Image-Based Crowd Rendering

Franco Tecchia[1], Céline Loscos[1], Yiorgos Chrysanthou[2]

[1] Virtual Environments and Computer Graphics, Computer Science Department, University College London, London, UK
[2] Computer Science Department, University of Cyprus, Nicosia, Cyprus

Abstract

Populated urban environments are very important in many applications such as urban planning and entertainment. However rendering in real time many people in a complex environment is still challenging. In this paper, we propose methods for rendering real time animated crowds in virtual cities. We take advantage of the properties of the urban environment, and the way a viewer and the avatars move within it, to produce fast rendering, based on positional and directional discretization. To allow the display of a large number of different individual people at interactive frame rates, we combined texture compression with multi-pass rendering. The results show that we can visualise in real time a city with thousands of animated people.

Keywords: Image-based rendering, crowd rendering, virtual cities simulation, texture compression.

Figure 1: The crowd rendering system.

1 Introduction

The wide use of computer graphics in games, entertainment, medical, architectural and cultural applications, has led it to becoming a prevalent area of research. At the current stage of technology, a user can interactively navigate through complex, polygon-based scenes rendered with sophisticated lighting effects and high quality antialiasing techniques. Animated characters (or agents) with which the users can interact are also becoming more and more common. However, the rendering of crowded scenes with thousands of different animated virtual people has still not been addressed sufficiently for real time use. In this paper we propose new methods for the rendering of these highly populated scenes. To render animated human avatars in a complex polygonal model will result in non-interactive frame rate. In this paper, we propose an image-based rendering approach for displaying multiple avatars.

In order to minimize geometrical complexity, each human is represented with a single adaptive impostor. Appropriate impostor images are selected depending on the viewpoint position and the frame of animation. A previous approach [14] has already proposed such a solution. However, in that approach the required texture memory is excessive, resulting in a simulation that included only one type of avatar, as we see in their example. They still generated animations in real time with thousands humans but they look all the same. Our current work boosts the quality of rendering using aggressive optimisations and adding important environmental effects such as shadows.

We analysed all the improvements needed to allow more variety with an increase of the visual quality, while keeping a real time frame rate. This led us to a set of new techniques presented in this paper, which combined together help to display crowd of humans with an undeniable improved quality while keeping the rendering cost low. We minimize the popping effect when changing views, by choosing the impostor representation that fits best for walking humans. However, the technique used to select the best fitting impostor can be applied to other kinds of objects. We also decided on a strategy to decrease the amount of texture memory required for one human, as well as finding new displaying methods to make every avatar look different. To minimize the memory consumption we drop the regular-grid organization of the images, removing all the unused space in the impostor images set, in this way reducing the memory requirements by about 3/4. This compression technique reduces the size of the required texture memory for each kind of human, allowing the addition of several kinds of humans. To enhance the crowd variety without increasing the memory usage, we use multi-pass rendering. An example of the final rendering system can be seen in Fig. 1.

In Section 2, we briefly introduce some of the previous work on virtual city simulation, as well as the previous image-based rendering approaches which are relevant to our work. We then describe an image-based rendering method to display humans in real time, explaining the impostor representation chosen in Section 3, detailing the texture compression algorithm in Section 4, and the multi-pass algorithm in Section 5. Finally we conclude with some results and a discussion.

2 Background

The rendering of populated urban environments requires the synthesis of what are often considered two separate problems: the real-time visualisation of large-scale static environments, and the visualisation of animated crowds and traffic. Because both are expensive to render it is essential to reduce the amount of time required for each frame to be displayed. In this paper, we focus on real-time display of an animated crowd. We believe that the effort made on the animated crowd display combined together with accelerating techniques for walkthrough in virtual environments, should allow high-quality visualisation of big cities.

