Improving Locality for Progressive Radiosity Algorithm: A Study based on the Blocking Transformation of the Scene.

J. R. Sanjurjo
Department of Electronics and Systems
Univertity of A Coruña
E–15071, A Coruña, Spain
josesan@udc.es

M. Amor
Department of Electronics and Systems
Univertity of A Coruña
E–15071, A Coruña, Spain
margamor@udc.es

M. Bóo
Department of Electronics and Computer Engineering
Univertity of Santiago de Compostela
E–15782, Santiago de Compostela, Spain
elmboo@usc.es

R. Doallo
Department of Electronics and Systems
Univertity of A Coruña
E–15071, A Coruña, Spain
doallo@udc.es

ABSTRACT

Efficient global illumination is an important challenge in computer graphics. The main problem of these algorithms is their associated execution time and storage requirements. This is a handicap for simulating large scenes and reducing these costs is a constant research objective of the computer graphics community.
In this paper we present a performance analysis of the progressive radiosity algorithm based on the blocking transformation of the scene according to a uniform partition. The data locality exploitation and the associated decrement in cache misses permit the reduction of the execution time of the algorithm. An extended analysis of the influence of the scene subdivision on the execution time requirements is presented. Because of our analysis, we conclude that important performance improvements in terms of execution time are obtained with this technique.

Keywords
Radiosity, data locality, scene partition, blocking transformation.

1. INTRODUCTION

The radiosity method [Coh93a] is a global illumination algorithm used to simulate light transfer in synthetic images. The main drawbacks of the radiosity method are its high storage requirements and long execution times [Sil94a]. Among the different radiosity algorithms we have focussed our analysis on the progressive radiosity algorithm [Coh88a, Coh93a], due to its simpler structure. Similar analyses can be performed on other radiosity strategies, e.g. for the hierarchy radiosity algorithm [Han91a].

In general, the performance of a code is influenced by the data reuse and data locality exploitation. It is considered data reuse when the next utilization of a data does not imply an access to the lower level of the memory hierarchy. In fact, there can be a wide performance gap between programs that are designed to optimize cache performance and those that are not, for example in the context of computer graphics [Cho06a].

In this work we present a theoretical data reuse analysis of the progressive radiosity algorithm. Based on the cache miss rate obtained, we have concluded that a data reuse technique can be employed to optimize the algorithm. Specifically, our optimization proposal is based on the blocking transformation of the scene through its uniform partition. This permits the optimization of data reutilisation and, with this, the acceleration of the algorithm.

Scene partitioning is a technique usually employed for distributed memory systems. On such platforms, the effective distribution of data among processors is very important. In some proposals the scene is replicated on all processors [Gar00a, Pad01a], but this limits the applicability of the solutions to scenes with low complexity. A valid method for complex scenes implies the distribution of the scene among the processors [Arn96a, Gib00a, Amo04a, Gue03a, San05a]. These proposals employ different methods for scene partition and distribution among processors.

Based on this idea, we work on one partitioning procedure [Gue03a, San05a] in our attempt to exploit and

analyze the data locality of the sequential algorithm. Scene partitioning implies different challenges. For example, the visibility technique employed to determine the visibility among elements associated with different partitions. This technique, as will be shown in the paper, also has an associated computational cost that has to be considered. We do not propose an accurate study of the cache behaviour, but an extended analysis of the benefits of the partitioning strategy. This analysis is validated with a set of test scenes.

The rest of this paper is organized as follows: in Section 2 we briefly introduce the progressive radiosity algorithm; in Section 3 we analyze the data reuse in the algorithm; in Section 4 we present the block transformation strategy based on a partitioning of the scene; experimental results are shown in Section 5; finally, in Section 6 we present the main conclusions.

2. THE PROGRESSIVE RADIOSITY ALGORITHM

The radiosity method is currently one of the most popular global illumination models in computer graphics. This method solves a global illumination problem expressed by the rendering equation [Kaj86a], simplified by considering only ideal diffuse surfaces. The resultant discrete radiosity equation is:

$$B_i = E_i + \rho_i \sum_{j=1}^{N} B_j F_{ij} \qquad (1)$$

where B_i is the radiosity of patch P_i, E_i is the emittance, ρ_i the diffuse reflectivity and N is the number of patches of the scene. The summation represents the contribution of the other patches of the scene and F_{ij} is the *form factor* between patches P_i and P_j. F_{ij} is an adimensional constant that only depends on the geometry of the scene and represents the proportion of the radiosity leaving patch P_i that is received by patch P_j. There is one radiosity equation per patch and one *form factor* among all pairs of patches. Therefore, interaction of light among the patches in the environment can be stated as a set of simultaneous equations:

$$\begin{bmatrix} 1-\rho_1 F_{11} & -\rho_1 F_{12} & \cdots & -\rho_1 F_{1N} \\ \rho_2 F_{21} & 1-\rho_1 F_{22} & \cdots & -\rho_1 F_{2N} \\ \vdots & \vdots & \vdots & \vdots \\ \rho_N F_{N1} & -\rho_N F_{N2} & \cdots & 1-\rho_N F_{NN} \end{bmatrix} \begin{bmatrix} B_1 \\ B_2 \\ \vdots \\ B_N \end{bmatrix} = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_N \end{bmatrix} \qquad (2)$$

where the matrix is called the *Interaction matrix*. Therefore, the radiosity equation system is liable to be very large with complexity $O(N^2)$.

Some algorithms, such as the progressive radiosity proposal [Coh88a], have been developed for the minimization of these computational requirements. The basic idea of the algorithm is to reduce the number of patches to be considered for each iteration and, with this, to reduce the number of computations.

```
1   while (not converged){
2       Pi = 0;
3       /* LOOP1: Select Patch with greatest ΔBiAi */
4       for ( P = scene→Patch_initial; P; P = P→next)
5           Pi = MaxPot(Pi, P);
6
7       /* LOOP2: Calculate (the first time that a patch
                is shoot ) Patches interaction with Pi */
8       for ( P = scene→Patch_initial; P; P = P→next)
9           Interaction(P, Pi);
10
11      /* LOOP3: Shoot Radiosity from patch Pi */
12      for ( Pj = Pi →Interaction; Pj;
            Pj = Pi → Interaction→ next) {
13          Vij = Visibility(Pi, Pj);
14          Gij = Geometry(Pi, Pj);
15          Fij = Gij · Vij;
16          Radiosity(Pi, Pj);
17          }
18      }
```

Figure 1: Pseudocode for progressive radiosity algorithm.

The structure of the progressive radiosity algorithm is outlined in Figure 1. The key idea is to *shoot* in every step the energy of the patch P_i with the greatest unshot radiosity (lines 2 to 5). As a result of that shot, the other patches, P_j, may receive some new radiosity and patch P_i left without un-shot radiosity (lines 7 to 17).

Each shooting step can be viewed as multiplying the value of radiosity to be "shot" by a column of the *interaction matrix*. Therefore at that moment the *form factors* between the shot patch and the rest of the patches have to be calculated (lines 13 to 15). V_{ij} (line 13) is the visibility factor and takes the value 1 if P_i is visible from P_j, and 0 otherwise. In our implementation we have used a method based on directional techniques [Hai94a, Pad03a] to calculate the visibility term. G_{ij} (line 14) captures all the geometric terms. In our algorithm we have used the analytic method called *differential area to convex polygon* [Coh93a] to compute it.

As soon as the radiosity of a patch has been shot, another patch is selected. A patch may be reselected for the shooting procedure if it has received light from other patches. Therefore, the amount of radiosity the patch has received since the last time, ΔB_i, can be shot. The algorithm iterates until a desired tolerance is reached (line 1). Thus, the complexity would be $O(k \times N)$, k being the number of steps performed, with $k \ll N$.

3. DATA REUSE ANALYSIS IN PROGRESSIVE RADIOSITY ALGORITHM

In this section, we make a quantitative analysis of the cache behaviour for the progressive radiosity algorithm.

The objective is to analyse the nature of the cache misses associated with the algorithm for a later optimization of the algorithm based on the data locality exploitation. As will be shown later in this section, the progressive radiosity algorithm requires that, for standard scene sizes and hierarchical memory structures, the same patch is retrieved multiple times from lower levels of the memory hierarchy.

As can observed in the pseudocode of Figure 1, the progressive algorithm consists of three main loops where read and write operations are performed on the sequential list of N patches of the scene. We will define the number of memory references (Nr) as the number of times a patch is referred. Taking into account this definition and assuming usual scene and memory sizes, the number of memory references (Nr) required by the different data structures to perform the calculation of radiosity for each loop is:

$$\begin{aligned} Nr_{Loop1} &= N \times k \\ Nr_{Loop2} &= N \times t \\ Nr_{Loop3} &= \overline{M} \times k \end{aligned} \tag{3}$$

where N is the number of patches, k the number of iterations, t the number of different patches that are shot, $t \leq k$, and being $\overline{M}$ the average number of accessed objects per iteration. Loop1 selects the patch P_i with maximum power and Loop2 the patches P_j that interact with it. Both loops present a sequential access nature. On the other hand, Loop3 exhibits a behaviour difficult to predict and exploit. This is due to the irregular structure of the algorithm employed to compute the visibility. In our case we have employed a directional technique that implies the access to potential candidates to include the interaction between two polygons. Taking into account this information, the total number of references is:

$$Nr = (N + \overline{M}) \times k + t \times N \tag{4}$$

As result, each patch has to be accessed multiple times and this leads to a large memory bandwith and a overhead of cache misses. As N is a very large number for usual scene sizes, the cache cannot contain all the blocks needed during the execution of the standard radiosity algorithm (code of Figure 1). Then, the replacement misses will occur in all loops (following the notation of [Gho99a] we call replacement misses capacity and conflict misses).

