Vis Comput (2013) 29:131–140
DOI 10.1007/s00371-012-0718-8

ORIGINAL ARTICLE

Real-time feature-based image morphing for memory-efficient impostor rendering and animation on GPU

Kamer Ali Yuksel · Alp Yucebilgin · Selim Balcisoy · Aytul Ercil

Published online: 9 May 2012

Abstract Real-time rendering of large animated crowds consisting of 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, methods using large numbers of precomputed image-based representations, called impostors, have been proposed. These methods take advantage of existing programmable graphics hardware to compensate for computational expense while maintaining visual fidelity. Thanks to these methods, the number of different virtual humans rendered in real time is no longer restricted by computational power but by texture memory consumed for the variety and discretization of their animations. This work proposes a resource-efficient impostor rendering methodology that employs image morphing techniques to reduce memory consumption while preserving perceptual quality, thus allowing higher diversity or resolution of the rendered crowds. Results of the experiments indicated that the proposed method, in comparison with conventional impostor rendering techniques, can obtain 38 % smoother animations or 87 % better appearance quality by reducing the number of key-frames required for preserving the animation quality via resynthesizing them with up to 92 % similarity on real time.

Keywords Crowd simulation · Impostor rendering · Image and view morphing · Graphical processing unit

This work has been partially supported by TUBITAK under project 109E134, VIPSAFE.

Electronic supplementary material The online version of this article (doi:10.1007/s00371-012-0718-8) contains supplementary material, which is available to authorized users.

K.A. Yuksel (✉) · S. Balcisoy · A. Ercil
Faculty of Engineering and Natural Sciences, Sabanci University, Istanbul, Turkey
e-mail: kamer@sabanciuniv.edu

S. Balcisoy
e-mail: balcisoy@sabanciuniv.edu

A. Ercil
e-mail: aytulercil@sabanciuniv.edu

A. Yucebilgin
Four Eyes Lab, Department of Computer Science, University of California, Santa Barbara, USA
e-mail: alp@cs.ucsb.edu

1 Introduction

Large computer-generated crowds have become a common feature for movies and games. However, rendering thousands of different characters is still challenging in real time at interactive frame rates considering 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 have attempted to minimize the geometrical complexity of each character using a single adaptive quad, where the appropriate texture is selected depending on viewpoint and animation frame. Impostor rendering is a technique in which a discrete set that is based on possible views of a character is pre-rendered and stored in memory; the nearest view from the set is used to display the character on a quadrilateral dynamically oriented toward the viewer [1, 2].

This technique is suitable for performing on a graphic processing unit (GPU), as it requires only a single quad for each character and simple texture lookups for their animations and transformations. Using impostor rendering

on GPU, Millan and Rudomin [3] have successfully rendered 65,536 characters at interactive frame rates of 30 fps. However, even the simplest behavioral simulation of the displayed characters restricts the number of characters to 10,000–30,000 at similar frame rates [4, 5].

The emergence of portable gaming market has made the texture-based rendering crucial because of their limited GPU capabilities. For instance, Apple's iPhone 3GS is equipped with PowerVR SGX at 200 MHz, rendering 7 million triangles per second while its 256 MB unified memory is able to support a maximum texture size of 2048 × 2048 pixels. Considering that the number of triangles required for each character is around one-thousand for mobile games, a maximum of hundred reasonably detailed characters can be rendered using polygonal rendering techniques on this GPU.

Furthermore, using texture-based methods to render three-dimensional characters is also limiting due to the capacity of texture memory. Since multiple frames of animations are stored for each viewpoint, the required memory for a single character is excessive. Hence, the contemporary memory supported by GPUs is insufficient for rendering detailed textures for each different characters. Thalmann et al. [6] state that the appearance and animation variety of characters are major challenges of the crowd simulation. The memory constraint generally forces the designers to use one basic avatar for all characters which is then diversified by assigning random colors to its various parts or by increasing the degree of discretization. However, this higher degree of discretization also suffers from popping artifacts while changing viewpoints or animation frames.

According to Aubel et al. [9], frame-to-frame changes in animations are small even when rendering complex, moving figures (e.g., virtual humans). The researchers stipulate that a few key postures are often sufficient to perceive an entire animation since human perception tends to reconstruct the missing frames when a sufficient amount of frames is shown [9]. 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 as they can exactly replicate the motion of associated geometric models.

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 operate at a much higher refresh rate than earlier hardware. Consequently, we propose reallocating some of the processing power dedicated for rendering to synthesizing intermediate textures procedurally through image morphing algorithms [7, 8].

