Parallel Progressive Precomputed Radiance Transfer

Luís Paulo Santos*
Departamento de Informática
Universidade do Minho
Portugal

Sérgio Valentim†
Escola Superior de Artes e Design
Instituto Politécnico de Leiria
Portugal

António Ramires Fernandes‡
Departamento de Informática
Universidade do Minho
Portugal

(a) 24 directions (b) 96 directions (c) 1536 directions

Figure 1: Stanford bunny lit with the Eucaliptus Grove HDR cross for different numbers of directions

Abstract

Precomputed Radiance Transport (PRT) was introduced as a technique to enable interactive navigation and distant environmental real time relighting of rigid scenes. Evaluating radiance transport is, however, a computationally very demanding task, which precludes PRT's utilization during the model design phase, since the user must wait for long periods of time before being able to light and navigate within the model. This paper proposes and validates an approach to provide visual feedback to the user as soon as possible, within PRT context. By resorting to parallel processing and progressive refinement, the user is quickly presented with a lower lighting resolution of the virtual model. This is then progressively refined by incrementally increasing the number of incident directions taken into account on transport computations. PRT is, however, a complex algorithm that requires frequent collective communications of huge volumes of data, thus constraining the maximum achievable speedup on a parallel system. This issue is analysed and an alternative workload distribution is proposed and evaluated on a 12 node dual processor cluster. The final solution ensures a good resource utilization rate, reducing response times from dozens of seconds to a few hundred milliseconds.

CR Categories: I.3.7 [Computing Methodologies]: Computer Graphics—Three-Dimensional Graphics and Realism; D.1.3 [Software]: Programming Techniques—Concurrent Programming – Parallel Programming

Keywords: parallel computing, progressive refinement, precomputed radiance transfer

*e-mail: psantos@di.uminho.pt
†e-mail: sergio.valentim@esad.ipleiria.pt
‡e-mail: arf@di.uminho.pt

1 Introduction

The ability to visualize a virtual world with physically based global illumination and dynamically changing view point, geometry and lighting is a major challenge for the computer graphics research community. Sloan *et al.* [Sloan et al. 2002] introduced Precomputed Radiance Transport (PRT) as a technique to achieve interactive changing view point and low frequency distant environmental real time relighting of rigid scenes for light transport effects that terminate on diffuse or glossy surfaces. The main idea behind PRT is that, if the set of possible lighting directions remains fixed, then light transport can be precomputed for static geometry, independently of the actual radiance values of the light sources. A transport function that converts arbitrary incident into exitant radiance is evaluated over sample points (e.g., vertices on a polygonal model) and stored as a transport matrix for later usage on rendering time. Rendering is later performed by computing the dot product of the transport function with the vector L of the actual values of incident radiance. By using appropriate compression and acceleration techniques interactive frame rates can be achieved [Sloan et al. 2002; Green 2003; Ng et al. 2003; Ng et al. 2004; deVore 1998; Stollnitz et al. 1995; Sloan et al. 2003]. PRT has been extended to allow all-frequency lighting effects [Ng et al. 2003; Ng et al. 2004; Liu et al. 2004; Wang et al. 2004], extended BRDFs [Wang et al. 2005], local light sources [Kristensen et al. 2005] and locally deformable geometry [Sloan et al. 2005].

Once the transport matrix is available, high quality navigation and

relighting is possible at interactive rates. But evaluating transport is a computationally very demanding task, whose time complexity is linear with the number of sampling points and incident directions. This huge computation times preclude PRT's utilization during the model design phase, since the user must wait for long periods of time before being able to light and navigate within the model. This paper's main contribution is the proposal and validation of an approach to provide visual feedback to the user as soon as possible, within PRT context.

Parallel processing is an obvious solution: by decomposing the transport computation workload and distributing it among several computing nodes, execution times are reduced. However, PRT is a multipass algorithm that requires collective communication of huge volumes of data between passes; this results on low computation to communication ratios, that limit the number of parallel resources that can be used efficiently, thus limiting the maximum achievable speedup. We study this problem and propose alternative distributions of workload among the processing nodes in order to maximise speedup.

The user's waiting time is further reduced by resorting to progressive refinement. Intermediate solutions of increasing quality are computed by incrementally increasing the number of incident lighting directions. The user is quickly presented with a lower lighting resolution of the virtual model, which is then progressively refined, as illustrated in figure 1. The combination of parallel processing with progressive refinement reduces response times from dozens of seconds to a few hundred milliseconds.

This paper starts by presenting PRT theoretical framework. It then proceeds to discuss and evaluate parallel solutions; progressive refinement is analysed on section 4. Section 5 concludes and presents some ideas for future work.

