Eurographics Symposium on Parallel Graphics and Visualization (2009)
J. Comba, K. Debattista, and D. Weiskopf (Editors)

Time-constrained High-fidelity Rendering on Local Desktop Grids

Vibhor Aggarwal[†], Kurt Debattista, Piotr Dubla, Thomas Bashford-Rogers and Alan Chalmers

The Digital Lab, University of Warwick, UK
[†]Vibhor.Aggarwal@warwick.ac.uk

Abstract

Parallel computing has been frequently used for reducing the rendering time of high-fidelity images, since the generation of such images has a high computational cost. Numerous algorithms have been proposed for parallel rendering but they primarily focus on utilising shared memory machines or dedicated distributed clusters. A local desktop grid, composed of arbitrary computational resources connected to a network such as those in a lab or an enterprise, provides an inexpensive alternative to dedicated clusters. The computational power offered by such a desktop grid is time-variant as the resources are not dedicated. This paper presents fault-tolerant algorithms for rendering high-fidelity images on a desktop grid within a given time-constraint. Due to the dynamic nature of resources, the task assignment does not rely on subdividing the image into tiles. Instead, a progressive approach is used that encompasses aspects of the entire image for each task and ensures that the time-constraints are met. Traditional reconstruction techniques are used to calculate the missing data. This approach is designed to avoid redundancy to maintain time-constraints. As a further enhancement, the algorithm decomposes the computation into components representing different tasks to achieve better visual quality considering the time-constraint and variable resources. This paper illustrates how the component-based approach maintains a better visual fidelity considering a given time-constraint while making use of volatile computational resources.

Categories and Subject Descriptors (according to ACM CCS): Computer Graphics [I.3.1]: Hardware Architecture—Parallel processing; Computer Graphics [I.3.7]: Three-Dimensional Graphics and Realism—Ray tracing; Computer Graphics [I.3.2]: Graphics Systems—Distributed/network graphics;

1. Introduction

Nowadays, high-fidelity rendering forms an integral part of areas such as product visualisation, archaeological reconstructions, architectural walk-throughs and movie special effects. These renderings are usually produced by solving the rendering equation [Kaj86] using a Monte Carlo simulation. This process is computationally intensive; hence parallel computing has been often employed to cut down the rendering time. Many parallel rendering algorithms have been devised exploiting either data parallelism by decomposing the computations in object space or task parallelism by decomposing in image space [CDR02]. These algorithms have been targeted at dedicated resources such as shared memory machines or dedicated clusters known as render farms. In contrast, local desktop grids provide a cheaper option to these expensive resources.

Desktop grids are based on the concept of harnessing idle CPU cycles of a desktop PC by cycle stealing. The origins of this idea can be traced back to the PARC Worm [SH82] and since then it has been effectively used by various BOINC projects [And04, ACK*02], GIMPS [GIM] and others to solve complex scientific problems. A desktop grid is formed by multiple resources which connect to a network without offering any guarantee of service. As these resources are undedicated, computational power offered by these desktop grids is volatile. A local desktop grid infrastructure is built by using institutional resources such as those in a lab or an enterprise. With the advent of multi-core CPUs, running a

DOI: 10.2312/EGPGV/EGPGV09/103-110

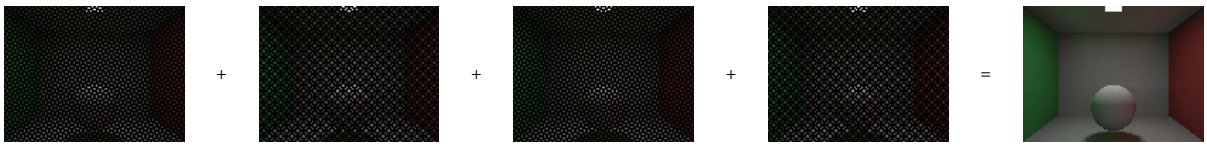


Figure 1: *Image of the Cornell Box divided into 4 groups of pseudo-randomly chosen pixels. Each of these groups can be independently computed in parallel. In practice, the image is divided into many more groups. Please note that the resolution of 256×192 used in these images is for illustrative purposes only.*

local desktop grid has become more viable as some of the cores may not be in constant use.