To clarify let us consider the example of Figures 2. This figure illustrates an interaction matrix for a given scene with $N = 16$ patches. In this figure, a symbol $\times$ indicates that there is an interaction between the patches associated with the row and the column. For example, Patch 0 interacts with patches $\{1,2,3,4,7,9,11,12,13,15\}$. We assume for this simple example a small cache with capacity of four data. We employ a fully associative1 cache because of its

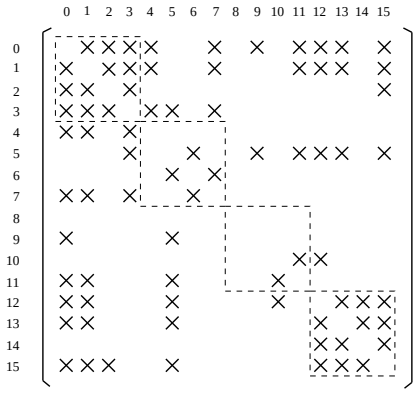


Figure 2: Interaction Matrix for a scene with 16 patches.

Address of reference	Hit/ /miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
-	-	12	13	14	15
1	miss	1	13	14	15
2	miss	1	2	14	15
3	miss	1	2	3	15
4	miss	1	2	3	4
7	miss	7	2	3	4
9	miss	7	9	3	4
11	miss	7	9	11	4
12	miss	7	9	11	12
13	miss	13	9	11	12
15	miss	13	15	11	12

Table 1: Cache contents for the example of Figure 2: Standard progressive radiosity algorithm.

good performance. This simple example with a small cache permits the illustration of the low exploitation of the data locality when the available storage resources are not sufficient to store the full scene.

Let us suppose patch 0 is the patch with the greatest unshot radiosity. A snapshot of the data accesses for the progressive radiosity algorithm is shown in Table 1 . The first row shows the contents of cache after the execution of Loop2 and the rest of the table illustrates the series of references for Loop3. As an example, note that when patch 4 is referenced it replaces patch 15. This is due to the associative nature of the cache and we have choosen the replace method of the least recently used block. Therefore, this set of references generates 10 misses for 10 accesses. This simple example shows the low exploitation of the data locality for this system.

The performance of a code is highly influenced by data reuse and the available memory resources. The number of times a patch is referenced during the full execution of the algorithm constitutes a tool to evaluate the data transfer in the memory hierarchy for large scenes. Specifically, the redundancy access for the progressive algorithm can be measured using the following equation:

$$Ra = \frac{Nr}{N} = \frac{(N + \overline{M}) \times k + t \times N}{N} = k + \frac{\overline{M}}{N} \times k + t \tag{5}$$

For the scenes we have employed in our tests and for the convergence criteria we have employed (line 1 of code

in Figure 1) we observe that $\overline{M} \simeq N^2/4$ and $t \simeq k$. The scenes employed in our tests have very different structures and sizes and, consequentially, we consider these approaches as representative of a generic case. Therefore:

$$Ra = \left(2 + \frac{N}{4}\right) k \tag{6}$$

Consequently, each patch is accessed an average of $\left(2 + \frac{N}{4}\right) k$ times. Taking into account this large number of references, it is important to maximize the accesses to each patch while it is in the cache. Reducing the cache misses permits the reduction of the number of times the same patch have to be loaded in the cache and, consequently, the reduction of the execution time of the algorithm.

4. DATA LOCALITY OPTIMIZATION BASED ON BLOCKING TRANSFORMATION

Transforming the procedures of a program might improve the spatial and temporal locality of the data. In this section we present a method to improve the data reuse based on the data locality exploitation. To increase the reutilization of the data, a blocking transformation strategy based on the partitioning of the scene is performed. Specifically, we propose the blocking transformation of the code to reduce the cache misses via improving temporal locality exploitation. The goal is to maximize accesses to the data loaded into the cache before the data is replaced.

To ensure that the data being accessed can fit in the cache, the original code is changed to work on subscenes of a lower size. To do this, the whole scene data is read and the bounding volume, which contains the entire scene, is computed. This volume is divided into p uniform disjoint subspaces, where p is the number of partitions. The partitioning strategy implies three main challenges: first, the partitioning procedure to be employed, second, the visibility technique to determine the visibility among patches in different sub-scenes and finally the scheduling procedure to sequentially process the different partitions.

