

Fast Adaptive Selection of Best Views

Pere-Pau Vázquez[†]

Mateu Sbert[‡]

[†] Dept. LSI, Universitat Politècnica de Catalunya
ETSEIB - Campus Sud, Diagonal 648, 8a planta, E-08028 Barcelona, Spain
pvazquez@lsi.upc.es

[‡] Institut d'Informàtica i Aplicacions
Universitat de Girona
Campus Montilivi, Edifici P1 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 the Internet; adding a good image to a model may give a good idea of it before deciding to download. To select the most interesting viewpoint of an object, we use the so-called *viewpoint entropy*. We assume 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 other approaches with ratios of up to 360:1 on the selection of best views, and therefore achieve a nearly interactive rate.

Keywords: Best View Selection, Entropy

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 to find models through the Internet. The simple addition of informative images of models might avoid to spend hours downloading (maybe translating) and inspecting objects. Furthermore, they can be applied to games if we guide the best view selection by weighting

interesting polygons (such as the ones of the main character) with importance values that favor the selection of a view that focuses on the interesting object.

In this paper we present 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 also uses viewpoint entropy. In Sec-

tion 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

In this Section we review previous work on viewpoint selection.

Kamada and Kawai 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 [1]. Intuitively, the viewer should be as orthogonal as possible to every face of the 3D object. Of course, this is not possible, and therefore 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 details[2].

Colin [3] 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.

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 set of views 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 func-

tion which combines both parameters is created. Barral *et al.* [2] present a method for the automatic exploration of objects or scenes. This method is based on the good view notion 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. They derive two operators (a density estimator and an orientation estimator) which are used to determine which parts of a scene will attract the user attention.

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 [8, 9, 10]. It is commonly used to disqualify scene interpretations that assume special viewpoints, thus, it can be used to avoid ambiguities [11]. The Design GalleriesTM (DG) system is a method to automatically set parameters for computer graphics and animation. The pa-

parameters studied 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 [12]. In Vázquez *et al.* [13] have presented a measure based on Information Theory. It is dubbed *viewpoint entropy* and is based on Shannon’s entropy [14]. Gumhold [15] 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 adaptive method for the automatic light positioning is also presented.

3 Brute-Force Method

As introduced in Vázquez *et al.* [13], given a certain viewpoint, the goodness can be evaluated by using the *viewpoint entropy* function, that is:

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

where the logarithms are taken in base two, and p_i is the relative projected area (A_i/A_t) of face i . 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 [13] uses a brute-force algorithm that analyzes the entropy for a dense set of viewpoints placed all around the object. The view with maximum entropy is selected as the best view.

