Fast Adaptive Selection of Best Views

Pere-Pau Vázquez[1] and Mateu Sbert[2]

[1] Dept. LSI, Universitat Politècnica de Catal
[illegible]

[2] IIiA, Universitat de Girona
Campus Montilivi, EPS, E-17071 Girona, Spain
mateu@ima.udg.es

Abstract. Automatic computation of best views of objects is very useful. For example, they can be used as the starting point of a scene exploration, or to enrich galleries of objects available through Internet by adding an image a model that may help to decide if it is worth downloading. To select the most interesting viewpoint of an object, we use the so-called *viewpoint entropy*. The best view is the one which gives the most information of the object being inspected. In this paper we present an adaptive method to compute best views. Our adaptive scheme allows to improve over previous approaches the time of the selection of best views by an order of magnitude, and achieve a nearly interactive rate.

1 Introduction

The automatic selection of best views of objects may be very useful. Best views can be set as the starting point of a scene exploration, for example in medical applications. They can also be used to reduce the time devoted inspecting 3D models and to understand them. The simple addition of informative images of models might avoid to spend longtime downloading (translating) and inspecting objects. Furthermore, applications to games to aid in camera control are also possible. We present in this paper a new method for the adaptive selection of best views. Our algorithm is fast and can obtain a good view of an object in a couple of seconds.

The rest of the paper is organized as follows. In Section 2 we analyze previous work on good view selection. Section 3 explains the previous brute-force approach that uses viewpoint entropy. In Section 4 we present our new algorithm. In Section 5 we show the results and discuss them, and finally, in Section 6 we conclude and point out possible future work.

2 Previous Work

Colin [1] presents a method to select a good view to observe a scene modeled with an octree. This method chooses the view which shows the highest amount of voxels, according to two different visibility measures, one exact and another one approximate. Kamada and Kawai [2] consider a viewing direction to be good if in a projection, parallel line segments on a plane in 3D project as far away from each other as possible. Intuitively, the viewer should be as orthogonal as possible to every face of the 3D object. As this is not possible, they suggest to minimize (over all the faces) the maximum angle deviation between a normal to the face and the line of sight from the viewer. However, this method fails when comparing scenes with equal number of degenerated faces and it does not ensure that the user will see a large amount of detail [3]. Plemenos and Benayada [4] extend Kamada's definition. They consider a direction to be good if it minimizes the maximum angle deviation between a normal to the face and the line of sight from the viewer *and* it also provides a high amount of detail. If only the first condition is satisfied, it does not ensure that the view will not hide important information of the scene. Therefore, they add another parameter to the maximizing function which accounts for the detail seen. The parameter added is the number of visible faces from a viewpoint. Then, a function which combines both parameters is created. Moreover, an adaptive method is implemented, but the refining criterion is the number of visible faces. Barral *et al.* [3] present a method for the automatic exploration of objects or scenes. This method is based on the good view concept definition presented in [4]. In this case, the goodness of a view is computed by defining a new importance function that depends on the visible pixels of each polygon. An extension of this method can be found in Dorme [5], where an extra parameter, the so-called *exploration parameter* is added in order to avoid taking into account faces that have already been shown to the user. However, they admit that they have not been able to determine a good weighting scheme for the different factors. This causes some problems with objects containing holes, as these are not captured properly by the algorithm.

