Using Image Morphing for Memory-Efficient Impostor Rendering on GPU

Kamer Ali Yuksel[1], Aytul Ercil[4]
Computer Vision and Pattern Analysis Laboratory
Faculty of Engineering and Natural Sciences
Sabanci University, Istanbul, Turkey
{kamer, aytulercil}@sabanciuniv.edu

Alp Yucebilgin[2], Selim Balcisoy[3]
Computer Graphics Laboratory
Faculty of Engineering and Natural Sciences
Sabanci University, Istanbul, Turkey
{ayucebilgin, balcisoy}@sabanciuniv.edu

Abstract— **Real-time rendering of large animated crowds consisting thousands of virtual humans is important for several applications including simulations, games and interactive walkthroughs; but cannot be performed using complex polygonal models at interactive frame rates. For that reason, several methods using large numbers of pre-computed image-based representations, which are called as impostors, have been proposed. These methods take the advantage of existing programmable graphics hardware to compensate the computational expense while maintaining the visual fidelity. Making the number of different virtual humans, which can be rendered in real-time, not restricted anymore by the required computational power but by the texture memory consumed for the variety and discretization of their animations. In this work, we proposed an alternative method that reduces the memory consumption by generating compelling intermediate textures using image-morphing techniques. In order to demonstrate the preserved perceptual quality of animations, where half of the key-frames were rendered using the proposed methodology, we have implemented the system using the graphical processing unit and obtained promising results at interactive frame rates.**

Keywords-real-time rendering; crowd simulation; impostor rendering; image morphing; graphical processing unit

I. INTRODUCTION

Large computer-generated crowds have become a common feature especially for movies and games. However, rendering thousands of different characters is still challenging in real-time at interactive frame rates concerning that the crowd should be heterogeneous and individuals should appear to be visually realistic and convincingly animated. Recent advances in graphics hardware have presented new opportunities to increase the size of real-time rendered crowds without decreasing their details.

Many researchers attempted to minimize the geometrical complexity of each character using a single adaptive quad, where the appropriate texture is selected depending on the viewpoint positing and the frame of animation. In this technique, a discrete set that based on possible views for a character is pre-rendered and stored in memory and the nearest view from the set is used to display the character on a quadrilateral dynamically oriented towards the viewer [1, 2]. This technique, which is often called as impostor rendering, is suitable for performing on graphic processing unit (GPU), as it requires only a single quad for each character of which animations and transformations using simple texture lookups. Using this technique on GPU, Millan and Rudomin [3] have been successfully rendered 65,536 characters at interactive frame rates of 30 fps. However, even the most simplistic behavioral simulation of the displayed characters already restricts this number to 10,000-30,000 at similar frame rates [4, 5].

Figure 1. Screenshot from the GPU implementation showing the overall quality of the morphed intermediate animation frames

Texture-based rendering techniques also became crucial for mobile games, where much less number of characters needs to be rendered, due to the emergence of portable gaming market driven by Smart-phones having slightly limited GPU capacities than desktop platforms. For instance, Apple's iPhone 3GS equipped with PowerVR SGX at 200Mhz that is able to render 7 million triangles per second, while having 256MB of unified memory that can be used with a maximum texture size of 2048x2048 pixels. Using polygonal rendering techniques, at most 100 reasonably detailed characters can be rendered considering that the number of triangle required for each of them is around 1 thousand for mobile games.

However, multiple frames of their animations should be stored for each viewpoint, which requires huge amount of texture memory while rendering 3-dimensional characters using texture-based methods. Accordingly, the required memory for each different character is excessive and the contemporary memory amount supported by GPUs is not sufficient for having different detailed textures for each discretization of different characters. Thallman et. al. [6] states the appearance and animation variety of characters as major challenges of the crowd simulation. The memory constraint generally causes using one type of avatar and diversifying its appearance for each character, often by assigning random colors to its different parts, or increasing the degree of discretization. However, higher degree of discretization also suffers from the popping artifacts while changing viewpoints or animation frames. Generating

intermediate frames of animations would artificially increase the degree of discretization and reduce popping effects. Also, it would improve the realism of existing texture-based characters since modern display devices are operating at a much higher refresh rate than earlier hardware. Consequently, we propose using some of the processing power dedicated for rendering for procedurally generating intermediate textures through image morphing algorithms [7, 8].