Computational problems which can be modelled as bag-of-tasks applications are suitable for execution on desktop grids as they can be broken into smaller independent subtasks. These subtasks can be executed in parallel on any of the available resources of a desktop grid using a task pull-model whereby any idle resource pulls off a task from a central server. Volatility of resources means that redundancy is needed for providing fault-tolerance. Ray tracing [Whi80] easily lends itself to this model as computation of one pixel is completely independent of any other pixel. However, scheduling each pixel as a different task would incur a high communication overhead and therefore pixels are generally grouped together in the form of image tiles when parallel computing is employed.

The use of time-constraints is an elegant way of employing changeable resources. However, such an approach introduces many challenges. Redundancy needs to be avoided while trying to meet a given time-constraint to obtain a better visual fidelity. A straightforward solution for adapting the ray tracing algorithm for computation on desktop grids is to group pixels chosen pseudo-randomly [Sob94, vdC35] over the complete image space instead of using tiles (see Figure 1). If a task fails to complete, image reconstruction techniques can be used to fill in the missing data. It would be difficult to reconstruct an image if a tile of pixels is grouped together as a task instead, as there would be holes in the image when redundancy is not used and a task would fail . By using image reconstruction algorithms, redundancy can be avoided while maintaining a time-constraint.

Generation of high-fidelity images using ray tracing requires multiple per pixel computations to account for global illumination effects. This paper presents a further enhancement to the novel way of using time-variant resources such as a local desktop grid, by breaking down these per pixel computations into components. This enables better visual quality to be achieved within a user defined time-constraint. In this component-based algorithm, direct and indirect light calculations are separated into different tasks. Furthermore, each task refines the solution progressively and consists of a set of pseudo-randomly chosen pixels in order to avoid redundancy as explained above.

This paper is organised as follows: Section 2 presents some previous work in the field of parallel rendering, time-constrained rendering and desktop grids. Section 3 discusses the details of the straightforward approach and the component-based algorithm. Section 4 contains the results comparing the component-based algorithm with the straightforward approach. The conclusion and future work are presented in Section 5.

2. Related Work

2.1. Parallel Rendering

The large computational complexity exhibited by high-fidelity rendering has encouraged researchers to tackle the rendering problem with parallel computing. A detailed survey of these techniques is presented by Chalmers et al. [CDR02]. Distributed ray tracing [CPC84] is relatively easy to parallelise if the entire scene description can be duplicated on each processor, as each processor can be designated to independently work on a part of the image space. However, load balancing remains a challenging issue. Heirich et al. [HA98] provided a good comparison of various load balancing techniques and showed that any static task subdivision strategy is affected by load imbalances. Badouel et al. [BP90] described a dynamic demand-driven load balancing based on the master-worker paradigm where by each worker is assigned a 3×3 tile of pixels when it becomes idle. But this approach suffered from scalability and therefore, Green et al. [GP90] used a hierarchy of masters to counter this. Reisman et al. [RGS00] devised a dynamic load balancing strategy for progressive ray tracing on distributed clusters.

One of the first parallel architectures designed for interactive ray tracing using 96 processors and shared memory was proposed by Muuss et al. [Muu95]. Another such volume rendering system was proposed by Parker et al. [PMS*99] using an optimised static load balancing approach and it supported image based rendering with realistic shadows. The disadvantage of these approaches is that they used expensive supercomputers. Wald et al. [WSBW01] enhanced the methods presented in [PMS*99] by using a an object dataflow strategy to run on a distributed cluster. Using a highly optimised SIMD code they were able to achieve interactivity. Wald et al. [WKB*02, BWS03] further enhanced this

by adding global illumination and handling a few dynamic scene changes by modifying instant radiosity [Kel97].

The previous approaches for parallel rendering have relied on utilising expensive dedicated equipment and have not looked at the issues of time-constraints, fault-tolerance and load balancing while computing in parallel on time-variant resources such as a desktop grid.

2.2. Time-constrained Rendering

An important characteristic of rendering on changeable resources is to obtain highest possible quality within a given time-constraint. Funkhouser et al. [FS93] devised a mechanism for predicting the level of detail and the shader to be used for maintaining constant frame-rates by using a greedy algorithm. Using a decision theoretic framework and flexible level of detail, real-time rendering was achieved by Horovitz et al. [HL97]. Dumont et al. [DPF03] added perceptual guidance to the decision theoretic framework for hardware-based global illumination. Debattista et al. [DSSC05] provided a framework for controlling the pixel quality in a time-constrained setting for generation of high-fidelity images without perceivable difference.