In this paper, we describe an impostor rendering method which uses image morphing techniques to reduce memory consumption while preserving perceptual quality, thus

Fig. 1 Screenshot from a large-scale crowd simulation scene using the proposed method for impostor rendering and animation

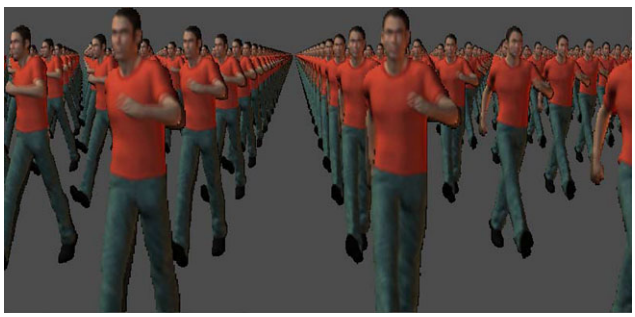

Fig. 2 Screenshot from the real-time rendering testbed showing the improved overall quality of the morphed intermediate animation frames

increasing diversity or resolution of the rendered crowds (Fig. 1). Using this technique, we cut the number of intermediate frames by half and regenerated them using image morphing techniques during the rendering phase, in order to save space from the texture memory of GPU. This extra space could then be employed for increasing the appearance quality or diversity of characters displayed at each scene, allowing the addition of several kinds of avatars (Fig. 2).

Consequently, we implemented a reference application using the graphical processing unit and evaluated the perceptual quality of the proposed method both visually and statistically. Experiments indicate that using the same amount of texture memory with state-of-art impostor rendering techniques, the proposed method obtains 38 % smoother animations or 87 % increased appearance quality. Either contribution is possible by reducing the number of keyframes required for preserving the animation quality by resynthesizing them on real time while achieving up to 92 % cross-correlation with the original image.

This paper, an extended version of an earlier publication [11], is organized as follows. Section 2 presents studies and technologies related to the proposed method. Section 3 describes in detail the different phases of the proposed methodology. Section 4 examines utilized image morphing techniques and algorithms both in theory and practice. Section 5 describes the application of image morphing to impos-

tor rendering for reducing the memory consumption. Section 6 offers a comprehensive evaluation of the proposed method via quantitative and qualitative experiments. Section 7 discusses the results of experiments that are described in the previous section. The last sections draw conclusions and offer suggestions for future works.

2 Related works

Several attempts have been made to increase the appearance quality and diversity of impostors during real-time rendering. Most of them suggested solving the problem of excessive memory consumption by removing wasted rectangular regions between poses and symmetric movements within animation frames (key-frames).

Firstly, Tecchia et al. [12, 13] combined texture compression with a multilayered 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. Then, Dobbyn et al. [14] 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. [15] 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. [16] adopted a caching system between multiple agents and across multiple frames, which enables skinned key-poses to be reused by multipass rendering. In this way, they exploited the temporal and intra-crowd coherencies that are inherent within a populated scene and stored the best set of skinned key-poses that can be shared amongst crowd members.

Researchers have also addressed the interpolation of image sequences. Motivated by insights of human perception, Stich et al. [17] presented an approach focusing on the properties important to visual motion perception and validated their approach using a user study. They utilized an optical flow-based warping refinement method and an adaptive nonlinear image-blending scheme to guarantee perceptional plausibility of the interpolated intermediate images. Lastly, they 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.

Meanwhile, Zamith et al. [18] 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. They demonstrated that their method is promising and scalable on the number of features. Moreover, Gao et al. [20] automated the feature specification process when 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. Also, Wong et al. [19] described an error measure based on epipolar geometry for morphing between images of different viewpoints. Finally, Karungaru et al. [21] detected optimum control points to automatically perform feature-based image metamorphosis using genetic algorithms and neural network for recognizing facial features.

3 Methodology

The proposed method complements conventional impostor rendering techniques by adding several substeps to preprocessing and rendering phases. Hence, it may be combined with many of previously proposed techniques, such as removal of unused regions and symmetric key-frames, and postprocessing, e.g., colorization of characters, in order to reduce memory consumption and increase crowd diversity.

In order to evaluate the efficiency of the proposed method, we have implemented a reference application using OpenGL and GLSL (Fig. 3) 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 subcomponents on both CPU and GPU (Fig. 4).