This paper describes an impostor rendering method, which uses image morphing techniques to reduce the memory consumption while preserving the perceptual quality, allowing higher diversity or resolution of the rendered crowds. Using this technique, we have cut the number of intermediate frames by half and re-generated them using image morphing during the rendering phase to save an extra space from the texture memory of GPU, which can later be employed for increasing the appearance quality or diversity of characters displayed at each scene, allowing the addition of several kinds of avatars (Fig. 1). According to Aubel et. al. [9], frame-to-frame changes in the animation are small while rendering complex, moving figures (e.g. virtual humans) and human perception tends to reconstruct the missing frames when sufficient amount of frames are being shown; thus, few key postures are often sufficient to perceive an entire animation. After a set of perceptual experiments, Hamill et. al, [10] found that impostors are an extremely effective substitute for detailed geometry on large-scale simulations involving complex scenes and they able to equivalently replicate the motion of associated geometric models. The paper is organized as follows. In section 2, we have presented works related to the proposed method. Section 3 describes the techniques in this research to reduce the memory consumption while preserving the perceptual quality of the impostor rendering. Finally, section 4 describes the evaluation of this technique and section 5 describes some conclusions and future works.

II. Related Works

Several attempts previously made to increase the appearance quality and diversity of Impostors during real-time rendering. Most of them also attacked the problem of the excessive memory consumption by removing wasted rectangular regions between poses, symmetric movements within animation frames (key-frames), etc. Firstly, Tecchia et. al. [11] combined texture compression with a multi-layered impostor rendering technique to increase number of different animated virtual humans by taking advantage of the alpha channel to select and color different regions of the body. Dobbyn et. al. [12] improved their algorithm by constructing final textures for characters through blending a set of image maps produced by normal maps, detail maps, and a set of customizable color materials to achieve visually realistic simulations. Furthermore, Kavan et. al. [13] generated intermediate images by automatically constructing 2D polygonal characters (referred as Polypostors) for a given direction from a segmented 3D character. Their algorithm based on dynamic programming shifts the vertices of the 2D polygons to approximate the actual rendered image. Whereas, Lister et. al. [14] adopted a caching system, which enables skinned key-poses to be re-used by multi-pass rendering, between multiple agents and across multiple frames. By this way, they have exploited the temporal and intra-crowd coherencies that are inherent within a populated scene and stored best set of skinned key-poses that can be shared amongst crowd members.

Besides, researchers have also addressed the interpolation of image sequences. Stich et. al. [15] presented an approach to that is motivated by insights of human perception by focusing on the properties important to visual motion perception and validated their approach using a user study. They have utilized an optical flow-based warping refinement method and an adaptive non-linear image-blending scheme to guarantee perceptional plausibility of the interpolated intermediate images. Finally, they have demonstrated how to continuously navigate the viewpoint between camera positions and shutter release times and how to animate still pictures and create smooth camera motion paths. More similar works have been also done previously in different contexts. Firstly, Zamith et. al. [16] have implemented the feature-based image metamorphosis algorithm in parallel on GPU using the Compute Unified Device Architecture (CUDA) to animate the appearance of a 3D character's face by morphing its texture map and shown that their method is promising and scalable on the number of features. Moreover, Wong et. al. [17] described an error measure based on epipolar geometry for morphing between images of different view points. Also, Gao et. al. [18] automated the feature specification process if the input images are sufficiently similar by assigning cost functions to the bilinear B-spline warp and pixel intensity recoloring and using the warp that has the global minimum sum of cost as the solution. Finally, Karungaru et. al. [19] detected optimum control points to automatically perform feature-based image metamorphosis using genetic algorithms and neural network for recognizing facial features.