Marchand and Courty have presented a general framework that allows the automatic control of a camera in a dynamic environment [6]. The proposed method is based on the *image-based control* or visual servoing approach. Bourque and Dudek [7] describe a system for the automatic construction of image-based virtual realities to represent a real environment. The criterion used to determine the importance of a view is based on human attention. In the robotics literature, the goal of selecting a small set of cameras which allow to observe all object surfaces has been studied under the name of sensor planning. For example Wong *et al* [8] present an algorithm that searches all possible viewpoints, and selects the next best view as the one that can carve the most empty space voxels. Their system computes the next best view by optimizing an objective function that measures the quantity of unknown information in each of a group of potential viewpoints. Although the system is effective, such an approach may also result in views that observe surfaces at very oblique angles, which is undesirable in some areas such as Image-Based Rendering, as it yields poor sampling of colour in those surfaces. Freeman has exploited the *generic viewpoint assumption* to address shape from shading problems. The *generic viewpoint assumption* states that an observer is not in a special position relative to the scene [9]. It is commonly used to disqualify scene interpretations that assume special viewpoints, thus, it can be used to avoid ambiguities [10]. The Design Galleries^{TM} (DG) system [11] is a method to automatically set parameters for computer graphics and animation. They focus on the problem of parameter tweaking. The parameters which are studied by this technique are: light selection and placement for image rendering; opacity and colour transfer-function specification for volume rendering; and motion control for particle-system and articulated-figure animation.

Vázquez *et al.* [12] have presented a measure, *viewpoint entropy*, based on Shannon's entropy [13]. Gumhold [14] has presented a method for the automatic light source placement which also uses an entropy-based function. The function is intended to measure the information coming from the illumination of a scene. Some experiments with users have been carried out in order to improve the measure with perceptual information. An optimization method for the automatic light positioning is also presented.

3 Best View Selection Algorithm

Given a certain viewpoint, the goodness is evaluated by using the *viewpoint entropy* function, that is:

$$I(S,p) = -\sum_{i=1}^{n} p_i \log p_i \tag{1}$$

where the logarithms are taken in base two, and p_i is the relative projected area (A_i/A_t) of face i (A_i is the projected area of face i and A_t is the total projected area). The best view of an object will be the most informative one, the one with maximum viewpoint entropy. Determining the best view is not easy. A first approach presented in Vázquez *et al.* [12] uses a brute-force algorithm that analyzes the entropy in a dense set of viewpoints placed all around the object. The view with maximum entropy is selected as the best view. This method is depicted in Algorithm 1.

Algorithm 1 Computes the view with the highest entropy of an object.

```
Select a set of points placed in regular positions all around the object
maxI ← 0
viewpoint← 0
for all the points do
  aux ← Compute the viewpoint entropy
  if aux> maxI then
    maxI ← aux
    viewpoint ← current point
  end if
end for
Write maxI and viewpoint
```

The use of a brute-force method is justified by the essence of the entropy function. It is not continuous and makes therefore difficult to predict maximum reachable entropy values in the neighborhood of already analyzed points. The denser the set of views, the smaller the probability of missing important views.

4 Adaptive Best View Selection

In Section 3, we have presented the brute-force algorithm to select the best view of an object. This algorithm ensures that the best view is not missed. More concretely, it makes sure that the possible error committed is under an user-defined threshold. This is, the smaller the distance between two points, the lower the difference in entropy, so the denser the set of views analyzed, the smaller error we will commit. This is true although the function is not continuous, as the number of different faces that we will see if we move the camera will be small if the camera movement is also small. The user decides the number of views to analyze, the higher the number of views the smaller the probability of missing the best view. However, this algorithm is very demanding, as every view computation the OpenGL buffer must be read back and processed. In order to accelerate the computation of best views, an adaptive method is compulsory.

In [14] an adaptive method that predicts the best position for an entropy-based measure is developed. In this case the authors define the lighting entropy as a method to measure the information captured from viewpoint coming from a lit scene. It is used to determine the best light positions in a scene given an user position and viewing direction. In order to use a global optimization method, the authors assume that the lighting entropy function at the resolution they use is Lipschitz continuous.