2 Precomputed Radiance Transfer

The rendering equation [Kajiya 1986] models light distribution within an environment with no participating media and allows the computation of the exitant radiance at a point x along direction w_0, $L'(x,w_0)$, as given by

$$L'(x,w_0) = \int_\Omega f_r(x,w,w_0)(w\cdot N_x)L(x,w)\delta_w$$

where $f_r(x,w,w_0)$ is the Bidirectional Reflectance Distribution Function (BRDF), $(w\cdot N_x)$ is the cosine of the normalised incident direction w and the normal of the surface at x, N_x, and $L(x,w)$ is the incident radiance at x along w. On purely diffuse environments, such as those considered on this paper, the exitant radiance does not depend on the viewing direction w_0 and the BRDF is independent of both w_0 and w. The previous equation can be rewritten as

$$L'(x) = \int_\Omega \frac{\rho(x)}{\pi}(w\cdot N_x)L(x,w)\delta_w$$

where $\rho(x)$ is the surface albedo at x. Let

$$T(x,w) = \frac{\rho(x)}{\pi}(w\cdot N_x) \tag{1}$$

be the diffuse transport operator. The rendering equation can thus be rewritten as

$$L'(x) = \int_\Omega T(x,w)L(x,w)\delta_w \tag{2}$$

The main idea behind Precomputed Radiance Transfer (PRT) consists on separating light transport from incident radiance. For static geometry, $T(x,w)$ can be precomputed for all relevant x and w, while exitant radiance is then parameterised by incident radiance, which is only known at rendering time. If the set of relevant incident directions is discretised, then by applying numerical quadrature equation 2 can be converted onto a summation

$$L'(x) = \sum_w T(x,w)L(x,w) \tag{3}$$

Let $T(x)$ denote the transport vector for point x with transport coefficients for all directions w and $L(x)$ be the incident radiance vector; equation 3 can be written in vector notation as

$$L'(x) = T(x)L(x)$$

If the incident radiance is an high dynamic range environment map surrounding the geometric model and if it represents infinitely distant light sources, then $L(x)$ is constant for all surface points x and is denoted by L. The objects' surface can be densely sampled to create transport vectors that map arbitrary incident lighting into exitant radiance. At rendering time, these transport vectors are applied to the actual vector L producing scalar exitant radiance for each sampled point. On a polygonal model $T(x)$ is precomputed for each vertex, resulting on a transport matrix T, which has as many rows as vertices on the model and as many columns as the number of incident directions. Rendering an image for an arbitrary view point consists on evaluating $L' = TL$, which is the dot product between $T(x)$ and L for each vertex. If this evaluation is efficient the user might navigate within the model at interactive rates and have dynamic lighting. The size of T might prevent an interactive frame rate, but there are compression techniques, such as linear approximations using an orthonormal spherical harmonics base [Sloan et al. 2002; Green 2003], non-linear approximations using an orthonormal wavelet base [Ng et al. 2003; Ng et al. 2004; deVore 1998; Stollnitz et al. 1995] and clustered principal components [Sloan et al. 2003] that reduce its size and the number of floating point operations required to compute the dot products. Rendering may also be speeded up by using well known techniques such as view-frustum and occlusion culling, that reduce the number of vectors that have to be processed for each frame. These will not be discussed here, since this paper focuses on T's precomputation.

The definition of the transport operator depends on the actual illumination model being used. For diffuse shadowed direct lighting (hereby denoted as DS), the visibility operator, $V(x,w)$, must be added to equation 1:

$$T_{DS}(x,w) = \frac{\rho(x)}{\pi}V(x,w)(w\cdot N_x) \tag{4}$$

$V(x,w)$ yields 0 if a ray from x in direction w intersects an object in the scene (meaning that the corresponding light source is occluded), and 1 otherwise. Exitant radiance is thus given by $L'_{DS} = T_{DS}L$.

To include diffuse interreflections (denoted as IR) the transport operator becomes recursive. Incident radiance on x along a direction w where an object is intersected depends on the radiance that object reflects towards x. The transport operator for a single level of interreflections and for each direction w incident on x is given by

$$\forall_{w'},\ T_{IR_1}(x,w') += \frac{\rho(x)}{\pi}(1-V(x,w))(w\cdot N_x)T_{DS}(x',w') \tag{5}$$

where x' is the point visible from x along w and w' iterates all directions incident on x'. T_{IR} for an arbitrary number n of interreflections can be recursively evaluated using the Neumann series expansion of

the transport equation [Jensen 2001; Dutré et al. 2003]:

$$T_{IR} = T_{DS} + T_{IR_1} + T^2_{IR_1} + T^3_{IR_1} + \cdots = T_{DS} + \sum_{m=1}^{n} T^m_{IR_1} \quad (6)$$

An interpretation of this series is that it sums terms representing radiance reflected $1, 2, 3, \cdots$ times. Exitant radiance for the diffuse interreflected model can thus be computed as $L'_{IR} = T_{IR} L$

2.1 Algorithm

To compute diffuse interreflected transport we use a breadth first algorithm similar to the one proposed by Sloan *at al.* [Sloan et al. 2002]. This is a multi-pass algorithm; transport values from the previous pass, $T_{IR_{b-1}}$, are reused to compute T_{IR_b}, by interpolating across the triangle's surface. On the algorithm description presented below b denotes a pass index, with $b = 0$ being the direct shadowed pass, denoted by DS pass, and $b > 0$ corresponding to successive interreflection passes, each denoted by IR$_b$ pass. The intersections matrix I is computed by ray tracing on this first pass and stores which triangle is visible for all x and for each direction w. All T_{IR_b} are initialized to 0 and the final result is accumulated onto T_{IR}. All vertices and all directions are evaluated on each step.