III. Methodology

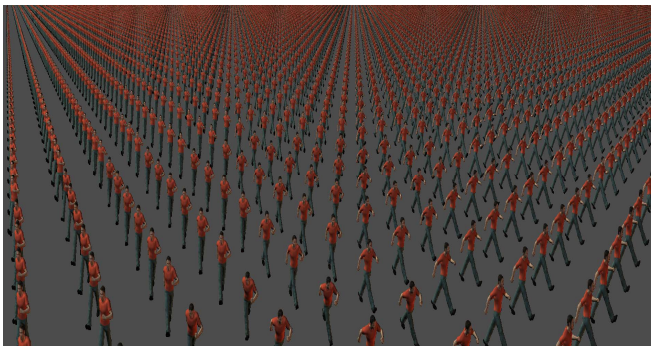

Figure 2. Densely populated scene of 16,384 characters that are rendered at 25 fps avg. by the application implemented using OpenGL and GLSL.

The proposed method contributes several additional sub-steps to two phases, pre-processing and rendering, of conventional impostor rendering techniques. Thus, it can be also thought as a complementary for existing impostor rendering techniques. Hence, it may be used together with

most of previously proposed techniques for reducing the memory consumption and increasing the crowd diversity, such as removal of unused regions and symmetric key-frames, and post-processing, e.g. colorization of characters. In order to evaluate the efficiency of the proposed method, we have implemented a reference application using OpenGL and GLSL (Fig. 2) and an additional interface in MATLAB. The architecture, which is designed to implement the proposed methodology, consists of offline and real-time processing phases having various sub-components on both CPU and GPU (Fig. 3). During the offline process, the impostor texture map is generated and features, which are necessary for the feature-based image-morphing algorithm, are manually selected through the MATLAB interface and then encoded to the texture map. During the online process, the reference application for crowd simulation processes user interaction and behavioral simulation on CPU. Then, it renders updated characters through the shader pipeline on GPU, considering encoded features and updated settings.

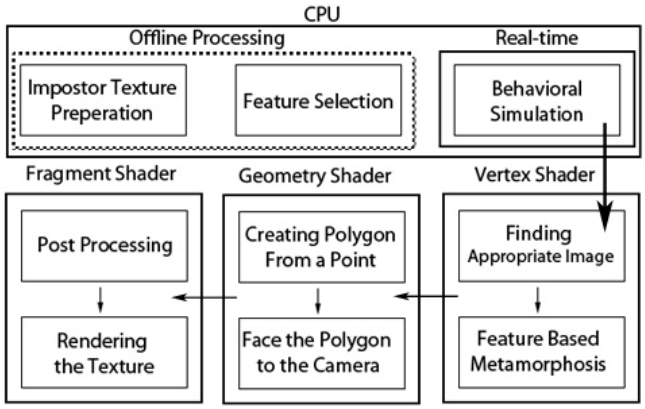

Figure 3. The overview of the architecture used to implement the proposed methodology describing the hiearchy between various sub-steps

After behavioral simulation, the position, heading and current key-frame of each character is continuously transferred to GPU along with the viewing frustum of camera. The vertex shader checks if the most appropriate frame is directly available and generate it using the morphing algorithm if necessary. The geometry shader generates the quad primitive according to the relative position of the character and adjusts its orientation to face the camera. Finally, the fragment shader applies optional post-processing effects, such as Gaussian blur to reduce possible artifacts of the morphing algorithm, and renders the final character by mapping the sub-texture to the impostor quad discarding invisible pixels via alpha-testing.

