NORTHWESTERN
UNIVERSITY

Computer Science Department

Technical Report
NWU-CS-04-47
Oct 28, 2004

Temporally Adaptive Frameless Rendering

Abhinav Dayal, Cliff Woolley, David Luebke, Benjamin Watson

Abstract

Recent advances in computational power and algorithmic sophistication have made ray tracing an increasingly viable and attractive algorithm for interactive rendering. Assuming that these trends will continue, we are investigating novel rendering strategies that exploit the unique capabilities of interactive ray tracing. Specifically, we propose a system that adapts rendering effort spatially, sampling some image regions more densely than others, and temporally, sampling some regions more often than others. Our system revisits and extends Bishop et al.'s frameless rendering with new approaches to sampling and reconstruction. We make sampling both spatially and temporally adaptive, using closed loop feedback from the current state of the image to continuously guide sampling toward regions of the image with significant change over space or time. We then send these frameless samples in a continuous stream to a temporally deep buffer, which stores all the samples created over a short time interval. The image to be displayed is reconstructed from this deep buffer. Reconstruction is also temporally adaptive, responding both to sampling density and color gradient. Where the displayed scene is static, spatial color change dominates and older samples are given significant weight in reconstruction, resulting in sharper images. Where the scene is dynamic, more recent samples are emphasized, resulting in a possibly blurry but up-to-date image. We describe a CPU-based implementation that runs at near-interactive rates on current hardware, and analyze simulations of the real-time performance we expect from future hardware-accelerated implementations. Our analysis accounts for temporal as well as spatial error by comparing displayed imagery across time to a hypothetical ideal renderer capable of instantaneously generating optimal frames. From these results we argue that the temporally adaptive approach is not only more accurate than frameless rendering, but also more accurate than traditional framed rendering at a given sampling rate.

Keywords: [**Computer Graphics**]: Display algorithms, Raytracing, Virtual reality, frameless rendering, adaptive rendering, reconstruction, sampling, global illumination.

Technical Report (NWU-CS-04-47)
Northwestern University (Evanston, IL)

Temporally Adaptive Frameless Rendering

Abhinav Dayal[†], Cliff Woolley[‡], David Luebke[‡], and Benjamin Watson[†]

† Department of Computer Science, Northwestern University
‡ Department of Computer Science, University of Virginia

Abstract

Recent advances in computational power and algorithmic sophistication have made ray tracing an increasingly viable and attractive algorithm for interactive rendering. Assuming that these trends will continue, we are investigating novel rendering strategies that exploit the unique capabilities of interactive ray tracing. Specifically, we propose a system that adapts rendering effort spatially, sampling some image regions more densely than others, and temporally, sampling some regions more often than others. Our system revisits and extends Bishop et al.'s frameless rendering with new approaches to sampling and reconstruction. We make sampling both spatially and temporally adaptive, using closed loop feedback from the current state of the image to continuously guide sampling toward regions of the image with significant change over space or time. We then send these frameless samples in a continuous stream to a temporally deep buffer, which stores all the samples created over a short time interval. The image to be displayed is reconstructed from this deep buffer. Reconstruction is also temporally adaptive, responding both to sampling density and color gradient. Where the displayed scene is static, spatial color change dominates and older samples are given significant weight in reconstruction, resulting in sharper images. Where the scene is dynamic, more recent samples are emphasized, resulting in a possibly blurry but up-to-date image. We describe a CPU-based implementation that runs at near-interactive rates on current hardware, and analyze simulations of the real-time performance we expect from future hardware-accelerated implementations. Our analysis accounts for temporal as well as spatial error by comparing displayed imagery across time to a hypothetical ideal renderer capable of instantaneously generating optimal frames. From these results we argue that the temporally adaptive approach is not only more accurate than frameless rendering, but also more accurate than traditional framed rendering at a given sampling rate.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation

1. Improving interactive rendering

In recent years a number of traditionally offline rendering algorithms have become feasible in the interactive realm. The sudden appearance of programmable high-precision graphics processors (GPUs) has drastically expanded the range of algorithms that can be employed in real-time graphics; meanwhile, the steady progress of Moore's Law has made techniques such as ray tracing, long considered a slow algorithm suited only for offline realistic rendering, feasible in real-time rendering settings [23]:. These trends are related; indeed, some of the most promising research on interactive global illumination performs algorithms such as ray tracing and photon mapping directly on the GPU [18, 19]. Future hardware should provide even better support for these algorithms, quickening the day when ray-based algorithms are an accepted and powerful component of every production rendering system. What makes interactive ray tracing attractive? Researchers in the area have commented on the ray tracing's ability to model physically correct global illumination phenomena, its easy applicability to different shaders and primitive types, and its output-sensitive running time, only weakly dependent on scene complexity [25]. We focus on another unique capability available in a ray-based renderer but not a depth-buffered rasterizer. We believe that the ability of interactive ray tracing to selectively sample the image plane enables a new approach to rendering that is more interactive, more accurate, and more portable. To achieve these goals, we argue that the advent of real-time ray tracing demands

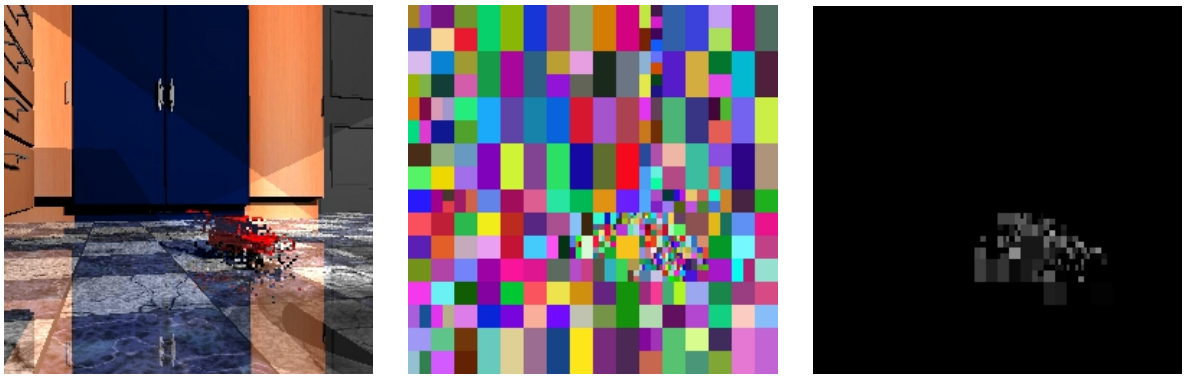

Figure 1: *Adaptive frameless sampling of a moving car in a static scene. Left, the newest samples in each pixel region. Middle, the tiles in the spatial hierarchy. Note small tiles over car and high contrast edges. Right, the per-tile derivative term, which very effectively focuses on the car.*

a rethinking of the fundamental sampling strategies used in computer graphics.