1. **DS pass –** Compute visibility and direct shadowed transport and store it, respectively, on I and T_{IR_0}:

$$T_{IR_0}(x,w) = \frac{\rho(x)}{\pi} V(x,w)(w \cdot N_x)$$

2. **DS pass –** Accumulate on T_{IR}

$$T_{IR}(x,w) = T_{IR_0}(x,w)$$

3. **IR$_b$ pass –** Increment the pass index b
4. **IR$_b$ pass –** For each pair (x,w) if a triangle's intersection is registered on $I(x,w)$:
   (a) compute $T_{IR_b}(x)$ by interpolating from that triangle's vertices using the appropriate rows of $T_{IR_{b-1}}$; this interpolation has to be done for the set of all incident directions w'
   (b) Multiply $T_{IR_b}(x,w')$ by x's BRDF and the appropriate cosine for the set of all incident directions:

$$\forall w' : T_{IR_b}(x,w') * = \frac{\rho(x)}{\pi}(w \cdot N_x)$$

5. **IR$_b$ pass –**Accumulate on T_{IR}

$$T_{IR}(x,w) + = T_{IR_b}(x,w)$$

6. **IR$_b$ pass –**For additional passes go back to step 3. Repeat adding light bounces until energy variation falls below a prespecified threshold or a given number of bounces is achieved

2.2 Results

The above algorithm has been implemented as a multithreaded program. Experiments were run on a single machine with two 3.2 GHz Intel Xeon processors and 2 GBytes of memory. Two different size geometric models, based on the Stanford bunny, were used: the small model has 17730 vertices and the large one has 76920 vertices. Due to the huge memory footprint of the intersections and transport matrices (see table 1) the number of incident directions that could be simulated was upper bounded to 1536 and 384 for the smaller and larger models, respectively. All experiments were run with 4 threads and 2 levels of diffuse interreflections (or light bounces). Precomputation times are presented in seconds (table 1) for the diffuse shadowed pass (DS), the interreflections passes (IR) and total time.

Model	#directions	Matrix size	DS	IR	**Total**
Small bunny	96	19,5 MB	14,1	1,2	**15,3**
	384	77,9 MB	55,8	16,1	**71,9**
	1536	331,6 MB	221,5	141,5	**363,0**
Large bunny	96	84,5 MB	264,8	8,0	**272,8**
	384	338,0 MB	1055,1	64,1	**1119,2**

Table 1: Single node multithreaded version: Precomputation times (in seconds) and memory requirements for each matrix

3 Parallel PreComputed Radiance Transfer

Transport precomputation times are large for medium size models and a significant number of lighting directions. This precludes its utilization on the design stage, where a change to the model requires transport to be reevaluated from the scratch. Parallel computing is thus an obvious approach to reduce computation times.

3.1 Parallel Decomposition and Organization

The workload associated with computing transport can be decomposed along two independent dimensions: the vertices and the incident directions. The former corresponds to a decomposition of the transport matrix into rows, having different processors compute different rows of T, while the latter corresponds to a decomposition into columns. In fact, for each pass, the computation of each element $T(x,w)$ is completely independent of any other element of the matrix; this can thus be the finest grain used on the parallel system. We selected a decomposition along the vertices dimension, because there are more vertices than directions thus providing a larger degree of parallelism to distribute tasks amongst the parallel system nodes. Tasks consist of subsets of vertices and are distributed to the nodes by the master node on a demand driven approach. An adaptive task partitioning strategy is applied within each pass, the few first tasks being larger than the last ones to increase the probability of proper load balancing, as suggested by Freisleben [Freisleben et al. 1997]. Plachetka [Plachetka 2004] proposes a formal model to compute tasks's sizes for a factoring task partitioning strategy, which could be applied if some statistics were known, such as task assignment latency and tasks' minimum and maximum execution times. For the current work we simplify task partitioning by applying Freisleben's adaptive strategy.

The algorithm, however, is decomposed onto multiple passes, and each pass can not start while the former has not finished completely (figure 2). The first pass corresponds to diffuse shadowed (DS) computations and evaluates both visibility and DS transport, stored respectively onto the I and T_{IR_0} matrix. Subsequent passes evaluate diffuse interreflections; these use the I matrix to look up which triangle is visible from each vertex along each direction, and the