2.3. Desktop Grids

The desktop grid is an emerging computing infrastructure providing a much cheaper alternative than current supercomputers, whilst still offering significant computational power. The current average throughput of the BOINC projects is 1.20 PetaFLOPS [AR09], which is more than 1.10 PetaFLOPS achieved by the current fastest supercomputer [TOP]. The desktop grid is an effective computing resource even at an institutional level and SZTAKI Desktop Grid framework [KPK06] is one such example. Bag-of-tasks applications are best suited to run on the desktop grids where communication between parallel processes is a major bottleneck for tightly-coupled applications as advised by Cirne et al. [CBS*03].

Desktop grid has been also known as volunteer computing, as the resource owners offer idle CPU cycles without any guarantee of service. This poses a big challenge when computing on desktop grids as the resources are changeable and the algorithms need to be able to cope with faults. One of the two primary fault-tolerance mechanisms, redundancy [ZZH*04] or checkpointing-and-restarting [DAS06] or a hybrid [AC05] of the two may be employed for masking the volatility of resources. However, when rendering with time-variant resources under time-constraints, image reconstruction techniques can be applied to cope with faults if an intelligent image sampling strategy is used. Previous algorithms for rendering on a computational grid can be found in [ACD08,CSL06], but these rely on the fault-tolerance provided by the grid middleware to tackle any faults that may arise. A time-critical visualisation method for grid computing presented by Gao et al. [GLH*08] also relied on redundancy.

3. Time-constrained Fault-tolerant Parallel Rendering Algorithms

In this section, details of two novel methods for time-constrained fault-tolerant parallel rendering are discussed. The goal of the presented algorithms is to break down the rendering computations into subtasks in an elegant manner such that each discrete subtask refines the solution progressively while taking care that in the case where one or more of them fail to complete, the missing data can be reconstructed. The description of various attributes that help the presented algorithms to achieve this objective are addressed in the following subsections.

3.1. Straightforward Approach

For rendering high-fidelity images, several computations are needed per pixel. As explained earlier in Section 1, these per pixel computations can be carried out in parallel, independently of one another. The idea behind this algorithm is to exploit this inherent parallelism of the rendering computations by utilising changeable resources.

3.1.1. Task Subdivision and Fault-tolerance

While rendering in parallel, an image is traditionally divided into groups of neighbouring pixels which are computed independently on parallel processors. The disadvantage of using such an approach while computing on variable resources such as a desktop grid is that if a task fails, image reconstruction techniques would not be able to approximate the missing data as there would be gaps in the image. Duplicate tasks would need to be scheduled in order to cope with faults, as is done in most applications which provide fault-tolerance using redundancy. However, the presented algorithms avoid such redundancy by subdividing the image into sets of pixels using a pseudo-random sampling strategy (see Figure 1) proposed by Sobol [Sob94] instead of using tiling. These groups of pixels are scheduled as different tasks which are completed in parallel. A well known image reconstruction technique, nearest-neighbour reconstruction, is then used to fill in the missing data. In this technique, data from the pixel nearest to the missing pixel is used for estimating the colour value. A single buffer is stored at the master to accumulate the light samples and if some pixels are missing then this buffer is reconstructed to obtain the final result.

3.1.2. Task Scheduling and Load Balancing

A master-worker paradigm is used for rendering in parallel. All the tasks to be completed are kept in a queue on the master. This is a pull-model where each idle worker sends a request to ask for work from the master. The master then

assigns it a task from the front of the task queue. Using fine granularity, this demand-driven dynamic task scheduling maintains a well-balanced load.

3.1.3. Time-constraint

The master keeps track of the time and whenever it assigns a task to the worker for the first time, it sends it a timestamp for the deadline. The worker checks if this timestamp is about to expire, while rendering the image. At a small Δt time before the timestamp is about to expire, it sends back the data for all the pixels for which the computations have been completed. The value of Δt is chosen such that the results from the worker would reach the master before the time-constraint expires. The master discards any results received after the deadline and stops scheduling any new tasks after the deadline has expired. The master then uses reconstruction methods, if need be, to generate the final image. The reconstruction is done in real-time using a commodity Graphics Processing Unit.