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

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 viewpoint entropy
  if aux > maxI then
    maxI ← aux
    viewpoint ← current point
  end if
end for
Write maxI and viewpoint

```

already analyzed points. The denser the set of views, the smaller the probability of missing important views. This brute-force method is depicted in Algorithm 1.

This method ensures that the best view is not missed. More concretely, it makes sure that the possible error committed is under an user-defined threshold, as the user decides the number of views to analyze. However, this algorithm is very demanding, as every time the OpenGL buffer must be read back and processed. In order to accelerate the computation of best views, an adaptive method is compulsory.

4 Adaptive Best View Selection

In [15] an adaptive method that predicts the best position for an entropy-based measure is developed. The authors define the lighting entropy as a method to measure the information captured from viewpoint coming from a lit scene. 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. Although the visibility of two different close positions may vary, this does not happen very often. Two nearby positions

usually *see* similar sets of polygons. We will take advantage of this fact to *estimate* entropy values of new positions by using the information on visibility of faces and already computed entropy of neighbor points. We have successfully 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, together with a bitmap that encodes the visibility of each face from the camera position. Once these six views are calculated, the set of points is triangulated and we predict the entropy at the middle points 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 from the two endpoints of the edge.
- Each predicted position sees all the faces under the same projected area.

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 the same set of faces seen by two already computed points. 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. The third assumption allows to cheaply estimate the total entropy from the predicted view. These conditions make our estimation conservative in the sense that

the error committed will be positive, that is we may predict a higher entropy than the real one but not otherwise. This helps not to miss important views. 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 threshold, for example five degrees. The algorithm stops when none of the estimated values is higher than the values of entropy correctly computed.

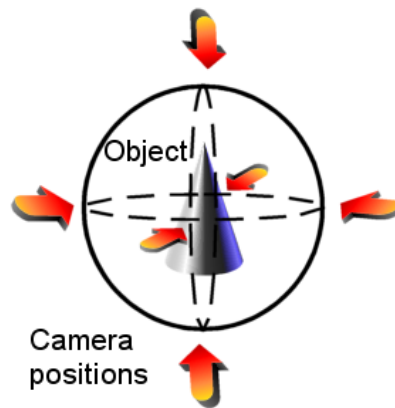


Figure 1: The six initial camera positions around an object.

4.1 Entropy estimation

For every initial camera position, we compute its entropy value (I) with equation 1. We also code and store in a bitmap the visibility of the faces from the camera position. To evaluate the entropy from each new view we need to know:

- The set of visible faces.
- How much area they project.

Both values are unknown, the unique data that we know from the new position is the

distance that separates it from the two initial views. As these data are very difficult to obtain, we assume that each predicted position shows the faces under the same projected area, and that a view placed in the middle of an edge *sees* all the faces seen by the two endpoints but at a better angle. 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.

Working under the assumption that the initial viewpoints show all the visible faces with the same projection area, we can simplify equation 1 to: $H(X) = -N_f p_i \log p_i$, where N_f is the number of faces visible from the viewpoint¹, and p_i is the projected area of each face. As we are seeing all the faces with the same projection area, $p_i = 1/N_f$.

So for each viewpoint we know now what the projected area of each face would be if all the faces were seen under the same projected area. Now we want to estimate the entropy that the same visible faces would yield if the model is rotated a certain α radians. As we have made the assumption that with the rotation all the visible faces will increase their projected area, we increase their probability proportionally to the rotation angle. To make the estimation conservative the predicted entropy H_p will be:

$$H_p = N_{fp}((1 + |\sin\alpha|/2)/N_f) * \log((1 + |\sin\alpha|/2)/N_f) \quad (2)$$

where N_{fp} is the estimated number of faces seen from the new position. However, this results in a too pessimistic estimator. The values of the predicted entropy are low enough to allow missing important points.

¹As the entropy takes into account the contribution to entropy of the background, the actual value of N_f is the number of visible faces plus one.

We have found empirically that the following one is better:

$$H_p = N_{fp}(1 + |\sin\alpha|/2) * \log((1 + |\sin\alpha|/2)/N_f) \quad (3)$$

Although strictly speaking this value cannot be considered an entropy, it is a good conservative estimator of viewpoint entropy value. The points where we predict the entropy are the middle points of the edges that connect neighboring camera positions. This prediction is calculated two times, each one taking into account the actual entropy of each of the correctly computed viewpoints. To build the final entropy estimation, we apply the following heuristic formula to these values:

$$H_p = (H_{p1} + H_{p2}) * \frac{N_{fp}}{N_{f1} + N_{f2} - 1} \quad (4)$$

where H_p is the total estimated entropy and H_{p1} and H_{p2} are the predicted values of the two neighbors. N_{fp} is the estimated number of visible faces from the predicted view. N_{f1} and N_{f2} the number of faces of the endpoints of the edge. Remember that the estimated number of views visible from the predicted point (N_{ft}) is the union of the sets of visible faces from the neighbors.

4.2 Algorithm

We have designed an algorithm that computes quickly the best view of an object. It performs basically three 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, according to equation 1.
3. If the highest estimated value is higher than the already computed values add the new view and 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. 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.

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

