# Frame Cache Management for Multi-frame Rate Systems

Stefan Hauswiesner, Philipp Grasmug, Denis Kalkofen, and Dieter Schmalstieg

Institute for Computer Graphics and Vision, Graz University of Technology

**Abstract.** Multi-frame rate systems decouple viewing from rendering in an asynchronous pipeline. Multiple GPUs can be used as frame sources, while a primary GPU is responsible for viewing and display update. Conventionally, the last rendering result is used for display. However, modern GPUs are equipped with a fairly large amount of memory which allows frames to be cached in video memory. As long as the data is static, caching allows for a more sophisticated reference frame selection that increases the output quality. With a growing frame database, images for most viewpoints can be queried from the cache and the system converges into a conventional image-based rendering system. However, multi-frame rate systems use purely virtual image sources. As a consequence, the rendering process can be actively steered by the viewing process, which allows for advanced strategies. Moreover, by picking multiple reference frames from the cache, we can avoid display artifacts arising from occlusions.

## 1 Introduction

Multi-frame rate rendering helps interactive systems to achieve little latency and high frame rates [1]. It decouples display updates from image generation in a pipeline with asynchronous communication across multiple GPUs. The display update stage can guarantee high frame rates and nearly latency-free response to user interaction, such as moving the viewpoint. At the same time, frame sources in the backend can produce new high quality images at their own, usually slower pace. Such systems employ image-based rendering (IBR) to hide the latency of its rendering nodes. Conventionally, the last rendering result is used as a reference frame to display the current viewpoint. This is a natural choice, because it requires little memory and provides acceptable quality. However, modern GPUs are equipped with a large amount of on-board memory, which can be used to improve the quality.

Therefore, we introduce frame caching strategies, which allow to reuse already rendered content. These strategies maximize the utility of available GPU memory by storing frames which are likely to improve future visual quality. Every frame which is received from a frame source is evaluated for its usefulness (section 4). By using more than a single reference frame for warping, the output quality can be increased especially at depth discontinuities. We suggest a heuristic for selecting multiple frames from the cache (section 5). Finally, we evaluate the visual quality (section 6).

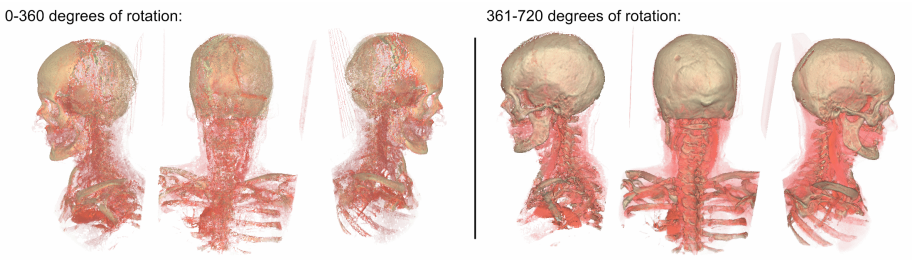



**Fig. 1.** The effect of caching reference frames when rotating a dataset around the Y axis. The first turn contains visible artifacts, while the second turn offers considerably better visual quality.

## 2 Related work

For coping with situations where the computation load exceeds the interactive rendering power of a parallel system, but some computations have to be performed with minimum latency, the concept of multi-frame rate rendering was introduced [1]. Multi-frame rate systems use a number of rendering nodes to independently render parts of the scene. The results are combined without waiting for updates. Thus, the display node (in our case: the *primary GPU*) is not slowed down by the rendering nodes, which enables the display node to perform computations with bounded latency to user input or other events. This approach is especially useful for applications, in which high interactivity has to be guaranteed, e. g., augmented reality, stereoscopic display or interaction with magic lenses etc.

Image warping [2] is a form of image based rendering that allows to extrapolate new views from existing images with per-pixel depth information. Our system uses forward image warping for latency compensation.

The work in [3–5] describes a multi-frame rate architecture, which is able to use the GPU to accelerate image warping by using vertex shader programs for the required scatter operation. However, this work is not suitable for transparency and volumetric datasets and does not employ an image cache. [6] support multi-frame rate volume rendering with transparency, but assume a fixed view point.