Large-scale environments are those containing millions of polygons. Although thousands of polygons can be displayed and visualised in a real-time frame rate, delays appear between frames for

a larger number of polygons decreasing the quality of the visualisation and the ability of the user to walk through. There has been a lot of published work on this subject. There are in general three different classes of methods that can be used for accelerating the rendering of large environments: visibility culling, imaged-based rendering and level-of-detail representation. In our case, we need to reduce the number of polygons to display as well as to take care of the real-time animation of the avatars. Visibility culling can be a very efficient acceleration in urban scenes, but still many polygons would need to be rendered. Think for example a crowded square, a user might visualise thousands of virtual avatars as well as view the surrounding city details. Even the additional use of level of details techniques results in too many polygons to display. Considering these limitations, an image-based rendering approach seemed more suitable for both the animation and the rendering of animated avatars. We focused on the lowest level, when the viewer is at a certain distance from the virtual humans, and potentially has in view a very large number of them. In applications for which a user needs to have a closer look, only the closer avatars can be rendered with polygons.

The principle of image based rendering techniques is to replace parts of the polygonal content of the scene with images. These images can be either computed dynamically or *a priori*. Maciel [8] uses pre-rendered images to replace polygonal parts of a static environment in a walkthrough application. This is done individually for single objects or hierarchically for clusters of objects. These images are used in a load balancing system to replace geometry which is sufficiently far away. A year later Schaufler et al. [12] and Shade et al. [13] independently presented the concept of dynamically generated impostors. In this case object images are generated at run-time and re-used for as long as the introduced error remains below a threshold. Numerous algorithms can be found in the literature that try to generate better approximations. For example Chen [3], Debevec [6] and McMillan [10] warp the images to adapt them to different viewpoints while Mark [9] and Darsa [5] apply the images on triangular meshes to better approximate the shape of the object. Schaufler [11] proposed a hardware assisted approach to image warping and Dally [4] used an algorithm that is very efficient in storing image data starting from a number of input images.

The literature on human modelling and rendering is also very extensive. However the largest part of it is concerned with achieving realistic approximations using complex and expensive geometric representations. Even with the help of level of detail techniques it would be almost impossible to use a very large number of such representations in a real-time system. An alternative which was recently employed by Aubel et al. [1, 2] is the use of impostors for the rendering of the virtual humans. In [1] each human is replaced by a single impostor while in [2] the authors take a much more detailed approach where each body part is replaced by an impostor, overall using 16 impostors for each human. In both of these methods the impostors are computed dynamically and used only for a few frames before being discarded. Tecchia and Chrysanthou [14] proposed a less accurate but much more scalable method which uses fully precomputed images. They showed results with only one individual replicated many times due to the excessive texture requirements.

Part of our work is based on the approach of Tecchia [14] which mapped an appropriate texture onto an impostor to display walking humans. In this previous approach, to generate the impostors, a set of textures is created, each corresponding to a frame of animation. Each texture is composed of a set of images of the character taken from different positions. A sampled hemisphere is used to capture the images, from 32 positions around the character and 8 elevations. At run time, depending on the view position with respect to each individual, the most appropriate image is chosen and displayed on an impostor, which is a single polygon dynamically oriented toward the viewpoint. No interpolation is used between views, as this would be too CPU-intensive. The appropriate texture to map is chosen depending on the viewpoint and the frame of animation. To improve the rendering speed, the humans are drawn frame by frame of animation and the textures are loaded only once per frame of rendering.

However several limitations can be observed in the technique used in [14]. First it needs a lot of texture memory because 32*8 samples images need to be stored in one texture. Popping between frames of animation can also be observed due to the sampling of the view position (there are 11.25 degrees of differences in the orientation of the object between each image). As a consequence of the cost of the texture memory, the authors showed animation only for a single type of character. However, we decided to use this method as a basis because it opens a great potential if efforts are made to reduce memory requirements.

In the next sections, we detail the different contributions of this paper. First we studied different ways of placing the impostor polygon to reduce popping effects (see Section 3). To reduce the amount of texture memory needed, we combined texture compression explained in Section 4 together with multi-pass rendering which helps to enable variety as described in Section 5.