3.2. Component-based Algorithm

The motivation for this algorithm is to be able to achieve better visual quality by limiting the reconstruction noise. The division of the computations at a pixel level allows a finer granularity meaning more pixels can be scheduled per job. Therefore, with each completed job in a given time-constraint more information is obtained, reducing the dependency on image reconstruction techniques. The lighting calculations at a pixel level are subdivided into components.

The radiance at a point p in direction -Θ is given by the rendering equation [Kaj86]:

$$L(p \rightarrow \Theta) =$$

$$L_e(p \rightarrow \Theta) + \int_{\Omega_p} f_r(p, \Theta \leftrightarrow \Psi) \cos(N_p, \Psi) L(p \leftarrow \Psi) \delta_{\omega_\Psi}$$

For a specific direction Ψ_i,

$$L_i(p \rightarrow \Theta) =$$

$$\int_{\Omega_p} f_r(p, \Theta \leftrightarrow \Psi_i) \cos(N_p, \Psi_i) L(p \leftarrow \Psi_i) \delta_{\omega_\Psi}$$

Traditionally, it is common to subdivide, the computation into direct(L_d) and indirect(L_{id}) computations [SSH*98, DSSC05], using Ψ_d to refer to the direction of the direct contribution of the light and Ψ_{id} for the indirect contribution:

$$L(p \rightarrow \Theta) = L_e(p \rightarrow \Theta) + L_d(p \rightarrow \Theta) + L_{id}(p \rightarrow \Theta)$$

Furthermore, the indirect computations can be broken into separate components such as indirect diffuse, indirect specular and indirect glossy. The component-based approach divides the light components into indirect diffuse which shall be referred to as indirect lighting and all the other non-indirect diffuse components are grouped together which shall be referred to as direct lighting.

3.2.1. Component-based Task Subdivision

Task subdivision in the straightforward approach is further enhanced by the component-based approach, by subdividing computations into components on a per pixel level as well. This is in addition to image space subdivision employed in the straightforward approach as explained in Section 3.1.1. The lighting calculations are primarily broken into two components, namely direct and indirect light calculations. The direct light is approximated by ray tracing the scene using different samples on area light sources. This computation can be split such that a group of tasks generating the complete image, samples a different point on the area light source. The direct lighting can then be obtained by averaging results from these tasks. The estimation of indirect lighting is done by sampling Virtual Point Lights (VPLs) [Kel97]. Light paths are traced by shooting rays from the light source and the VPLs are created at the points where these intersect with the scene. These VPLs are then sampled with visibility rays to obtain the indirect lighting. If α VPLs are to be sampled for generating the image, they can be grouped into β sets where each set contains α / β VPLs. Each group of tasks generating the whole image can then sample one of the β subsets independently of the other groups. The images generated by the different groups of tasks, then need to be averaged to obtain the complete estimation of the indirect light calculation.

3.2.2. Fault-tolerance

The fault-tolerance mechanism used is the same as explained in Section 3.1.1, wherein reconstruction techniques are used for dealing with faults. A single buffer is stored at the master to accumulate the direct light samples and if some pixels are missing then this buffer is reconstructed. On the other hand, multiple buffers are stored for indirect light samples, one for each set of VPLs being sampled together. Each set of VPLs generates a considerably different image than another set. Therefore, to be able to reconstruct the indirect lighting with a better visual fidelity, multiple buffers are used for each set of VPLs. Furthermore, an indirect buffer is discarded if it does not contain a minimum percentage of data and there are other buffers which contain more than the minimum amount. This is done to prevent the reconstruction noise from having a big impact on the visual quality of the image.

3.2.3. Task Scheduling and Load Balancing

A similar approach is used for task scheduling and load balancing as explained in Section 3.1.2. Each task computes a part of one of the lighting components (direct or indirect) for a group of pseudo-randomly chosen pixels. Two task queues are maintained at the master, one contains the tasks for direct lighting computations and the other for indirect lighting computations. Tasks are chosen alternatively from the two queues when the workers send a request for more work. A set of tasks computing the direct lighting for different groups of

a) Sibenik (80,483 Polygons) b) Conference Room (190,953 Polygons) c) Cornell Box (63,055 Polygons)