Recent implementations of the render cache [7] utilize GPUs for improved performance. In contrast to the render cache notion, our system does not operate on single points. Modern GPUs are efficient at data parallel execution. Our system therefore manages the creation, storage and transfer of frames, or images. The Tapestry system [8] also elaborates on exploiting frame-to-frame coherence by utilizing view-dependent meshes. However, these meshes are not suitable for transparent objects, like volumetric data visualizations.

## 3 Multi-frame rate rendering with caching

This work is based on a multi-frame rate architecture [9] that is capable of rendering scenes consisting of meshes as well as volumetric objects. Volumetric objects are layered to account for motion parallax withing the object. Rendering can be performed on the GPUs of a single PC, or even on remote machines. The display stage employs image-warping, because the reference frame that is used for display was generally not rendered exactly from the desired viewpoint. It is therefore able to compensate for the latency of the image-generating GPUs.

From the users point of view, our system looks like a simple model viewer. The camera is able to orbit the scene objects and move towards the model or away from it, thus enabling zoom operations. Currently, our system does not support viewpoints inside the scene. This is not a general limitation, but simplifies the frame cache operations.

## 4 Cache Organization

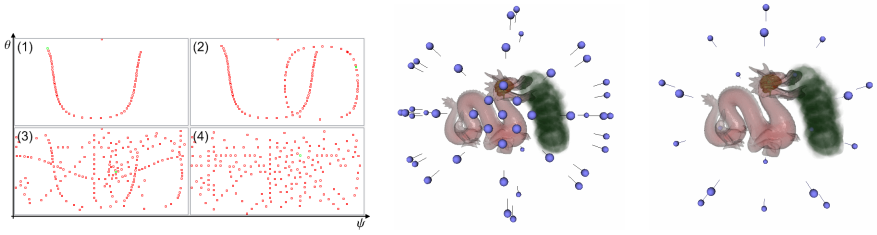
In our previous work [9], we use the last known rendering result of a frame source GPU for image warping and display. However, the visual quality directly scales with the number and density of available frames along the navigation path [10]. Therefore, our new system stores as many valuable reference frames as possible, but also extends the image cache automatically during idle phases. Note, that reusing rendered images introduces a limitation: the scene's objects are not allowed to deform arbitrarily. Animations can only be performed when they can be described by a single transformation matrix per object.

### 4.1 Retaining frames

The maximum number of frames that a reference frame cache can store is limited by the size of GPU memory. A simple round-robin strategy will not create a database with good memory utilization. For example, a reference frame, which shows approximately the same features as another, does not contribute to the display result and therefore wastes memory.

When receiving a new reference frame, its viewpoint distance to the closest neighbor is compared to a threshold. New frames below this threshold do not add enough information to the cache database and are discarded. Above the threshold, the frame is stored if a free slot in the cache is available. If no free slot is available, a trade-off has to be found: from the closest pair of frames, one is replaced. Figure 2(a) illustrates how the frame cache improves over time. Figure 1 shows the result of this strategy.

Finding an appropriate sampling density by setting the mentioned threshold requires knowledge of the dataset and available GPU memory. We use the angle between two viewing directions to determine the distance of two frames. For the datasets shown in this paper, we used a threshold of 4.5°.




(a) (b) (c)

**Fig. 2.** (a) spherical coordinates $\theta$,$\psi$ of reference frame viewpoints during user navigation. Image (1) shows 65 reference frames and is captured after a complete rotation around the object. (2) shows 109 reference frames after another rotation. (3) shows the situation when the cache is full (200 frames). After a long and diverse navigation session the camera positions are evenly distributed around the object according to our update strategy (4). (b) and (c) show different prerendering strategies with camera positions augmented as spheres.

### 4.2 Estimating subsequent camera locations

When the system is not in continuous motion, and no user interaction is performed for a period of time, no redraw is necessary. During these idle periods, the rendering nodes can be scheduled to render new viewpoints, without interfering with the display output. The primary GPU stores the new information as described above, which helps to improve future image quality. We call this process *prerendering*.