3 Choosing the impostor representation

When image-based representation are used to render complex objects, two common forms of artefacts may arise: missing data due to inter-occlusion may cause black regions to appear, and popping effects may occur when the image samples are warped and/or blended to obtain the final image. As we introduced earlier, we try to maximize the rendering speed using a minimal geometric complexity for each impostor; this leads us to use a single polygon as the plane on which to project a sample. In this scenario, the main perceived artefact is the popping between different samples as the viewpoint changes; unfortunately, the multi-pass algorithm used in our system to improve the crowd variety prevents us from blending together different samples, solution that could have mitigated the problem. Another way to reduce the popping effect could be to augment the number of samples. However since we still want to minimize the memory consumption, we preferred to use some other methods. Taking into account that we have only a limited number of image samples available, we decided to accept this popping effect up to a certain extent, while putting some effort to minimize it.

The popping artefact is due to the fact that all the points on the surface of the sampled object are projected onto the same plane, from the direction that the camera is facing when the sample is created. Obviously, as the camera position changes, the projection of such points on the impostor cannot change, and the current impostor is no longer an exact replica of the object appearance. The amount of error for a generic point on the object surface is proportional to the distance of the point from the projection plane. This is demonstrated in Fig. 2.

The plane commonly used in literature as the projection plane for an impostor is usually the one perpendicular to the view direction from which the sample image was taken. This plane does not take into account the shape of the object nor any kind of special occlusion that could be present in the image. We then decided to try a different approach: given an object and the camera position from where the sample image is created, we search for the projection plane passing trough the object that minimises the sum of the distances of the sampled points and the projection plane.

To apply such idea, we need to project back in the 3D space the points visible in the impostor image, to get 3D visible samples of the object. We apply a Principal Component Analysis (PCA) to the set of 3D points and identify the two principal eigenvectors as directions describing the projection plane. In the case of samples

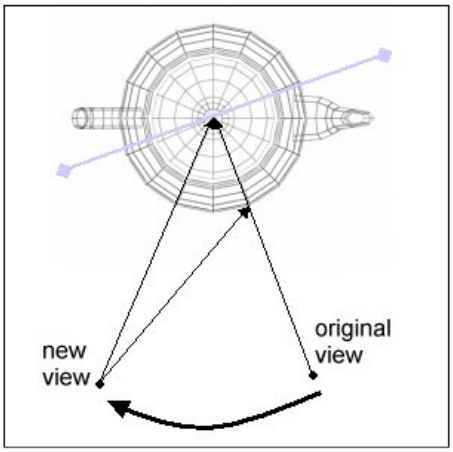

Figure 2: Error introduced when changing the viewpoint. This error is proportional to the distance between the 3D point and the projection plane.

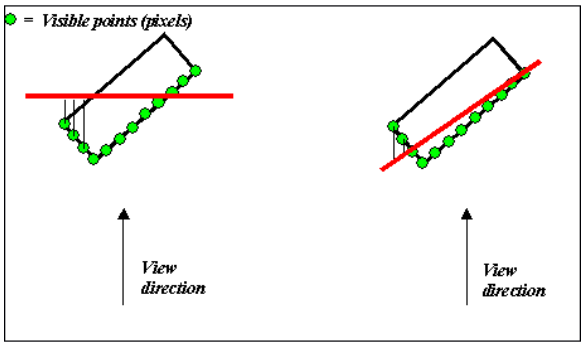

Figure 3: Distance of the visible samples from the impostor plane. Left: The plane is perpendicular to the camera direction. Right: The best fitting plane is chosen to minimise the distance between sample points and the projection plane.

of human polygonal models, such a plane results in a significantly better approximation of the position of the visible pixels in respect to the actual point positions in 3D. Unfortunately, other visual artefacts arose using the best fitting plane as the impostor plane; in fact, the new special orientation of this new plane produces an asymmetric warping of the image depending on which direction the camera moves away from the sampling position. For some extreme cases, for a particular plane orientation and a certain distance of the camera, perspective distortions can also become too evident. This is more visible when moving upwards rather that around since our object (the avatar) is longer along that dimension.