During the offline process, the impostor texture map is generated, and features that 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 processes user interaction and behavioral simulation on CPU. Then, the application renders updated characters through the shader pipeline on GPU, based on the encoded features and updated settings.

After behavioral simulation, the position, heading, and current key-frame of each character are continuously transferred to GPU along with the viewing frustum of camera. The vertex shader checks if the most appropriate frame is directly available and generates 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 postprocessing effects, such as Gaussian blur to reduce possible artifacts of the morphing algorithm, and renders the final character by discarding invisible pixels on the subtexture via alpha-testing and mapping it to the impostor quad.

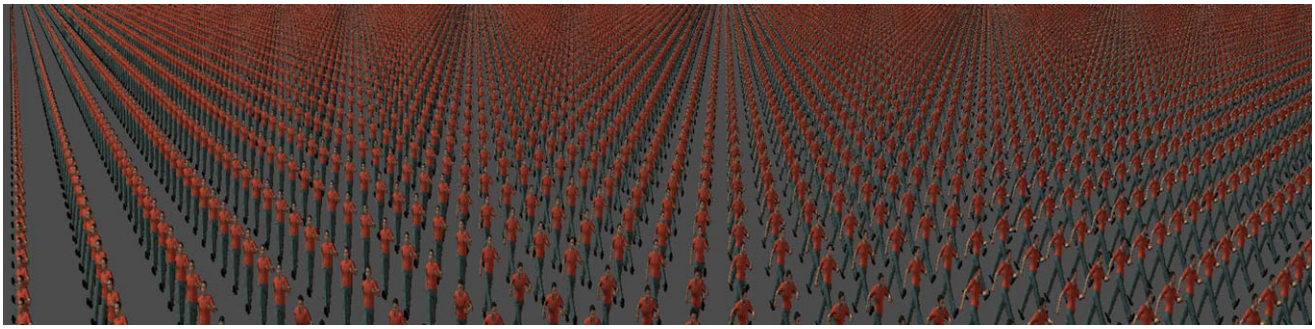

Fig. 3 Densely populated scene of 16,384 characters that are rendered at 25 fps on average by the application implemented using OpenGL and GLSL

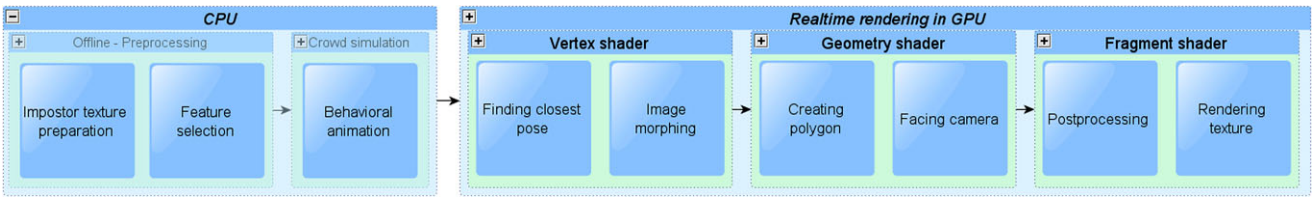

Fig. 4 The overview of the architecture used to implement the proposed methodology describing the hierarchy between various substeps

Fig. 5 Discrete set of captured 16 by 4 view-points which are uniformly distributed over the surface of a bounding hemisphere

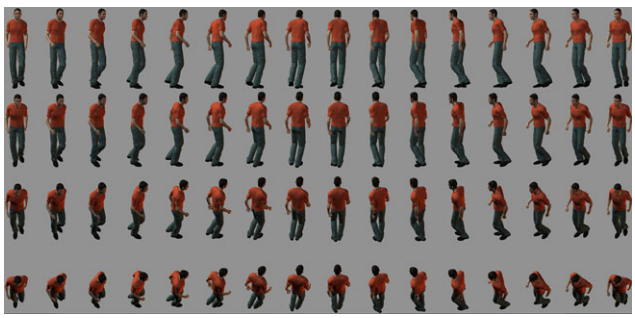

Fig. 6 The partial view of the final impostor texture map including images from different view-points for only one animation frame

3.1 Generating impostor textures

In order to produce impostor textures, we prerendered a discrete set of (16 × 4) viewpoints uniformly distributed over the surface of a bounding hemisphere (Fig. 5) using an orthographic projection for each key-frame and appended them into an overall texture. Using 3D Studio Max, we captured ray-traced images of each viewpoint by rotating the camera around horizontal and vertical axes. Each capture consisted of 256 × 256 pixels, and there were 16 consecutive key-frames that resulted in 8192 × 8192 × 4 bytes to be consumed for a specific avatar (Fig. 6), which is near the maximum memory supported by most of contemporary GPUs. Additional characters or animations requires reducing either the resolution of each view or the number of discretized poses; however, both would decrease the visual or perceptual quality of the characters.