A. Generating Impostor Textures

In order to produce impostor textures, we have pre-rendered a discrete set of (16x4) viewpoints, which are uniformly distributed over the surface of a bounding hemisphere (Fig. 4), using an orthographic projection for each key-frame and appended them into an overall texture. Using 3D Studio Max, we have automatically captured ray-traced images of each viewpoint by rotating the camera around horizontal and vertical axes. Each capture consisted of 256x256 pixels and there were 16 consecutive key-frames that resulted in 8192x8192x4 bytes to be consumed for a specific avatar (Fig. 5), which is almost the maximum memory supported by most of contemporary GPUs. In order to add additional characters or animations, one should either reduce the resolution of each view or the number of discretized poses, which would decrease the visual or perceptual quality of the characters.

Figure 4. Discrete set of captured 16 by 4 view-points which are uniformly distributed over the surface of a bounding hemisphere.

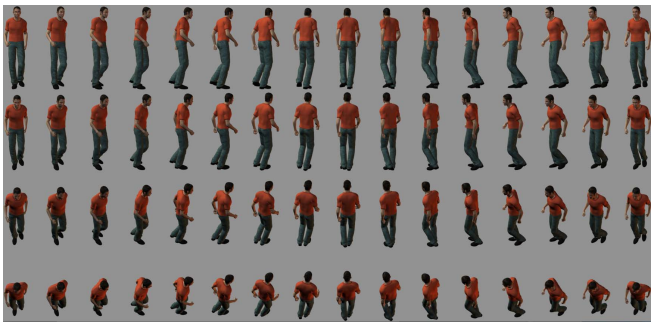

Figure 5. The partial view of the final impostor texture map including images from different view-points for only one animation frame.

B. Selecting and Encoding Features

After generating impostor textures, we have manually selected line pairs as control points, which are required by the image-morphing algorithm that we have employed (Fig. 6). In order to use the pixel buffer effectively, selected features are encoded as color components of pixels within each sub-texture where the alpha channel is equal to zero. Control points have been selected manually using an external interface in MATLAB, where the resulting intermediate images and their correlation coefficients with original images are presented after each update. In order to fasten the selection process, hints of features, which are be detected using Scale-invariant feature transform (SIFT) algorithm [20], can be displayed on both images and users may be asked to select optimum features that would lead to best morphed image.

The visual quality of the generated intermediate image is mainly dependent on the proper selection and the number of

line pairs, concerning the similarity of the morphed images (Table 2). Although the selection of line pairs has been done manually, it would be possible to automatically optimize those features using the quantitative measures and techniques mentioned in related-works section. Using more control points would often lead to a better result but it would reduce the performance of the proposed method by increasing the complexity of the morphing algorithm (Table 1). Hence, we have decided to use only 8 line pairs per image after couple of preliminary experiments (Fig. 6). Decisions of optimum features for key-frames were slightly easier because pairwise control points sometimes were not available on rotated images. Thus, selecting 8 line pairs for 8 key-frames on 16x4 viewpoints have approximately taken two hours. Those experiments also shown that perceivable artifact occurs only on faces of characters especially during the frontal views, due to the limited number of control points and the naturally developed facial feature detection skills of humans (Fig. 9). For that reason, it might be advisable to use some of the extra memory for faces of intermediate textures and paste them on top of procedurally generated images. This would enable increasing the crowd diversity using faces of different characters while it would still be resource-efficient, as faces would cover only a minor area of the texture, in comparison with their bodies.

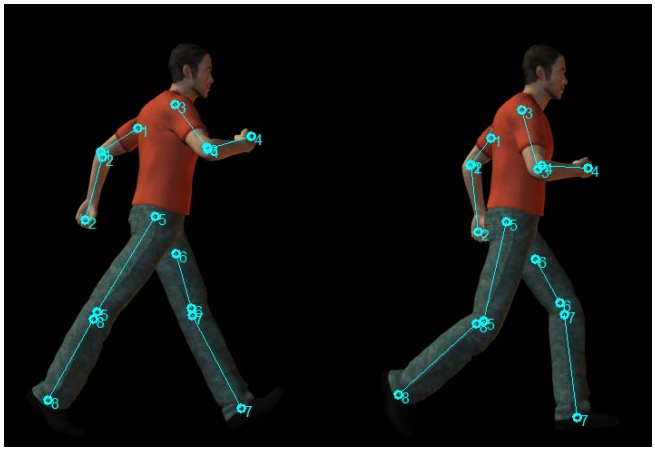

Figure 6. Selection of line pairs (control points) of two consecutive animation frames for feature-based image metamorphosis.

