Automatic Keyframe Selection for High-Quality Image-Based Walkthrough Animation Using Viewpoint Entropy

Pere-Pau Vázquez

Mateu Sbert

Institut d'Informàtica i Aplicacions
Universitat de Girona
Campus Montilivi, Edifici P1 EPS, E-17071 Girona, Spain
{pvazquez|mateu}@ima.udg.es

ABSTRACT

The computation of high quality animation sequences is expensive. Generation time for each frame can take a few hours. Recently, Image-Based Rendering methods have been proposed to solve this problem. As these techniques obtain new arbitrary views from precomputed ones at low cost, walkthroughs may be computed faster. Consequently, the selection of the precomputed images is a very important step. The initial set of keyframes should fulfill two requirements, it must be small but provide as much information as possible on the scene. In this paper we review several keyframe selection strategies and then we propose a new method based on entropy that achieve similar, and in some cases better, results.

Keywords: High-Quality Walkthrough, Image-Based Rendering, Keyframe Selection, Entropy

1 INTRODUCTION

Simulating realistic images with the classical approaches (radiosity, raytracing...) is a very complex process, it requires a lot of time and computational resources to obtain high quality images. Improvements in rendering algorithms (multiresolution rendering, textures, visibility preprocess, and so on) are continuously challenged by increasing scene complexity.

In contrast to this, there is a group of methods named Image-Based Rendering (IBR), that are able to generate in real time arbitrary views from precomputed images. IBR techniques rely on the use of images instead of a geometric representation. This allows to break the rendering computation time dependency on the number of polygons. Instead, the frame rate depends mainly on the image resolution.

In this paper we review several previously used strategies for selecting keyframes for an Image-Based walkthrough animation methods [1], and then we propose a new method based on entropy which yields similar and in some cases better results. The rest of the paper is organised as follows. In Section 2 we present a high quality image-based walkthrough method, in Section 3 we explain previous initial keyframe placement strategies, later, in Section 4 we present our new hardware-based method. In Section 5 we show the results and, finally, in Section 6 we conclude and point out possible future work.

2 HIGH QUALITY IMAGE-BASED WALKTHROUGH

One of the most relevant stages in walkthrough rendering techniques which use keyframing is the generation of in between frames. Our work is strongly based on the high quality walkthrough that uses IBR techniques by Myszkowski *et al* [1]. It consists of three main steps:

- First it performs a 3D warping [2] of the data, which avoids occlusion artifacts.
- Then, an adaptive *splatting* technique, similar to the one in [3], is used to reduce the existing gaps.
- And finally, the holes produced by occluded objects are removed by blending two different keyframes as in Mark *et al* [4].

Once this process has finished there remains a couple of problems to solve: pixels that are occluded in both images and specular and glossy effects due to the change in view direction. The first problem is solved by tracing the corresponding rays. Specular and strongly glossy effects are usually of high contrast, thus they attract the viewer's attention, so they must be handled very accurately. In order to enhance the quality of the image, the perceivable errors caused by these pixels are computed with the aid of a perception-based Animation Quality Metric [5, 6]. The pixels selected by the AQM method are added to the ones which are occluded in both images and are also ray traced. Figure 1 depicts the processing of inbetween frames.

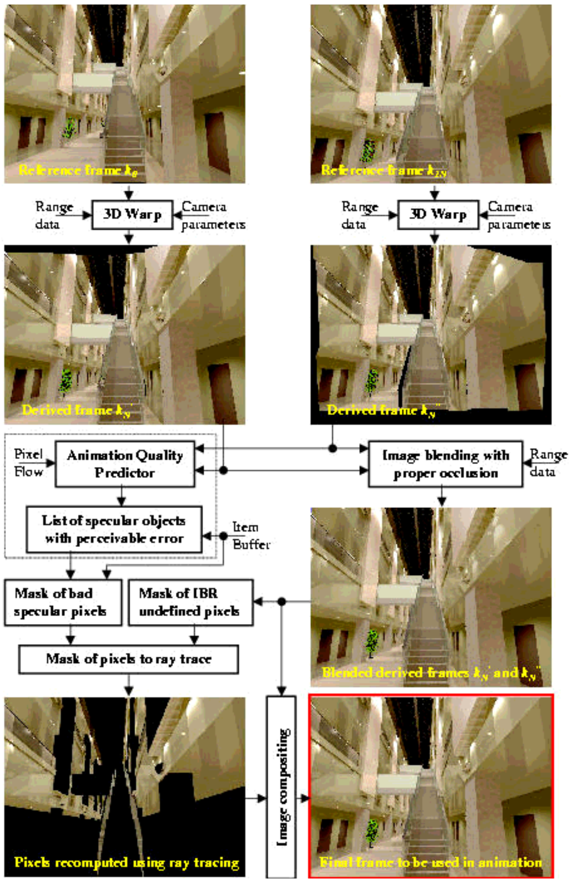

Figure 1: Frames generation process. Courtesy of Karol Myszkowski and Takehiro Tawara from the MPI for Computer Science of Saarbrücken.