The partitioning strategy should assure resulting subscenes with a convex structure. This permits the simplification of the visibility determination between patches of the same sub-scene and between patches of different sub-scenes. Within the wide range of existing partitioning techniques, we have focused on static graph partitioning algorithms, since the progressive radiosity method (unlike, for instance, hierarchical radiosity method) does not use an evolving data structure. This way, it is not necessary to apply more complex dynamic graph partitioning methods. Within static techniques we have chosen geometric techniques, as we are dealing with geometric scenes and these techniques obtain an extremely fast partition based mainly on geometric information. A similar strategy was employed in [Amo04a] in the context of the parallelization of the progressive algorithm on distributed memory systems.

The version we have employed makes a geometric uniform partition of the input scene. It splits the scene domain into a set of disjoint subspaces of the same shape and size. This kind of partitioning is computed quickly and straightforwardly and, as will be shown later, it also provides high data locality. Taking into account that all subspaces are equal-sized, we cannot divide a scene for an odd number of sub-scenes into two halves and then, recursively, split one of the sub-scenes into two new halves, because the resulting subspaces would not be of equal size. So the implemented algorithm is not recursive and the number of splitting planes that are needed to split the scene along each coordinate axis is computed in advance. Finally, each partition has a disjoint set of polygons of the whole scene.

The visibility computation among patches is a challenge when the patches are in different sub-scenes. We have used a visibility mask technique [Arn96a, Gue03a] to solve this problem. This mask is a structure that stores all the visibility information encountered during the processing of a selected patch in its local subscene. This local visibility information can be employed for the computation of the other sub-scenes. In other words, the other sub-scenes do not require accessing the patch information associated to this scene, but to its visibility mask. This permits the reduction of data accesses with low locality.

The partitioning and visibility mask techniques have to be employed cautiously because both algorithms have associated computational costs that could result in an increment of the execution time of the algorithm. However, as will be shown in the result section, for a specific number of partitions these costs are completely overcome by the benefits associated with the data locality exploitation.

With respect to the new scheduling procedure, the general structure of the new algorithm is illustrated in Figure 3. The partitioning of the scene into sub-scenes, divides the calculations associated with each iteration. On one side, there is the gathering of local radiosity of the sub-scene, on the other side the gathering of remote radiosity coming from the rest of the scene. In each iteration, the sub-scenes are processed sequentially (line 2). For each sub-scene, the patch with the greatest unshot energy is chosen P_i^l (lines 4 to 7) and the patches in the same sub-scene that interact with it are identified (lines 9 to 11). After this, the local gathering of radiosity is computed following a similar procedure to the standard progressive radiosity algorithm but, as will be shown in the following, with slight modifications (lines 13 to 23).

```
1  while (not converged){
2        for( sub-scene_l = scene→subcene_initial;
             sub-scene_l; sub-scene_l= sub-scene_l →next){
3
4            P_i^l = 0;
5            /* LOOP1: Select Patch with greatest ΔB_i A_i
                                  for subcene l */
6            for ( P = sub-scene_l →Patch_initial; P;
                   P = P→next)
7                P_i^l = MaxPot(P_i^l, P);
8
9            /* LOOP2: Calculate (the first time that a
             patch is shoot ) Patches interaction with P_i^l */
10           for ( P = sub-scene^l →Patch_initial; P;
                   P = P→next)
11               Interaction(P, P_i^l);
12
13           /* LOOP3: Shoot Radiosity from patch P_i^l */
14           for ( P_j = P_i^l →Interaction; P_j;
                   P_j = P_i^l → Interaction→ next) {
15               if ( P_i^l ∈ sub-scene_l )
16                 V_ij^local = Visibility_local(P_i^l, P_j);
17               else
18                 V_ij = Visibility_global(P_i^l, P_j)
19               G_ij = Geometry(P_i, P_j);
20               F_ij = G_ij · V_ij;
21               Radiosity(P_i, P_j);
22
23           }
24           if ( Mask(P_i^l) > 0 )
25             Gathering(P_i^l);
26       }
27 }
```

Figure 3: Pseudocode for progressive radiosity algorithm after blocking transformation.

Once the local radiosity is updated in the sub-scene its energy is propagated to the whole scene (lines 24 to 25). In this stage, the patch is copied to the sub-scenes of the adjacent partitions and the visibility information is calculated. As previously mentioned, we have employed the mask technique [Arn96a, Gue03a] that computes a visibility map per patch. This map stores the visibility of the patch with respect to the neighbouring scene, taking into account the occlusions associated with its local sub-scene.