The topic of sampling in ray tracing, and related approaches such as path tracing, may seem nearly exhausted, but almost all previous work has focused on *spatial sampling*, or where to sample in the image plane. In an interactive setting, the question of *temporal sampling*, or *when* to sample with respect to user input, becomes equally important. Temporal sampling in traditional graphics is bound to the frame: an image is begun in the back buffer incorporating the latest user input, but by the time the frame is swapped to the front buffer for display, the image reflects stale input. To mitigate this, interactive rendering systems increase the frame rate by reducing the complexity of the scene, trading off fidelity for performance. We consider this tradeoff in terms of *spatial error* and *temporal error*. Spatial error is caused by rendering coarse approximations for speed, and includes such factors as resolution of the rendered image and geometric complexity of the rendered models. Temporal error is caused by the delay imposed by rendering, and includes such factors as how often the image is generated (frame rate) and how long the image takes to render and display (latency).

In this paper we investigate novel sampling schemes for managing the fidelity-performance tradeoff. Our approach has two important implications. First, we advocate *adaptive temporal sampling*, analogous to the adaptive spatial sampling that takes place in progressive ray tracing [16, 2, 14]. Just as spatially adaptive renderers display detail *where* it is most important, adaptive temporal sampling displays detail *when* it is most important. Second, we advocate *frameless rendering* [3], in which samples are not collected into coherent frames for double-buffered display, but instead are incorporated immediately into the image. Frameless rendering, which requires a per-sample rendering algorithm such as real-time ray tracing, decouples spatial and temporal updates and thus enables very flexible adaptive spatial and temporal sampling.

Our prototype adaptive frameless render is broken into three primary sub-systems. An *adaptive sampler* directs rendering to image regions undergoing significant change (in space and/or time). The sampler produces a stream of samples scattered across space-time; recent samples are collected and stored in a temporally *deep buffer*. An *adaptive reconstructor* repeatedly reconstructs the samples in the deep buffer into an image for display, adapting the reconstruction filter to local sampling density and color gradients. Where the displayed scene is static, spatial color change dominates and older samples are given significant weight in reconstruction, resulting in sharper images. Where the scene is dynamic, only more recent samples are emphasized, resulting in a possibly blurry but up-to-date image.

We describe the design of an interactive system built on these principles, and show in simulation that this system achieves superior rendering accuracy and responsiveness. We evaluate our system with a "gold standard" analysis that compares displayed imagery to the ideal image that would be displayed by a hypothetical ideal renderer, evaluating the image difference at using mean RMS error and and show that it outperforms not only the pseudorandom frameless sampling of Bishop et al. [3], but also traditional framed sampling strategies with the same overall sampling rate. Since our approach is self-monitoring, we also argue that it can achieve a new level of portability and adaptivity to changes in platform and load.

2. Related work

2.1. Interactive ray tracing

Recent years have seen interactive ray tracing go from an oxymoron to a reality. Interactive ray tracers have been demonstrated on supercomputers [17], PC clusters [26], on

the SIMD instruction sets of modern CPUs [24], and on graphics hardware [18] [4]. Wald et al. provide a good summary of the state of the art [25]. To build an interactive ray tracer, while hardly simple, is becoming a matter of engineering. Looking forward, hardware and software systems are being developed [21] [10] that help harness the rendering power of PC clusters, and consumer graphics hardware is growing more powerful and more flexible at an astonishing rate. Real-time ray tracing is currently feasible; we believe it will soon become commonplace even on commodity desktop hardware.

2.2. Interruptible rendering

Recent work on temporally adaptive sampling includes a new approach to fidelity control called *interruptible rendering* [30] that adaptively controls frame rate to minimize the sum of spatial and temporal error. They propose a progressive rendering framework that renders a coarse image into the back buffer and continuously refines it, while tracking the error introduced by subsequent input (such as changes in viewpoint). When this temporal error exceeds the spatial error caused by coarse rendering, there is no longer any reason to refine further, since any improvement to the appearance of objects in the image will be overwhelmed by their incorrect position and/or size. In other words, further refinement becomes pointless when the error due to the image being late is greater than the error due to the image being coarse. The front and back buffers are then swapped and rendering begins again into the back buffer for the most recent viewpoint. The resulting system produces coarse, high frame-rate display when input is changing rapidly, and finely detailed, low frame rate display when input is static.

2.3. Frameless rendering

Interruptible rendering retains a basic underlying assumption of interactive computer graphics: all pixels in a given image represent a single moment in time (or possibly a fixed duration surrounding that moment, e.g. the "shutter speed" used for motion blur). When the system swaps buffers, all pixels in the image are simultaneously replaced with pixels representing a different moment in time. In interactive settings, this coherent temporal sampling strategy has several unfortunate perceptual consequences: temporal aliasing, delay, and temporal discontinuity. Temporal aliasing results when the sampling rate is inadequate to capture high speed motion. Motion blur techniques can compensate for this aliasing, but are generally so expensive that in interactive settings they actually worsen the problem. Delay is a byproduct of double buffering, which avoids tearing (simultaneous display of two partial frames) at the cost of ensuring that each displayed scene is at least two frames old before it is swapped out. Even at a 60 Hz frame rate, this introduces 33 ms of delay – a level that human factors researchers have consistently shown can harm task performance [29] [20]. Finally, when frame rates fall below 60 Hz, the perceptual sensation of image continuity is broken, resulting in display of choppy or "jerky" looking motion.

Interruptible rendering performs adaptive temporal sampling to achieve higher accuracy, but that sampling is still coherent: all pixels (or, more generally, spatial samples) still represent the same moment in time. We have since focused our research on the unique opportunities for temporally adaptive rendering presented by Bishop et al.'s *frameless rendering* [3]. This novel rendering strategy replaces the coherent, simultaneous, double-buffered update of all pixels with stochastically distributed spatial samples, each representing the most current input when the sample was taken. Frameless rendering thus decouples spatial and temporal sampling, so that the pixels in a frameless image represent many moments in time.

We build on and improve the original frameless rendering approach. Our rendering system samples rapidly changing regions of an image coarsely but frequently to reduce temporal error, while refining static portions of the image to reduce spatial error. We improve the displayed image by performing a reconstruction step, filtering samples in space and time so that older samples are weighted less than recent samples; reconstruction adapts to the local sampling density and color gradient to optimize the filter width for different parts of the scene.

In the following sections we describe our temporally adaptive sampling and reconstruction strategies, and discuss their implementation in a simulated prototype system. We next describe the "gold standard" evaluation technique and analyze our prototype against traditional rendering as well as an "oracle" system that knows which samples are valid and which contain stale information. We close by discussing future directions and argue that frameless, temporally adaptive systems will ultimately provide more interactive, accurate, portable rendering.

