Empirical Evaluation of Hierarchical Higher Order Face Cluster Radiosity

Leonardo Spanò and Enrico Gobbetti

25th February 2003

TR-xxx-yyy

Visual Computing Group

CRS4

Sesta Strada Ovest ZI Macchiareddu

09010 Uta

Italy

Contents

1 Introduction

1.1 Purpose

We want to investigate how our hierarchical higher order vector radiosity system *radiator [8][3]* behaves in terms of time and memory requirements under various conditions.
The tests were aimed to verify that our software has the following properties:

i. Vector irradiance is more effective than scalar radiosity when the environments to solve have an illumination detail coarser than geometric detail;

ii. Higher order bases significantly reduce the number of energy transfer links to compute;

iii. Shooting saves memory, for the decreasing number of links after a few iterations;

iv. It is possible to solve and display glossy enviroments at interactive frame rates.

1.2 Method

In order to have environments with illumination less complex than geometry we built a few test scenes using mainly non-planar objects each made up of hundreds of thousands connected triangles. For properties (i) and (iii) we compared our method against the volume clustering [6][7] scalar radiosity method as implemented in the public Renderpark[5] package. To verify property (iii) the comparison was only meant to show the difference in number of links with respect to the gathering method. For the last property, we associated a glossy material to some of the objects making up the test scenes and we built a multithreaded program running the higher-order solver in parallel with a viewer exploiting recent programmable graphics hardware in order to shade the polygons using non-constant bases for irradiance and full BRDF.

For an easy reproduction of our tests, we used publicly available complex objects[2] and Greg Ward's MGF format to compose the scenes.
To study geometric scalability, we prepared different versions of the same complex scene, each one with a different polygon count.
Together with geometric complexity, the parameters we changed in the simulations were the error threshold and irradiance basis order.

In the next section we detail the test framework. In Section 3 we show the various results obtained with radiator compared, where possible, to renderpark. In the same section we visually compare radiator's solutions to renderpark's. The conclusions follow in Section 4.

2 Test framework

2.1 Introduction

Our tests aimed to investigate time and memory performance of the hierarchical higher order vector radiosity system we have implemented for processing realistic illuminated indoor scenes containing highly tessellated objects. The measures interesting to us are time, number of energy transfer links, memory usage, number of solution leaf elements and overall capabilities with glossy scenes. We also wanted to grasp the performance of different orders of irradiance basis and to stress the importance of the *decouple-light-from-geometry* approach by comparison with an older standard method (we chose volume clustered Galerkin radiosity as implemented in the Renderpark software).

With the above goals in mind we varied some of the possible input parameters, namely:

- *Geometric complexity.* This is represented by the number of input primitives (triangles or quads) in each scene. The details for each scene are in section 2.2. The same scenes were tried with renderpark, although we interpolated some results for intermediate complexity scenes in order to speed the comparison process.
- *Relative error threshold.* It is the fraction of the total emitted power in the scene. The values we used are $10^{-4}, 10^{-5}, 10^{-6}$. With Renderpark we tried only a subset of the possible scene/threshold combinations.
- *Order of irradiance basis.* In our software, the tests were run using constant, linear, quadratic and cubic bases for the vector irradiance distribution while we left the standard scalar radiosity settings in renderpark.

Table 1 lists the type of experiments we made. With radiator, the order of radiosity basis was constant in all runs while irradiance was approximated with constant, linear, quadratic and cubic bases. In renderpark, we left the default linear basis. Beacuse of the coupling between visibility samples and kernel integration in radiator, we also ran some tests disabling visibility queries in order to emphasize the performance of higher order bases. We did not analyse other input factors such as number of light sources, number of complex objects and reflectance.

We chose to compare our solver with renderpark's volume clustered scalar Galerkin radiosity for its wide diffusion and to emphasize the efficiency of a higher-order vector irradiance approach to the global illumination problem. The measures compared are time and number of energy transfers. The comparison is not particularly fair because of the intrinsic difference in the methods and because the two packages do not share the same code basis (e.g. the visibility queries are computed using two completely different methods).

To quantify the efficiency of our refiner, we analyzed the number of energy transfers and the number of leaf elements in a solution hierarchy of illuminated patches. For a given error threshold, a small number of transfers and leaves is obviously desired, meaning that no over-refinement happened.

In what follows we describe in greater detail the composition of the scenes and how the test were performed.

vs→	Geometric complexity	Error threshold	Iteration	Compared to Renderpark (volume clustering)
Time	with and w/out visibility	yes	yes	vs error thresh; vs geometric complexity
Energy transfers	yes	yes	small/medium/large error threshold	geometric complexity
Memory	yes	yes	yes	-
Leaf elements	yes	yes	yes	-
Glossy scenes	-	-	-	qualitative

Table 1: The experiments made. Radiator was run with constant, linear, quadratic, cubic vector irradiance basis for 10 iterations.

2.2 Test scenes

The scenes we tested are nine different versions of a museum scene similar to Willmott's[10]. Our museum contains three complex scanned models, one curvy pedestal and one planar light source in the middle of the ceiling. The scene is shown in **Figure 1**. The various scenes differ in the number of faces only, roughly ranging from 164000 to 1500000. All the scanned models are freely available[2] and the scenes are written in the standard MGF format. In **Table 3** the contents of each scene are listed.

Figure 1: The museum scene used in the tests.