3 PREVIOUS KEYFRAMES PLACEMENT STRATEGIES

In this Section we compare several previous keyframes placement strategies. Along this Section we assume a fixed number of initial keyframes and the goal will be to minimize the number of pixels that cannot be properly derived from the keyframes due to visibility problems.

3.1 Uniform Placement

The simplest choice that can be tested is an uniform time step. That is, we divide the steps of the path by the spacing we decide and we choose the first frame of each segment. Its cost is null and yields surprisingly good results. However, this method is only suitable for scenes where Pixel Flow variation from frame to frame is low, in these cases, choosing the keyframes placement according to PF results in a lower number of wrong pixels.

3.2 Pixel Flow

The occlusion problems that appear when warping two images together in a new one are somehow related to the differences between the camera parameters of the new position and the reference images. A measure of such differences can be the Pixel Flow[1] between the two frames. When the reference images minimize the pixel dislocations, the number of pixels that cannot be computed by warping is also reduced.

Pixel Flow computation is carried out by using McMillan's warping technique [2], which makes its calculation cost negligible [1]. Computing initial keyframes by using the Pixel Flow can be done in the following way: the average of PF for all frames along the animation path is accumulated. Then, the total accumulated PF is split into equal intervals, which determine the keyframe placement. This results in a well balanced per frame distribution of pixels, which cannot be derived by warping. However, this method is only suitable when the variance of PF in a sequence is high. If the sequence has little difference of PF between frames, uniform placement behaves better. This leads to define a new strategy which starts from an uniform placement which is modified according to PF.

[1]Pixel Flow is computed as the motion vector field that measures where each pixel moves to from frame to frame in an animation sequence [1].

3.3 Uniform placement modified by Pixel Flow

As previous strategies were found to be useful only for a particular case of walk-through, Myszkowski *et al* [1] investigated a new method that combines the strengths of both. This method uses a constant initial spacing between frames equal to Δ. Then, this spacing is changed according to the PF, but the change of spacing is limited to a maximum size of $\pm A\Delta$, where A is a fixed coefficient, usually taking values of $0.25 < A < 0.75$. Computing the spacing between frames is performed in the following way. Let $\overline{p}_i$ denote the actual accumulated PF magnitude between the currently considered pair of frames and let $\overline{P}$ denote the average accumulated PF for all pairs of keyframes in the sequence. The fixed spacing Δ is assumed for $\overline{p}_i$ and $\overline{P}$ computation. A new reference location will be placed at distance δ_i of the previous one, with $\delta_i = (1 + \alpha_i)\Delta$. Where α_i is computed as:

$$\alpha_i = \frac{\overline{P} - \overline{p}_i}{\overline{P}},$$

and α_i is

$$\alpha_i = \begin{cases} -A & if \quad \alpha_i < -A, \\ \alpha_i & if \quad -A \leq \alpha_i \leq A, \\ A & if \quad \alpha_i > A. \end{cases}$$

This method allows to reduce the number of pixels which have to be recomputed at each frame. The cost of placing the new reference images is negligible if we take into account that the Pixel Flow must be computed anyway as some of the subsequent steps of the animation rendering (such as AQM [7] processing and pixel reprojetion) will use it.

4 KEYFRAME SELECTION USING VIEWPOINT ENTROPY

Although the computation of Pixel Flow can be done using IBR techniques and therefore its cost is low, there are some scenarios where this information would not encode real changes in scene. For example, a camera tilt would produce the Pixel Flow magnitude to grow although all the pixels previously seen may be visible in the next image. Thus, Pixel Flow could produce undesired effects in our keyframe selection algorithm. In this Section we present a new technique which addresses the problem of keyframe selection from a very different point of view. The selection of good views is done by only using an ID-rendering and analyzing the information contained in the frame-buffer with a measure called *viewpoint entropy*. This measure can be interpreted as the amount of information obtained from a point. Computing the difference of information between two frames can somehow indicate the amount of pixels to recompute from one frame to another. This difference is used to determine which reference images are selected. We will see that this measure performs well and its computation can be accelerated by using OpenGL. First we introduce the foundaments of the measure.