Generating valuable reference frames for future user interaction requires to predict future navigation. In typical exploration scenarios, the user is only interested in a relatively small range of viewing angles, whereas in navigation tasks all angles may be visible at some point. If the application follows an easily predictable animation, it is also possible to extrapolate the last known movements.

Our system supports these three types of scenarios: for examination, we assume that future viewing angles are close to the current one, and therefore start with a fine sampling around it. The more different the viewing angles, the sparser we sample. See Figure 2 (b) for a visualization of such a prerendering strategy. For navigation applications, camera positions for prerendering are spaced evenly on the bounding sphere of the scene. This can be achieved by placing the reference camera positions at the vertices of a regular polyhedron, which guarantees an equal distance between neighboring positions (see Figure 2 (c)). If the viewpoint follows a certain path, this information can greatly improve image quality. However, it is domain specific, so we provide just a simple motion prediction in addition to the strategies above. This prediction places reference frame viewpoints along the extrapolated path of the last user-induced camera motion with a step size that accounts for the speed of the motion. When a steady rotation around the object is started, this method is capable of producing reference frames just in time for the display process.

### 4.3 Handling transfer function changes

Multi-frame rate rendering is often employed to improve the performance of viewing volumetric datasets, which are especially common in medical applications. In such applications, the transfer function that maps density values to color values can be changed. This invalidates all frames in the cache, reducing the usefulness of the suggested approach. However, we employ an algorithm that transforms all images in the frame cache to approximate the change of the transfer function.

As described above, every frame in the cache consists of a set of layers. Each layer consists of color, depth and normal values at each pixel. Using this information alone, it is not possible to transform the images to reflect a change of the transfer function. In addition to our previous work [9], we therefore also store representative density values and the layer thickness per pixel, which help to reclassify every pixel using the new transfer function. The representative density value for each ray segment is selected from the sample that has maximum impact on the output color: the sample with the highest opacity weight after classification.

The process of reclassifying a pixel starts by classifying the pixel using its representative density value and the new transfer function. The queried color value is accumulated according to the thickness (or length) of the ray segment. The resulting color represents a homogeneously filled ray segment, which is not necessarily a good approximation. However, there is more information available to improve the quality: the representative density value can be classified by the old transfer function. The resulting color value is then compared to the actual color value of the ray segment. The opacity ratio between these two colors describes how strong the opacity of a ray increased during accumulation. This ratio is multiplied with the color resulting from the new classification to obtain the final approximation.

Results of this reclassification can be seen in figure 3. We evaluated the usefulness of the algorithm by examining the reclassification results between several transfer functions. We observed that the layering criterion that is used to separate the volume dataset into view-dependent layers has strong impact on the visual quality of the result. It controls how well features are distributed between the layers. If more features are mixed in a single layer, the chosen density value is not a good representative. To accommodate for this fact, we modified the layering criterion of [9] to create layer boundaries whenever the density variance of a ray segment becomes higher than a threshold. A threshold of zero variance maximizes the visual quality, because only one density value per layer remains. This requires many layers. Therefore the resulting quality mainly scales with the number of layers. Increasing the number of layers improves the reclassification quality, but decreases overall performance. The reclassification itself takes less than 2 milliseconds per layer on the evaluation system (see below) and is therefore not a limiting factor.

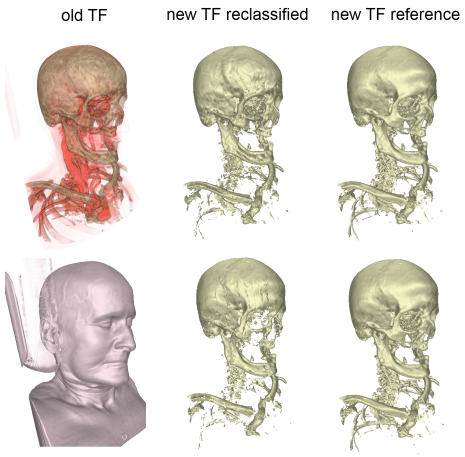



**Fig. 3.** Reclassification results: the left column shows the initial transfer functions (TF). The right column shows renderings of the new TF. The middle column results are approximated from the left TFs. Ten layers are used for all of the renderings.