As the best fitting plane computed with PCA cannot be used as is, we reduce the artefacts by combining the two candidate orientations of the plane. Starting from an impostor plane purely perpendicular to the camera, we ”perturb” its orientation using the result plane obtained from a PCA of the sample image. We therefore minimise the popping while limiting the introduction of other artefacts. In practice we found that averaging the two directions, worked well.

4 Image compression

Although the hardware texture memory available has increased, still, efforts should be made to reduce the amount used for displaying virtual humans. First it can be noticed that because humans are walking, the movement is symmetric. Instead of 32 samples, we can then reduce it to 16 and get the others 16 by mirroring the texture. Such symmetry can be noticed in other objects, such as for cars or bicycles and this approximation may be useful for these as well.

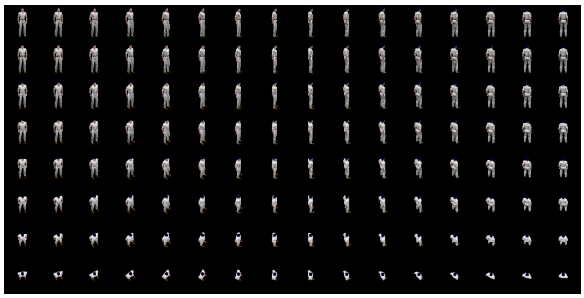

Figure 4: Impostor texture for a frame of animation as done in Tecchia and Chrysanthou [14].

Figure 5: Illustration of the rendering of impostors using a compressed texture. On the bottom right, texture after compression. The images are packed occupying only one fourth of the previous texture as shown in Fig. 4.

Second, all the images are of the same size. This results in a considerable waste of space, since for some images the human fits a restricted area. We start placing the samples on a texture using a regular grid as was done in [14]. Each sample is a prerendered ray-traced image of 256*256 pixels [1] of the character, using an orthographic projection. An example of the resulting image is shown in Fig. 4. In this way, extracting a particular sample is a trivial and fast operation, but there is a lot of unused texture space around each of the samples that gets wasted. To minimise the amount of such unused regions, during the pre-processing phase we compute the smallest rectangle containing the character for each sample. Then, we combine all these samples in a single image, reorganising them in order to minimise the unused space. Thanks to this process, the resulting new image is much smaller than the

[1] using 3D Studio Max

original without any loss, in terms of image quality. With our current reorganisation strategy, we can reduce the amount of texture memory used to store our human images to 25% of the original value. In our case, in order to have a good trade-off between the quality and the memory required for the samples, each frame of animation was stored using a single image of 512*512 pixel as a total size. An example of the resulting texture is shown in Fig. 5. The texture was then stored using the OpenGL compressed format ($GL\_COMPRESSED\_RGBA\_S3TC\_DXT3$) [17], which gives a further memory compression ratio of 1:4. This ratio is extremely efficient, although the image looses part of the quality. The compression format allows us to keep alpha values, and encodes them in 4 bits. Once loaded in texture memory, each frame of animation for a single human model requires 256 Kbytes.

Because we no longer have a regular grid, appropriate texture coordinates now need to be precomputed and stored for each sample. Then, at rendering time we need to compute on the fly the right size and orientation for the impostor to avoid the introduction of distortions of the sample image. It is important to notice that, because of our optimal samples placement strategy, these parameters generally vary for different frames of animation even considering a fixed point of view. An example of the mapping and the choice of the impostor is shown in Fig. 5.

5 Increasing the variety of avatars

With the texture compression described above we gain texture memory which we can use to simulate more humans than in [14]. To simulate 10 different types of human, with 10 frames of animation each, we need about 25MB of texture memory. Although we improved the possibility of variety, ten different avatars are not enough to populate a city. Because we were limited by the texture memory, we decided to modify the texture on the fly using multi-pass rendering. As we cannot change the shape and the kind of human, we changed the colour of significant parts of the body, like cloth, hairs, and skin colour.