4.1 Viewpoint Entropy

Viewpoint entropy [8] is based on an Information Theory measure, the Shannon Entropy [9, 10]. The Shannon Entropy of a discrete random variable X with values in the set $\{a_1, a_2, ..., a_n\}$ is defined as

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

where $p_i = Pr[X = a_i]$, the logarithms are taken in base 2 and when $p_i = 0$ we take $p_i \log p_i = 0$. 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*.

Let a scene S consist of a set of N_f faces. We are going to use as probability distribution the relative area of the projected faces over the sphere of directions centered in the viewpoint p, as in Fig. 2. Thus, the *viewpoint entropy* of a point p from a scene S is defined as:

$$I(S,p) = -\sum_{i=0}^{N_f} \frac{A_i}{4\Pi} \log \frac{A_i}{4\Pi}$$

where A_i is the projected area of face i and 4Π is the solid angle of the sphere. Hence, $\frac{A_i}{4\Pi}$ represents the *visibility* of face i with respect to the point p. When $i = 0$, the area projected is the background. This is required because when computing entropy we must use a probability distribution function, otherwise the entropy measure would not be consistent. On the other hand, this is not the only reason, a probability distribution could be built by normalizing the measures without using the background. However, such a measure cannot handle with distances: projecting the scene under the same direction but at a different distance would give the same value. The use of background gives the objects which are near higher entropy than the ones which are far. In exceptional cases this could lead to small errors, but this can only happen if we compare two different views, one showing all the faces, including background, with the same projected area and the second showing all the faces but the background with the same projected area. The error commited will be of $log(N_f+1)-logN_f$, which for a big value of N_f is negligible. However, this does not happen in practice because usually a number of faces are not visible.

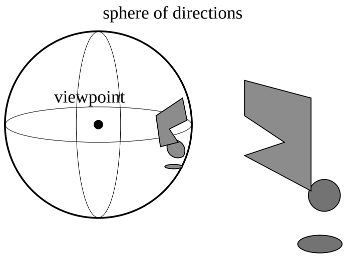

Figure 2: Computation of the viewpoint entropy.

4.2 Perspective frustum entropy

In many cases what we really want to measure is the amount of information provided from a single image that does not cover all the sphere of directions, as this is the way we usually obtain the 2D representations. In order to do this we must consider the case where we have a perspective projection. We only take into account the objects that are inside the perspective frustum (see Fig. 3). In this case the measure is called *perspective frustum entropy* and it is defined as [8]:

$$I_{F_P}(S,p) = -\sum_{i=0}^{N_f} \frac{A_i}{A_f} \log \frac{A_i}{A_f}, \qquad (1)$$

where A_i is the relative projected area of face i over the sphere that falls into perspective frustum F_P, which is a pyramid of rectangular basis, and A_f is the total area of the projected image. The projected areas of each face can be computed by weighting the pixels of the projection plane by the solid angle they subtend and summing up all the pixels belonging to the same face. Consequently, the resulting function will be:

$$I_{F_P}(S,p) = -\sum_{i=0}^{N_f} \frac{\sum_{j=0}^{N_{pix_i}} A_j * SA_j}{A_f} * \\ * \log \frac{\sum_{j=0}^{N_{pix_i}} A_j * SA_j}{A_f}, \qquad (2)$$

where N_{pix_i} is the number of pixels that project face i and SA_j is the solid angle subtended by pixel j.

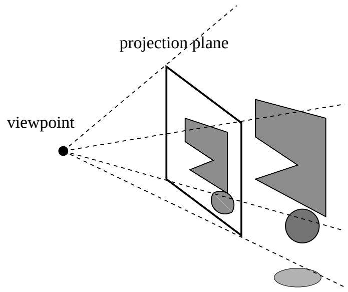

Figure 3: Only the objects inside the frustum are considered.

4.3 Keyframe selection using perspective frustum entropy

To compose our set of keyframes we place the camera on the steps of the walkthrough. We select the initial frame, and the next one will be the one which shows a difference of information with the initial one higher than a certain threshold. We have tuned the threshold in order to obtain the same number of keyframes to be able to compare the results with the ones obtained with other strategies. Next we see how we evaluate the difference on information between two frames.