We address the problem by a totally different perspective by the design of an adaptive system. We will start from a coarse set of views that will be recursively refined depending on the visible faces of each view. We will take advantage of the fact that two nearby positions usually *see* similar sets of polygons to *estimate* entropy values of new positions. The program will estimate an entropy in a new position by using the entropies of its neighbors and the set of faces that can be seen from the set of neighbor points. We have applied our method for single objects but a similar adaptive scheme can be designed for indoor scenes. Initially, six cameras are placed around the object in orthogonal positions (see Figure 1). At these positions the entropy is measured and stored. Moreover, we also store an array with the projected area of each face visible from the camera position. Once these six views are calculated, the set of points is triangulated and we predict the entropy

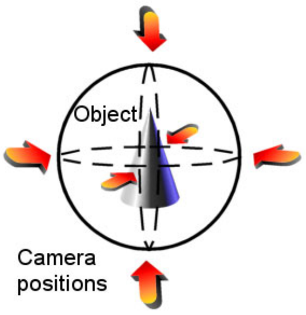

Fig. 1. The six initial camera positions around an object.

that could be captured from the middle point of the edges using a conservative estimator. To build this estimator we make the following assumptions:

– The set of visible faces from the new position is the union of faces that can be seen from the two endpoints of the edge.
– We see the faces at a better projection than in the neighbors.

The first assumption is used to ensure that we are not going to miss any face from the new view. For nearby points, a new view placed in the middle of them is likely going to see almost all the faces seen by two views. The second assumption is justified by the fact that in some situations, the number of visible faces does not increase but the amount of projected area of those faces does grow. These conditions make our estimation conservative in order not to miss important points. Although a new position could show faces that are not seen from the neighbors, some of the previously seen faces will hide. Moreover, the projected area of some of the faces will also decrease. As we estimate as if all of them grew, this will compensate for the apparition of new faces. Moreover, because estimated entropy could be always higher than the entropy of the two nearby points, we avoid the selection of infinite in between views by adding a constraint: a new view will be analyzed only if the angle between the new view and the already computed ones is larger than a given threshold (we use five degrees). The algorithm stops when none of the estimated values is higher than the values of entropy correctly computed.

4.1 Entropy Estimation

For every initial camera position, we compute its entropy value (I). We also code and store in an array the contributions of the visible faces from the camera position to the entropy. To evaluate the entropy from each new view we need to know the relative projected areas of all the visible faces from that point. This information is unknown, and the only information we have relative to the new position is the distance that separates it from the two initial views, and the entropy of these neighbors (together with the contributions to entropy of each visible face from the initial views). For the purpose of building our estimator, we assume that the new view will *see* the union of faces that can be seen from the two neighbors, and that the new position sees these faces at a better projection. As the solid angle depends on the rotation angle between the initial positions and the currently estimated viewpoint, we will use this angle to estimate the new projected areas. A good estimator for the projected area A could be then: $A = (1 + \left|sin\alpha\right|/2)$, where α is the angle from the new position and the neighbor. However, this results in a too pessimistic estimator in the sense that it yields too high values and therefore many views have to be then analyzed. After some tests, we found empirically that a good entropy estimator (I_p):

$$I_p = -\sum_{i=0}^{n} Nf_p((1 + \left|sin\alpha\right|/2) * \log((1 + \left|sin\alpha\right|/2)/N_f) \tag{2}$$

where N_{f_p} is the *estimated number* of faces seen from the new view and N_f is the number of faces visible from one neighbor. When two neighbors see the same face, we select the term which adds a higher value to entropy.

4.2 Algorithm

We have designed an algorithm that computes quickly the best view of an object. It performs basically four steps:

1. Evaluate the viewpoint entropy of the initial views. Build a triangular mesh with the views as its vertices.
2. Predict the entropy of the middle points of each edge, using equation 1.
3. If the highest estimated value is higher than the already computed values, evaluate its real entropy and add the new view. Go to step 2.

We have chosen a set of six initial views, placed on the intersecting points of the X, Y, and Z axis with a bounding sphere that contains the object (see Figure 1). Furthermore, instead of using a triangular mesh, we use a mesh of spherical triangles, as we want all the views to be placed at the same distance from the object (see Figure 2a). The edges will be arcs, and the middle points of the edges are the views whose entropy is estimated adaptively. This method is sketched in Algorithm 2. The estimation of entropy is only performed for edges whose endpoints are placed at a distance over a threshold (we have used a threshold of five degrees). The algorithm stops when none of the predicted values is higher than the already computed ones.