3.2 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 three-dimensionality using two-dimensional representations, each character is rendered to a textured quad facing continuously toward the camera. The texture coordinates of the quad are 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 prerendered images. Later, the geometry shader is employed to generate quads from point primitives received from the

graphics pipeline. These quads are then translated to specified positions. Following this step, the most approximate image to the corresponding key-frame and viewpoint is found or synthesized and finally passed to the fragment shader for postprocessing and rendering.

4 Image morphing

In this work, view morphing is used to generate a smooth 3D animation from one viewpoint to another by smoothly interpolating their camera matrices, in order to synthesize high-quality and perceptually convincing transitions between existing poses. For each reference view, a set of 3D correspondences are established to generate in-between frames by triangulating the set of matched feature points in each image, e.g., using Delaunay triangulation.

As the 3D points are reprojected into their intermediate views, pixels can be mapped from their original source images to their new views using affine or projective mapping. The final image is then composited using a linear blend of the two reference images, and each image is warped toward the other image before blending instead of simply cross-dissolving between two images. No ghosting results if the correspondences have been properly set up and corresponding features are aligned.

The image deformation is calculated as reverse mapping that goes through the destination image pixel by pixel and samples the correct pixel from the source image. Displacements for local deformations are specified through corresponding oriented line segments. Pixels along each line segment are transferred from source to destination as specified, and other pixels are warped using a smooth interpolation of these displacements.

Each line segment correspondence specifies a similarity transform (e.g., translation, rotation, and scaling) for pixels in its vicinity. Line segments influence the overall displacement of the image using a weighting function that depends on the minimum distance to the line segment. For each pixel, the target location for each line correspondence is computed along with a weight that depends on the distance and the line segment length. The weighted average of all target locations then becomes the final destination location.

In order to sample the source image, local transformations of each segment feature are blended based on an inverse distance weighted interpolation to reconstruct the intermediate image. For each pixel location, a weighted average displacement is calculated. Weights are assigned to each line according to their distance to the pixel location, so that the weight is heaviest when the pixel is exactly on the line and weaker the further the pixel is from it.

4.1 Selecting and encoding features

After generating impostor textures, we manually selected control points to identify line pairs as required by the image-morphing algorithm employed (Fig. 7). In order to use the pixel buffer effectively, selected features are encoded as color components of pixels within each subtexture where the alpha channel is equal to zero. Control points have been selected manually using an external interface in MATLAB. Resulting intermediate images and their correlation coefficients with original images are presented after each update.

The visual quality of the generated intermediate image is mainly dependent on the proper selection and the number of line pairs, as shown in Table 2. Using more control points often leads to a better result but reduces the performance of the proposed method by increasing the complexity of the morphing algorithm (Table 1). After a couple of preliminary experiments, we decided to use only eight line pairs per image (Fig. 7). Decisions of optimum features for key-frames were slightly easier because pairwise control points sometimes were not available on rotated images. Thus, selecting eight line pairs for eight key-frames on 16×4 viewpoints took approximately two hours.

Although the selection of line pairs was done manually, it is possible to automatically optimize those features using the quantitative measures and techniques mentioned in Sect. 2.

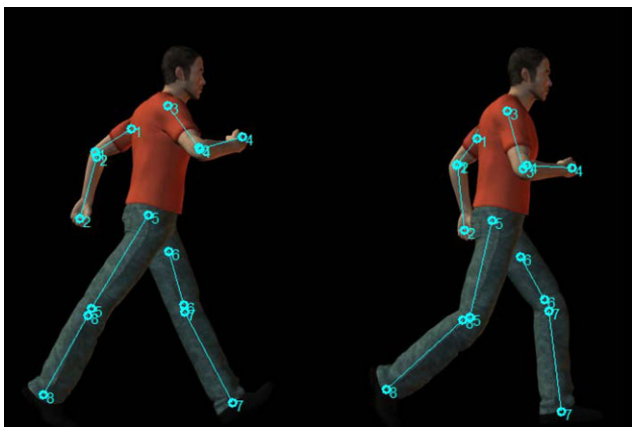

Fig. 7 Selection of line pairs (control points) of two consecutive animation frames for feature-based image metamorphosis

Table 1 Real-time rendering performance statistics

# 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 2 Image similarity statistics as cross-correlation

# of Features	CC-Mean	CC-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

The scale-invariant feature transform (SIFT) algorithm [22] can be utilized to speed up the selection process by providing hints for features. Those hints can be displayed on both images, and users may be asked to select optimum features that would lead to best morphed image.