C. *Rendering Characters using Impostors*

The position and heading information of each character is continuously transferred to the graphics memory using a pixel buffer before the rendering of each frame. In order to create the illusion of 3-dimensionality using 2-dimensional representations, each character is rendered to a textured quad facing continuously towards the camera of which texture coordinates changed according to the closest available view in texture map when the animation frame or the heading, which is relative to the current viewing position of the camera, of any character is updated. Firstly, the viewing frustum from the current view is obtained by the vertex shader and used to calculate texture coordinates to obtain the most suitable image for the current view and animation pose from the discrete set of pre-rendered images. Then, the geometry shader is employed to generate quads, which are then translated to specified positions, from point primitives received from the graphics pipeline. Then, the most approximate image to the corresponding key-frame and viewpoint is found or synthesized, and finally passed to the fragment shader for post-processing and rendering.

D. *Generating Intermediate Animations*

In order to cut the total amount of texture memory consumed for each animation frame into half, we have re-generated intermediate frames using image morphing techniques. Differences in viewpoint of the rendered characters might have caused unnatural distortions while generating intermediate textures through morphing. For high quality and perceptually convincing transitions between existing poses, we have first employed View Morphing algorithm [2] to describe the scene appearance for a more continuous range of viewpoints using two basis views of a static scene. This technique works by pre-warping two images prior to computing a morph and then post-warping the interpolated images. This allows determining the set of all views on the line between the optical centers of these two basis views; thus, rectifying them and then interpolating corresponding pixels can synthesize the intermediate perspective. View morphing is suitable for impostor rendering due its monotonicity property, which means that the camera configurations are known and the fundamental matrix is available.

Figure 7. Intermediate keyframe generated by the proposed method (Left) and original image (Right) of the same animation frame and viewpoint.

After view-morphing, we have employed Feature-based Image Metamorphosis algorithm [3] where a set of pairs of corresponding features previously specified as line segments, in order to reconstruct after a combination of weighted local affine transformations, instead of direct linear interpolation of pixels. Local transformations of each segment feature are blended based on an inverse distance weighted interpolation to reconstruct the intermediate image. Hence, the synthesized image represents a view from a viewpoint or key-frame halfway between the two

consecutive frames similar to the expected intermediate frame. Upon the request of existing key-frames, the method acts totally same with the conventional impostor rendering. In this method, intermediate frames of animations are actually removed from the impostor texture and generated using image morphing. When they are requested, feature-based image metamorphosis algorithm is applied using the existing previous and next frames of the animation containing coordinates of line pairs in their transparent pixels (Fig. 6). The previous and next key-frames are warped using the computed transformations and blended to produce the intermediate key-frame (Fig. 7).

E. Generating Intermediate Viewpoints

Figure 8. Examples of consecutive viewpoints that are utilized for generating their intermediate viewpoint using the proposed method.

Figure 9. Intermediate viewpoint generated using same number of (eight) features focusing on body parts (Left) and the face (Right) of the character.