To identify the areas to change, we pre-compute an alpha-channel image[2] with a different alpha value for each part to modify (see Fig. 6).Tuning the alpha channel we can define up to 256 different regions in the texture if no compression is used, or up to 16 if using the s3tc compression [17] since only 4 bits are available for the alpha channel. During the rendering, we use the alpha channel to select parts to be rendered while a multi-pass rendering. For each pass, the impostor polygon colour is changed to the desired colour and the texture is applied using the flag $GL\_MODULATE$ and setting the alpha threshold of the alpha test to the one associated with the part of interest.

Because we compute a texture modulation, the shading is preserved as it is already included into the texture. In our experiments, we draw up to 3 passes, thus changing only the colour of the shirt and the trousers for each individual. More passes can be done, as we said, since we can identify up to 16 regions. However, the multi-pass rendering might slow down the overall rendering rate, and a trade off between the variety and the rendering time must be done.

6 Implementation details and results

We implemented and tested the methods described above. For the simulation, we added some elements that allow better quality for the results.

In order to control the motion of the virtual humans, we subdivide the floor of the environment into *tiles* of regular size. While humans move around, they check information corresponding to the tile they occupy. Any information can be stored [16] but we use only this information for collision detection and shadowing. The tiles are subdivided so that an avatar can be at different positions into the same tile. If an avatar stays in the same tile in the next frame, it just continues to move in its current direction and no decision needs to be taken. In our system, the collision detection map is a binary map. When the tile is encoded by black, it is impassable and the avatar needs to change direction. We performed as well inter-collision detection between humans, by checking if a destination tile is already occupied. We also use a binary shadowing map, which encodes regions on the ground covered by the shadows of the buildings, to determine if humans are in shadows relatively to the buildings. When a human reaches a dark cell of the 2D representation of the positions, its impostor is darkened.

Figure 6: Example of avatars used. On the left hand, an avatar as rendered with ray tracing in 3D Studio Max. On the middle, an avatar with alpha-channels to identify parts to modify. On the right hand, an avatar rendered with multi-passes regarding to the alpha-channel. Notice how these four avatars look different although they are built from the same 3D model.

Using the impostor approach, we can also compute and display the shadows of the moving humans [7]. We use another polygon to be displayed on the floor with a shadow texture. The polygon is the projection of the human impostor on the floor level with regards to the light direction. When rendering, the shadows are displayed using the appropriate frame, which corresponds to the position of the human. Depending on the light position, the appropriate texture is applied and the projected polygon is scale consistently to the texture compression. To fake a shadow, the shadow impostor polygon is darkened so that the texture is modulated and darkened as well.

We develop the system on a PC Pentium III - 800Mhz with NVIDIA GeForce GTS2 video card. We populated our environment with 6 different avatars, and perform 3 passes to draw different colours, chosen randomly. For the results shown on Fig. 8 and Fig. 9, we displayed $2,000$ for each of four types of humans and $1,000$ for each of the two last type, thus displaying $10,000$ of different humans. One of these types is a jogger, thus having an animation different from the others. These humans move in a village modelled with $41,260$ polygons. The display is updated between 12 and 20 frames per second depending on the displayed polygonal complexity. Although the rendering is done in real time, there is no trade off made to decrease the quality. The rendering quality is as good as if no optimisation algorithms had been done. Some videos are shown in

[2] Using 3D Studio Max. The anti-aliasing must be turned off to avoid the borders of two adjacent regions to becoming blurred.

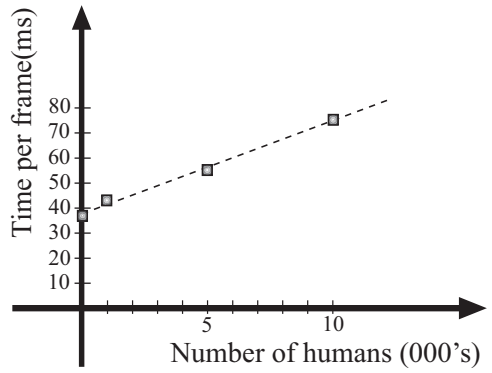

Figure 7: Number of humans against time per frame.

http://www.cs.ucl.ac.uk/staff/Y.Chrysanthou/crowds/CGA/.