Note that the shot patches of a sub-scene and their corresponding visibility masks are copied to the neighbour sub-scenes. This means that for the neighbour sub-scenes we have to process "local" and "copied" patches. This has to be taken into account for computing the gathering of radiosity inside each sub-scene (lines 13 to 23). In the case of a "copied" patch being processed, the gathering of energy is computed similarly but the non-local visibility information (line 18) is calculated by casting rays from patch to sub-scene through its visibility mask. The procedure follows for the adjacent sub-scenes with the same strategy, shooting patches are copied and visibility masks are computed/updated taking into account the occlusion information local to already processed sub-scene.

Address of reference	Hit/ /miss	Contents of cache blocks after reference			
		Block 0	Block 1	Block 2	Block 3
-	-	0	1	2	3
1	hit	0	1	2	3
2	hit	0	1	2	3
3	hit	0	1	2	3

Table 2: Cache contents for the example of Figure 2: Modified progressive radiosity algorithm.

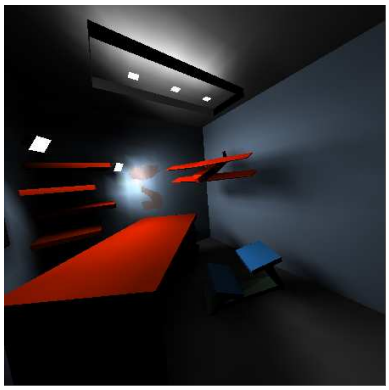
This technique permits the evaluation of the progressive radiosity algorithm following a partitioning strategy. As will be shown in the following, benefits in terms of the cache misses reduction can be obtained. As an example, Table 2 shows the cache accesses after applying the blocking strategy to the example of Figure 2. In this example we consider that the scene was subdivided in four partitions, each sub-scene with 4 patches (indicated with dashed lines in the Figure 2). The first row of table shows the content of the cache after Loop2 and the remaining rows show the consecutive references for Loop3 associated with the local computations of the first sub-scene (patches 0, 1, 2, 3). The cache memory, with capacity for four data, stores the patches associated with the sub-scene and the local computations can be performed without cache misses. Note that for this example the set of references generates 3 hits for 3 accesses.

This simple example emphasizes the potential benefits that can be obtained with the partitioning technique. This way, if each sub-scene can be fully stored in the cache, the local computations can be performed without accessing the lower levels of the hierarchical memory. In more detail, let us compute the total number of memory references for the modified algorithm:

$$Nr = \sum_{l=1}^{p} \left(\underbrace{B^l \times k}_{t1} + \underbrace{B^l \times t}_{t2} + \underbrace{\overline{M^l} \times k}_{t3} \right) + (p-1) \times N_m \times k \tag{7}$$

where p is the number of sub-scenes, B^l the number of patches in each sub-scene, $\overline{M^l}$ is the average number of accessed objects in each sub-scene per iteration and each mask consists of N_m patches. Note that the last term of equation is associated to the masks computation. But accordingly to its definition Nr is lower when the B^l patches fit into the cache level. In this case, once the sub-scene is stored in the cache for Loop1, the information is re-employed for the Loop2 and Loop3 without requiring more information. This means that terms $t2$ and $t3$ can be eliminated from the Nr equation. Therefore, the redundancy access is

$$Ra = \sum_{l=1}^{p} \frac{B^l \times k}{N} = k \tag{8}$$



(a)

(b)

Figure 4: Illuminated scenes: (a) *room-model* and (b) *armadillo*.

Comparing this result with that obtained for the original algorithm (see Equation 6) we potentially have $\left(2+\frac{N}{4}\right)$ times less cache misses than with the original algorithm using the partitioning strategy. Note that the term associated to the masks computation is omitted as in this analysis we analyze only the redundancy access for the patches.

5. EXPERIMENTAL RESULTS

In this section, we present performance results and analysis of the proposed progressive radiosity method. We have analyzed the performance benefits of the partitioning strategy in terms of execution time and the reduction of cache misses. We have also verified that the quality of the final images is not affected by the techniques employed.

To evaluate our proposal we have used an Athlon 64 3.2 GHz with a 64 + 64 KB L1 cache, a 512 KB L2 cache and a 1 GB RAM. We have tested our proposal with different scenes. Here we include the result for three representative scenes, two of them depicted in Figure 4: *room-model* with 5306 patches (see Figure 4.a), *armadillo* with 3999 patches (see Figure 4.b) and *largeroom-model* with 7070 patches. The last scene, not included in the figure, is a larger *room-model* scene with more objects.