Finally, we have attempted to generate intermediate viewpoints using the proposed method. The view-morphing algorithm almost successfully produces the pre-warped images where the perspective of the consecutive viewpoints is matched to the target intermediate viewpoint (Fig. 8). However, manual selection of optimal line pairs at consecutive viewpoints is more difficult because some features may not be visible in both images due to the rotation of the character. Furthermore, the distribution of limited number of features over different body parts affects their quality on the resulting image. The facial area of the character requires often more consideration as it is slightly more inconsistent between consecutive viewpoints than consecutive key-frames (Fig. 9). Moreover, the artifacts of generated intermediate viewpoints are more perceivable than intermediate key-frames because users do not change their viewpoints as much as animations, which continuously cycle key-frames. Hence, the perceptual quality of the animation is preserved in contrast, as the morphed key-frames are only shown between the original key-frames (Fig. 8). Consequently, we have not employed the generation of intermediate viewpoints while evaluating the proposed method.

IV. EVALUATION AND DISCUSSION

The proposed technique was evaluated on an Intel Core2 Quad Q6600 PC with 2GB RAM using a GeForce GTX 460 graphics card with 1 GB of memory. Different number of characters and features, which were selected as line pairs, was employed to evaluate the performance of the reference application of the proposed methodology. Real-time rendering performances of various combinations are presented in Table 1, using the average frame rate as an evaluation metric. Finally, we have employed statistical metrics to measure the similarity between original and morphed intermediate (Int) images using different number of features. Then, we compared them with the similarity of consecutive (Cons) images, in order to quantitatively evaluate the visual quality of the generated images. The mean and standard deviation (Stdev) of cross-correlations between those set of target images are shown in Table 2 below.

Figure 10. Impostors from different texture maps of equal number of key-frames and memory consumption. The proposed (right) has higher resolution due to re-generated animation frames via image morphing.

The overall graphics memory is shared between different characters that need to be rendered in each scene. In order to fit each character into the limited memory, one should decrease either their resolution or key-frames. Charging off the resolution of impostors would decrease the quality of their appearance (Fig. 10). Similarly, reducing the number of key-frames would decrease the smoothness and reality of

the animated characters. Instead, the proposed method generates intermediate frames by morphing them in run-time. Achieved quality of the resulting images, which are morphed in real-time using limited amount of features, are promising considering that impostor rendering is often used as a level of detail technique.

TABLE I. REAL-TIME RENDERING PERFORMANCE STATISTICS

Number of Characters/Features vs. Avg. FPS		
# of Characters	***# of Features***	***Avg. FPS***
8,192	4	59.2
8,192	8	34.5
8,192	12	19.8
16,384	4	43.4
16,384	8	25.7
16,384	12	14.9

TABLE II. IMAGE SIMILARITY STATISTICS

Target Images (# of Features)	**Cross-Correlation**	
	Mean	***Stdev***
Cons. key-frames	0.56	0.16
Cons. view-points	0.47	0.12
Int. key-frames (8)	0.85	0.10
Int. viewpoints (8)	0.78	0.06
Int. key-frames (12)	0.92	0.07
Int. viewpoints (12)	0.83	0.05

V. CONCLUSION AND FUTURE WORKS

In this work, we have proposed a resource-efficient impostor rendering methodology, which employs image morphing techniques to reduce the memory consumption while preserving the perceptual quality, allowing higher diversity or resolution of the rendered crowds. Then, we have demonstrated the proposed method by implementing a reference application of crowd simulation using GPU. The proposed method is evaluated visually by replacing only the intermadiate frames of animations with procedurally generated ones. However, it is applicable also for generating intermediate viewpoints, which was not evaluated due to the relatively higher difficulty of selecting their optimum features manually. Promising results obtained for the animation case insights that it would be able to generate them as well once automatic feature specification techniques are implemented. Finally, we would also like to try the proposed method for further increasing the diversity of the crowd by changing body masses of individuals, as a future work. Please note that, the supplementary video material of the proposed method presenting the reference application can be accessed from: http://goo.gl/1sdWP

REFERENCES

[1] G. Schaufler and W. Sturzlinger, "A Three-Dimensional Image Cache for Virtual Reality," Computer Graphics Forum, vol. 15, no. 3, Sept. 1996, pp. C227-C235.