## 5 Reference frame selection

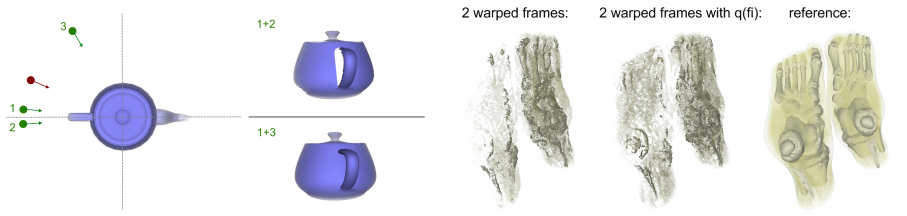
When previously unseen parts of the scene become visible, no information is available for these parts. This is called a disocclusion (see the top right of Figure 4(a) for an example). To better deal with disocclusions, we use multiple reference views that are selected from the cache. Warping from multiple reference frames increases the chance that suitable information is found for every pixel.

As a quality criterion for selecting reference frames, the similarity of the reference frame's view to the current view seems obvious. Similarity can be computed as a function of the Euclidean distance of the camera positions, or the viewing angle between them. However, reference frames which are closest to the desired camera position are not necessarily the best. This is especially true for an unstructured viewpoint database, which is created by our system. Figure 4(a) illustrates how warping from the two closest cameras may not contain enough information for a properly generated image. Unfortunately, problems such as the one depicted in figure 4(a) occur frequently.

Diverse viewpoints are less likely to suffer from the same occlusions. Therefore, a suitable way to avoid occlusions is to introduce a penalty factor depending on the distance between two views of the selected set. This factor enforces a certain distance between the selected reference frame viewpoints, while the set stays close to the current view. The quality of a reference frame $f_i$ for the current frame $c$ can therefore be expressed as:

$$q(f_i) = -\alpha * dist(f_i, c) + \beta * dist(f_i, closest(f_i))$$

with $closest(x)$ searching for the closest reference frame to $x$ in the set of selected frames, and $\alpha$ and $\beta$ representing weighting coefficients. The ratio $\alpha/\beta$




(a) (b)

**Fig. 4.** (a) in this example, the two closest frames (green 1 and 2) to the current camera position (red) do not include information behind the handle of the teapot. Selecting more distant, but more diverse views can lead to a better result in such a case. (b) screenshots of a latency compensated rendering after a large viewpoint motion. Two warped frames selected by distance only (left) and by our $q(f_i)$ metric (middle). Right: reference rendering of the VIX feet dataset.

defines how the distance to the current view and the distance to the nearest neighbor relate. A high $\alpha/\beta$ favors nearby frames, whereas a low $\alpha/\beta$ favors diverse views. To find the set of selected frames, we start with an empty set and evaluate $f_i$ for all frames. We pick the frame with the highest $f_i$ and add it to the selection set. Then, $f_i$ is evaluated again for all remaining frames and again the highest is added to the set. This is performed until the desired amount of reference frames are in the set for warping.

When using angles as the distance measurement and $\alpha/\beta = 3$, the example of figure 4(a) ranks the reference frames from best to worst $1 > 3 > 2$, thus resolving the problem. During our evaluations a ratio $\alpha/\beta$ in the range $[2, 3]$ worked out best. Figure 4(b) illustrates the benefit of our metric.

When warping from more than one reference frame, several fragments are transformed to the same pixel location in most cases. Using a depth test to dissolve fragment collisions was suggested in [11], but the depth test does not necessarily sort out fragments of low quality (inaccurately positioned). Also, the depth test possibly may not produce homogeneously generated surfaces, especially when the source frames contain fragments with similar depth, which is a situation prone to numerical issues. We found that blending all or several fragments of a pixel is not a good option either, because wrongly positioned fragments from lower similarity views interfere with the object's final appearance.

As a remedy, reference frames are sorted by their $q(f_i)$, and then rendered from low to high quality without using any depth buffer test. Layers are not mixed, so the overall depth order stays intact. This way fragments from a potentially better (closer) reference frame overwrite bad fragments. The surfaces appear more homogeneous (smoother), while still being able to fill holes.