Building the scenes The scene with $id{=}n$ has approximately $n \cdot 164000$ faces. The models in the various scenes were obtained one by one from the full resolution mesh using Garland's qslim[4] simplification tool with explicit number of target faces, compactness ratio 1 and default values for the other parameters. The memory footprint of a scene is the storage reserved to geometry, materials and pointers to nodes for the whole set of hierarchical surfaces in the scene. Note that the museum walls are not given as highly tessellated objects, they are refined as needed at runtime.

Materials The materials used for the three statues also have directional reflectance, which is taken into account by our solver and displayed using hardware programmable shading facilities.

	$\rho_d = .5, \rho_s = .15, roughness = .2$			$\rho_d = .25$	$\rho_d = .75\,(grey), .4\,(green), .14\,(red)$		
Scene ID	Buddha	Isis	Igea	Pedestal	Walls and source	Total input faces total faces	Memory Footprint [KB]
1	100000	37500	26800	510	7	164816	55691
2	200000	75000	53600	1020	7	329624	111382
3	300000	112500	80400	1530	7	494431	167073
4	400000	150000	107200	2040	7	659237	222765
5	500000	187500	134000	2550	7	824027	278451
6	600000	225000	160000	3060	7	988009	333863
7	700000	262500	187000	3570	7	1153009	389620
8	800000	300000	214000	4080	7	1317975	445366
9	900000	337000	241200	4590	7	1482474	500955

Table 3: Composition of the test scenes. Diffuse and specular reflectances are indicated above the object name. The light source emits 20000 cd. (A face is a triangle or a rectangle, KB=1000 bytes).

2.3 Runtime settings

In **Table 4** are listed some of the world parameters after loading. Before running the tests, we ensured that these parameters were the same in our software and in renderpark. The average reflectance in the scenes is 0.67 and the total emitted power is 109.25 W.

Scene ID	Total area	Estimated Avg. Radiance [W/m^2/sr]
1	148.161	0.718309
2	148.200	0.718365
3	148.234	0.718766
4	148.243	0.719497
5	148.206	0.721137
6	147.998	0.723780
7	148.014	0.727345
8	147.699	0.719352
9	146.720	0.733591

Table 4: World statistics after loading. In renderpark the values are the same.

2.3.1 Radiator settings

The input parameters interesting to us were given as command line options. The settings common to all the radiator runs are listed below:

- (Scalar) Radiosity basis order = constant
- Maximum iteration = 10
- Minimum residual ratio = 0

The minimum residual ratio was set to zero to run exactly ten iterations in each test. The following is an example of radiator invocation:

```
radiator --solver-radiosity-basis constant --solver-irradiance-basis linear
 --solver-relative-transport-error-threshold 0.00001 --solver-maximum-iteration 10
 --solver-minimum-residual-ratio 0 --output-radiosity-statistics-file scene_id_stat.txt
 scene_id.mgf
```

The input parameters explored with radiator are shown in **Table 5**. In the following we will indicate as "default test set" the tests described by the first row of the table.

Scene ID	Relative error threshold	Order of vector irradiance basis	With visibility
$1 \div 9$	$10^{-6}, 10^{-5}, 10^{-4}$	constant, linear, quadratic, cubic	yes
$1 \div 9$	10^{-5}	constant, linear, quadratic, cubic	no

Table 5: The space of input parameters spanned by radiator experiments. The first row indicates the "default test set".

2.3.2 Renderpark settings

The settings common to all the renderpark runs are indicated below:

```
-radiance-method Galerkin
-gr-iteration-method Jacobi
-gr-hierarchical
-gr-link-error-threshold 1e-5
-gr-lazy-linking
-gr-clustering
-gr-link-error-threshold
-gr-no-importance
```

The above settings select the Galerkin scalar radiosity solver with volume clustering. Theoretically this method is intrinsically less performant than ours in both space and time, because the workload strongly depends on the input geometric complexity rather than on illumination gradients. We chose such settings to easily emphasize the efficiency of the face clustering approach. We expect sub-quadratic performance from renderpark tests.

The parameters we changed with renderpark were the input scene and the error threshold (**Table 6**). The combinations we chose are not intended for a comprehensive evaluation of renderpark, but only to make some significative comparisons with our software.

Scene ID	Relative error threshold
$1 \div 9$	10^{-5}

Table 6: The space of input parameters spanned by the renderpark experiments. All the runs were done with the default Galerkin radiosity settings. The results for scenes 6,7,8 were obtained by interpolation.

2.4 Measurements

Radiator tests were all run for ten iterations, although we observed that the visual convergence is usually reached before the sixth iteration. Renderpark tests were run for ten iterations too.

2.4.1 Time

The time measurements in radiator and renderpark refer to the total CPU time. For both programs, we report the cumulative time at the tenth iteration.

2.4.2 Energy transfers

Given a scene, we define number of energy transfers as the sum of the established links between elements over ten iterations. This measure is compared to the cumulative number of interactions in renderpark (i.e. the sum of interactions between volumetric clusters, between surfaces, surface to cluster and cluster to surface). We want to check that the number of transfers our method generates is not constant over the iterations and that is vanishes at the end of the lighting simulation.

2.4.3 Solution memory

To study memory usage, we consider the solution memory footprint. The memory footprint is the amount[1] of storage needed to describe an irradiance distribution for all the distinct objects in the scene. For each object we sum up the memory reserved to the following properties:

- hierarchical structure (tree of illuminated nodes)

[1] In the graphs, KB stands for 1000 bytes.