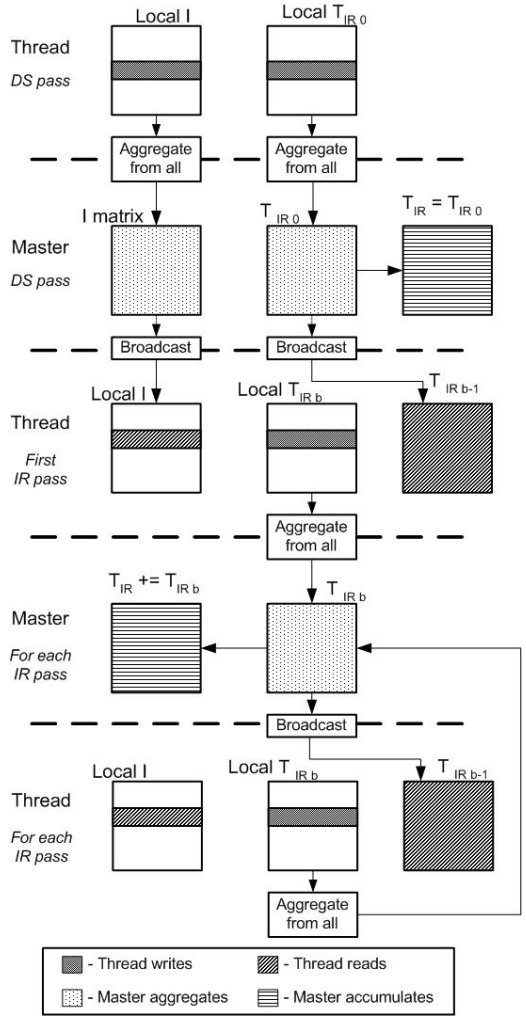


Figure 2: Parallel algorithm communication and data dependency patterns.

transport matrix from the previous pass, $T_{IR_{b-1}}$, in order to interpolate transport. The master must thus aggregate the results from all nodes and:

- broadcast I at the end of the DS pass;
- broadcast $T_{IR_{b-1}}$ at the beginning of each IR pass;
- accumulate T_{IR_b} on the final result matrix T_{IR}.

Since the matrices can be of considerable size (depending on the number of vertices and incident directions – table 1), collective communication operations and synchronisation at the end of each pass represent a significant overhead. The size of I and T is the same, since the former contains both the visible triangle identifier and the barycentric coordinates of intersection, while the latter contains transport coefficients for the three channels R, G and B. Arguably, the intersection matrix broadcast could be avoided if the set of vertices processed by each node on each pass was kept constant; however, the workload associated with the DS and the IR passes is very different (visibility and interpolation computations, respectively) and this would result on considerable load imbalances. The transport matrix must be broadcast, since each node requires the transport values from the previous iteration not for the vertices it is evaluating, but for those that are visible along each direction; these were probably computed on a different node. Besides the significant collective communications overheads, the sheer size of the matrices to broadcast may, and will for medium complexity data sets, exceed the maximum message size allowed by the underlying message passing middleware being used (MPICH 1.2.7 in our particular settings). To overcome this problem broadcasted messages are first compressed at the master node and later uncompressed at worker nodes using the standard compression library `zlib`.

3.2 Results

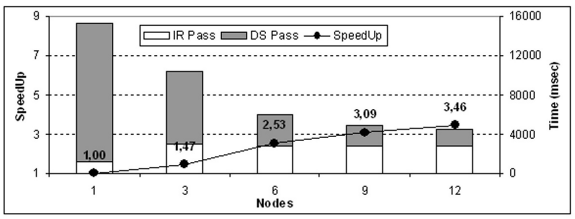
Experiments were run on a distributed cluster with 12 nodes, each with two 3.2 GHz Intel Xeon processors and 2 GB of memory. Nodes are interconnected using fully switched Gigabit Ethernet. All results presented on this paper correspond to a simulation of 2 levels of diffuse interreflections and 4 threads per node. On these experiments the master node acts only as a system controller; it does not compute any transport values, but is, nevertheless, taken into account to compute speedup. Figure 3 presents the execution times and speedups for the five data sets. The execution time is decomposed onto time spent on the DS and IR passes. The times for one node correspond to the multithreaded version presented on the previous section. The overall execution time decreases with the number of processors, as expected, but the speedup is too low; the larger geometric model presents better speedups, confirming that increasing the problem size allows a more efficient utilization of a larger number of resources, as predicted by Gustafson's Law [Gustafson 1988]. This rapid decrease in efficiency is due to the broadcasts and consequent poor performance of the parallel IR passes.

Figure 4 presents the DS and IR passes times separately for three data sets, with adapted time axis scales to allow a better perception of the respective scalability. It can be seen that the DS pass scales quite well, while the time required to compute interreflections increases in most cases. For both models and 96 directions IR times increased with the number of nodes, suggesting that exploiting parallelism is not a good option due to the low computation to communication ratio. However, for 384 directions and more than 3 nodes IR times decrease slightly with the number of nodes (at least up to 12): the larger workload associated with this data set increases the computation to communication ratio, justifying the parallel approach and overcoming the communication overheads associated with the broadcasts.

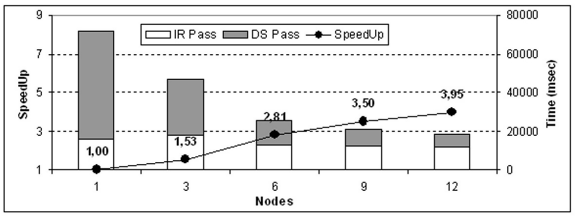
3.3 Reorganizing IR Calculations

The realization that IR passes are not good candidates for distributed memory parallel processing suggests that this component could be computed on the master rather than on the slave nodes. We modified our implementation such that the DS pass is done on the slaves, its results are aggregated on the master and then this node performs all IR passes, using a multithreaded implementation. No broadcasts are required since all threads share the same global data memory address space; the threads are required to synchronise at the end of each pass, but this overhead is minimal if the workload is well balanced among the threads. However, with this approach the master is required to maintain in memory two complete transport matrices on each IR pass: $T_{IR_{b-1}}$ and T_{IR_b}; these huge memory requirements restrict the size of tractable data sets: the small model can only be processed with up to 384 directions and the larger one with 96.

