see that the measured times are quite low even if the complexity is increased, mainly because we make one single rendering pass and the buffer read operation has a very low cost.

In Table 3, we show the results obtained with the Occlusion Query technique. In this table we can clearly observe that the measured times increase proportionally in relation to the complexity of the analyzed model. In the same way as the previous technique, the ratio remains unchanged here because the number of rendering passes is proportional to the number of triangles. A complete rendering of the object is only done at the first pass.

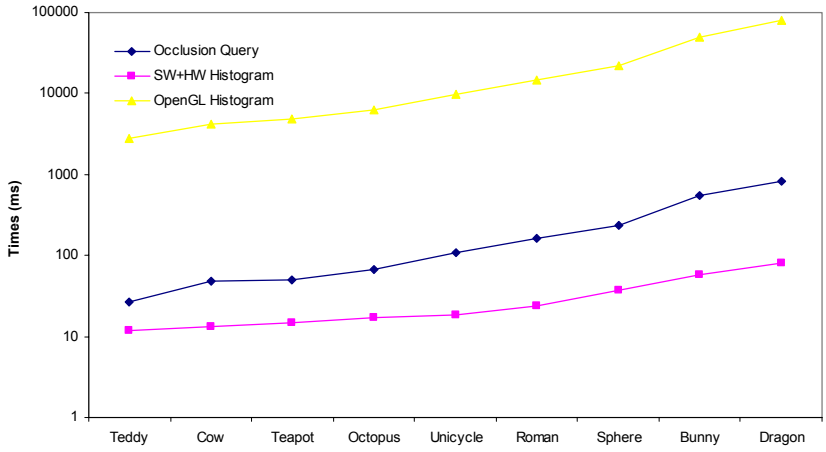


Fig. 3. Comparison of results obtained with the different analyzed techniques.

Finally, in Figure 3 we show as a summary a performance comparison among the different techniques. These results were obtained with the previously described NVIDIA card. We used an NVIDIA card because it fully supports all the techniques. Anyway, if we examine the proportions among the techniques with the ATI card, we can see that they are practically the same as with the NVIDIA card.

These results show clearly that by using the hybrid software and hardware histogram we can calculate the entropy in real time and even for complex objects (100,000 triangles), because times increase very slowly as complexity goes up. The next best technique is the Occlusion Query. Note that its cost grows as the object complexity increases, being unapproachable for complex objects for real time. Lastly, the OpenGL histogram technique is worst than the two others. This technique is useless for real time, unless we use objects of low complexity (1,000 triangles).

4 Conclusions

The viewpoint entropy is a metric that has been mainly used to determine the best viewpoint of a 3D object. In this paper we studied several hardware assisted techniques to allow computing the viewpoint entropy in an efficient way. Among the different analyzed techniques, the viewpoint entropy calculation with the hybrid software and hardware histogram has the best performance, followed by the occlusion query based technique. By using the hybrid software and hardware histogram technique we can practically achieve the entropy calculation in real time even for complex objects, while occlusion query technique allows us to obtain only interactivity.

We must take into account that the performance of the hybrid software and hardware histogram technique depends on the analysis of pixels done by the CPU and the read operation of the PCI Express bus. We also did some tests using higher resolutions, for example: 960x960, and we observed that the times for the occlusion query

are constant, but even in higher resolutions the hybrid software and hardware histogram technique gets better results than occlusion queries. The proportion is not as higher as before but still is significantly better. For our goal, we think that the resolution used in our experiments (256x256) is enough to obtain accurate results in the viewpoint entropy calculation.

References

1. C. Andújar, P. P. Vázquez, M. Fairén, Way-Finder: guided tours through complex walkthrough models, Computer Graphics Forum (Eurographics 2004), 2004
2. P. Barral, G. Dorme, D. Plemenos. Scene understanding techniques using a virtual camera. Eurographics 2000, Interlagen (Switzerland), August 20-25, 2000, Short papers proceedings
3. 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
4. V. Jolivet, D. Plemenos, M. Sbert, A pyramidal hemisphere subdivision method for Monte Carlo radiosity, Eurographics 99 Short Papers proceedings
5. M. Feixas, An Information Theory Framework for the Study of the Complexity of Visibility and Radiosity in a Scene. PhD thesis, Technical University of Catalonia, 2002
6. D. Plemenos. Exploring Virtual Worlds: Current Techniques and Future Issues. International Conference GraphiCon'2003, Moscow (Russia), September 5-10, 2003
7. 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
8. P. P. Vázquez, M. Feixas, M. Sbert, and A. Llobet. Viewpoint Entropy: A New Tool for Obtaining Good Views for Molecules. VisSym '02 (Eurographics - IEEE TCVG Symposium on Visualization) (Barcelona, Spain), 2002
9. P. P. Vázquez, PhD thesis, On the Selection of Good Views and its Application to Computer Graphics. Technical University of Catalonia, 2003
10. 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, December 2003
11. 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)
12. P. P. Vázquez and M. Sbert. Automatic indoor scene exploration. In International Conference on Artificial Intelligence and Computer Graphics, 3IA, Limoges, May 2003
13. 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
14. J. Rigau, M. Feixas, and M. Sbert. Entropy-Based Adaptive Sampling. Graphics Interface 2003 (Halifax, Canada), June 2003
15. 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
16. M. Segal, Kurt Akeley. The OpenGL Graphics System: A Specification (Version 2.0 – October 22, 2004). Silicon Graphics, Inc., 2004
17. A.Wilen, J.Schade, R.Thornburg. Introduction to PCI Express. A Hardware and Software Developer's Guide. Intel Press, 2003