### 5.1 Resolution of the reference frames

The above explanations assume that all reference frames have sufficient resolution for warping. This implies that the user's viewpoint lies on a sphere around

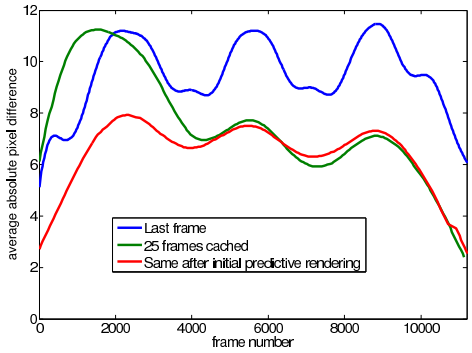



**Fig. 5.** Pixel error comparison of multi-frame rate rendering from the previous frame or from a reference frame cache (blue and green graphs). After initialization by pre-rendering, also the first seconds have improved quality (red graph).

the object, effectively prohibiting any zooming operations. To remove this limitation, all viewpoints that are used for rendering lie on a sphere with an as-small-as-possible radius. Therefore, the user can only freely select the viewpoint of the display node. The viewpoints of the frame source nodes always lie on the smallest sphere, but copy the selected viewing angle of the user. This ensures that all images have a sufficiently high resolution for future reuse, as long as the viewpoint stays outside this sphere. The radius can be trivially computed from the scene bounding box's extents.

## 6 Performance and quality evaluation

We recorded the viewpoint motion during a user session that contains a rotation around the Y-axis. Viewpoint location, orientation and time-stamps are stored for each frame during recording. When a session is replayed, the current viewpoint is interpolated from the recorded data and passed to our system as input. To obtain ground truth images, we rendered the recorded session using an offline raycaster. We used volumetric objects for evaluation, because they are highly complex even for modern GPUs.

For evaluation we used a single desktop PC equipped with 2 GPUs: a GeForce GTX 480 (Fermi generation) for latency compensation and display, and a GeForce GTX 275 as the frame source. The rendering resolution of all GPUs was set to 580x610. The frame rate of the output GPU was stable between 27 and 30 frames per second, while the frame source GPU performed raycasting at a stable frame rate between 0.3 and 0.5 frames per second.

For evaluation, we replay the recorded user session in real-time. The output images are stored with time stamps for an offline quality comparison. For every image in the evaluation track, the closest image from the reference track is found by comparing timestamps. From these two images, the average absolute pixel difference (AAPD) is computed as a quality metric.

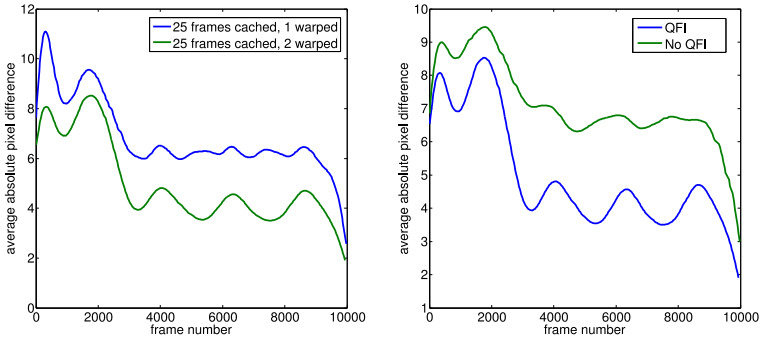



**Fig. 6.** AAPD comparison of our system using either one or two reference frames (a), and a comparison between our $q(f_i)$ metric and the angular distance.

Our first evaluation uses the MANIX head dataset (see figure 1) from the OsiriX database [12]. Figure 5 shows a pixel error comparison of warping from the previous frame or from a reference frame cache, which is dynamically updated during the evaluation. The camera orbits around the dataset for 4 full rounds. Around frame number 2800 the first round is completed, and the frame cache succeeds in improving the output quality. This proves that the reference frame cache is effective when coherence is present.