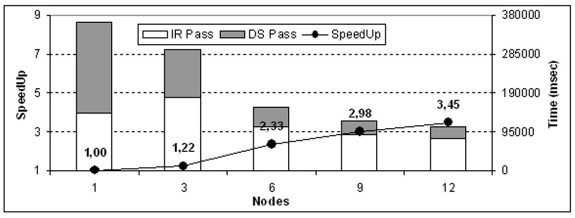
Figure 5 presents the results obtained with this approach. For the data sets corresponding to 96 directions (figures 5(a) and 5(c))



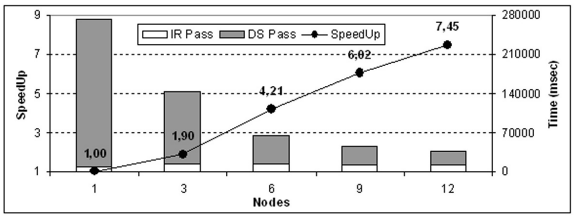
(a) Small bunny - 96 directions



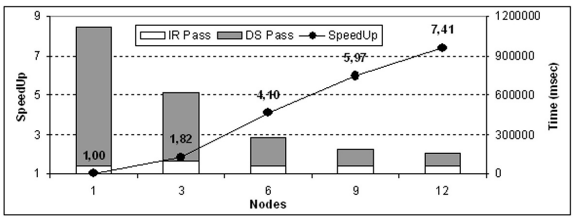
(b) Small bunny - 384 directions



(c) Small bunny - 1536 directions



(d) Large bunny - 96 directions

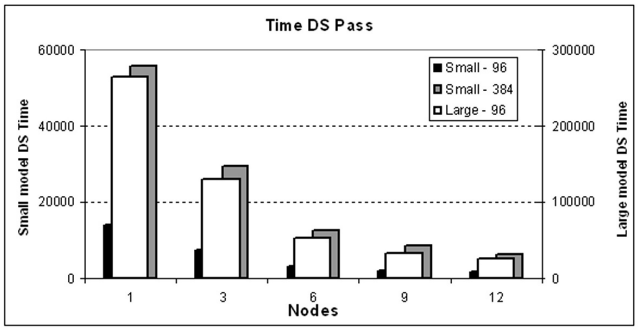


(e) Large bunny - 384 directions

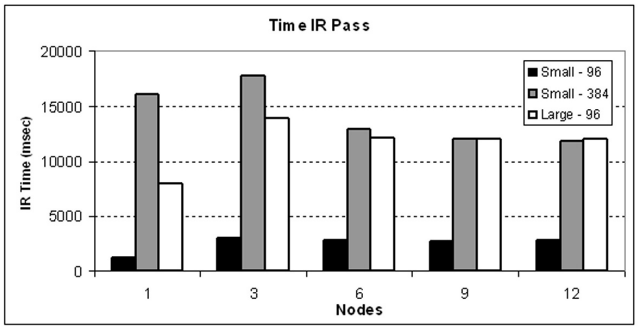
Figure 3: Results for parallel implementation.

there is a significant improvement in execution time. The simulation with 384 incident light directions got worst performance; the IR passes execution times increased for the same number of nodes, when compared to the previous version. This was expected, since it was already shown (see figure 4(b)) that above a certain workload exploiting parallelism for IR passes might result on a performance improvement.

Table 1 shows that with 384 directions IR amounts to 22,4% of total execution time, while for 96 directions it represents only 7,8% and 2,9% for the small and large models, respectively. This suggests



(a) DS pass



(b) IR passes

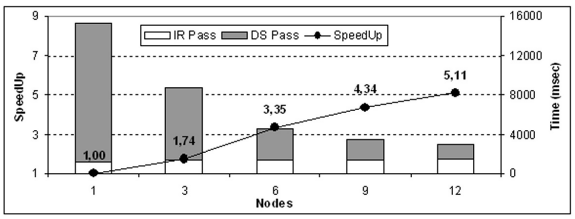
Figure 4: Separate DS and IR passes computing times

that there is a threshold above which the parallel approach to the diffuse interreflections passes is more efficient than the single node approach. This threshold depends not only on the number of vertices and the number of directions taken into account on the simulation, but on the characteristics of the geometrical model itself. An open space model, for example, is bound to have less object light interreflections than a closed space one, since rays have less probability of intersecting another object; the workload associated with IR passes is thus related to the ratio of occluded to non-occluded directions. This ratio is related to the sparsity of the I matrix, which may be used as a parameter on a decision making model. Nevertheless, making this decision may still prove to be a challenge, since the relative times of each component are known only after execution. However, these results indicate that this approach is effective up to a certain number of incident directions.