- the vector irradiance distribution at the leaves of the tree
- ΔB and $\Delta B'$ (current unshot radiosity, next iteration unshot radiosity)
- the lists of potentially occluded and unoccluded shooters

2.5 Leaf elements

The output of our hierarchical higher order solver is a hierarchy of illuminated patches[3]. Leaf nodes in the hierarchy are different from inner ones in that they have associated irradiance vectors representing the irradiance distribution. This distribution is represented with a number of samples depending on the order of the approximating basis. Measuring such leaves gives information about the quality of the solver oracle and expressive power of the chosen basis. We measure the total number of leaf elements at each iteration.

2.6 Hardware and software platform

All the tests were performed on a dual AMD Athlon1900+ system running Redhat Linux 8 (kernel 2.4.18) equipped with 2GB of RAM and NVidia Geforce4 4600Ti 128MB graphics board.

3 Results

In this section we show and comment results about time, memory, energy transfer links, leaf elements and glossy scenes. For our program *radiator*, we always show measurements obtained using constant, linear, quadratic and cubic bases for vector irradiance. The various measures are plotted against scene complexity, relative error threshold and iteration number. The error thresholds used to test our software are listed in **Table 5**. Some radiator tests were done without visibility queries, to stress the efficiency of the various bases.

Where meaningful, we made comparisons with renderpark: the performance of our method is opposed to the default linear scalar radiosity approximation found in renderpark's Galerkin solver with volume clustering. For tractability, we used the scenes and the settings indicated in **Table 6**. All the experiments were run for ten iterations.

3.1 Time

A crucial measure for any radiosity solver is the running time of a simulation. With the large scenes and cheap powerful graphics facilities available today, a naive solver could not handle such complexity, otherwise the users could find the software pointless for interactive applications. In our software, the visibility queries are coupled to the kernel integration, making the impact of visibility queries important, thus we also conducted the same experiments disabling visibility queries.

Geometric complexity. To see how our method depends on the geometric complexity of the input scene, we ran the default test set. The tests with error threshold[2] 10^{-5} were used for the graph in **Figure 2 (left)**. The graph confirms that hierarchical vector radiosity is independent on the geometric complexity of the input scene when this is detailed enough, as Willmott emphasized in his work[10]. All the tested bases have very similar convergence rates, with constant basis a little faster than the other bases. Quadratic and linear basis have present very similar running times.

Impact of visibility. The graph in **Figure 2 (right)** confirms that the performance is much better (about 10 times on average!) when ignoring inter-object occlusion. This result also stresses the importance of higher order bases: without visibility queries in our scenes, quadratic irradiance basis is more than two times faster than constant basis, while linear and cubic are still

[2]Looking at the images produced in the various experiments, we found that 10^{-6}, 10^{-5}, 10^{-4} are respectively high, medium and low precision relative error thresholds for our scenes.

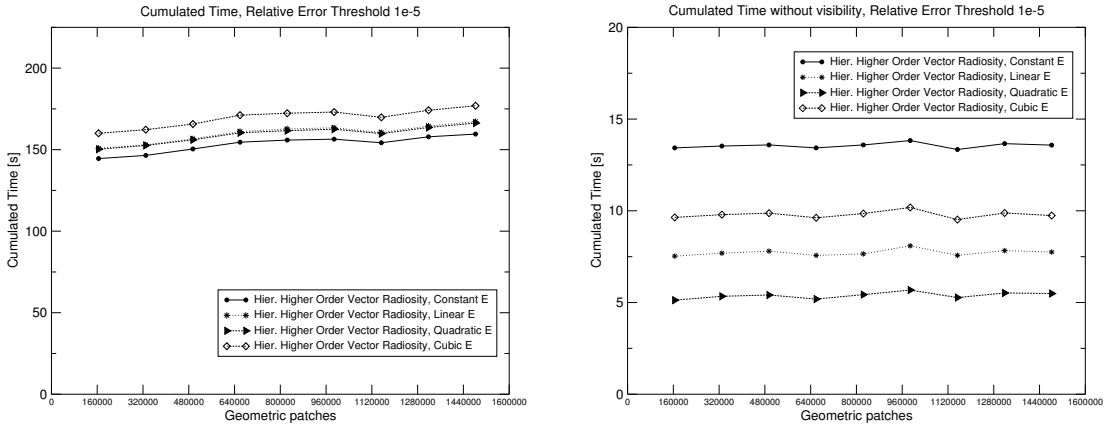

Figure 2: Time. The running time of the solver is independent on geometric complexity. Left: with visibility queries; right: without visibility.

significantly slower than quadratic. The fact that higher order bases are faster without visibility sampling is explained noting that in our system, the number of visibility samples is coupled to kernel integration.

Accuracy. The dependence of time on relative error threshold is shown in **Figure 3**. We ran scene 5 with the different irradiance bases and three relative error thresholds. The results show that the three bases perform in the same way. It is also clear that a threshold ten times smaller requires a time about 10 times longer. Higher order bases appear slightly slower for very small thresholds. Obviously, in choosing an approximating basis, it is important to also consider the rate of qualitative convergence, as shown in Section 3.5.

Time trend during a simulation. Radiosity solvers have a workload proportional to the number of links to evaluate. The number of links in turn grows with the geometric configurations of the (potentially) involved patches and with the energy exchanged over the links. Our system uses a shooting approach, thus we expect a bigger amount of work during the first iterations, for the shooted energy decreasing between successive iterations. This behaviour is confirmed by **Figure 4**, where the slowest iteration is the second, with any basis. Higher order bases are a bit slower than lower order.