3. Temporally adaptive, closed-loop sampling

While traditional frameless sampling is unbiased, we make our frameless renderer adaptive to improve rendering quality. Sampling is both spatially adaptive, focusing on regions where color changes across space; and temporally adaptive, focusing on regions where color changes over time (Figure 1). As in previous spatially adaptive rendering methods [16] [2] [14] [9], adaptive bias is added to sampling with the use of a spatial hierarchy of *tiles* superimposed over the view. However, while previous methods operated in the static context of a single frame, we operate in a dynamic frameless context. This is has several implications. First, rather than operating on a frame buffer, we send samples to a temporally *deep buffer* that collects samples scattered across space-time. Our tiles therefore partition a space-time volume using planes parallel to the temporal axis. As in framed

schemes, color variation within each tile guides rendering bias, but variation represents change over not just space but also time. Moreover, variation is not monotonically decreasing as the renderer increases the number of tiles, but constantly changing in response to user interaction and animation. Therefore the hierarchy is also constantly changing, with tiles continuously merged and split in response to dynamic changes in the contents of the deep buffer.

We implement our dynamic spatial hierarchy using a K-D tree. Given a target number of tiles, the tree is managed to ensure that the amount of color variation per unit space-time in each tile is roughly equal: the tile with the most color variation is split and the two tiles with the least summed variation are merged, until all tiles have roughly equal variation. As a result, small tiles are located over buffer regions with significant change or fine spatial detail, while large tiles emerge over static or coarsely detailed regions (Figure 1).

Sampling then becomes a biased, probabilistic process. Since time is not fixed as in framed renderers, we cannot simply iteratively sample the tile with the most variation per unit space-time – in doing so, we would overlook newly emerging motion and detail. At the same time, we cannot leave rendering unbiased and unimproved. Our solution is to sample each tile with equal probability, and select the sampled location within the tile using a uniform distribution. Because tiles vary in size, sampling is biased towards those regions of the image which exhibit high spatial and/or temporal variance. Because all tiles are sampled, we remain sensitive to newly emerging motion and detail.

This sampler is in fact a closed loop control system [7], capable of adapting to user input with great flexibility. In control theory, the *plant* is the process being directed by the *compensator*, which must adapt to external *disturbance*. Output from the plant becomes input for the compensator, closing the feedback loop. In a classic adaptive framed sampler, the compensator chooses the rendered location, the ray tracer is the plant that must be controlled, and disturbance is

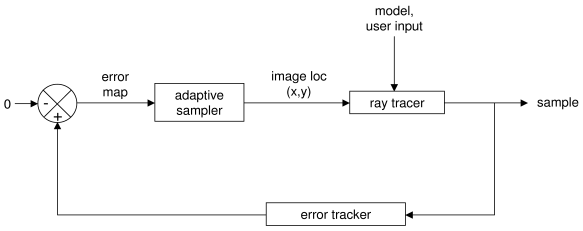

Figure 2: *Adaptive frameless sampling as closed loop control. Output sample from the ray tracer (plant) is sent to an error tracker, which adjusts the spatial tiling or error map. As long as the error map is not zero everywhere, the adaptive sampler (compensator) selects one tile to render, and one location in the tile. Constantly changing user input (disturbance) makes it very difficult to limit error.*

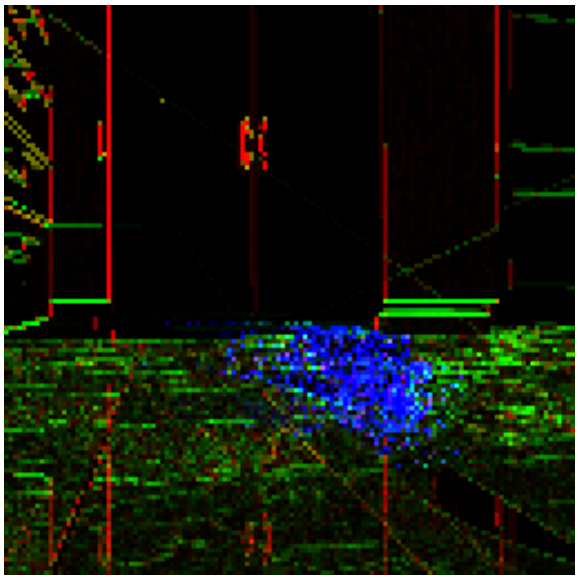

Figure 3: *A snapshot of color gradients in the car scene. Green and red are spatial gradients, blue is the temporal gradient. Here the spatial gradients dominate, and the number of tiles is fairly high.*

provided by the scene as viewed at the time being rendered. Our frameless sampler (Figure 2) faces a more difficult challenge: view and scene state may change after each sample. Unfortunately, a ray tracer is extremely nonlinear and highly multidimensional, and therefore very difficult to analyze using control theoretic techniques.

Nevertheless, more pragmatic control engineering techniques may be applied. One such technique is the use of *PID* controllers, in which control may respond in proportion to error itself (P), to its integral (I), and to its derivative (D). In our sampler, error is color variation – if it were small enough, we could assume that rendering was complete. In biasing sampling toward variation, we are already responding in proportion to error. However, we have also found it useful to respond to error's derivative. By biasing sampling toward regions in which variation is changing, we compensate for delay in our control system and direct sampling toward changing regions of the deep buffer, such as the edges of the car in Figure 1. We accomplish this by tracking variation change d, and adding it to variation itself p to form a new summed control error: $e = kp + (1-k)d$, where k in the range [0,1] is the weight applied to the proportional term. The right image in Figure 1 visualizes d for each tile by mapping high d values to high gray levels.

Our prototype adaptive sampler will be less effective when the rendered scene is more dynamic. In control theory, one way of compensating for varying rates of change in the target signal is to adjust *gain*, thereby damping or am-

plifying the impact of control on the plant. A similar sort of compensation is possible in our sampling control system by restricting or increasing the ability of the sampler to adapt to deep buffer content. We implement this approach by adjusting the number of tiles in the K-D tree according to the ratio of color change over time to color change across space. We achieve this by ensuring that $dC/dsS = dC/dtT$, where dC/ds and dC/dt are color change over space and time (Figure 3), S is the average width of the tiles, and T the average age of the samples in each tile. By solving for S we can derive the current number of tiles that would be appropriate.

4. Space-time reconstruction for interactive rendering

Frameless sampling strategies demand a rethinking of the traditional computer graphics concept of an "image", since at any given moment the samples in an image plane represent many different moments in time. The original frameless work [3] simply displayed the most recent sample at every pixel, a strategy we refer to as *traditional reconstruction* of the frameless sample stream. The result is a noisy, pixelated image which appears to sparkle or scintillate as the underlying scene changes (see Figure 4). Instead we store a temporally deep buffer of recent frameless samples, and continuously reconstruct images for display by convolving the samples with a space-time filter. This is similar to the classic computer graphics problem of reconstruction of an image from non-uniform samples [14], but with a temporal element: since older samples may represent "stale" data, they are treated with less confidence and contribute less to nearby pixels than more recent samples. The resulting images greatly improve over traditional reconstruction (see again Figure 4).