[2] J. Shade, D. Lischiniski, D. Salesin, T. DeRose, and J. Snyder, "Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments," Computer Graphics Proc. Ann. Conf. Series (Proc. Siggraph '96), pp. 75-82, Aug. 1996.

[3] E. Millan and I. Rudomin, “Impostors and pseudo-instancing for GPU crowd rendering,” in Proceedings of the 4th international conference on Computer graphics and interactive techniques in Australasia and Southeast Asia, 2006, p. 49–55.

[4] C. Reynolds, “Big fast crowds on ps3,” in *Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames*, 2006, p. 113–121.

[5] A. Treuille, S. Cooper, and Z. Popović, “Continuum crowds,” in *ACM SIGGRAPH 2006 Papers*, 2006, p. 1160–1168.

[6] D. Thalmann, H. Grillon, J. Maim, and B. Yersin, “Challenges in Crowd Simulation,” in International Conference on CyberWorlds 2009. CW’09. 2009, p. 1–12.

[7] S. M. Seitz and C. R. Dyer, “View morphing,” in Proceedings of the 23rd annual conference on Computer graphics and interactive techniques, 1996, p. 21–30.

[8] T. Beier and S. Neely, “Feature-based image metamorphosis,” ACM SIGGRAPH Computer Graphics, vol. 26, no. 2, p. 35–42, 1992.

[9] A. Aubel, R. Boulic, and D. Thalmann, “Real-time display of virtual humans: Levels of details and impostors,” Circuits and Systems for Video Technology, IEEE Transactions on, vol. 10, no. 2, p. 207–217, 2000.

[10] J. Hamill, R. McDonnell, S. Dobbyn, and C. O’Sullivan, “Perceptual evaluation of impostor representations for virtual humans and buildings,” in *Computer Graphics Forum*, 2005, vol. 24, p. 623–633.

[11] F. Tecchia, C. Loscos, and Y. Chrysanthou, “Visualizing Crowds in Real-Time,” in Computer Graphics Forum, 2002, vol. 21, p. 753–765.

[12] S. Dobbyn, J. Hamill, K. O’Conor, and C. O’Sullivan, “Geopostors: a real-time geometry/impostor crowd rendering system,” in Proceedings of the 2005 symposium on Interactive 3D graphics and games, 2005, p. 95–102.

[13] L. Kavan, S. Dobbyn, S. Collins, J. Žára, and C. O’Sullivan, “Polypostors: 2D polygonal impostors for 3D crowds,” in Proceedings of the 2008 symposium on Interactive 3D graphics and games, 2008, p. 149–155.

[14] W. Lister, R. G. Laycock, and A. M. Day, “A Dynamic Cache for Real-Time Crowd Rendering,” in Computer Graphics Forum, 2010.

[15] T. Stich, C. Linz, C. Wallraven, D. Cunningham, and M. Magnor, “Perception-motivated interpolation of image sequences,” ACM Transactions on Applied Perception (TAP), vol. 8, no. 2, p. 11, 2011.

[16] M. Zamith et al., “Real Time Feature-Based Parallel Morphing in GPU Applied to Texture-Based Animation,” in Systems, Signals and Image Processing, 2009. IWSSIP 2009. 16th International Conference on, p. 1–4.

[17] T. Y. Wong, P. Kovesi, and A. Datta, “Towards quantitative measures of image morphing quality,” in Digital Image Computing: Techniques and Applications, 2005. DICTA’05. Proceedings 2005, p. 19–19.

[18] P. Gao and T. W. Sederberg, “A work minimization approach to image morphing,” The Visual Computer, vol. 14, no. 8, p. 390–400, 1998.

[19] S. Karungaru, M. Fukumi, N. Akamatsu, and A. Takuya, “Automatic human faces morphing using genetic algorithms based control points selection,” International Journal of Innovative Computing, Information and Control, vol. 3, no. 2, p. 1–6, 2007.

[20] D. G. Lowe, “Object recognition from local scale-invariant features,” in *iccv*, 1999, p. 1150.