To evaluate the scalability of our simulation, we tried to simulate $1,000$, $5,000$, and $10,000$ people. We run the simulation on a chosen camera path identical for each simulation.

Simulation	0 people	$1,000$ people	$5,000$ people	$10,000$ people
Frames per sec.	26.20	24.08	18.26	13.35

As can be seen, rendering the city model itself already uses a lot of the resources since the average display is of only 26 frames per seconds. In Fig. 7 we plot the same data but as time per frame vs. number of humans and we can see clearer that the relation is almost linear. We believe that an occlusion culling algorithm performed on the static model could help to accelerate the rendering. The frame rate then decreases as the number of polygons (those for the avatars) displayed increases. It is to be noticed that these timings include as well the collision detection performed for each of the virtual humans simulated. A visibility test as well as an occlusion culling algorithm applied to both the collision detection and the display of the humans could accelerate the frame rate.

7 Conclusion and future work

We have presented a system that allows real time rendering of densely populated large scaled environments. The method used allows real-time rendering of a high number of different animated people, using multi-layered animated impostors. The rendering speed is independent of the complexity of the avatar model, although rendering the same number of humans would be impossible if using polygonal models. We improved already existing methods by three main contributions. First the choice of the impostor is adapted to the object to render, thus minimising popping effects when changing view. Second the amount of texture memory has been reduced, allowing the load of a higher number of different kinds of people. Finally we presented a multi-pass algorithm, taking advantage of the alpha channel to select and colour different regions of the body, thus allowing a wide variety (the 10,000 people simulated in our experiment all look different). With the continuous increase of the texture memory available in common machines we predict that the use of such IBR approaches will provide solutions to crowd visualisation.

Although we have already achieved very good results, we believe that there is great scope for further improvement and developments. In addition in our implementation we made a number of assumptions that could be re-examined.

We assumed that the viewer will be at a certain distance away from the avatars. This allowed us to use impostors which are created with orthographic projection and limited texture dimensions. If we allow the user to get very close then we will notice the artefacts. One way round this is to use a hybrid approach where polygonal human models are used instead of impostors for the few avatars that come right up to the viewpoint.

Since we used the same textures for the generation of the avatar shadows we implicitly also made the assumption that the position of the light source is at infinity. This was not a limitation for our examples since the only source was the sun. However if we want to simulate the city by night, with street lights, then we would have to 'warp' the textures before applying them.

The impostors are currently shaded while rendered in 3D Studio Max. This effectively fixes the shading to the particular light defined at the time, making the sources static. However, we believe that we could shade on the fly the impostors using normal maps indicating the orientation of each pixel. This would allow for the shading to be consistent if the light source is modified.

A very interesting extension to our system, that could greatly improve it's impact, would be to use real photographical images of humans instead of synthetic models. Of course the problem here would be the acquisition of all the sample images. Possibly a way to do it is by scanning the person in colour and then taking the samples from the scan.

Methods to cull polygons could speed up rendering times, selecting for display both polygons from the static environments and the moving people [15].

The current algorithms could certainly be used to render different kinds of objects such as cars, pets, children, or groups of people (children holding the hand of an adult). Also several animations could be possible, with an appropriate load of the texture. Particular care should be taken for transitions in between animations. Moreover using the multi-pass rendering algorithm we could simulate simple animation such as turning the head to left or right.

Finally, for each of this new type of objects more work should be done on developing appropriate behaviour. Although a lot of work has been done on behaviour in cities, there are still a lot of problems to solve, especially for real time simulation of thousands of agents.

Acknowledgement

This work was in part supported by the EPSRC project GR/R01576/01 and the EPSRC Interdisciplinary Research Centre *equator*.