The key question is what shape and size filter to use. A temporally narrow, spatially broad filter (i.e. a filter which falls off rapidly in time but gradually in space) will give very little weight to relatively old samples; such a filter emphasizes the newest samples and leads to a blurry but very current image. Such a filter provides low-latency response to changes and should be used when the underlying image is changing rapidly (Figure 4, right member of leftmost pair). A temporally broad, spatially narrow filter will give nearly as much weight to relatively old samples as to recent samples; such a filter accumulates the results of many samples and leads to a finely detailed, antialiased image when the underlying scene is changing slowly (Figure 4, right member of rightmost pair). However, often different regions of an image change at different rates; for example, in a stationary view in which an object is moving across a static background. A scene such as this demands spatially adaptive reconstruction, in which the filter width varies across the image. What should guide this process?

We use local sampling density and space-time gradient information to guide filter size. The sampler provides an estimate of sampling density for an image region, based on the overall sampling rate and on the tiling used to guide sampling. We size our filter – which can be interpreted as a space-time volume – as if we were reconstructing a regular sampling with this local sampling density, and while preserving the total volume of the filter perturb the filter widths according to local gradient information. We reason that a large spatial gradient implies an edge, which should be resolved with a narrow filter to preserve the underlying high frequencies. Similarly, a large temporal gradient implies a "temporal edge" such as an occlusion event, which should be resolved with a narrow filter to avoid including stale samples from before the event.

What function to use for the filter kernel remains an open question. Signal theory tells us that for a regularly sampled bandlimited function, ideal reconstruction should use a sinc function, but our deep buffer is far from regularly sampled and the underlying signal (an image of a three-dimensional scene) contains high-frequency discontinuities such as occlusion boundaries. We currently use an inverse exponential filter so that the relative contribution of two samples does not change as both grow older; however, the bandpass properties of this filter are less than ideal. We would like to investigate multistage approaches inspired by the classic Mitchell filter [14].

Our implementation of a deep buffer stores the last n samples within each pixel; typical values of n range from 1 to 8. As samples arrive they are bucketed into pixels and added to the deep buffer, displacing the oldest sample in that pixel; average gradient information is also updated incrementally as samples arrive. At display time a reconstruction process adjusts the filter size and widths at each pixel as described (using gradient and local sample density) and gathers samples "outwards" in space and time until the maximum possible incremental contribution of additional samples would be less than some threshold ε ($\varepsilon = 1\%$ in our case). The final color at that pixel is computed as the normalized weighted average of sample colors. This process is expensive – our simulation requires reconstruction times of a few hundred ms for small (256×256) image sizes – so we are investigating several techniques to accelerate reconstruction. One currentlykey technique will be to implement the reconstruction process directly on the graphics hardware, and we have a prototype implementation of a GPU-based reconstructor in which the deep buffer is represented as a texture with samples interleaved in columns; samples are added to the buffer by rendering points with a special pixel shader enabled. At display time the system reconstructs an image by drawing a single screen-sized quad with a (quite elaborate) pixel shader that reads and filters samples from the deep buffer texture. Though not yet fully adaptive, our initial GPU implementation provides a promising speedup (more than an order of magnitude) over the CPU version. We plan to revisit this implementation, which is far from optimized, and hope for another order of magnitude to allow the system to achieve interactive frame rates on realistic image resolutions.

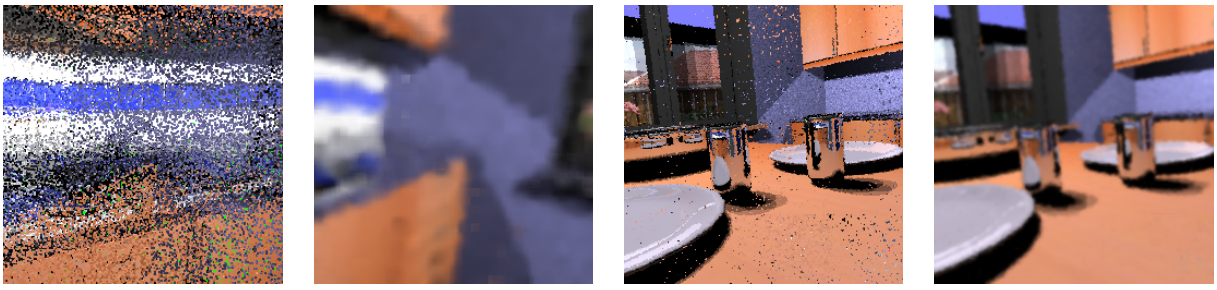

Figure 4: *Adaptive reconstructions. The left pair shows a dynamic scene, with the traditional frameless reconstruction on the left and the adaptive reconstruction on the right. The right pair shows a static scene, with the traditional reconstruction once more on the left and the adaptive reconstruction on the right.*

5. Gold standard evaluation

Using the *gold standard* validation described in [30], we find that our adaptive frameless renderer consistently outperforms other renders that have the same sampling rates.

Gold standard validation uses as its standard an *ideal renderer I* capable of rendering imagery in zero time. To perform comparisons to this standard, we create n ideal images I_j (j in [1,n]) at 60 Hz for a certain animation A using a simulated ideal renderer. We then create n more images R_j for animation A using an actual interactive renderer R. We next compare each image pair (I_j,R_j) using an image comparison metric *comp*, and average the resulting image differences: $1/n\Sigma_1^n comp(I_j,R_j)$. Note that if this comparison metric is root mean squared (RMS) error, this result is very closely related to the peak signal-to-noise ratio (PSNR), a commonly used measure of video quality.

We report the results of our gold standard evaluation using PSNR in Figure 5 below. In the figure, we compare several rendering methods. Two framed renderings either maximize Hz at the cost of spatial resolution (lo-res), or spatial resolution at the cost of Hz (hi-res). The traditional frameless rendering uses a pseudorandom non-adaptive sampler and simply displays the newest sample at a given pixel. The adaptive frameless renderings come in two groups: one that uses a fixed number of tiles (256), and one that uses a variable number of tiles, as determined by balance between spatial and temporal change in the sample stream. In both the fixed and variable groups, there are three biases in response to sampling: biased toward color change itself (k=.8) (P), toward the derivative of color change (k=.2) (D), or balanced. Rendering methods were tested in 3 different animations: the publicly available BART testbed [12]; a closeup of a toy car in the same testbed, and a dynamic interactive recording. All of these animations were rendered at sampling rates of 100K pixels per second, two at 1M pixels per second.

Adaptive frameless rendering is the clear winner, with high PSNRs throughout. In the largely static Toycar stream, it made almost no difference whether adaptive frameless rendering used a fixed or variable number of tiles. But in the other more dynamic streams, the variable number of tiles consistently had a small edge. Responding to the derivative of color change was slightly more effective when the scene was static, but the pattern here was less clear. In all cases however, adaptive frameless rendering was better than framed or traditional frameless rendering.