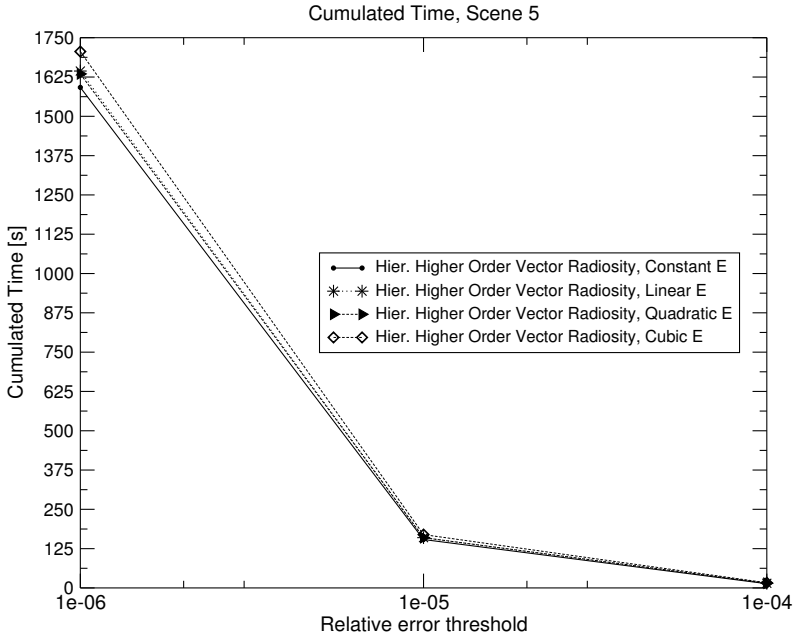

Figure 3: Time. The various bases behave similarly.

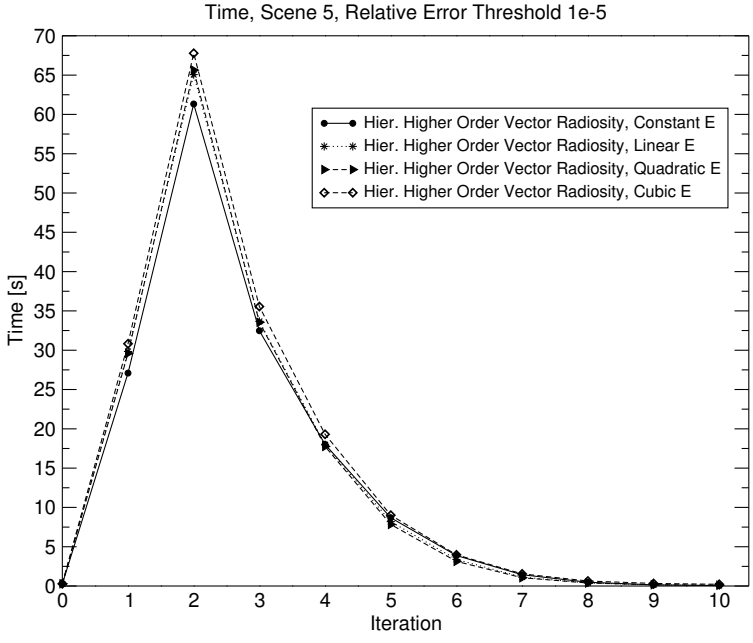

Figure 4: Time. As expected, the shooting approach spends more time at the beginning of the simulations.

Comparison with renderpark. The comparison with renderpark's volume clustering was

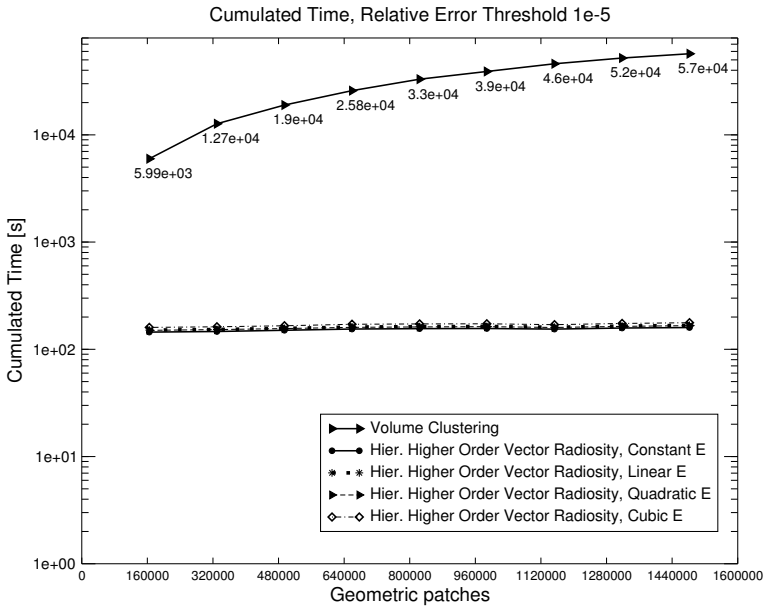

Figure 5: Time. Comparison with volume clustering. Volume clustering strongly depends on input geometric complexity.

made running all the scenes with error threshold 10^{-5}. The results are shown in **Figure 5**. We compared the cumulated time after ten iterations of our solver to ten iterations of renderpark. Renderpark presents a noticeable linear dependence on the scene complexity. From the graph, it is evident that even the solution of the simplest scene is one order of magnitude slower than radiator. Volume clustering takes from 6000 to 57000s for solving the museum scene at various resolutions, while radiator solves any scene in less than 180s. The constant irradiance basis is the fastest, taking 144s instead of the 177s needed by the cubic basis. Linear and quadratic basis are in between.