Figure 2: *Scenes used for experiments.*

Scene	Time-constraint (in seconds)	Component-based Approach				Straightforward Approach	
		Number of Tasks Completed		Percentage of Reconstructed Pixels		Number of tasks completed	Percentage of Reconstructed Pixels
		Direct Light	Indirect Light	Direct Light	Indirect Light		
Conference Room	10.0	727	726	0.00	29.10	1586	22.6
	8.0	515	514	0.00	0.00	991	51.66
	4.0	214	214	0.00	52.45	479	76.66
Sibenik	10.0	653	653	0.00	36.34	1440	29.75
	5.0	154	154	0.00	52.34	663	67.68
	3.0	95	94	0.00	51.66	298	85.5
Cornell Box	6.0	377	377	0.00	50.91	1125	45.12
	4.0	124	123	0.00	51.95	565	72.46
	2.0	21	21	67.18	67.23	130	93.7

Table 1: *Component-based approach versus Straightforward approach*

pixels by sampling the same point on the area light sources are queued together. A group of tasks which computes the indirect lighting from a given set of VPLs is scheduled in two equal parts. The first part of the group which calculates fifty percent of the pixels is scheduled before another group of tasks which calculates the complementary half of the pixels using another set of VPLs [KH01]. The second part of the group which uses the first set of VPLs is scheduled after one half of all the tasks have been scheduled. This is done so as to give a better visual quality by sampling more sets of VPLs within a given time-constraint, as the indirect lighting can be reconstructed if the time-constraint does not permit the completion of all tasks.

The same approach as mentioned in Section 3.1.3 is used for handling the time-constraints.

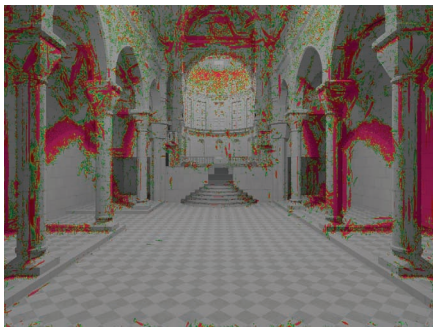
4. Results

The presented algorithms have been implemented using the Condor Master-Worker framework [GKYL01] and run on a local desktop grid formed by connecting twenty four machines with two dual-core AMD Opteron processors running at 2.6Ghz at each node. Each machine also had 8GB RAM shared among the four CPU cores. Each core was used independently as the number of idle CPUs in a machine vary with the load on it. Such a testbed was chosen because it made it possible to compare the visual quality of the two proposed algorithms. Experiments were conducted for time-constraints and fault-tolerance and the results obtained are detailed in the following section.

4.1. Time-constraints

The presented algorithms have been compared and the results are shown in Table 1. Figure 2 contains the three scenes that were chosen for comparison. The image resolution used was 1024×768 and 16 samples per pixel were used for direct lighting while the indirect lighting was calculated using 256 VPLs. The image was broken into 64 groups of pixels for the component-based approach. Each of these groups computed


a) Component-based Approach


b) VDP for Component-based Approach


c) Straightforward Approach

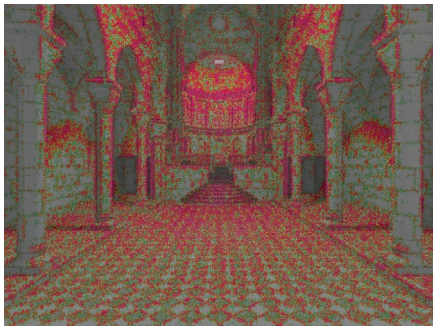

d) VDP for Straightforward Approach

Figure 3: *VDP comparison for images of Sibenik model generated with a time-constraint of 10 seconds. In the VDP output, the grey pixels depict no perceivable difference. The green pixels are used to represent low probability of noticeable difference, while the red pixels are used for depicting high probability.*