A quick glance at Figure 6 confirms this impression. These graphs show frame by frame comparisons using RMS between many of these rendering techniques and the ideal rendering. Adaptive frameless rendering is the consistent winner with traditional frameless and framed renderings almost always with more error. The lone exception is in the BART 100K stream, when the animation begins with a very dynamic content and uses sampling rate 20x too slow to support 60Hz at full resolution.

In addition to evaluating the ability of our entire adaptive renderer to approximate a hypothetical ideal renderer, we can evaluate the ability of our reconstruction sub-system to approximate a hypothetical ideal reconstructor. This *sample oracle* evaluation generates crucial and detailed feedback for the development of both the reconstruction sub-system and by extension, the entire rendering system. Sample oracle evaluation is identical to gold standard evaluation in all respects save one: rather than comparing the reconstructed interactive image R_j to a corresponding ideal image I_j, it

Render Method	Animation / Sampling Rate				
	Toycar 100K	Toycar 1M	BART 100K	BART 1M	Dynamic 100K
Framed: lo-res	11.26	15.82	10.16	12.69	10.35
Framed: hi-res	9.01	15.38	4.58	7.51	7.54
Traditional frameless	16.71	18.07	6.31	9.97	10.54
Adaptive: fixed (k=0.2)	18.11	19	10.35	15.08	13.95
Adaptive: fixed (k=0.5)	18.01	18.99	10.4	15.39	14.17
Adaptive: fixed (k=0.8)	17.86	18.93	10.21	15.58	14.13
Adaptive: var (k=0.2)	18.15	18.97	10.86	15.07	14.04
Adaptive: var (k=0.5)	18.01	18.96	11	15.42	14.31
Adaptive: var (k=0.8)	17.85	18.92	10.95	15.7	14.36

Figure 5: *A comparison of several rendering techniques to an ideal rendering using peak signal to noise ratio (PSNR), across three animations and two sampling rates.*

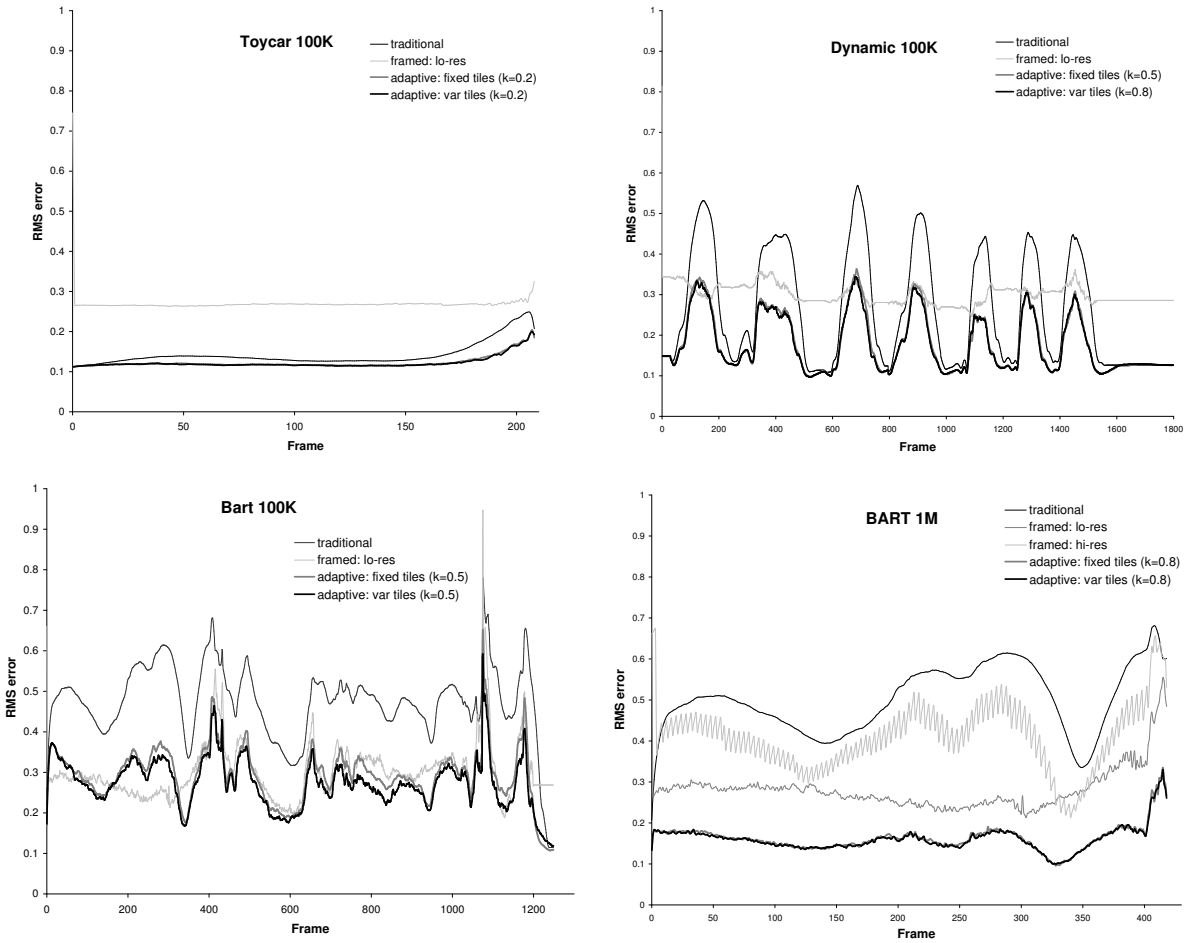

Figure 6: *A more detailed peak at the frame-by-frame root mean squared (RMS) error for some of the rendering techniques and streams from the above table.*

will compare R_j to a corresponding oracle image O_j that was ideally reconstructed from the sample stream used to produce R_j. O_j will be approximated by first eliminating any samples in the deep buffer that are more than a distance of ε_o (e.g. 10) in the perceptual color space *Lab* from the nearest pixel in I_j, leaving only those samples that haven't become "stale" due to occlusions, lighting changes, etc. These samples then become input for a sparse sample reconstructor, whose output is the O_j approximation. A sample of this evaluation is shown in 7.