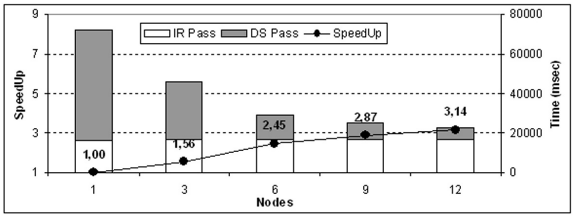
The main drawback of this workload distribution is the poor utilization of resources. During the DS phase the slave nodes are busy and the master only distributes tasks and collects results; on the other hand, during the IR phase all the slaves are idle and the master is busy computing diffuse interreflections. Some form of computation overlap between these two stages is required in order to increase resource utilization.

4 Progressive PreComputed Radiance Transfer

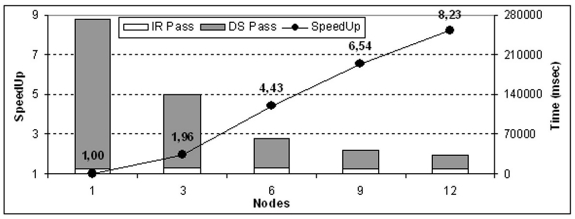
The main goal of the current work is to provide visual feedback to the user as soon as possible. Even though parallel processing reduces transport computation times significantly, these are still large for geometrical models of medium complexity, preventing the utilization of this technique during the modelling stage. Precomputa-



(a) Small bunny - 96 directions



(b) Small bunny - 384 directions



(c) Large bunny - 96 directions

Figure 5: Results for parallel DS and multithreaded IR.

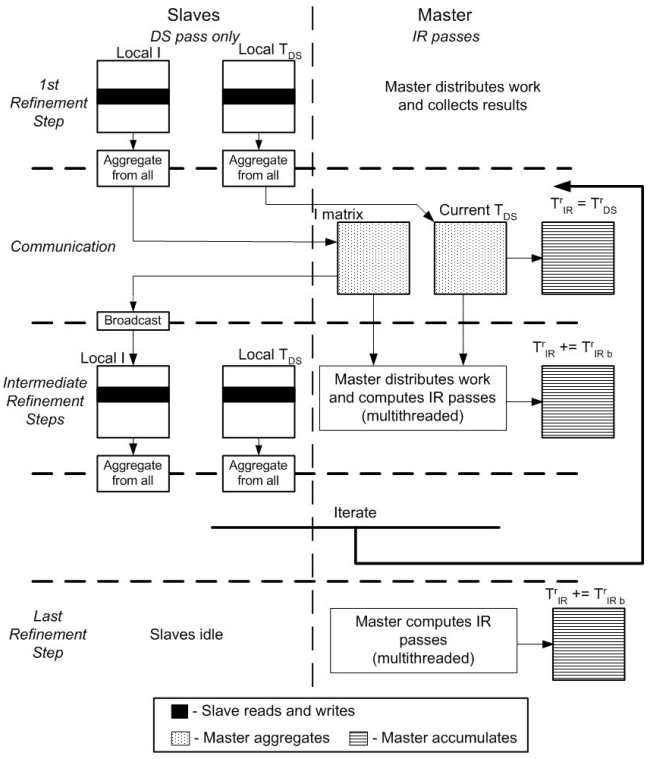


Figure 6: Progressive algorithm communication and data dependency patterns.

tion times are a function of the number of incident directions considered on the simulation. Faster visual feedback might be given to the user if transport is initially computed for a reduced number of directions and these are then increased progressively. The user will be quickly presented with a poor quality solution, which is then progressively refined as higher resolution transport matrices become available. Figure 1 illustrates this concept for the Stanford bunny with 24, 96 and 1536 directions. The bunny's shadow smoothness and shading quality increase with the number of directions as does the precomputation time.

4.1 Algorithm

Two versions of the progressive transport computation program have been developed: one with all passes running in parallel on the slaves, the other with the IR passes running on the master. The former is just a small extension of the application presented in section 3.1: an outer loop is added with the number of directions being progressively increased and ranging from 6, 24, 96, 384 up to the maximum user defined value. Intersections information from previous refinement steps is reused in order to avoid tracing rays repeatedly along the same directions. For example, on the first refinement step 6 directions are traced per vertex; on the next step only 18 additional rays must be traced per vertex, the information regarding the remaining 6 being reused.

The second progressive version has the DS passes running on the slaves and the IR passes running on the master. The main insight here is that interreflections' computations for progressive refinement step r, running on the master, may be overlapped with visibility calculations for refinement step $r+1$, which run on the slaves.