```

Select the initial six points
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
    Insert new view at the point of maximum predicted entropy and compute its real entropy
    Estimate the entropy on the middle point of the new edges (when the middle point is not too close)
    maxPred ← maximum of predicted entropies
end while
Select the view with maximum entropy

```

The estimation of entropy is only performed for edges whose endpoints are placed at a distance over a threshold (we have used an angle of five degrees). The algorithm stops when none of the predicted values is higher than the already computed ones. After choosing the point with maximum entropy, we subdivide the edges where it belongs and capture the corresponding views. We do this only once, in order to detect any possible maximum that might be close to our detected maximum and that was missed. In Fig-

ure 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.

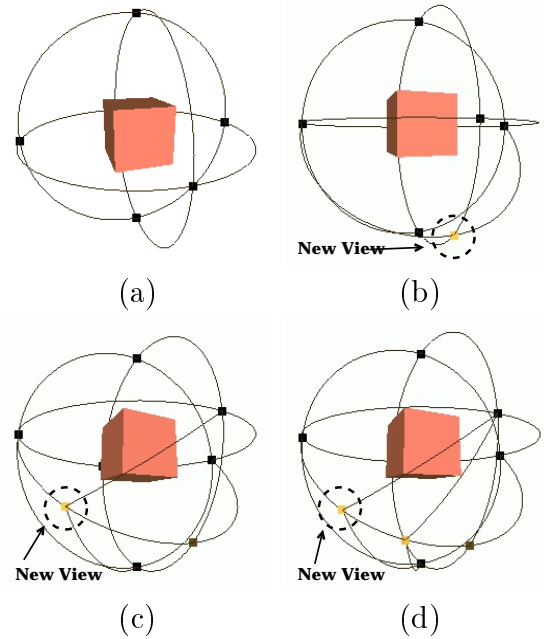


Figure 2: The four initial steps of the best view selection for the scene of the cube. In *a* the initial six views are shown. *(b)* to *d* show the initial three selected points.

4.3 Improving performance

The performance of our algorithm strongly depends on the amount of views that have to be analyzed, that is, the number of viewpoints 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 some improvements can be introduced:

- Reduce the size of the image.
- Reduce the amount of pixels that have to be read and analyzed.

The first solution can be achieved by reducing the size of the window. Unfortunately, this would cause to worsen the precision of the entropy computation. Although the loss in precision could be unimportant, (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. An easy manner of computing such area is by using a bounding box of the object. Hence, our algorithm is improved in the following way:

- Compute a bounding box of the object when it is being loaded.
- For every view computation, project the points of the bounding box to the viewing plane.
- Read and analyze only the pixels where the bounding box is projected.

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%, as it is shown in the following Section.

5 Results

The method presented above adequately computes the best views of objects in less than one second, while the brute-force

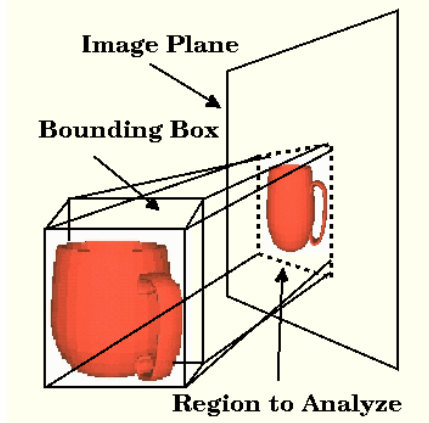


Figure 3: Projection of the bounding box of a mug. The dashed line denotes the region that will be read to main memory and then analyzed.

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 martini cup, a mug, and a candlestick.

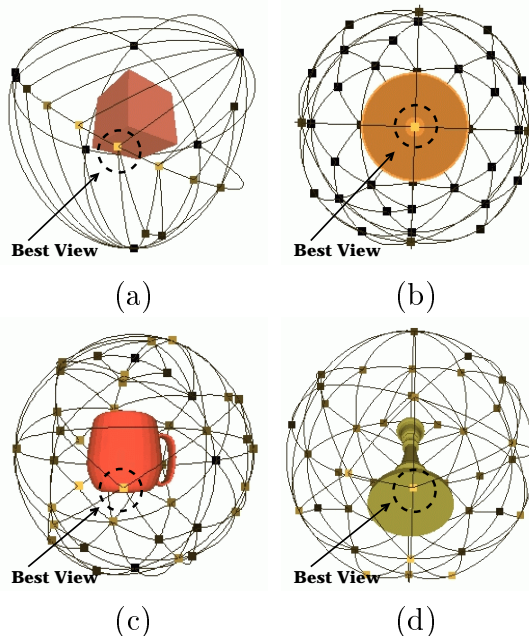


Figure 4: The different analyzed views for each of the objects. The colour of the viewpoint encodes the entropy, the lighter the colour the higher the entropy.

The results of the new method are compared with the brute-force method in Table 1.

<i>Model</i>	<i>Brute-Force Time (ms)</i>	<i>Adaptive Time (ms)</i>
<i>Cube</i>	47.54	0.13
<i>Martini</i>	60.35	0.94
<i>Mug</i>	59.69	1.09
<i>Candlestick</i>	61.28	0.75

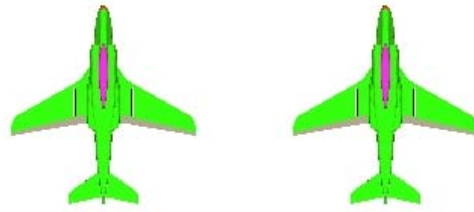
Table 1: Comparison of results of the adaptive method and the brute-force method.

The first column contains the name of the model analyzed. 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 365:1 for the case of the cube. In Table 2 we can see the improvement that is achieved thanks to the bounding boxes strategy.