6. Ongoing and Future Work

6.1. Sampling

As does any closed loop controller, our sampler will require extensive tuning. Several parameters are currently set based on empirical estimates from inadequate testing. Among these are the parameter k above, which controls the relative importance of the proportional and derivative terms in guiding sampling bias. Results indicate that the derivative term is useful when the sampling rate is low, and the scene fairly static. It may also prove useful to introduce an asymmetry in gain control that favors color change across space, or over time. This might take the form of a factor w in [0,1]: $dC/dsS = wdC/dtT$. Adjustments to the spatial hierarchy do not occur after every sample, but rather after every c samples (these c samples form a *chunk*). Currently c has rather arbitrarily been set at 100. Other parameters that must be tuned include the coefficients of the function used to assign weights to samples used in calculating variance, the minimum variance possible in each tile (variance never falls to zero, so tiles are always monitored for change), and the temporal depth of the deep buffer. We plan to study the optimal values for and interactions between these parameters by systematically varying them, recording the images that each parameter configuration produces both with and without re-

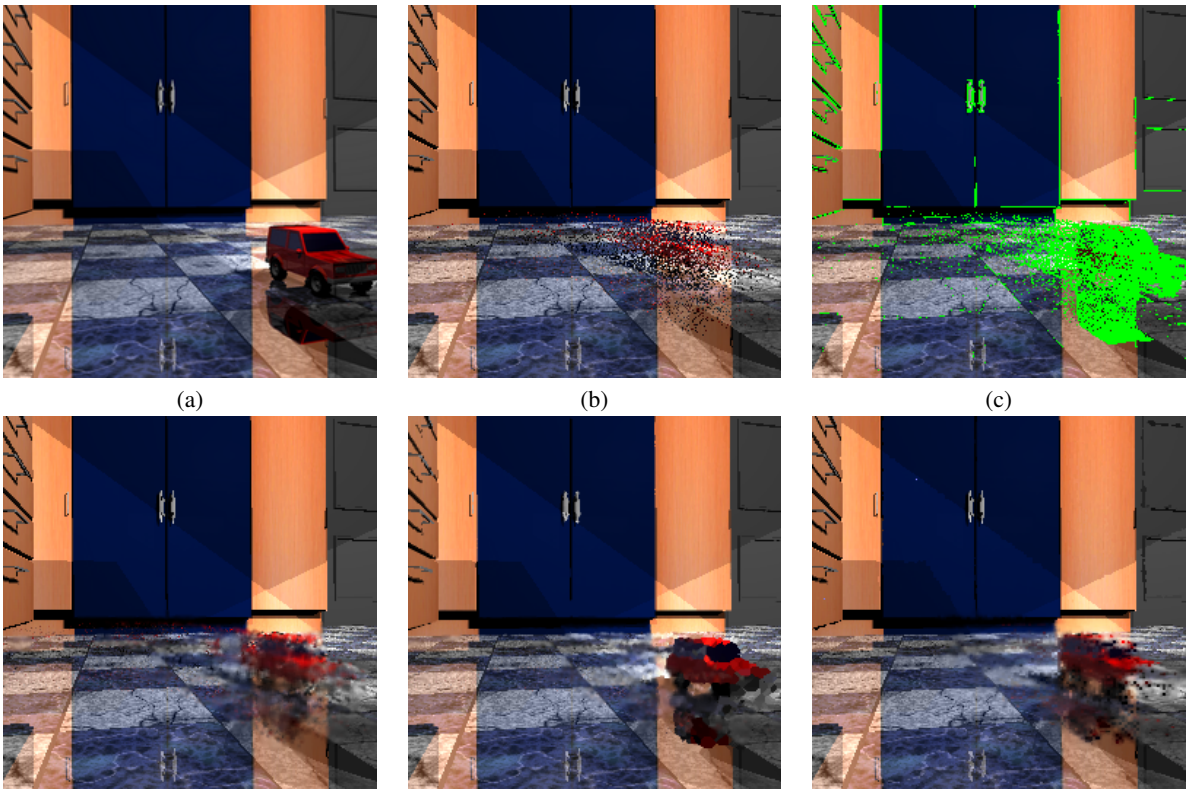

Figure 7: *Sample oracle evaluation of reconstruction. (a) The "oracle image" is generated with zero delay and very high supersampling to simulate the image that would be created by a hypothetical perfect sampler in response to instantaneous input. (b) shows an actual sample stream, drawn here with traditional reconstruction. (c) shows in green those samples that are invalid if compared to the oracle image (we compared to a threshold $dE* = 20$ in the CIELab color space, which is intended to provide a perceptually linear color distance metric). Most of these invalid pixels are due to temporal edges from the moving car. An ideal reconstructor would not include these samples at all. In practice, of course, our reconstructor does does not have access to the oracle image at run-time and thus does include these samples. The resulting image is shown in (d). In (e) we see the higher quality that an ideal reconstructor would be able to achieve by not including the samples. We believe that we should be able to approach the quality of (e) by employing more sophisticated reconstruction strategies, for example using edge-finding techniques based on the trilateral filtering approach of [5]. For example, (f) shows a preliminary implementation of bilateral filtering using the Lorentz filter [6].*

construction, and then evaluating each configuration using gold standard validation.

The methods we use to build the spatial hierarchy are currently quite simple, and could certainly be improved. Tiles are split with planes oriented not according to deep buffer content, but according to the level of the K-D tree (against X at one level, against Y at the next). Planes might instead be oriented against the prevailing direction of color change, found using methods adapted from image processing. Not only orientation but position of the splitting planes might be controlled to minimize variation in each child tile using the quadric techniques described by [8]. We might even use knowledge about scene geometry as it currently projects into the deep buffer to guide splitting. Any improvements these techniques bring in splitting would have to be balanced against the competing demands of algorithmic simplicity and efficiency: simpler algorithms are much easier to embed into hardware, while efficient algorithms introduce less delay (often an overriding concern) into the control system.

6.2. Reconstruction

While our results with reconstruction of temporally deep buffers are encouraging, and certainly improve over the traditional frameless reconstruction strategy of simply replacing pixels as new samples arrive, many avenues of further

research remain. To begin with, we plan to acclerate the reconstruction algorithm using programmable graphics hardware. More fundamentally, we need to continue to improve the automatic adaptation of filter shape to the underlying signal. We have begun investigating applying concepts from bilateral and trilateral filtering [6] [5]. These techniques perform nonlinear filtering by weighting samples according to their distance in luminance as well as in space. The bilateral filtering approach, which is related to anisotropic diffusion, tends to preserve edges; this behavior is enhanced by only considering samples within a certain "window", possibly asymmetric, about the reconstructed pixel. A number of parameters control the selection of this window and the size of the spatial and luminance filters. Trilateral filtering effectively supplies these parameters automatically by using another bilateral filtering step in the gradient domain. Incorporating these ideas into our reconstruction algorithm is non-trivial: we must extending the bilateral/trilateral filtering concept to include a third temporal dimension and to operate on a nonuniformly sampled domain. In particular, the core concept of bilateral filtering at a pixel P is to compare the luminance of the sample at P to neighboring samples; however when reconstructing an image from the deep buffer, there may be no recent sample at that pixel. In our experiments to date we use instead the nearest sample in (x, y, t) for comparison; Figure 7 illustrates our preliminary results.