As illustrated in figure 6, as soon as the master receives the DS pass results from the slaves, it can:

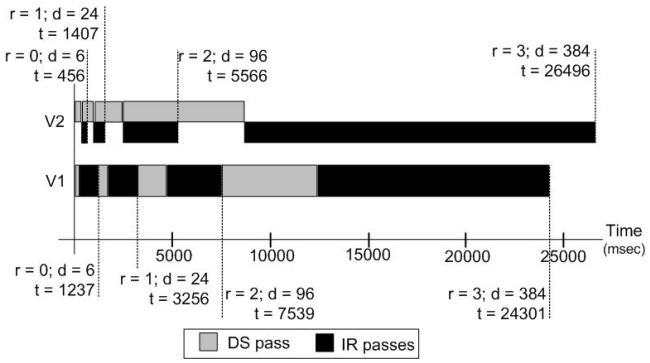
- start IR passes locally, using a multithreaded architecture;
- broadcast the aggregated intersections information to the slaves;
- distribute tasks regarding the DS pass of the next refinement step.

Since the number of directions evaluated on each refinement step grows exponentially with r ($d = 2^r * 6$), the master will always finish the IR passes for the current step before the slaves finish computing the additional visibility information (this is true for a reasonable number of IR passes). Consequently, the master may make T_{IR} for step r available to the renderer and then start receiving the slaves' results for step $r+1$. Computation of DS and IR passes is thus overlapped, and all nodes are kept busy most of the time. Broadcasting the intersection matrix to the slaves between refinement steps still represents an overhead, but this avoids reevaluating visibility for directions that have been processed previously.

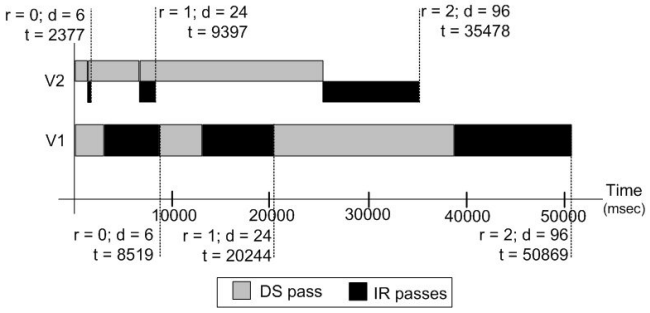
4.2 Results

Figure 7 presents time diagrams for both versions and both geometrical models, obtained using 12 nodes. Light gray blocks represent time spent by the slaves computing visibility and diffuse shadowed transport, while black blocks represent time spent on the master computing interreflections. Whenever a black block finishes, a particular refinement step is complete, meaning that the corresponding

transport matrix can be made available to the renderer; these time instants are labelled with the time elapsed since execution began. Data series labelled with V1 correspond to the all parallel versions, i.e., DS and IR passes running on the slaves; V2 corresponds to evaluating IR passes on the master and overlapping with the next refinement step DS pass. These results are also presented on table 2.



(a) Small bunny



(b) Large bunny

Figure 7: Time diagram for progressive versions and 12 nodes (V1 - all parallel; V2 - IR passes on the master).

		Time for T_{IR}			
Model		6	24	96	384
Small bunny	V1	1237	3256	7539	24301
	V2	456	1407	5566	26496
Large bunny	V1	8519	20244	50869	—
	V2	2377	9397	35478	—

Table 2: Transport computation times (msec) for progressive refinement with 12 nodes

Analysis of these time diagrams allows the following conclusions:

- for up to 96 directions V2 provides intermediate solutions much faster than V1. Overlapping evaluation of the DS and IR passes enables a more effective use of the available resources and avoids communication overheads associated with broadcasting intermediate transport matrices between successive IR passes;
- for the last refinement step, with the small model and 384 directions, V1 finishes faster than V2. The reasons for this result were discussed in sections 3.2 and 3.3: for this data set the all parallel version is more effective than processing all the diffuse interreflections on the master. Note, however, that faster feedback is still achieved by V2 for the previous refinement steps;
- it is quite clear from these diagrams that resources are not well exploited on the last refinement step. The master processes all the workload, while the many slaves remain idle.

The above remarks suggest that V2 performs better and gives faster feedback to the renderer, but above a given threshold a hybrid version should be used, where the last refinement step is performed by the slaves, not by the master. This would avoid having the slaves idle at the end of the computation, while the master struggles to evaluate the final transport matrix. Predicting this threshold remains an unsolved problem, since, as already stated, the relative timings of each component are only known after execution. Nevertheless, the overall trend suggests that a medium size data set would benefit from this approach.

5 Conclusion

This paper proposes progressive parallel Precomputed Radiance Transfer (PRT) as a means of providing fast visual feedback to users within a model design stage.

PRT is a multipass algorithm, where two different stages can be identified: the visibility and direct shadowed stage (DS) and the interreflections evaluation stage (IR). Evaluating IR on a distributed memory parallel system requires collective communication of huge volumes of data, which results on a low computation to communication ratio that limits the system's efficiency. To overcome this limitation a parallel version of PRT with DS computations assigned to slave nodes and IR computations performed on a master node has been developed and evaluated; results have shown that this approach outperforms an all parallel approach, up to a certain threshold related to workload size. Characterising this threshold would be useful for deciding which approach to follow for a given data set. This remains an open problem, since the relative time weights of the DS and IR stages are known only after execution; the sparsity of the intersections' matrix, I, is probably a good indicator of the relative workload of the IR passes and could be used as a criterium to predict this threshold. Parallel processing is, nevertheless, able to reduce computation times up to a significant degree of parallelism.