3.2 Energy transfers

Probably the most important quantity when solving a scene with given precision constraints is the number of links over which energy is exchanged and evaluated. Time and memory actually are strongly dependent on this key measure. Moreover, the energy transfers help to judge the efficiency of the refiner and approximation bases. In what follows we describe our experiments related to energy transfers.

Geometric complexity. The default test set with error threshold 10^{-5} was used to obtain the results plotted in **Figure 6**. The cumulative number of transfers is shown against the number of input primitives, exhibiting again a constant trend over the whole range of scenes. The best performing irradiance basis order is quadratic, closely followed by linear and cubic basis. The most expensive basis is constant, with an average overhead of about 20% transfers with respect to quadratic.

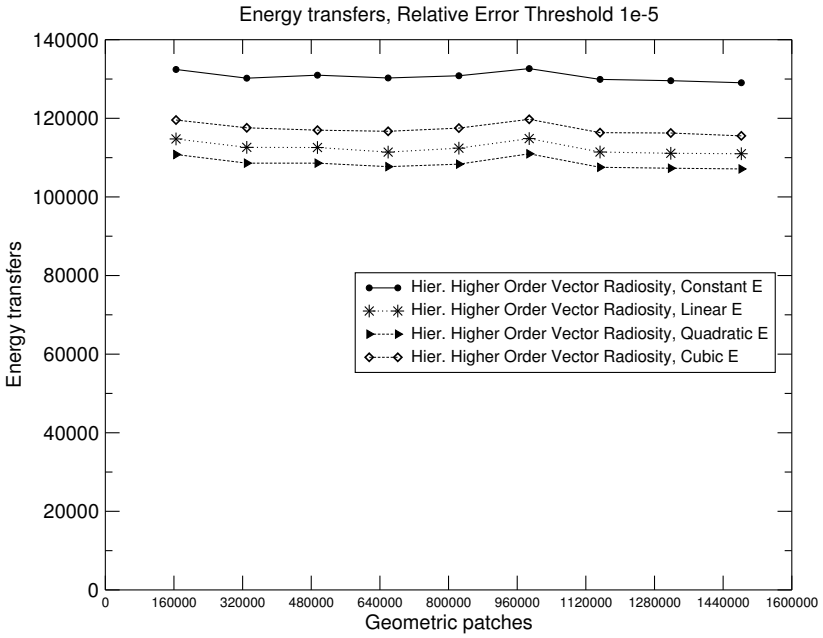

Figure 6: Energy transfers. The number is independent on geometric complexity.

Accuracy. The graph shown in **Figure 7** shows how the order of irradiance basis influences the number of links in Scene 5. The number of transfers is plotted against three error thresholds, giving a good overview. Over the whole range of thresholds, quadratic linear basis gives the smallest number of transfers, closely followed by linear and cubic. Constant basis is significantly more expensive (about 20% more than quadratic with any threshold). Such savings are independent on scene complexity, as seen in Figure 6.

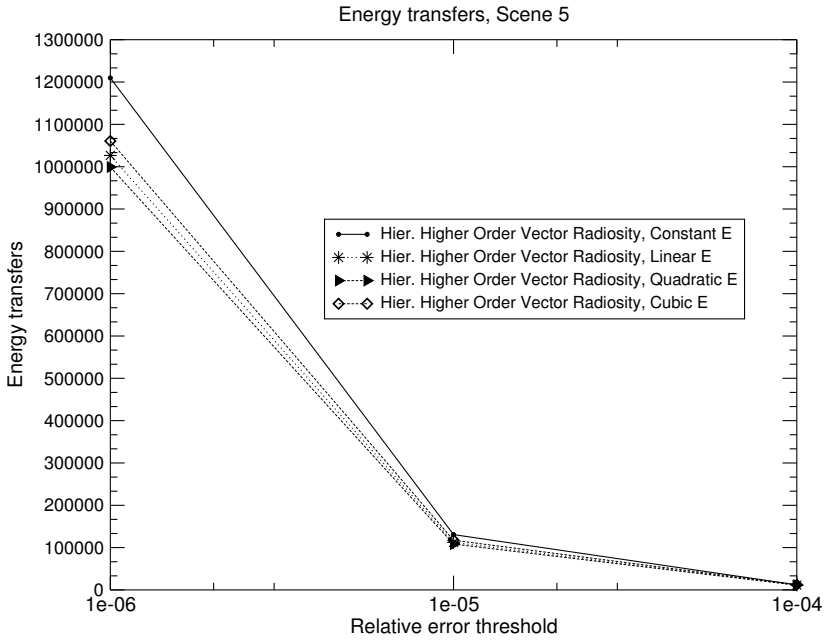

Figure 7: Energy transfers. Comparison between basis: as expected simpler bases generate more energy transfer links.