Bilateral/trilateral filtering can be thought of as filtering an image in a way that surmises the location of edges, using differences in luminance and the gradientWe domain, and avoids filtering samples across edges. Bala and co-workers have demonstrated the dramatic improvement possible in reconstruction of sparse samples by incorporating a priori knowledge of image discontinuities, such as object silhouettes and shadow boundaries [1]. We also plan to explore how we can exploit such a priori information in our application. Of course, we are concerned not only with spatial edges but also with temporal edges, such as occlusion events. When an occlusion event occurs in a region of the image, all samples in that region from prior to the occlusion are no longer valid and should be discarded. We will investigate ways to detect or surmise this sort of situation, in the reconstructor alone or with the assistance of the sampler. For example, if the geometry of the scene is known occlusion events might be detected a priori (computational geometers know this to be a difficult problem in its full generality, but we can perhaps simplify it for our purposes). A less demanding approach would simply have the sampler track object IDs at each pixel and flag samples where the underlying object appears to have changed; the trick is to avoid confounding spatial variation sampled over time with temporal variation. Given some knowledge or estimates of edges, we must encode this information so as to enable efficient reconstruction, perhaps extending concepts such as the *edge-point-image* (EPI) of Bala *et al.* to include temporal edges.

Finally, we plan to explore how our purely image-based reconstruction approach interacts with sample reprojection techniques such as the Render Cache [28] [27] and the Shading Cache [22]. These techniques employ three-dimensional knowledge of the scene to reuse samples from previous frames as the viewpoint moves, reprojecting samples to the appropriate screen-space location for the new viewpoint. While sample reprojection comes with its own costs and is not always appropriate (for example, for deforming or highly specular objects), it is an attractive strategy that dovetails well with frameless sampling. However, we need to consider how reprojection meshes with our temporal weighting scheme: are properly reprojected samples less "stale" than non-projected samples? Should their weighting be a function of material properties (e.g. specularity) as well as age? Do all samples need to be reprojected for each reconstruction, or would it suffice to reproject only some samples? What is the right way to combine static and reprojected samples?

6.3. Global illumination

One important research avenue we plan to explore is how to extend our temporally aware sampling and reconstruction ideas beyond simple ray tracing to more sophisticated global illumination algorithms. For example, our scheme would seem to apply naturally to path tracing: each path traced simply becomes another sample, and the reconstruction from our temporally deep buffer accumulates the effects of many paths to produce the appropriate color at every pixel. However, this straightforward extension suffers from an ambiguity: different samples (paths traced) at the same image location may have very different color and brightness. How can we disambiguate differences due to stochastic sampling of BRDF and light field from differences due to temporal edges? One possibility might involve re-tracing some previous paths to detect temporal discontinuities. We also expect path tracing to emphasize the importance of sample reprojection. Finally, we plan to investigate more fundamental ways of restructuring global illumination computations for interactive rendering using our ideas. For example, we envision a frameless version of Jensen's *photon map* structure [11] for efficient computation and representation of indirect illumination. Such a frameless photon map would be updated continuously and stochastically with photons shot from the light sources using spatially and temporally adaptive sampling strategies, and would be temporally deep, with a reconstruction that produces low-resolution illumination emphasizing recent photons where the scene or lighting is changing, and high-resolution illumination including old photons in static regions.

6.4. Perceptually driven rendering

With its ability to alter sampling and reconstruction across both space and time, our adaptive frameless renderer is an ideal platform for implementing and experimenting with

perceptually driven rendering in interactive settings. There are several obvious opportunities that might improve results. First, we might incorporate the effects of early stages in visual processing using the contrast sensitivity function (CSF), which describes threshold visual sensitivity to contrast at various spatial frequencies. The CSF could modulate the error calculations that bias sampling, or shape reconstruction filters. Sensitivity to fine detail drops as viewed objects increase their retinal velocity, or move into the visual periphery [13]. Our renderer could exploit these characteristics of our visual system by decreasing spatial sampling density and increasing spatial filter widths in our visual periphery, and by emphasizing temporal over spatial sampling density and narrowing temporal filter widths where objects are in motion. Finally, we might incorporate models of attention [31] into our renderer, reducing sampling density and increasing filter widths in regions of the scene that the model predicts will be uninteresting to the viewer.

6.5. Display hardware

A long-term avenue for future research would be to develop principles of new display hardware based on streams of samples rather than on raster-order image frames. Currently, reconstruction takes place in the graphics hardware at every display refresh, turning our frameless rendering scheme into framed images to meet the needs of today's display hardware. We would ideally like to move reconstruction into the display hardware itself, reducing the task of the graphics hardware to producing samples (and perhaps reprojecting old samples). One can imagine future displays or "smart framebuffers" in which the pixels operate like a sort of systolic array through which incoming samples diffuse and compete. Such a concept may seem quite foreign, accustomed as we are to notion of framebuffers that contain a grid of pixels continuously scanned out to a monitor in raster order; however, future display technology may not be as tied (or suited) to the raster scan-out developed for cathode ray tubes over fifty years ago. For example, organic LED technology lends itself to in-pixel circuitry such as our hypothesized smart framebuffers; printable displays (based on "electronic ink") are poorly suited for continuous raster update because of their slow refresh, and digital micromirror device (DMD) projectors are microelectromechanical systems (MEMS) which do not intrinsically depend on scanning; in principle one can individually address each of the microscopic vibrating mirrors that form the picture elements. A frameless, asynchronous approach to rendering could prove to be the key ingredient for realizing the potential of next-generation display technology.

The prospect of moving reconstruction into the display highlights one benefit of our approach: it enables asynchronous parallel graphics. Current parallel graphics architectures such as *sort-first*, *sort-middle*, and *sort-last* [15] are built around the common constraint that the results from all parallel rendering nodes must ultimately be combined into a single frame. If the display performed temporally adaptive reconstruction on incoming stream samples, the nodes could simply be responsible for producing samples as fast as possible and sending those samples to the display. By breaking the frame, we also break an underlying assumption and restrictive constraint in graphics architecture.

7. Conclusion

In conclusion, we advocate a new approach to interactive rendering, based on temporally adaptive sampling and reconstruction, and enabled by recent advances in interactive ray tracing. This approach improves traditional framed and frameless rendering by focusing sampling on regions of spatial and temporal change, and by performing adaptive reconstruction that emphasizes new samples when things are changing quickly and incorporates older samples for high spatial detail when things are static. Samples are streamed through a temporally deep buffer from which displayed images are reconstructed. We use a "gold standard" comparison to ideal renderings to argue that the resulting system displays greater accuracy than framed and frameless rendering schemes at comparable sampling rates. Based on these results, we believe that a temporally adaptive frameless approach shows great promise for future rendering algorithms and display hardware.

8. Acknowledgements

We would like to thank Edward Colgate and Kevin Lynch (Department of Mechanical Engineering, Northwestern University) for their fruitful discussions on control systems, the Saarland University Graphics Group for providing us with the OpenRT ray tracer and Peter Lindstrom (Georgia Tech.) for providing the ltdiff metric to measure perceptual image differences. This work was supported in part by the NSF grants 0092973, 0093172, 0112937, and 0130869. The 3D model(kitchen) is the courtesy of the BART (A Benchmark for Animated Ray Tracing) project at Chalmers University of Technology, Sweden. Finally we would like to extend our thanks to ATI and nVidia for their hardware support.