Those experiments also showed that perceivable artifacts occur 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. 10). For that reason, dedicating part of the extra memory to the faces of intermediate textures that can be pasted on top of procedurally generated images would be advisable. Variations in the faces of characters would increase crowd diversity and still be resource-efficient as faces, as opposed to bodies, cover only a minor area of the texture.

5 Real-time generation using parallel morphing

In order to cut the total amount of texture memory consumed for each animation frame into half, we regenerated intermediate animation frames and viewpoints using image morphing techniques. The parallelization of these algorithms using GPU is straightforward since they are processing all pixels independently through reverse mapping.

5.1 Generating intermediate animations

Differences in viewpoint of the rendered characters might cause unnatural distortions while generating intermediate textures through morphing. For that reason, we have first employed the view morphing algorithm [7] to describe the scene appearance for a more continuous range of viewpoints using two basis views of a static scene.

This technique works by prewarping two images prior to computing a morph and then postwarping the interpolated images. This method allows determining the set of all views on the line between the optical centers of these two basis views. The intermediate perspective is synthesized by rectifying them and then interpolating corresponding pixels. View morphing is suitable for impostor rendering due to its monotonicity property, which means that the camera configurations are known and the fundamental matrix is available.

Fig. 8 Intermediate key-frame generated by the proposed method (*left*) and original image (*right*) of the same animation frame and viewpoint

After view-morphing, instead of direct linear interpolation of pixels, we employed the feature-based image metamorphosis algorithm [8] where pairs of corresponding features, previously specified as line segments, are reconstructed after a combination of weighted local affine transformations. The synthesized image represents a view from a viewpoint or key-frame halfway between the two consecutive frames similar to the expected intermediate frame.

In the proposed method, intermediate frames of animations are actually removed from the impostor texture and generated using image morphing. When they are requested, the feature-based image metamorphosis algorithm is applied using the existing previous and subsequent frames of the animation containing coordinates of line pairs in their transparent pixels (Fig. 7). The previous and next key-frames are warped using the computed transformations and blended to produce the intermediate key-frame (Fig. 8). When existing key-frames are requested, the method acts exactly as the conventional impostor rendering.

5.2 Generating intermediate viewpoints

Besides, we attempted to generate intermediate viewpoints using the proposed method. The view-morphing algorithm almost successfully produces the prewarped images where the perspective of the consecutive viewpoints is matched to the target intermediate viewpoint (Fig. 9). However, manual selection of optimal line pairs at consecutive viewpoints is more difficult because some features may not be visible in both images when the character rotates. Hence, we have not employed the generation of intermediate viewpoints while evaluating the proposed method.

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. Therefore,

Fig. 9 Examples of consecutive viewpoints for generating their intermediate viewpoint using the proposed method

Fig. 10 Intermediate viewpoints that are generated using the same number of (eight) features focusing on body parts (*left*) and the face (*right*) of the character

the perceptual quality of the animation is preserved, as the morphed key-frames are only shown between the original key-frames (Fig. 9). Furthermore, the distribution of a limited number of features over different body parts affects their quality on the resulting image. For instance, the facial area of the character often requires more consideration as it is slightly more inconsistent between consecutive viewpoints than between consecutive key-frames (Fig. 10).

6 Evaluation

The proposed technique was evaluated on an Intel Core2 Quad Q6600 PC with 2 GB RAM using a GeForce GTX 460 graphics card with 1 GB of memory. Different numbers of characters and features, which were selected as line pairs, were 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.