References

[1] A. Aubel, R. Boulic, and D. Thalmann. Animated impostors for real-time display of numerous virtual humans. In Jean-Claude Heudin, editor, *Proceedings of the 1st International Conference on Virtual Worlds (VW-98)*, volume 1434 of *LNAI*, pages 14–28, Berlin, July 1–3 1998. Springer.

[2] A. Aubel, R. Boulic, and D. Thalmann. Lowering the cost of virtual human rendering with structured animated impostors. In *Proceedings of WSCG 99*, Plzen, Czech Republic, 1999.

[3] Shenchang Eric Chen and Lance Williams. View interpolation for image synthesis. In James T. Kajiya, editor, *Computer Graphics (SIGGRAPH '93 Proceedings)*, volume 27, pages 279–288, August 1993.

[4] William J. Dally, Leonard McMillan, Gary Bishop, and Henry Fuchs. The delta tree: An object-centered approach to image-based rendering. Technical Memo AIM-1604, Massachusetts Institute of Technology, Artificial Intelligence Laboratory, May 1996.

Figure 8: The crowd visualisation. Notice the number of different people. Using the optimisation techniques presented in the paper, we visualise thousands of different humans in real time.

Figure 9: A view closer to the humans.

[5] Lucia Darsa, Bruno Costa Silva, and Amitabh Varshney. Navigating static environments using image-space simplification and morphing. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 25–34. ACM SIGGRAPH, April 1997. ISBN 0-89791-884-3.

[6] Paul E. Debevec, Camillo J. Taylor, and Jitendra Malik. Modeling and rendering architecture from photographs: A hybrid geometry- and image-based approach. In *ACM SIGGRAPH 96 Conference Proceedings*, pages 11–20, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[7] C. Loscos, F. Tecchia, and Y. Chrysanthou. Real time shadows for animated crowds in virtual cities. In *ACM Symposium on Virtual Reality Software and Technology*, pages 85–92, November 2001.

[8] Paulo W. C. Maciel and Peter Shirley. Visual navigation of large environments using textured clusters. In Pat Hanrahan and Jim Winget, editors, *ACM Computer Graphics (Symp. on Interactive 3D Graphics)*, pages 95–102. ACM SIGGRAPH, April 1995. ISBN 0-89791-736-7.

[9] William R. Mark, Leonard McMillan, and Gary Bishop. Post-rendering 3D warping. In Michael Cohen and David Zeltzer, editors, *1997 Symposium on Interactive 3D Graphics*, pages 7–16. ACM SIGGRAPH, April 1997. ISBN 0-89791-884-3.

[10] Leonard McMillan and Gary Bishop. Plenoptic modeling: An image-based rendering system. *Computer Graphics*, 29(Annual Conference Series):39–46, November 1995.

[11] G. Schaufler. Per-object image warping with layered impostors. In *9th Eurographics Workshop on Rendering '98*, pages 145–156, Vienna, Austria, April 1998. EUROGRAPHICS. ISBN 0-89791-884-3.

[12] Germot Schaufler and Wolfgang Sturzlinger. A three-dimensional image cache for virtual reality. *Computer Graphics Forum*, 15(3):C227–C235, C471–C472, September 1996.

[13] Jonathan Shade, Dani Lischinski, David Salesin, Tony DeRose, and John Snyder. Hierarchical image caching for accelerated walkthroughs of complex environments. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 75–82. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[14] F. Tecchia and Y. Chrysanthou. *Real-Time Rendering of Densely Populated Urban Environments*, pages 83–88. Springer Computer Science, 2000. Rendering Techniques 2000.

[15] F. Tecchia, C. Loscos, and Y. Chrysanthou. Real time rendering of populated urban environments. In *ACM Siggraph Technical Sketch*, August 2001.

[16] F. Tecchia, C. Loscos, R. Conroy, and Y. Chrysanthou. Agent behaviour simulator (abs): A platform for urban behaviour development. In *GTEC'2001*, January 2001.

[17] OpenGL texture compression. *http://oss.sgi.com/projects/ogl-sample/registry/EXT/texture_compression_s3tc.txt*.