Algorithm 2 Adaptive computation of the best view of an object.

```
Select the initial six points on the three edges
Evaluate the entropy on these points
Triangulate the set of points
Predict the entropy on the middle point of all edges
maxPred← maximum of predicted entropies
while maxPred > maximum of computed entropies do
    Compute the real entropy of the highest predicted value and insert the new view
    Predict the entropy on the middle point of the new edges
    maxPred← maximum of predicted entropies
end while
Select the view with maximum entropy
```

In Figure 2 we can see how the viewpoints are progressively inspected. In Figure 2a we see the initial mesh, and in Figures 2b to d we see how the views are inserted (the lighter the point, the higher its entropy). The performance of our algorithm strongly depends on the amount of views from which we have to correctly compute viewpoint entropy. This is due to the fact that view analysis requires reading back the colour and the depth buffer to main memory and then inspecting it. The complexity of both operations depends on the size of the image. In order to accelerate the computation of the viewpoint entropy we may reduce image size. This could be accomplished using two different strategies:

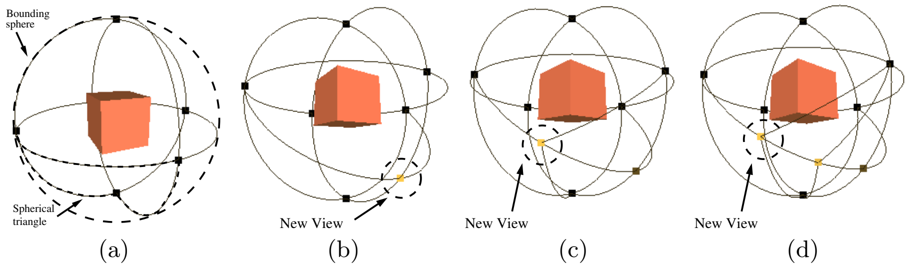

Fig. 2. The four initial steps of the best view selection for the scene of the cube.

– Reduce the resolution of the image.
– Reduce the amount of pixels read and analyzed.

The first solution could cause to worsen the precision of the entropy computation. Although the loss in precision could be unimportant, as only very small faces would disappear, we have decided to use the second strategy. A safe effective way to reduce the amount of pixels to be read back to main memory consists in predicting how much of the object *really* projects to the resulting image. This can be simply achieved by the use of a bounding box of the object. The bounding box is computed while the scene is being loaded and then used every frame that has to be analyzed to predict the real size of the projection. This process is depicted in Figure 3. The region to analyze corresponds to the projection of the bounding box of the object on the viewing plane (dashed line). This way we avoid reading all the pixels of the rendered view, which is a slow operation, and we also analyze a smaller image. This method dramatically improves the performance of each entropy evaluation thus reducing the time of the overall algorithm up to a 80%.

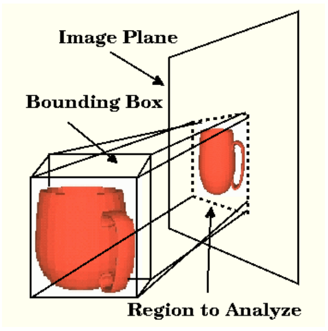

Fig. 3. Projection of the bounding box of a mug. The dashed line denotes the region that will be read and analyzed.

5 Results

The method presented above adequately computes the best views of objects of several thousands of polygons in less than one second, while the brute-force method needs one or two minutes depending on the model and the number of views analyzed. In general, the views selected by both methods are the same. Figure 4 shows the views analyzed for a set of objects: a cube, a cow, and two chairs. The results of the new method are compared with the brute-force method in Table 1. The second column shows the computation time for the brute-force method and the last one the timings for the adaptive strategy. To achieve a similar result than with the adaptive method, the brute-force method needs to analyze about 1500 views. In this case we obtain speed-ups of up to 396:1 for the case of the cube.