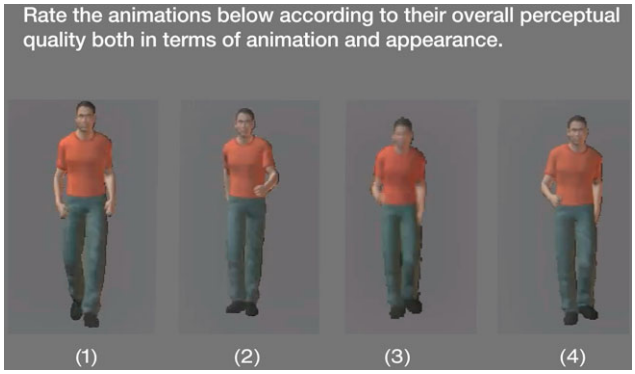

Fig. 11 Screenshot from the session Q3, where the participants was asked to evaluate the overall perceptual quality of the models M1–M4 having different animation and appearance qualities while depicting the same amount of texture memory

6.1 Quantitative experiments

In order to analyze the visual performance of the proposed method, we employed statistical metrics that measure the similarity between original and morphed intermediate (Int) images using numbers of features. The mean and standard deviation (Stdev) of cross-correlations between those sets of target images are shown in Table 2. The similarity of the generated images is quantitatively evaluated by taking the similarity of consecutive (Cons) images as the base performance. The results show that the generated intermediate images are significantly closer to the original images than their consecutive images while the similarity is proportional with the number of features utilized.

6.2 Qualitative experiments

In order to evaluate the contribution of the proposed method over the perceptual quality of impostors, we performed a user study that involved 16 participants. The participants are all undergraduate students in the Computer Science and Engineering Department of the Sabanci University. The study were composed of three sessions (Q1–Q3) where participants was asked to rate (out of 10) impostors (Fig. 11) according to their animation (Q1), appearance (Q2), and overall (Q3) qualities, respectively.

All of the impostors (M1–M4) were rendered in real time, consuming the same amount of texture memory, using either conventional (M1 and M3) or proposed method (M2 and M4). The impostor M1 had 8-key frames less than the impostor M3, which had 16 key-frames but half texture resolution. Using the texture of M1, we have rendered the impostors M2 and M4 and generated 8 additional intermediate key-frames with 12 and 8 features, respectively, via the proposed method.

To interpret the significance of enhancement achieved by the proposed method, we performed analysis of variance

Fig. 12 In-detail view of the session Q2 where impostors from different texture maps have an equal number of key-frames and memory consumption. M4 (*right*) has a higher resolution due to regenerated animation frames using the proposed method

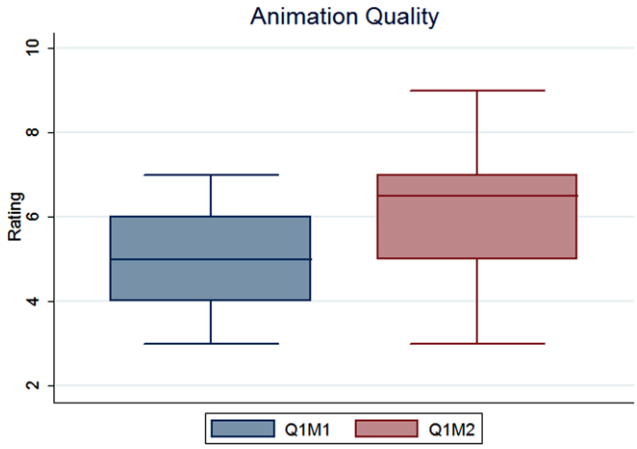

Fig. 13 Box-plot of the animation quality ratings collected from 16 participants during Q1 where M2 is rendered using the proposed method and rated 38 % higher on average

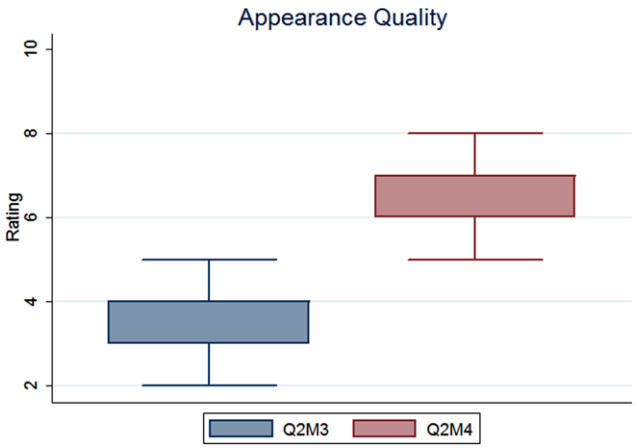

Fig. 14 Box-plot of the appearance quality ratings collected from 16 participants during Q2 where M4 is rendered using the proposed method and rated 87 % higher on average

(ANOVA) over the t-test of the sample that was gathered by the answers of 16 individuals who compared the different models. ANOVA uses statistical procedures to partition the observed variance in a particular variable into components attributable to different sources of variation. When the values follow a normal distribution, there is only one independent variable, and the objective is to detect a significant difference in the means of two sets of values; therefore, ANOVA reduces to Student's two-sample t-test. The results of the ANOVA indicated that the proposed method increases the mean of all three ratings significantly within a 1 % confidence level.