Ideally, the difference on viewpoint entropy between two frames can be computed using the *Kullback-Leibler distance* [9, 10], but this is not possible when having probability values of 0, as it is our case. Moreover, we only want to take into account increases on information. A face seen in frame i and not seen in frame $i+1$ should not increase the value of the difference, only the faces which were not seen in previous frame or the ones which are seen better in the new frame. Our algorithm computes the perspective frustum viewpoint entropy for the first camera position of the segment and, for each frame, the difference is computed by summing up the increases of contribution to entropy for each face. A high value of difference means that there is a high amount of information appearing in the current frame that does not appear in the first one. If it is over a threshold the current frame is selected, and set as the initial of the segment and the process is started again.

Algorithm 1 Selects the reference images of the walkthrough.

```
Select a set of points placed in a path
infdiff ← 0; iniImg ← 0;
Store the contribution to viewpoint entropy of each face of point 0
for pos = 1..maxViewp do
  if pos − iniImg > maxFramesSeg then
    iniImg ← pos; Write pos;
  else
    Store the contribution to viewpoint entropy of each face of point pos
    for all the faces do
      if diffContribFaces(curFace, pos, iniImg) > 0 then
        infodiff ← infodiff + diffContribFaces(curFace, pos, iniImg)
      end if
    end for
    if infodiff > threshold then
      iniImg ← pos; Write pos
    end if
  end if
end for
```

Algorithm 1 presents the pseudocode that we have used to select the keyframes for the animation. Function *diffContribFaces(curFace, pos, iniImg)* returns the difference of the contribution to the entropy of face $curFace$ for point pos and $curFace$ for point $iniImg$. If it is positive it means that in the current position, face $curFace$ is providing a larger value of information than in position $iniImg$ (i.e. the projected area of this face at the new frame is larger than at the initial frame of the segment).

Projections are computed using graphics hardware. The complexity of the method depends on the number of frames to analyze and the resolution of the images. Ideally, the resolution of images should be the same than the used to render, but if necessary it can be reduced to improve performance. Analyzing an image is fast, and for a resolution of 400×400 a frame rate of 13 to 14 frames per second is achieved.

5 RESULTS

Several tests have been made with the different strategies. Table 1 depicts the average percentage of errors for these methods. Although uniform placement of keyframes gives good results, the better ones are obtained either with viewpoint entropy differences or with a suitable flexibility coefficient with the Uniform Placement modified by Pixel Flow variation method. On the other hand, some flexibility coefficient values can yield better results than our method in some walkthroughs, as we can see in Table 1 for value .75. However, to select the best one, they have to be tested before, as walkthroughs with different variations on PF could require a different coefficient. In contrast to this, with our method the same threshold will yield similar results regardless the walkthrough. Efficiency improvements can be obtained by reducing the level of detail. For instance, reducing the window size to 200×200 leads to an improvement of four in computation time, with similar results than full resolution as the polygons which would disappear represent low amount of information in the view.

In Figure 4 we can see two frames of the walkthrough. The initial keyframes of the first walkthrough (Figures (a) and (c)) were generated using the uniform placement plus PF (with a δ of 0.75) modification. Figures (b) and (d) come from a walkthrough generated from initial images computed by our method. The white regions denote the pixels that have to be recomputed because they do not appear in any of the keyframes of the segment (the images were modified with xv to obtain a good B/W printing).

6 CONCLUSIONS AND FUTURE WORK

We have developed a new method for the selection of keyframes for walkthrough renderings. Its cost is low, keyframes are selected in seconds, as most of the process can be accelerated using graphics hardware. A further acceleration will come from the reduction of image size, which would speed up analysis, with a cost in accuracy, but we will also seek for measures that help predict the behaviour of viewpoint entropy according to the camera movement.

Acknowledgements

The authors want to thank K. Myszkowski and T. Tawara for the data and the renderings they did for us. We also want to thank M. Feixas, Ph. Bekaert, and H. Schirmacher for their continuous support and advice. This work has been partially supported by SIMULGEN Open ESPRIT project #35772, grant BR98/1003 of Universitat de Girona, and TIC-2001-2416 of the Spanish government.

REFERENCES

[1] Karol Myszkowski, Przemyslaw Rokita, and Takehiro Tawara. Perception-based fast rendering and antialiasing of walkthrough sequences. *IEEE Transactions on Visualization and Computer Graphics*, 6(4):360–379, October 2000.

[2] L. McMillan. *An Image-Based Approach to Three-Dimensional Computer Graphics, Ph.D. Dissertation.* PhD thesis, April 1997.

[3] L. He J. Shade, S. Gortler and R. Szeliski. Layered depth images. In *Computer Graphics Proceedings (Proc. SIGGRAPH '98)*, pages 231–242, July 1998.