Trend of transfers during a simulation. This experiment points out how the number of energy transfer links changes during a simulation. We tried various basis orders and two error thresholds (medium and high precision) with Scene 5 (no other scenes need to be studied for the independence of transfers on scene complexity). The graphs in **Figure 8** show the number of transfers plotted against the iteration. Both runs have similar behaviour: the number of transfers grows, reaching the maximum value at the second iteration, and then diminishes to a positive minimum. As expected, the maximum number of links depends on the error threshold: the precise threshold generates a peak number of transfers about 7 times bigger than the medium precision threshold. The minimum value of links reached in the final iterations is a fraction of the squared number of root patches in the scene, which in turn depends on the average degree of occlusion between the objects. In these experiments, quadratic, linear and cubic bases perform in the same way, generating significantly less links than constant basis. In the example shown in Figure **8**, quadratic basis saves about 20% of the constant basis transfers, with both thresholds.

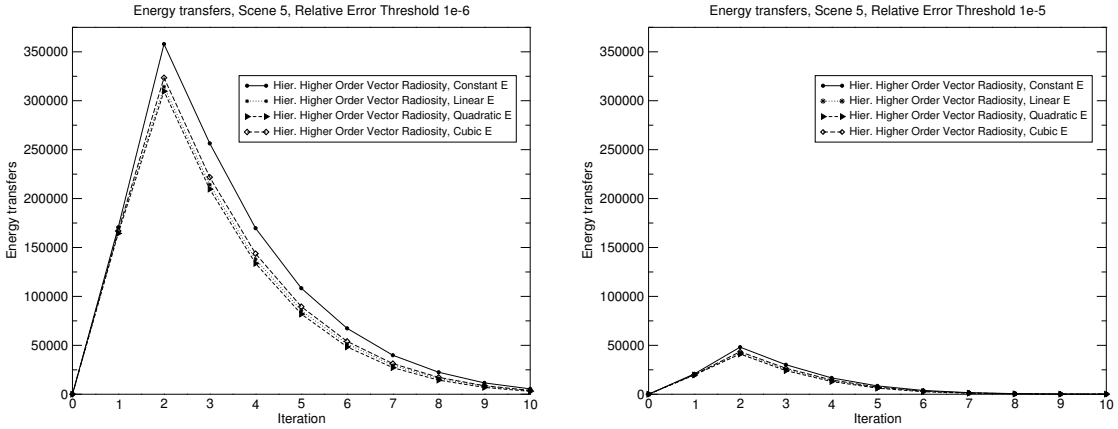

Figure 8: Energy transfers in Scene 5 plotted against iteration number with small (left) and medium (right) error threshold. In both experiments, quadratic irradiance basis saves about 20% of the total number of transfers generated by the constant basis.

Comparison with renderpark. To see how many transfers are avoided with respect to volume clustering, we used Scene 3 and threshold $10^{-5}, 10^{-4}$ to make the comparison. **Figure 9** shows that our method takes significantly less transfers, namely one order of magnitude less than renderpark. Again, this is due to the very different approaches of the methods.

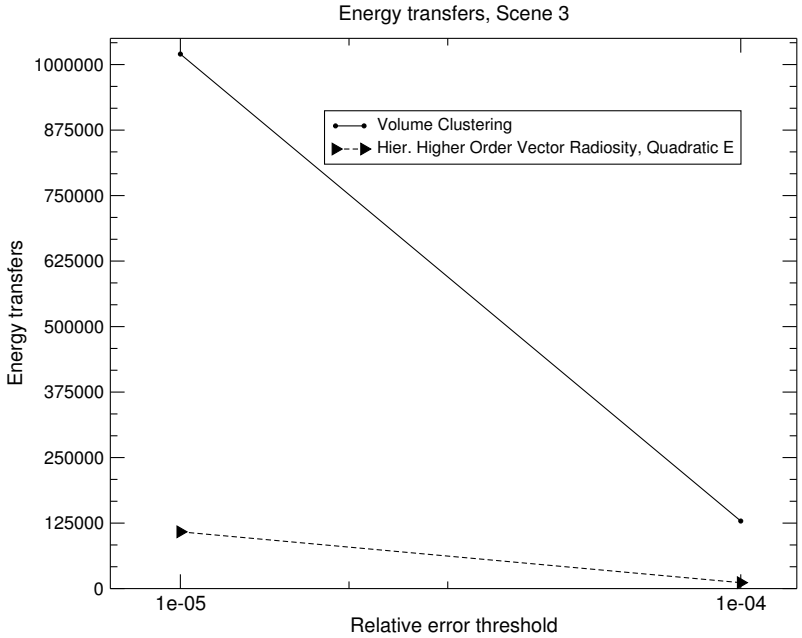

Figure 9: Energy transfers. Comparison with renderpark's volume clustering with changing target precision.

3.3 Solution memory

Because of the heavy storage needs of higher order approaches[9], we wanted to be sure about a careful handling of the memory used in our system. In the following we show results about the memory needs for the irradiance distribution on the objects. (The solution memory is defined in section 2.4.3).

Geometric complexity. To see memory usage we tried radiator on the standard test set. The solution memory footprint at convergence with error threshold 10^{-5} is presented in **Figure 10**. Storage needs for the solution is independent on the input geometric complexity and it is always a small fraction of the input memory footprint. The different bases require significantly different amounts of memory: constant irradiance obviously wins, while linear, quadratic, cubic bases require two, four and eight times the amount needed by constant basis, respectively. In our scenes the storage needs are always sorted like the order of the correspondent irradiance basis. Other authors[1] found different results with different scenes.