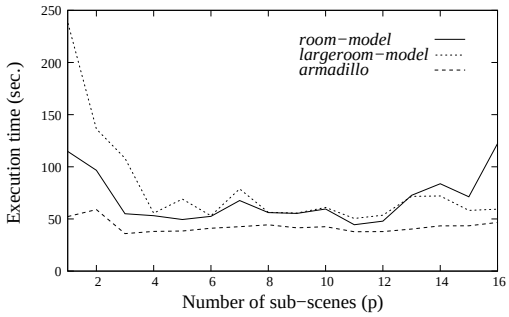


Figure 5: Execution time for *room-model*, *armadillo* and *largeroom-model*.

First, we evaluate performance in terms of execution time. The results for the test scenes on the *Athlon* system are depicted in Figure 5. No data for $p > 16$ are included as execution time increases in all cases. We observe that the partitioning strategy achieves a reduction in the required time when p is incremented for low p values, but the times increase for large p values. For example, we observe that for the *largeroom-model*, the execution time for the original algorithm is 238.63 seconds, while for the partitioned version with $p = 4$ this time is lowered to 55.54. Therefore, a high speedup of 4.30 is obtained for this example. This large reduction in the execution times is mainly due to two reasons: Firstly, the reduction in the cache misses due to data locality exploitation associated with the partitioning strategy and, secondly, the visibility mask technique, required to make the partitioning strategy viable, also has an inherent reduction in the computational time. This visibility technique implies an approximation to the visibility determination and, as a consequence, the reduction of the computational complexity.

To verify the reduction in cache misses we have chosen a hardware measurement [Amm97a] that can accurately obtain a broad range of performance metrics for a program. Specifically, we have used the PAPI (*Performance Application Programming Interface*) [Moo01a] library that has the benefit of a cross-platform interface to the counters, allowing the maintenance of a common source for a wide variety of architectures. Figure 6 shows the number of cache misses for the *Athlon* system for our test scenes for the first level (see Figure 6.a) and for the second level (see Figure 6.b). For reasons of clarity the data miss rates are also depicted: for the first cache in Figure 7.a and for the second cache in Figure 7.b.

As can be observed in Figure 6.b and Figure 7.b the sub-scenes fit into the second cache level for the three scenes even for small p values. This can be deduced from the figures as increments in the p values imply reductions in the number of cache misses and the cache miss rate. This way, smaller sub-scenes fit into the memory and the data can be reutilized, reducing the

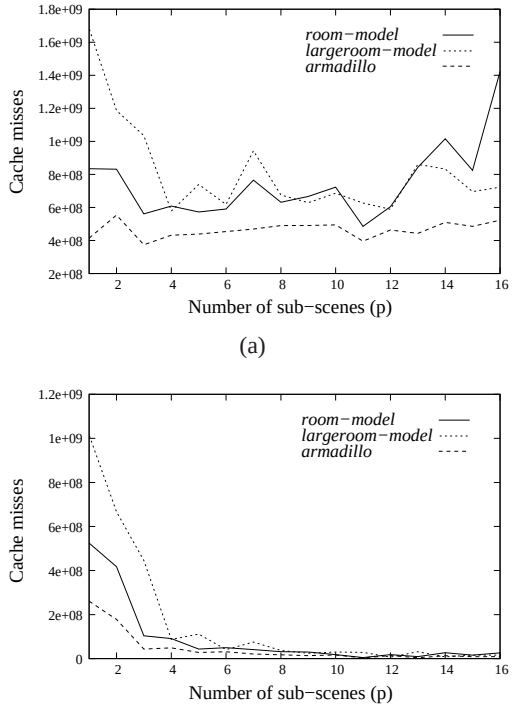


Figure 6: Cache misses (a) first level (b) second level.

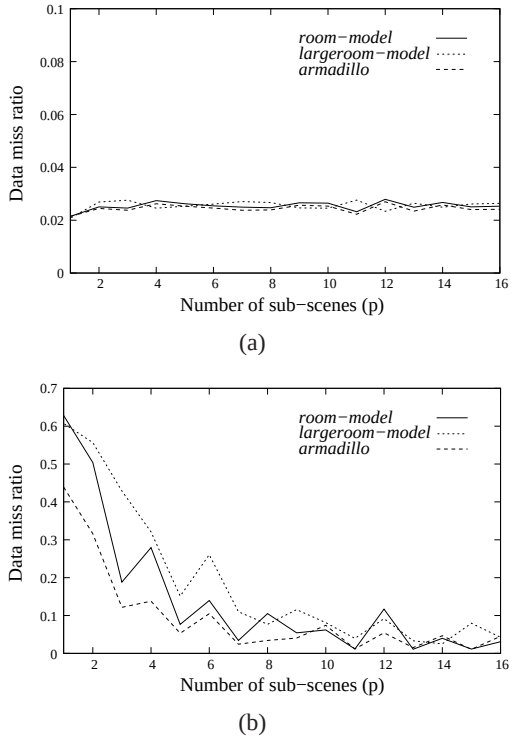


Figure 7: Data miss ratio (a) first level (b) second level.

data traffic between memory levels. Note that, as can be observed in these figures, large p values do not eliminate the cache misses. These are compulsory misses, i.e. misses associated with the initial sub-scene loading each time a new sub-scene is processed.