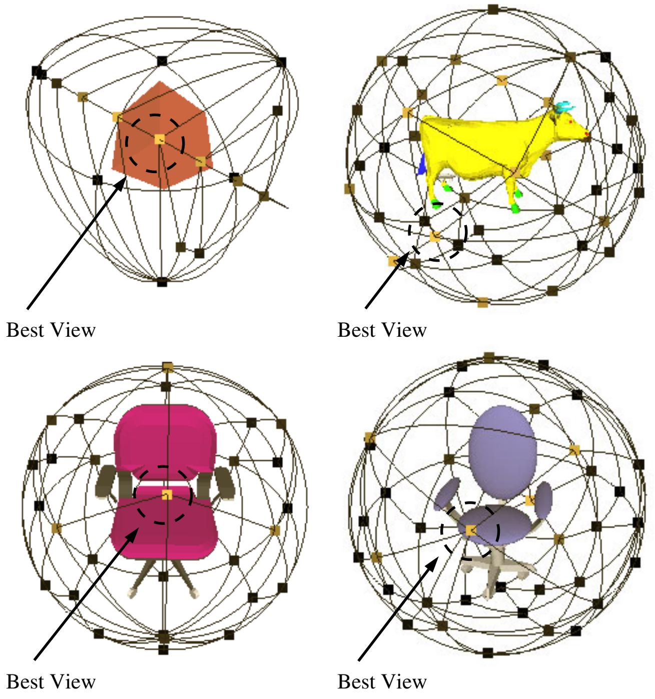

Fig. 4. The different analyzed views for each of the objects. The colour of the viewpoint encodes the entropy, the lighter the colour the higher the entropy.

Another example gives more curious results. In Figure 5*a* we can see the best view of a lorry selected with the brute-force method. Its entropy is 0.192. On the other hand, the best view using the adaptive method is the one that appears in 5*b*, whose entropy is 0.1957 slightly higher. As the lorry is practically symmetric the views are very similar. However, with the adaptive method even the second best view is better (exactly 0.1948) than the brute-force one. In this case its position is very close to the one of the best view as selected with the previous algorithm. This view is shown in 5*c*. Although throughout the calculation process we use a conservative estimator, the resulting values of estimated entropy are in a feasible range (we do not generate estimated values which are very high in comparison with the real values of entropy captured), and consequently, the system performs very well, only a small number of images has to be actually analyzed. We obtain the same best view positions than with the

Model	*Brute-Force Time (ms)*	*Adaptive Time (ms)*	*Reduction ratio*
Cube	47540	120	396:1
Cow	28320	1490	19:1
Chair 1	22170	720	31:1
Chair 2	29700	1730	17:1

Table 1. Comparison of results of the adaptive method and the brute-force method.

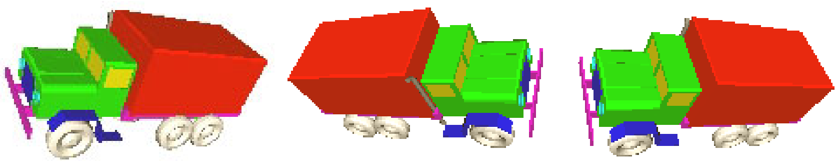
(a) $I = 0.192$ (b) $I = 0.196$ (c) $I = 0.195$

Fig. 5. The best views selected by our two algorithms. *(a)* was selected with the brute-force method, *(b)* and *(c)* with the adaptive. Note that even the second best view chosen by the adaptive method, *(c)*, is better than *(a)*.

brute force method, and sometimes some views that the previous algorithm had missed. Moreover we reduce the time of computation from one or two minutes for a high number of views to less than one second in most cases. As this estimation is conservative, in the sense that the predicted value will be slightly higher than the resulting entropy, the probability of missing important positions will be low. Though we have not been able to find any particular case, our system could fail if the object had such a shape that some of the faces were only visible from a special point, and this was the point of maximum entropy. If the positions close to this point saw a small number of faces and had a low entropy value, we could miss this position. However, this is not likely to happen, as some faces that are only visible from a very concrete region will project to a small area and therefore will have a low entropy rate.