7 Discussion

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. 12). 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 real time. Achieved quality of the resulting images, which are morphed in real time using a limited amount of features, are promising considering that impostor rendering is often used as a level of detail technique.

The user study verified our hypothesis that the proposed method can be utilized to obtain 38 % smoother animations (Fig. 13) using the same amount of texture memory (Q1), or to increase the appearance quality by 87 % (Fig. 14) via reducing the number of key-frames required for preserving the animation quality (Q2). Moreover, the analysis of Q3 indicated that the number of features utilized is affective on the overall quality of the impostors by comparing the performances of M2 and M4 (Fig. 15), a finding which is highly consistent with the quantitative experiments regarding the image similarities calculated by their cross-correlation. Interestingly, M1 and M4 performed almost equal on Q3, which might be because of facial artifacts in M4 due the limited amount of features in our belief; still, M4 performed significantly better than M3 did (Fig. 15).

8 Conclusion

In this work, we proposed an alternative real-time impostor rendering method that reduces memory consumption by generating compelling intermediate textures using image-morphing techniques. To demonstrate the preserved perceptual quality of animations, where half of the key-frames

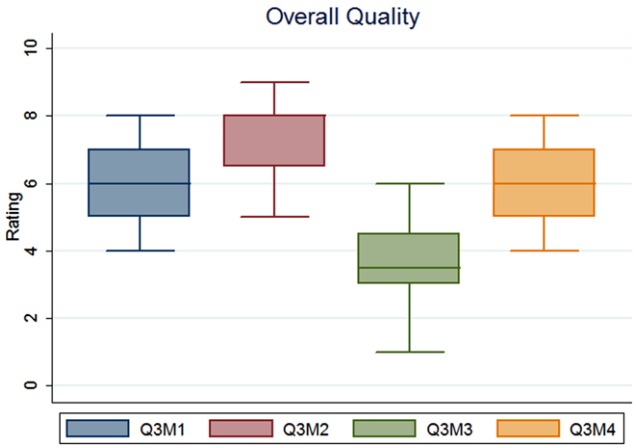

Fig. 15 Box-plot of the overall quality ratings collected from 16 participants during Q3 where M2 and M4 are rendered using the proposed method

were rendered using the proposed methodology, we implemented a system using the graphical processing unit and obtained promising results at interactive frame rates.

The proposed method is visually evaluated by replacing only the intermediate frames of animations with procedurally generated ones. However, it is also applicable 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 suggest 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 and synthesizing faces for them, 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/VqhGG.

Acknowledgements The authors would like to thank anonymous reviewers at CW2011 and TVCJ for their comments that helped us to improve the quality of this manuscript.