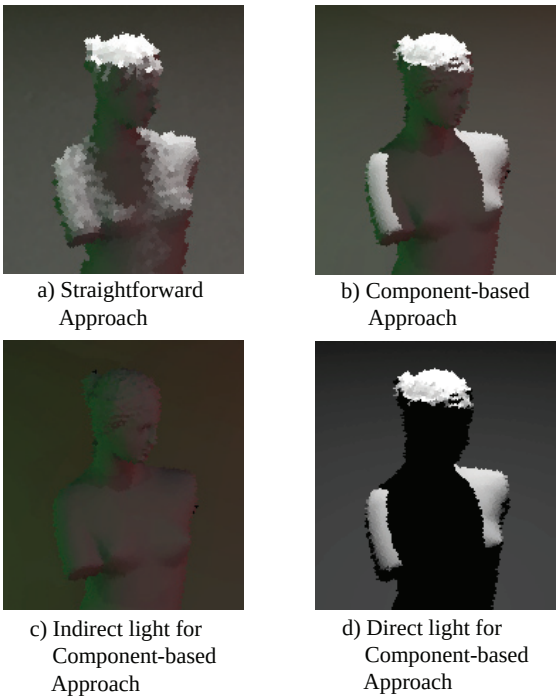
a single sample for the direct lighting and hence 1024 tasks were used in total for computing 16 samples per pixel. For the indirect lighting computations, 256 VPLs were split into 16 sets of 16 VPLs each. Hence 16 buffers were used at the master for storing the indirect computations. Here, also the image was subdivided into 64 groups of pixels and a total of 1024 tasks was used to calculate the indirect lighting. For the straightforward approach the whole image was subdivided into 2048 groups of pixels so that the number of tasks to be completed for the whole solution remained the same for both approaches. For each of these 2048 groups, the task computed 16 samples per pixel for the direct lighting and sampled all 256 VPLs for the indirect lighting.

The visual quality of the images generated using the two approaches was compared using a well-known Visual Difference Predictor(VDP) metric [Dal93]. This metric is used for predicting the probability with which two images would be perceived differently by a human. The VDP (P>75%) denotes the percentage of pixels in an image that would be perceived differently with a probability higher than 75%. The images rendered without any time-constraint were used as the gold standard while using the VDP for error prediction. It was found that the time needed for generating the gold standard images for both approaches was almost the same.

Scene	Time-constraint (in Seconds)	VDP (P>75%)	
		Component-based Approach	Straightforward Approach
Conference Room	4.0	25.98	30.47
	8.0	12.31	17
	10.0	9.55	11.54
Sibenik	3.0	17.58	54.35
	5.0	15.78	30.71
	10.0	1.55	7.89
Cornell Box	2.0	13.43	13.99
	4.0	6.72	7.27
	6.0	4.41	4.44

Table 2: *VDP results*

The VDP results are shown in Table 2. It can be seen from Table 2, that the error in visual quality as predicted by VDP for a given time-constraint has been found to be always less for the images computed using component-based approach as compared to the straightforward approach. The difference between the two approaches is considerably higher for a shorter time-constraint. This variation is more evident for the Sibenik Model than the Cornell Box because the impact of



a) Straightforward Approach

b) Component-based Approach

c) Indirect light for Component-based Approach

d) Direct light for Component-based Approach

Figure 4: *A part of Cornell Box image is shown to compare visual quality for the component-based approach with the straightforward approach at a time-constraint of 2 seconds.*

reconstruction noise on visual quality is higher for a complex scenes. The visual quality of the two approaches appears to converge as the time-constraint is increased. Figure 3, depicts the images of Sibenik model at a time-constraint of 5 seconds for both approaches along with the VDP output images.

An important thing to note here is that, as advised by Ramanarayanan et al. [RFWB07], VDP is a good metric but not a substitute for the human eye. To illustrate this fact, Cornell Box images generated using the two presented approaches with a time-constraint of two seconds are shown in Figure 4. The values of VDP(P>75%) obtained are 13.93% and 13.43% (as shown in Table 2) whereas the image generated by the component-based approach is much more visually appealing than the one generated by the straightforward approach as it has lower reconstruction noise.

4.2. Fault-tolerance

In order to illustrate the fact that the presented algorithms are capable of producing results on volatile resources handling both addition and removal of resources, the resources were varied in a controlled fashion. In one run, the resources were increased from 14 machines (56 cores) to 24 machines (96 cores) linearly with one machine being added each second and the time constraint used was 10 seconds. The VDP (P>75%) was found to be 11.15% when compared to the gold standard. In the second run, the resources were decreased from 24 machines (96 cores) to 14 machines (56 cores) linearly with one machine was being removed each second and the time-constraint used was again 10 seconds. The VDP (P>75%) was found to be 11.31% when compared to the gold standard. This is close to VDP (P>75%) value of 9.55% for the image of the Conference Room generated without varying the resources with a time-constraint of 10 seconds. The component-based algorithm was used for these results as it provides better visual fidelity.