References

[1] K. Bala, B. Walter, and D. P. Greenberg. Combining edges and points for interactive high-quality rendering. *ACM Trans. Graph.*, 22(3):631–640, 2003. 9

[2] L. Bergman, H. Fuchs, E. Grant, and S. Spach. Image rendering by adaptive refinement. In *Proceedings of the 13th annual conference on Computer graphics and interactive techniques*, pages 29–37. ACM Press, 1986. 2, 3

[3] G. Bishop, H. Fuchs, L. McMillan, and E. J. Scher Zagier. Frameless rendering: Double buffering consid-

ered harmful. *Computer Graphics*, 28(Annual Conference Series):175–176, 1994. 2, 3, 5

[4] N. A. Carr, J. D. Hall, and J. C. Hart. The ray engine. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 37–46. Eurographics Association, 2002. 3

[5] P. Choudhury and J. Tumblin. The trilateral filter for high contrast images and meshes. In *Proceedings of the 14th Eurographics workshop on Rendering*, pages 186–196. Eurographics Association, 2003. 8, 9

[6] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 257–266. ACM Press, 2002. 8, 9

[7] K. Dutton, S. Thompson, and B. Barrachlough. *The Art of Control Engineering*. Addison-Wesley Pub Co, first edition, 1997. 4

[8] M. Garland and P. S. Heckbert. Surface simplification using quadric error metrics. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, pages 209–216. ACM Press/Addison-Wesley Publishing Co., 1997. 8

[9] A. Glassner. *Principles of Digital Image Synthesis*. Morgan Kaufmann, first edition, 1995. 3

[10] G. Humphreys, M. Houston, R. Ng, R. Frank, S. Ahern, P. D. Kirchner, and J. T. Klosowski. Chromium: a stream-processing framework for interactive rendering on clusters. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 693–702. ACM Press, 2002. 3

[11] H. W. Jensen. *Realistic Image Synthesis Using Photon Mapping*. AK Peters, 2001. 9

[12] J. Lext, U. Assarsson, and T. MÃűller. Bart: A benchmark for animated ray tracing. Technical report, Department of Computer Engineering, Chalmers University of Technology,Parametric Technology Corporation, 2000. http://www.ce.chalmers.se/BART. 6

[13] D. Luebke, M. Reddy, J. D. Cohen, A. Varshney, B. Watson, and R. Huebner. *Level of Detail for 3D Graphics*. Morgan Kaufmann, first edition, 2002. 10

[14] D. P. Mitchell. Generating antialiased images at low sampling densities. In *Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 65–72. ACM Press, 1987. 2, 3, 5

[15] S. Molnar, M. Cox, D. Ellsworth, and H. Fuchs. A sorting classification of parallel rendering. *IEEE Computer Graphics and Applications*, 14(4):23–32, July 1994. 10

[16] J. Painter and K. Sloan. Antialiased ray tracing by adaptive progressive refinement. In *Proceedings of the 16th annual conference on Computer graphics and interactive techniques*, pages 281–288. ACM Press, 1989. 2, 3

[17] S. Parker, W. Martin, P.-P. J. Sloan, P. Shirley, B. Smits, and C. Hansen. Interactive ray tracing. In *Proceedings of the 1999 symposium on Interactive 3D graphics*, pages 119–126. ACM Press, 1999. 2

[18] T. J. Purcell, I. Buck, W. R. Mark, and P. Hanrahan. Ray tracing on programmable graphics hardware. *ACM Transactions on Graphics*, 21(3):703–712, July 2002. ISSN 0730-0301 (Proceedings of ACM SIGGRAPH 2002). 1, 3

[19] T. J. Purcell, C. Donner, M. Cammarano, H. W. Jensen, and P. Hanrahan. Photon mapping on programmable graphics hardware. In *Proceedings of the ACM SIGGRAPH/EUROGRAPHICS conference on Graphics hardware*, pages 41–50. Eurographics Association, 2003. 1

[20] M. J. P. Regan, G. S. P. Miller, S. M. Rubin, and C. Kogelnik. A real-time low-latency hardware light-field renderer. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 287–290. ACM Press/Addison-Wesley Publishing Co., 1999. 3

[21] G. Stoll, M. Eldridge, D. Patterson, A. Webb, S. Berman, R. Levy, C. Caywood, M. Taveira, S. Hunt, and P. Hanrahan. Lightning-2: a high-performance display subsystem for pc clusters. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 141–148. ACM Press, 2001. 3

[22] P. Tole, F. Pellacini, B. Walter, and D. P. Greenberg. Interactive global illumination in dynamic scenes. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 537–546. ACM Press, 2002. 9

[23] I. Wald, C. Benthin, A. Dietrich, and P. Slusallek. Interactive distributed ray tracing on commodity pc clusters – state of the art and practical applications. *Lecture Notes on Computer Science*, 2790:499–508, 2003. Proceedings of EuroPar 2003. 1

[24] I. Wald, C. Benthin, M. Wagner, and P. Slusallek. Interactive rendering with coherent ray tracing. In A. Chalmers and T.-M. Rhyne, editors, *Computer Graphics Forum (Proceedings of EUROGRAPHICS 2001*, volume 20, pages 153–164. Blackwell Publishers, Oxford, 2001. available at http://graphics.cs.uni-sb.de/ wald/Publications. 3

[25] I. Wald, T. J. Purcell, J. Schmittler, C. Benthin, and

P. Slusallek. Realtime ray tracing and its use for interactive global illumination. In *Eurographics State of the Art Reports*, 2003. 1, 3

[26] I. Wald, P. Slusallek, and C. Benthin. Interactive distributed ray tracing of highly complex models. In S.J.Gortler and K.Myszkowski, editors, *Rendering Techniques 2001 (Proceedings of the 12th EUROGRAPHICS Workshop on Rendering*, pages 277–288. Springer, 2001. available at http://graphics.cs.uni-sb.de/ wald/Publications. 2

[27] B. Walter, G. Drettakis, and D. P. Greenberg. Enhancing and optimizing the render cache. In *Proceedings of the 13th Eurographics workshop on Rendering*, pages 37–42. Eurographics Association, 2002. 9

[28] B. Walter, G. Drettakis, and S. Parker. Interactive rendering using render cache. In *Proceedings of the 10th Eurographics workshop on Rendering*, pages 19–30. Springer Wien, 1999. 9

[29] C. Wickens. *The Effects of Control Dynamics on Performance*, volume II. K. Boff, L. Kaufmann and J. Thomas. New York, NY, Wiley, 1986. 3

[30] C. Woolley, D. Luebke, B. Watson, and A. Dayal. Interruptible rendering. In *Proceedings of the 2003 symposium on Interactive 3D graphics*, pages 143–151. ACM Press, 2003. 3, 6

[31] H. Yee, S. Pattanaik, and D. P. Greenberg. Spatiotemporal sensitivity and visual attention for efficient rendering of dynamic environments. *ACM Trans. Graph.*, 20(1):39–65, 2001. 10