Figure 5 additionally shows the effect of prerendering. The frame cache has been filled automatically prior to the evaluation run. We used the prerendering strategy that places viewpoints regularly on the bounding sphere. During the first camera orbit, the predictively filled frame cache provides superior visual quality. During further revolutions, the dynamic frame cache without prior rendering catches up and the implementations behave similarly.

The second evaluation uses the VIX feet dataset. We performed a pixel error comparison of multi-frame rate rendering from a frame cache using either one or two reference frames. Two frames help to cover disocclusion artifacts. When rendering from two reference frames, our $q(f_i)$ metric improves the quality when compared to a simple two-nearest-viewpoints selection strategy. Figure 6 shows the measured AAPDs. Rendering from more reference frames does not generally improve the visual quality. When the selected frames were rendered from too distant viewpoints, the quality even declines.

## 7 Conclusions and future work

We have described a system that is suitable for displaying arbitrarily rendered static scenes at high frame rates using a multi-frame rate approach in combination with image warping. Because scenes are static, our system can improve over time by creating and maintaining an image cache, which is updated either by user interaction or prerendering during idle phases. The suggested update

strategy attempts to maximize the memory utilization. The cache can be preserved even when the transfer function of the volume dataset changes. The visual quality of the system is further improved by warping layers from several reference frames that are selected according to our quality metric. This metric favors diverse views of the scene over too similar, nearby views. While the described method assumes that the viewpoint orbits around the scene, it is not limited to such a scenario: future work may utilize both the viewpoint and the viewing angle to cache and select images.

## Acknowledgements

This work was funded by the Austrian Science Fund (FWF) under contract P-24021-N23.

## References

1. Springer, J.P., Beck, S., Weiszig, F., Reiners, D., Froehlich, B.: Multi-frame rate rendering and display. In Sherman, W.R., Lin, M., Steed, A., eds.: VR, IEEE Computer Society (2007) 195–202
2. Mark, W.R., Mcmillan, L., Bishop, G.: Post-rendering 3d warping. In: In 1997 Symposium on Interactive 3D Graphics. (1997) 7–16
3. Smit, F.A., van Liere, R., Fröhlich, B.: An image-warping vr-architecture: design, implementation and applications. In: VRST '08: Proceedings of the 2008 ACM symposium on Virtual reality software and technology, New York, NY, USA, ACM (2008) 115–122
4. Smit, F.A., van Liere, R., Fröhlich, B.: The design and implementation of a vr-architecture for smooth motion. In: VRST '07: Proceedings of the 2007 ACM symposium on Virtual reality software and technology, New York, NY, USA, ACM (2007) 153–156
5. Smit, F.A., van Liere, R., Beck, S., Fröhlich, B.: An image-warping architecture for vr: Low latency versus image quality. In: VR, IEEE (2009) 27–34
6. Springer, J.P., Lux, C., Reiners, D., Froehlich, B.: Advanced multi-frame rate rendering techniques. In: VR, IEEE (2008) 177–184
7. Velázquez-Armendáriz, E., Lee, E., Bala, K., Walter, B.: Implementing the render cache and the edge-and-point image on graphics hardware. In: Proceedings of Graphics Interface 2006. GI '06, Toronto, Ont., Canada, Canada, Canadian Information Processing Society (2006) 211–217
8. Simmons, M.: Tapestry: An Efficient Mesh-based Display Representation for Interactive Rendering. PhD thesis, EECS Department, University of California, Berkeley (2001)
9. Hauswiesner, S., Kalkofen, D., Schmalstieg, D.: Multi-frame rate volume rendering. In: EGPGV, Norrköping, Sweden, Eurographics Association (2010)
10. Mark, W.: Post-rendering 3d image warping: Visibility, reconstruction, and performance for depth-image warping. Technical report, Chapel Hill, NC, USA (1999)
11. Chen, S.E., Williams, L.: View interpolation for image synthesis. In: SIGGRAPH '93: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, New York, NY, USA, ACM (1993) 279–288
12. OsiriX: Dicom sample image sets. OsiriX Imaging Software, http://pubimage.hcuge.ch:8080/ (2009)