5. Conclusion and Future Work

This paper presents two novel algorithms for rendering in parallel on volatile resources in a user defined time-constraint. It demonstrates the use of local desktop grids for computing high-fidelity images. Furthermore, it illustrates that time-constraints are a good way of using resources which are dynamic in nature. Fault-tolerance is also handled without using redundancy especially under time-constraints.

A comparison between the two presented approaches has been carried out and it has been shown that the component-based approach offers a better visual quality over the straightforward approach. This can be attributed to the fact that the component-based approach increments the visual quality in steps and is less dependent on image reconstruction techniques when compared to the straightforward approach. These algorithms can be further extended to render high-fidelity animations on local desktop grids as well, where computations for each frame of the animation can be parallelised using the presented approaches and tasks from all the frames can be queued up on the master. Although care must be taken to filter reconstruction noise between frames to prevent flickering.

It has been observed that the current frameworks for task management on desktop grids are developed with an aim of trying to provide guarantee of service for the applications on volatile resources by using traditional fault-tolerant mechanisms. The ramp-up time of Condor master-worker framework was also found to be on the high side. These issues would make it difficult to achieve interactive rates. Off-line rendering on the other hand, can cope with faults without relying on such fault-tolerance techniques as explained in Section 3.1.1. Hence, a task management framework with dynamic task scheduling which would focus on achieving maximum throughput would enhance the performance of desktop grids even further from the rendering perspective.

Future work should look at incorporating perceptual metrics in task scheduling to further improve the visual quality. In addition, work needs to be done to use better reconstruction algorithms and to predict and include the reconstruction time within the time-constraint.

6. Acknowledgements

The authors wish to thank Greg Ward for the Conference Room model, Marko Dabrovic for the Sibenik model and Stanford Computer Graphics Laboratory for the Bunny model which was used as part of the Cornell Box Scene.

References

[AC05] ANGLANO C., CANONICO M.: Fault-tolerant scheduling for bag-of-tasks grid applications. In *Advances in Grid Computing - EGC 2005* (2005), Lecture Notes in Computer Science.

[ACD08] AGGARWAL V., CHALMERS A., DEBATTISTA K.: High-Fidelity Rendering of Animations on the Grid: A Case Study. In *EGPGV* (2008).

[ACK*02] ANDERSON D. P., COBB J., KORPELA E., LEBOFSKY M., WERTHIMER D.: Seti@home: an experiment in public-resource computing. *Commun. ACM 45*, 11 (2002).

[And04] ANDERSON D. P.: Boinc: A system for public-resource computing and storage. In *GRID '04* (2004).

[AR09] ANDERSON D. P., REED K.: Celebrating diversity in volunteer computing. In *Hawaii International Conference on System Sciences (HICSS)* (2009).

[BP90] BADOUEL D., PRIOL T.: An efficient parallel ray tracing scheme for highly parallel architectures. In *Eurographics Hardware Workshop* (1990).

[BWS03] BENTHIN C., WALD I., SLUSALLEK P.: A Scalable Approach to Interactive Global Illumination. *Computer Graphics Forum 22*, 3 (2003).

[CBS*03] CIRNE W., BRASILEIRO F., SAUVE J., ANDRADE N., PARANHOS D., SANTOS-NETO E., MEDEIROS R.: Grid computing for bag of tasks applications. In *Proc. of the 3rd IFIP Conference on E-Commerce, E-Business and EGovernment* (2003).

[CDR02] CHALMERS A., DAVIS T., REINHARD E. (Eds.): *Practical Parallel Rendering*. A. K. Peters, Ltd., 2002.

[CPC84] COOK R. L., PORTER T., CARPENTER L.: Distributed ray tracing. In *SIGGRAPH '84* (1984).

[CSL06] CHONG A., SOURIN A., LEVINSKI K.: Grid-based computer animation rendering. In *GRAPHITE '06* (2006).

[Dal93] DALY S.: The visible differences predictor: an algorithm for the assessment of image fidelity.

[DAS06] DOMINGUES P., ANDRZEJAK A., SILVA L. M.: Using checkpointing to enhance turnaround time on institutional desktop grids. In *E-SCIENCE '06* (2006).

[DPF03] DUMONT R., PELLACINI F., FERWERDA J. A.: Perceptually-driven decision theory for interactive realistic rendering. *ACM Trans. Graph. 22*, 2 (2003).