In order to provide even shorter response times a progressive refinement approach to PRT has been proposed: the number of incident lighting directions considered on transport computation is increased incrementally, providing the user with solutions of increasing quality as time flows. The parallel progressive refinement application is essentially a loop of the original non-progressive parallel program. This loop allows overlapping IR computations on the master node for refinement step r with DS computations on the slaves for refinement step $r+1$. This solution proved to result on a good resource utilization rate, reducing response times from dozens of seconds to a few hundred milliseconds, which was this work's original goal.

There are two main issues associated with the proposed progressive parallel PRT system, which remain to be solved.

Above a certain workload size IR passes are more effective if performed in parallel by the slave nodes than if performed on a single node. This is specially significant for the last refinement step, since the slaves remain idle while the master struggles to finish its task. Predicting whether this last step should be performed by the master or by the remaining nodes would allow even shorter response times.

The whole algorithm is very memory consuming; this is specially true when a single node computes all IR passes, since it has to keep two complete transport matrices in memory, $T_{IR_{b-1}}$ and T_{IR_b}, corresponding to two consecutive light bounces. Memory requirements limit the size of the data sets than can be handled efficiently. A

reorganization of the data structures is required in order to reduce memory consumption, thus enabling processing of more complex geometrical models and/or a larger number of incident directions.

Acknowledgements

The hardware used for all experiments belongs to "SEARCH - SErvices and Advanced Research Computing with HTC/HPC clusters", a project partially funded by the Portuguese Fundação para a Ciência e Tecnologia.

References

DEVORE, R. 1998. Nonlinear Approximation. *Acta Numerica 7*, 51–150.

DUTRÉ, P., BEKAERT, P., AND BALA, K. 2003. *Advanced Global Illumination*. AK Peters.

FREISLEBEN, B., HARTMANN, D., AND KIELMANN, T. 1997. Parallel raytracing: a case study on partitioning and scheduling on workstation clusters. In *Hawaii Int. Conf. on System Sciences*, IEEE Computer Society Press, vol. 1, 596–605.

GREEN, R. 2003. Spherical Harmonic Lighting: The Gritty Details. Tech. rep., Sony Computer Entertainment America, January.

GUSTAFSON, J. 1988. Reevaluating Amdahl's Law. *Communications of the ACM* (May), 532–533.

JENSEN, H. W. 2001. *Realistic Image Synthesis Using Photon Mapping*. AK Peters.

KAJIYA, J. 1986. The Rendering Equation. In *ACM SIGGRAPH*, vol. 20, 143–150.

KRISTENSEN, A. W., AKENINE-MOLLER, T., AND JENSEN, H. W. 2005. Precomputed Local Radiance Transfer for Real-Time Lighting Design. *ACM Transactions on Graphics (SIGGRAPH'2005)* (July), 1208–1215.

LIU, X., SLOAN, P.-P., SHUM, H.-Y., AND SNYDER, J. 2004. All-Frequency Precomputed Radiance Transfer for Glossy Objects. In *Eurographics Symposium on Rendering*, EG Press.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2003. All-Frequency Shadows Using Non-Linear Wavelet Lighting Approximation. *ACM Transactions on Graphics (SIGGRAPH'2003) 22*, 3 (July), 376–381.

NG, R., RAMAMOORTHI, R., AND HANRAHAN, P. 2004. Triple Product Wavelet Integrals for All-Frequency Relighting. *ACM Transactions on Graphics (SIGGRAPH'2004)* (July).

PLACHETKA, T. 2004. Tuning of algorithms for independent task placement in the context of demand-driven parallel ray tracing. In *5th Eurographics Parallel Graphics and Visualization*, 101–109.

SLOAN, P., KAUTZ, J., AND SNYDER, J. 2002. Precomputed Radiance Transfer for Real-Time Rendering in Dynamic, Low-Frequency Lighting Environments. In *SIGGRAPH'2002*, 527–536.

SLOAN, P., HALL, J., HART, J., AND SNYDER, J. 2003. Clustered Principal Components for Precomputed Radiance Transfer. In *SIGGRAPH'2003*, 382–391.

SLOAN, P.-P., LUNA, B., AND SNYDER, J. 2005. Local, Deformable Precomputed Radiance Transfer. *ACM Transactions on Graphics (SIGGRAPH'2005)* (July), 1216–1224.

STOLLNITZ, E., DEROSE, T., AND SALESIN, D. 1995. Wavelets for Computer Graphics: A Primer, Part 1. *IEEE Computer Graphics and Applications 15*, 3 (May), 76–84.

WANG, R., TRAN, J., AND LUEBKE, D. 2004. All-Frequency Relighting of Non-Diffuse Objects using Separable BRDF Approximation. In *Eurographics Symposium on Rendering*, EG Press.

WANG, R., TRAN, J., AND LUEBKE, D. 2005. All-Frequency Interactive Relighting of Translucent Objects with Single and Multiple Scattering. *ACM Transactions on Graphics (SIGGRAPH'2005)* (July), 1202–1207.