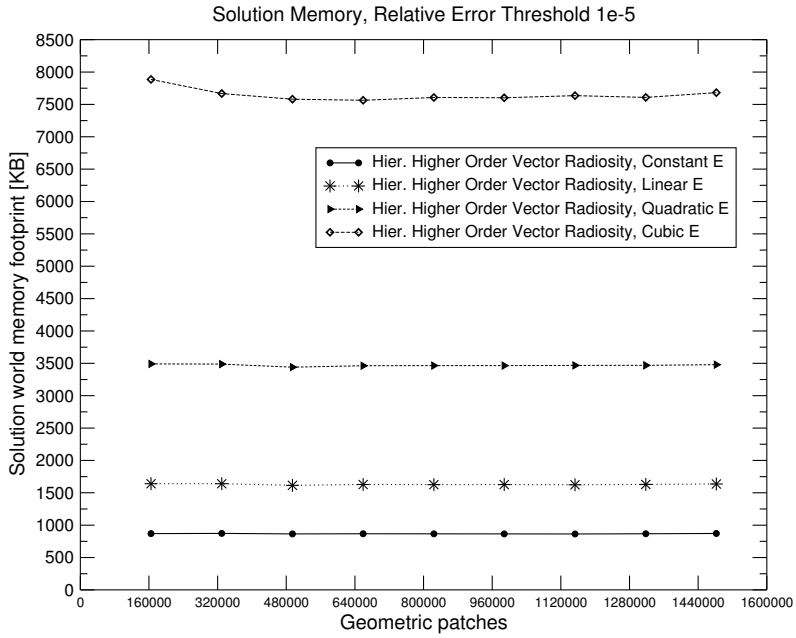

Figure 10: Solution memry. Storage needs are independent on scene complexity.

Accuracy. When changing the target precision, the differences in memory needs are significant. Plotting the results for Scene 5 in **Figure 11**, we notice that higher order bases are a concern for high precision simulations, suggesting to use quadratic or linear basis for small thresholds.

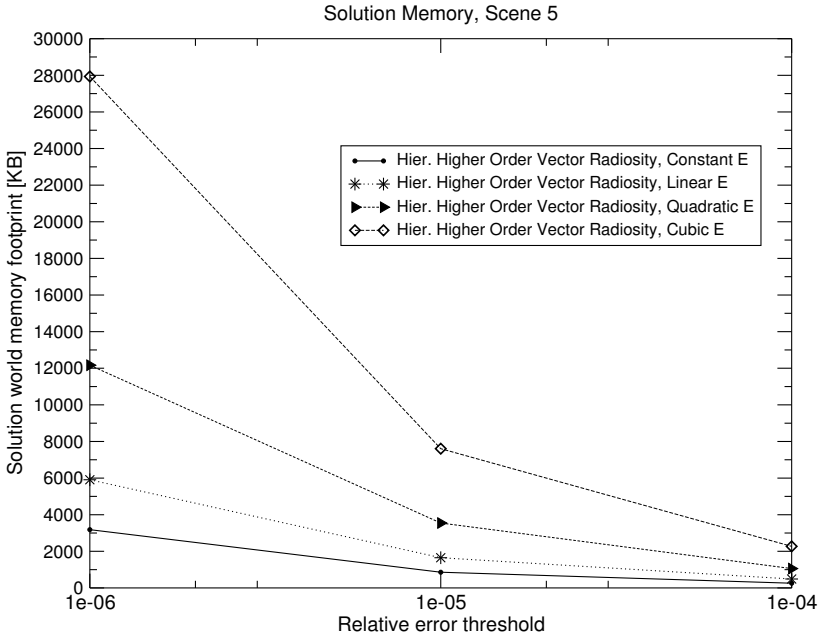

Figure 11: Solution memory. Trade-off for the various bases with respect to target precision.

Memory trend during a simulation. The use of memory during the iterations of the solver is shown in **Figure 12**. In our scenes, the used memory grows in the first two iterations, becoming constant from the second iteration on, where no more refinement happens. Similar behaviours are present in all the test scenes and with different thresholds. The early convergence is due to the shooting nature of our solver and obviously on the characteristics of our scene.

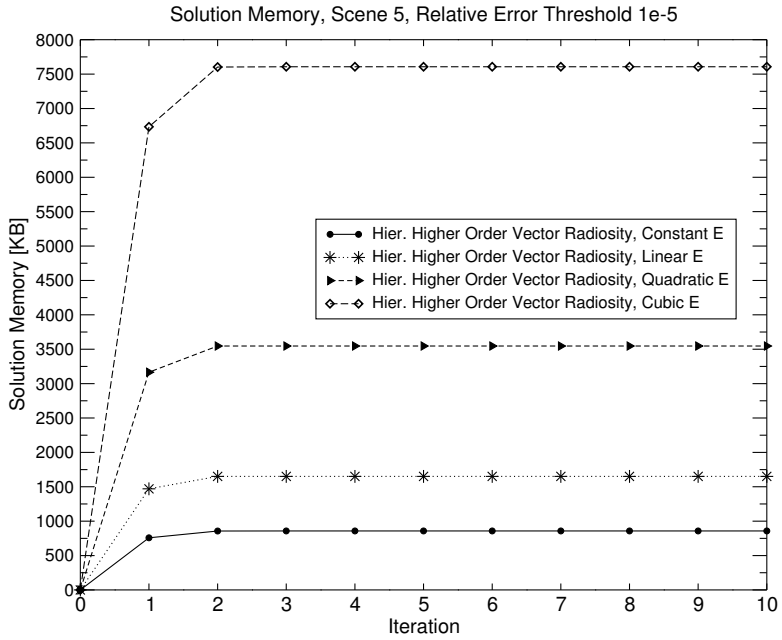

Figure 12: Solution memory. After the first iterations, the used storage stops growing.