[DSSC05] DEBATTISTA K., SUNDSTEDT V., SANTOS L. P., CHALMERS A.: Selective component-based rendering. In *GRAPHITE '05* (2005).

[FS93] FUNKHOUSER T. A., SEQUIN C. H.: Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments. In *SIGGRAPH'93* (1993).

[GIM] Great Internet Mersenne Prime Search. `http://www.mersenne.org/`.

[GKYL01] GOUX J.-P., KULKARNI S., YODER M., LINDEROTH J.: Master–worker: An enabling framework for applications on the computational grid. *Cluster Computing 4*, 1 (2001).

[GLH*08] GAO J., LIU H., HUANG J., BECK M., WU Q., MOORE T., KOHL J.: Time-Critical Distributed Visualization with Fault Tolerance. In *EGPGV* (2008), Favre J. M., Ma K.-L., (Eds.).

[GP90] GREEN S. A., PADDON D. J.: A highly flexible multiprocessor solution for ray tracing. *The Visual Computer 6* (1990).

[HA98] HEIRICH A., ARVO J.: A competitive analysis of load balancing strategies for parallel ray tracing. *The Journal of Supercomputing* (1998).

[HL97] HORVITZ E., LENGYEL J.: Perception, attention, and resources: A decision-theoretic approach to graphics rendering. In *Proc. of the 13th Conference on Uncertainty in Artificial Intelligence (UAI-97* (1997).

[Kaj86] KAJIYA J. T.: The rendering equation. In *SIGGRAPH '86* (1986).

[Kel97] KELLER A.: Instant radiosity. In *SIGGRAPH '97* (1997).

[KH01] KELLER A., HEIDRICH W.: Interleaved Sampling. In *Eurographics Workshop on Rendering Techniques* (2001), Springer-Verlag.

[KPK06] KACSUK P., PODHORSZKI N., KISS T.: Scalable desktop grid system. In *VECPAR* (2006).

[Muu95] MUUSS M. J.: Towards real-time ray-tracing of combinatorial solid geometric models. In *Proc. Ballistic Research Laboratories Computer-Aided Design (BRL-CAD) Symposium* (1995).

[PMS*99] PARKER S., MARTIN W., SLOAN P.-P. J., SHIRLEY P., SMITS B., HANSEN C.: Interactive ray tracing. In *I3D '99* (1999).

[RFWB07] RAMANARAYANAN G., FERWERDA J., WALTER B., BALA K.: Visual equivalence: towards a new standard for image fidelity. In *SIGGRAPH '07* (2007).

[RGS00] REISMAN A., GOTSMAN C., SCHUSTER A.: Interactive-rate animation generation by parallel progressive ray-tracing on distributed-memory machines. *J. Parallel Distrib. Comput. 60*, 9 (2000).

[SH82] SHOCH J. F., HUPP J. A.: The "worm" programs—early experience with a distributed computation. *Commun. ACM 25*, 3 (1982), 172–180.

[Sob94] SOBOL I. M.: *A Primer for the Monte Carlo Method.* CRC–Press, 1994.

[SSH*98] SLUSALLEK P., STAMMINGER M., HEIDRICH W., POPP J.-C., SEIDEL H.-P.: Composite lighting simulations with lighting networks. *IEEE Comput. Graph. Appl. 18*, 2 (1998).

[TOP] TOP500 Project. `http://www.top500.org/list/2008/11/100/`.

[vdC35] VAN DER CORPUT J. G.: Verteilungsfunktionen I and II. In *Proc. Nederl. Akad. Wetensch.* (1935).

[Whi80] WHITTED T.: An improved illumination model for shaded display. *Commun. ACM 23*, 6 (1980).

[WKB*02] WALD I., KOLLIG T., BENTHIN C., KELLER A., SLUSALLEK P.: Interactive Global Illumination using Fast Ray Tracing. In *EGSR'02* (2002).

[WSBW01] WALD I., SLUSALLEK P., BENTHIN C., WAGNER M.: Interactive rendering with coherent ray tracing. In *Eurographics 2001 Proceedings*. 2001.

[ZZH*04] ZHANG X., ZAGORODNOV D., HILTUNEN M., MARZULLO K., SCHLICHTING R. D.: Fault-tolerant grid services using primary-backup: feasibility and performance. In *CLUSTER '04* (2004).