[4] L. McMillan W. R. Mark and G. Bishop. Post-rendering 3d warping. In *Proc. of 1997 Symposium on Interactive 3D Graphics*, pages 7–16, New York, April 1997. ACM Press.

[5] Alan Chalmers, Ann McNamara, Scott Daly, Karol Myszkowski, and Tom Troscianko. *Image Quality Metrics.* ACM SIGGRAPH, July 2000.

[6] Karol Myszkowski. The visible differences predictor: Applications to global illumination problems. In G. Drettakis and N. Max, editors, *Rendering Techniques '98 (Proceedings of Eurographics Rendering Workshop '98)*, pages 233–236, New York, NY, 1998. Springer Wien.

[7] Karol Myszkowski, Przemyslaw Rokita, and Takehiro Tawara. Perceptually-informed accelerated rendering of high quality walkthrough sequences. In Dani Lischinski and Greg Ward Larson, editors, *Rendering Techniques '99*, Eurographics, pages 5–18. Springer-Verlag Wien New York, 1999.

[8] Pere-Pau Vázquez, Miquel Feixas, Mateu Sbert, and Wolfgang Heidrich. Viewpoint selection using viewpoint entropy. In T.Ertl, B. Girod, G.Greiner, H. Niemann, and H.-P. Seidel, editors, *Vision, Modeling, and Visualization 2001*, 2001.

[9] R.E. Blahut. *Principles and Practice of Information Theory.* Addison-Wesley, 1987.

[10] T.M. Cover and J.A. Thomas. *Elements of Information Theory.* Wiley, 1991.

Methods	**% Wrong Pixels**
Uniform placement	4.04213
Uniform Pixel Flow	4.19657
Uniform Placement + 0.25 $*$ PF	3.39578
Uniform Placement + 0.48 $*$ PF	2.00395
Uniform Placement + 0.75 $*$ PF	1.95706
Uniform Placement + 0.91 $*$ PF	4.97146
Perspective frustum entropy	2.71660

Table 1: Average wrong pixels using different strategies. 0.25 to 0.91 are different values for the flexibility coefficient of the segment length. Note that frustrum entropy behaves similarly or better than Unif. Placement plus Pixel Flow methods.

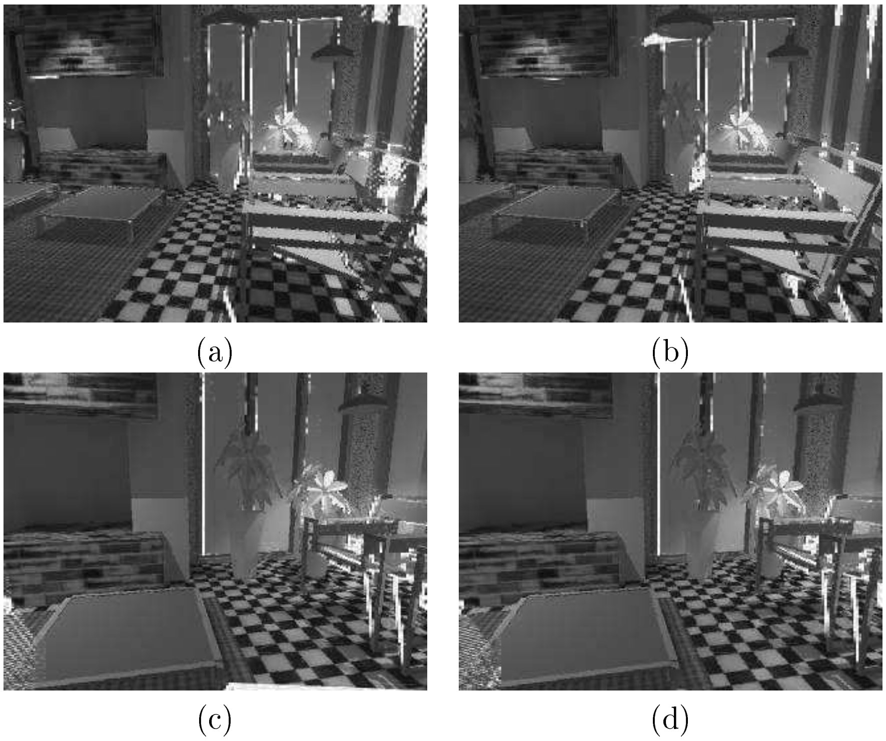

Figure 4: Comparing errors from different strategies. Figures *(a)* and *(c)* were generated from the initial sets computed by the Uniform Placement modified by PF. Figures *(b)* and *(d)* were computed using the frames generated by the entropy-based algorithm. Completely white regions denote the wrong pixels.