6 Conclusions and Future Work

In this paper we have presented a new method for the automatic selection of best views of objects. Best views can be used as a technique to improve galleries of objects providing images of the models that help the user decide which of those are interesting. Moreover, they can be used as starting points for navigation systems. In the future we want to address the problem of view selection using perceptual issues. Instead of taking into account only the geometry of the scene, best views could also be selected according to the illumination of a scene.

This can help in automatic camera positioning for video capture or automatic navigation in realistic rendering systems.

Acknowledgements

This project has been partially supported by TIC-2001-2416-C03-01 of the Spanish government, SGR2001-00296 grant from Catalan Government, and financial support from the Ministry of Universities, Research and the Information Society from Catalan Government.

References

[1] C. Colin. Automatic computation of a scene's good views. In *Proc. MICAD*, February 1990.

[2] T. Kamada and S. Kawai. A simple method for computing general position in displaying three-dimensional objects. *Computer Vision, Graphics, and Image Processing*, 41(1):43–56, January 1988.

[3] P. Barral, G. Dorme, and D. Plemenos. Scene understanding techniques using a virtual camera. In A. de Sousa and J.C. Torres, editors, *Proc. Eurographics'00, short presentations*, 2000.

[4] D. Plemenos and M. Benayada. Intelligent display in scene modeling. new techniques to automatically compute good views. In *Proc. International Conference GRAPHICON'96*, July 1996.

[5] G. Dorme. *Study and implementation of 3D scenes comprehension techniques.* PhD thesis, Université de Limoges, 2001. In French.

[6] E. Marchand and N. Courty. Image-based virtual camera motion strategies. In P. Poulin S. Fels, editor, *Proc.of the Graphics Interface Conference, GI2000*, pages 69–76, Montreal, Quebec, May 2000. Morgan Kaufmann.

[7] E. Bourque and G. Dudek. Automatic creation of image-based virtual reality. In *Sensor Fusion and Decentralized Control in Autonomous Robotic Systems*, volume 3209, pages 292–303, Bellingham, WA, October 1997. SPIE - The International Society for Optical Engineering. ISBN 0819426415.

[8] L. Wong, C. Dumont, and M. Abidi. Next best view system in a 3-d object modeling task. In *Proc. International Symposium on Computational Intelligence in Robotics and Automation (CIRA)*, pages 306–311, 1999.

[9] W. T. Freeman. Exploiting the generic view assumption to estimate scene parameters. In *Proc. 4th International Conference on Computer Vision*, pages 347–356, Berlin, Germany, 1993. IEEE.

[10] A. L. Yuille, J. M. Coughlan, and S. Konishi. The generic viewpoint constraint resolves the generalized bas relief ambiguity. In *Proc. of Conference on Information Scienes and Systems (CISS 2000)*, Princeton University, March 15–17 2000.

[11] J. Marks, B. Andalman, P. A. Beardsley, W. Freeman, S. Gibson, J. Hodgins, T. Kang, B. Mirtich, H. Pfister, W. Ruml, K. Ryall, J. Seims, and S. Shieber. Design galleries: A general approach to setting parameters for computer graphics and animation. In T. Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 389–400. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[12] P.-P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint selection using viewpoint entropy. In T. Ertl, B. Girod, G. Greiner H. Niemann, and H.-P. Seidel, editors, *Proceedings of the Vision Modeling and Visualization Conference (VMV-01)*, pages 273–280, Stuttgart, November 21–23 2001. IOS Press, Amsterdam.
[13] T. M. Cover and J. A. Thomas. *Elements of Information Theory.* Wiley, 1991.
[14] S. Gumhold. Maximum entropy light source placement. In *Proc. of the Visualization 2002 Conference*, pages 275–282. IEEE Computer Society Press, October 2002.