<i>Model</i>	<i>Adaptive Time (ms)</i>	<i>B. Boxes Time (ms)</i>
<i>Cube</i>	660	130
<i>Martini</i>	1400	940
<i>Mug</i>	1780	1090
<i>Candlestick</i>	1690	750

Table 2: Acceleration due to the use of bounding boxes. The results obtained with the simple adaptive method are in the middle column, and the last column shows the timings obtained with the strategy that uses bounding boxes.

Other examples are shown in Figure 5. In 5a we see the best view of the airplane computed with the brute-force method, and in 5b the best view has been computed with the adaptive method. Note that these views are the same.



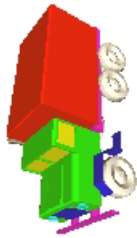
(a) Brute-force (b) Adaptive

Figure 5: The best views selected by our two algorithms.

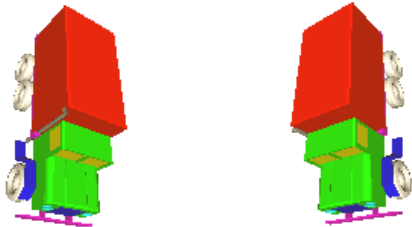
Another example gives more curious results. In Figure 6a we can see the best view of a lorry selected with the brute-force method. Its entropy is 0.191855. On the other hand, the best view using the adaptive method is the one that appears in 6b, whose entropy is 0.1957 slightly higher. As the lorry is practically symmetric the views are very similar. However, with the adaptive method even the second best view is better (it has exactly an entropy of 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 6c.

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 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 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 impor-

Brute-force



$$\frac{(a) I(S, p) = 0.191855}{\text{Adaptive}}$$



$$(b) I(S, p) = 0.1957 \quad (c) I(S, p) = 0.1948$$

Figure 6: The best views selected by our two algorithms. Note that the two first views chosen by the adaptive method ((b) and (c)) are better than (a).

tant positions will be low.

Although 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 ob-

jects 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, and SGR2001-00296 grant from Catalan Government.

REFERENCES

- [1] 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.
- [2] 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.
- [3] C. Colin. Automatic computation of a scene's good views. In *Proc. MICAD*, February 1990.
- [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] 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.
- [9] W. T. Freeman. Exploiting the generic viewpoint assumption. *Intl. Journal Computer Vision*, 20(3):243–261, 1996.
- [10] W. T. Freeman. The generic viewpoint assumption in a framework for visual perception. *Nature*, 368(6471):542–545, April 1994.
- [11] 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 Sciences and Systems (CISS 2000)*, Princeton University, March 15–17 2000.
- [12] J. Marks, B. Andalman, P. A. Beard-sley, 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 Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 389–400. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.
- [13] 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 2001 (VMV-01)*, pages 273–280, Berlin, November 21–23 2001. Aka GmbH.
- [14] T. M. Cover and J. A. Thomas. *Elements of Information Theory*. Wiley, 1991.
- [15] Stefan Gumhold. Maximum entropy light source placement. In *Proceedings of the Visualization 2002 Conference*, page ??? IEEE Computer Society Press, October 2002.