With respect to the first cache level, Figure 7.a indicates that the cache miss rate is practically constant while Figure 6.a indicates that the number of cache misses decreases for intermediate p values. This means that there is a decrease of references for these p values. The reduction of loads and stores in the program is due to two reasons: the blocking helps register allocation and the utilization of visibility masks minimizes the accesses in Loop2.

The visibility masks technique has an important influence in the behaviour of the algorithm. This is due to the fact that, when a visibility mask of a patch has only zeros (line 23 of Figure 3), the patch is not further gathered to the other partitions. This contrasts with the original algorithm, in which interactions of this patch with all the patches of the scene would be tested, with the consequent increment in the cache misses. On the other hand, the utilization of visibility masks reduces the number of accesses to determine the visibility between two polygons. We have used a method based on directional techniques which calculates a list of potential candidates to occlude the interaction between two polygons. The idea is to remove from the list those patches which cannot be pierced by any ray cast between both polygons. Then, to determine the visibility between patches it is not necessary to access the list of patches that belongs to other sub-scenes and this permits a notable reduction of the accesses. At the same time, the computational requirements associated with the visibility mask technique are dependent on the p value. For low p values, the overhead associated with the visibility masks technique is clear, mainly for scenes with small number of triangles (see execution time values for two partitions of the *armadillo* in Figure 5). For large p values the computational requirements associated with the large number of patches that has to be "copied" to the other scenes can be prohibitive. Note that each time a patch is "copied" its visibility mask has to be computed/updated. As a result, an increment in number of sub-scenes implies an increment in the number of patches that have to be gathered to other sub-scenes.

In summary, the execution time of the modified algorithm is minimum for a given intermediate p value. The influence of the p value can be also deduced from Equation 7. For larger p values the last term of equation is larger while the t_3 is smaller ($\overline{M^l}$ is smaller). In consequence, the optimum p value depends on the scene, however, we have found in our tests that a $p \in [4,6]$ is a good partitioning value for the tests we have performed. Larger scenes would result in higher p values. With this optimum p value, the data locality is exploited because the sub-scenes fit into the second cache level. However, larger p values imply an increment in the computational

complexity associated with the massive visibility masks computation and, as a consequence, an increment in the processing time.

Finally and with respect to the quality of the final images, we have not found any noticeable difference between the images generated with the standard algorithm and those generated with the modified algorithm. To further test the quality of the resulting images, we have used the mean residual error of the radiosity as the error metric. The numerical results indicate that the quality of the images is close to being the same for the original and the modified algorithms. For example, the result for the *largeroom-model* for the original algorithm is 0.000209 while for the partitioned version with $p = 4$ this error is 0.000240. In both cases the residual errors are very small. The visibility mask technique permits the elimination of computations and avoids testing the interaction between many pairs of patches. As an example, as was previously mentioned, a visibility mask with all zeros discards the interaction of the associated patch with any other patch in the following sub-scenes. However, although the number of visibility tests are reduced, no modifications are actually performed in the nature of the algorithm. Consequently, no changes in quality are expected.

6. CONCLUSIONS

In this paper we have proposed a technique for accelerating the progressive radiosity algorithm based on data locality exploitation. Specifically, we have employed a scene partitioning technique that permits the reduction of cache misses and, with this, the reduction of the execution time of the algorithm.

The simplicity of the partitioning strategy employed, permits an efficient partition of the scene with a very low computational complexity. On the other hand the visibility mask technique we have used allows the computation of the visibility information between patches in different sub-scenes. We have shown that this technique not only solves the problem of computing the visibility in a partitioned scene, but also reduces the visibility computation requirements without noticeable loss of quality in the final images.

Therefore, our modified progressive radiosity algorithm permits the obtaining of important reductions in the execution time with respect to the original algorithm, while keeping its quality properties. The benefits of our proposal have been experimentally evaluated for a set of representative scenes.

As future work, we plan to extend our analysis to a non-uniform partitioning strategy such as that proposed in [San05a]. We also plan to extend the analysis to other radiosity strategies, for example to the hierarchy radiosity algorithm [Han91a].

ACKNOWLEDGEMENTS

This work was partially supported by the Ministry of Science and Technology of Spain under contract MCYT TIN2004–07797–C02 and by the Secretaría I+D de Galicia (Spain) under the project PGIDT03TIC10502PR.