References

1. Schaufler, G., Stürzlinger, W.: A three dimensional image cache for virtual reality. Comput. Graph. Forum **15**(3), 227–235 (1996)
2. Shade, J., Lischinski, D., Salesin, D.H., DeRose, T., Snyder, J.: Hierarchical image caching for accelerated walkthroughs of complex environments. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, New York, NY, USA, pp. 75–82 (1996)
3. Millan, E., Rudomin, I.: 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, pp. 49–55, New York, NY, USA (2006)
4. Reynolds, C.: Big fast crowds on PS3. In: Proceedings of the 2006 ACM SIGGRAPH symposium on Videogames, New York, NY, USA, pp. 113–121 (2006)
5. Treuille, A., Cooper, S., Popović, Z.: Continuum crowds. In: ACM SIGGRAPH 2006 Papers, New York, NY, USA, pp. 1160–1168 (2006)
6. Thalmann, D., Grillon, H., Maim, J., Yersin, B.: Challenges in crowd simulation. In: International Conference on CyberWorlds, CW '09, pp. 1–12 (2009)
7. Seitz, S.M., Dyer, C.R.: View morphing. In: Proceedings of the 23rd Annual Conference on Computer Graphics and Interactive Techniques, pp. 21–30 (1996)
8. Beier, T., Neely, S.: Feature-based image metamorphosis. SIGGRAPH Comput. Graph. **26**(2), 35–42 (1992)
9. Aubel, A., Boulic, R., Thalmann, D.: Real-time display of virtual humans: levels of details and impostors. IEEE Trans. Circuits Syst. Video Technol. **10**(2), 207–217 (2000)
10. Hamill, J., McDonnell, R., Dobbyn, S., O'Sullivan, C.: Perceptual evaluation of impostor representations for virtual humans and buildings. Comput. Graph. Forum **24**(3), 623–633 (2005)
11. Yuksel, K.A., Yucebilgin, A., Balcisoy, S., Ercil, A.: Using image morphing for memory-efficient impostor rendering on GPU. In: International Conference on CyberWorlds, CW, pp. 197–202 (2001)
12. Tecchia, F., Chrysanthou, Y.: Real-time rendering of densely populated urban environments. In: Proceedings of the Eurographics Workshop on Rendering Techniques 2000, London, UK, pp. 83–88 (2000)
13. Tecchia, F., Loscos, C., Chrysanthou, Y.: Visualizing crowds in real-time. Comput. Graph. Forum **21**(4), 753–765 (2002)
14. Dobbyn, S., Hamill, J., O'Conor, K., O'Sullivan, C.: Geopostors: a real-time geometry/impostor crowd rendering system. In: Proceedings of the 2005 Symposium on Interactive 3D Graphics and Games, New York, NY, USA, pp. 95–102 (2005)
15. Kavan, L., Dobbyn, S., Collins, S., Žára, J., O'Sullivan, C.: Polypostors: 2D polygonal impostors for 3D crowds. In: Proceedings of the 2008 Symposium on Interactive 3D Graphics and Games, New York, NY, USA, pp. 149–155 (2008)
16. Lister, W., Laycock, R.G., Day, A.M.: A dynamic cache for real-time crowd rendering. Comput. Graph. Forum (2010)
17. Stich, T., Linz, C., Wallraven, C., Cunningham, D., Magnor, M.: Perception-motivated interpolation of image sequences. ACM Trans. Appl. Percept. **8**(2), 11 (2011)
18. Zamith, M., et al.: Real time feature-based parallel morphing in GPU applied to texture-based animation. In: 16th International Conference on Systems, Signals and Image Processing, IWSSIP 2009, pp. 1–4 (2009)
19. Wong, T.Y., Kovesi, P., Datta, A.: Towards quantitative measures of image morphing quality. In: Digital Image Computing: Techniques and Applications, DICTA'05, Proceedings, pp. 19–20 (2005)
20. Gao, P., Sederberg, T.W.: A work minimization approach to image morphing. Vis. Comput. **14**(8–9), 390–400 (1998)
21. Karungaru, S., Fukumi, M., Akamatsu, N.: Automatic human faces morphing using genetic algorithms based control points selection. Int. J. Innov. Comput. Inf. Control **3**, 247–256 (2007)
22. Lowe, D.G.: Object recognition from local scale-invariant features. In: The Proceedings of the Seventh IEEE International Conference on Computer Vision, vol. 2, pp. 1150–1157 (1999)

Kamer Ali Yuksel is a Ph.D. candidate in Computer Science and Engineering program of Sabanci University, Istanbul, Turkey, where he has also obtained his B.Sc. degree. Previously, he worked in reputable industry-connected research institutions such as Deutsche Telekom Laboratories, Berlin, Germany; and Microsoft Research, Cambridge, UK. He is currently working in the Computer Vision and Pattern Analysis Laboratory (VPALAB) to develop next-generation technologies in computer vision, computer graphics, and human-computer interaction.

Alp Yucebilgin received his B.Sc. degree in Computer Science and Engineering from Sabanci University, Istanbul, Turkey. He is currently M.Sc. student in Computer Science program of University of California, Santa Barbara, CA. He is member of Four Eyes Lab and his research interests are physically based animation, realistic image synthesis, non-photorealistic rendering, and real-time rendering.

Selim Balcisoy obtained his B.S. in Electronical Engineering from ETH, Zurich in 1996. He received his Ph.D. on Computer Science in 2001 from EPFL. Between 2001 and 2004, he was a Senior Research Engineer at Nokia Research Center, USA, where he conducted research on mobile graphics. His research interests include augmented reality, virtual environments, cultural heritage, and information visualization. Dr. Balcisoy co-authored over 30 publications at refereed international journals and conferences, and has been granted with one U.S. Patent.

Aytul Ercil received the M.Sc and Ph.D. degrees in applied mathematics from the Brown University, Providence, USA. She is a faculty member in the Faculty of Engineering and Natural Sciences at Sabanci University, Istanbul, Turkey. Her research interests are invariant object recognition, shape modeling, texture analysis, pattern recognition, biometrics, and multimodal classification.