REFERENCES

[Amm97a] Ammons, G., Ball, T., and Larus, J. Exploiting Hardware Performance Counters with Flow and Context Sensitive Profiling. In *ACM SIGPLAn'97 Conf. Programming Language Design and Implementation (PLDI'97)*, pp. 85–96, 1997.

[Amo04a] Amor, M., Sanjurjo, J.R., Padrón, E.J., and Doallo, R. Progressive Radiosity Method on Clusters Using a New Clipping Algorithm. *Int. J. of High Performance Computing and Networking (IJHPCN)*, Vol. 1, Nos. 1/2/3, pp. 55–63, 2004.

[Arn96a] Arnaldi, B., Priol, T., Renambot, L., and Pueyo, X. Visibility Masks for Solving Complex Radiosity Computations on Multiprocessors. In *Proc. of the First Eurographics Workshop on Parallel Graphics and Visualization*, pp. 219–232, 1996.

[Cho06a] Choi, M.-H., Park, W.-C., Neelamkavil, F., Han, T.-D., and Kim, S.-D. An Effective Visibility Culling Method Based on Cache Block. *IEEE Transactions on Computer*, Vol. 55, No. 8, pp. 1024–1031, 2006.

[Coh88a] Cohen, M., Chen, S.E., Wallace, J.R., and Greenberg, D.P.. A Progressive Refinement Approach to Fast Radiosity Image Generation. *ACM Computer Graphics (Proc. of SIGGRAPH '88)*, Vol. 22, No. 4, pp. 31–40, 1988.

[Coh93a] Cohen, M. and Wallace, J.R. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, 1993.

[Gar00a] Garmann, R. Spatial Partitioning for Parallel Hierarchical Radiosity on Distributed Memory Architectures. In *Proc. of the Third Eurographics Workshop on Parallel Graphics and Visualization*, Girona, Spain, September 2000.

[Gho99a] Ghosh, S., Martonosi, M., and Malik, S. Cache Miss Equation: A Compiler Framework for Analyzing and Tuning Memory Behavior. *ACM Trans. Programming Languages and Systems*, Vol. 21, No. 4, pp. 703–746, 1999.

[Gib00a] Gibson, S., and Hubbold, R.J. A Perceptually-Driven Parallel Algorithm for Efficient Radiosity Simulation. *IEEE Transactions on Visualization and Computer Graphics*, Vol. 6, No. 3, pp. 220–235, 2000.

[Gue03a] Guerra, A.P., Amor, M., Padrón, E.J., and Doallo, R. A High–Performance Progressive Radiosity Method Based on Scene Partitioning. *Lecture Notes in Computer Science*, Vol. 2565, pp. 537–548, 2003.

[Hai94a] Haines, E.A., and Wallace, J.R. Shaft Culling for Efficient Ray-Traced Radiosity. In *Proc. of Photorealistic Rendering in Computer Graphics (Second Eurographics Workshop on Rendering)*, pp. 122–138, 1994.

[Han91a] Hanrahan, P., Saltzman, D., and Aupperle, L. A Rapid Hierarchical Radiosity Algorithm. In *Computer Graphics (SIGGRAPH'91 Proc.)*, pp. 197–206, 1991.

[Kaj86a] Kajiya, J.T. The Rendering Equation. *Computer Graphics*, Vol. 20, pp. 143–150, 1986.

[Moo01a] Moore, S., and Smeds, N. Performance Tuning Using Hardware Counter Data. In *SuperComputing 2001*, 2001.

[Pad01a] Padrón, E., Amor, M., Touriño, J., and Doallo, R. Hierarchical Radiosity on Multicomputers: a Load–Balanced Approach. In *Proc. of the Tenth SIAM Conference on Parallel Processing for Scientific Computing*, 2001.

[Pad03a] Padrón, E.J., Troncoso, R., Amor, M., Sanjurjo, J.R., and Doallo, R. Estudio Comparativo de Algoritmos de Visibilidad en los Modelos de Iluminación Global. In *XII Congreso Español de Informática Gráfica (CEIG 2003)*, pp. 141–154, 2003.

[San05a] J.R. Sanjurjo, J.R., Amor, M., Bóo, M., and Doallo, R. Parallel Global Illumination Method Based on a Non-Uniform Partitioning of the Scene. In *Proc. of the 13th Euromicro Conference on Parallel, Distributed and Network-based Processing (PDP'05)*, pp. 251–257, 2005.

[Sil94a] Sillion, F.X., and Puech, C. *Radiosity. Global Illumination.* Morgan Kaufman Publishers, Inc., 1994.