3.4 Leaf elements

A measure of quality for the solver oracle and for the different bases is the number of leaf elements at convergence time. A good solver should solve a scene with a reduced number of leaves, for their impact on storage and time. (The leaf elements were described in section 2.5).

Dependence on geometric complexity. Running the default tests with error threshold 10^{-5}, we produced **Figure 13**. Again we are glad to see independence on number of input elements. The most efficient basis is quadratic, very closely followed by the other bases. The difference is not dramatic (because the irradiance patterns are smooth in our museum), but the number of leaves is really low when compared to the geometric complexity (the simplest scene has a number of primitives about 26 times the number of leaves generated by the constant solver, with the indicated error threshold).

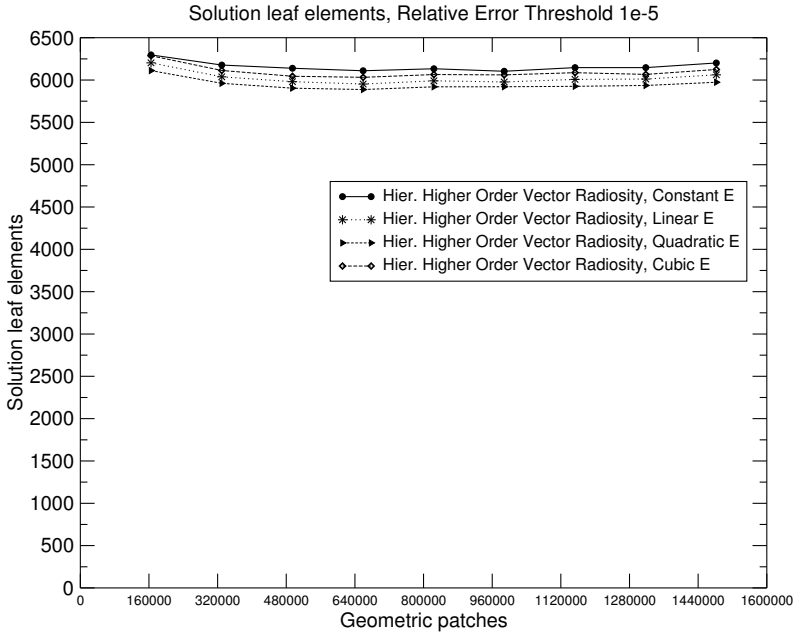

Figure 13: Solution leaf elements. Also this measure is independent on scene complexity. Quadratic is the cheapest.

Accuracy. It is interesting to see how the number of leaves changes with the target precision. Such dependence is shown in **Figure 14**: the number of leaves grows from about 2000 to about 6000 and then reaches 20000 with threshold 10^{-4},10^{-5},10^{-6} respectively. The various bases perform similarly over the whole range of thresholds. In our scenes, higher order bases appear more efficient for error thresholds smaller than 10^{-6}.

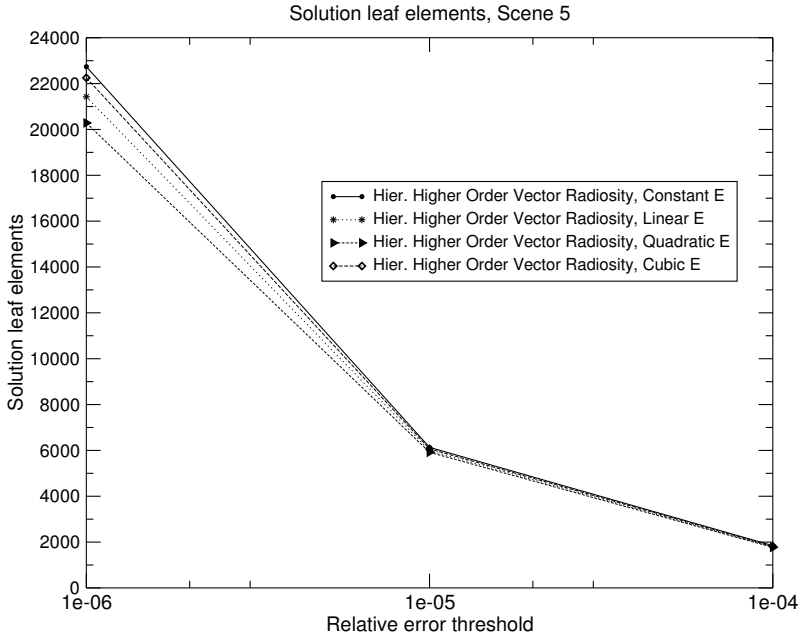

Figure 14: Leaf elements. The dependence on the error threshold is shown. The total number of solution leaves is always a fraction of the input triangle count. Quadratic basis wins.

Solution leaves to input triangle count ratio.

To help understanding how efficient the refinement oracle is, we consider the ratio between the input triangles count and the solution leaf count. In **Figure** 15 the results for all the scenes are shown, with three different error thresholds using the quadratic irradiance basis. The three curves all show a clear linear growth, because as seen in the previous sections, the number of final leaves depends solely on the error threshold. Fixing the threshold generates a number of solution leaves independent on the scene resolution, thus giving the linear trend over the whole range of scenes. The ratios range from 100:1 (threshold=10^{-6}) to about 850:1 (threshold=10^{-4}) with the chosen basis. At the moment we do not know how good these figures are.

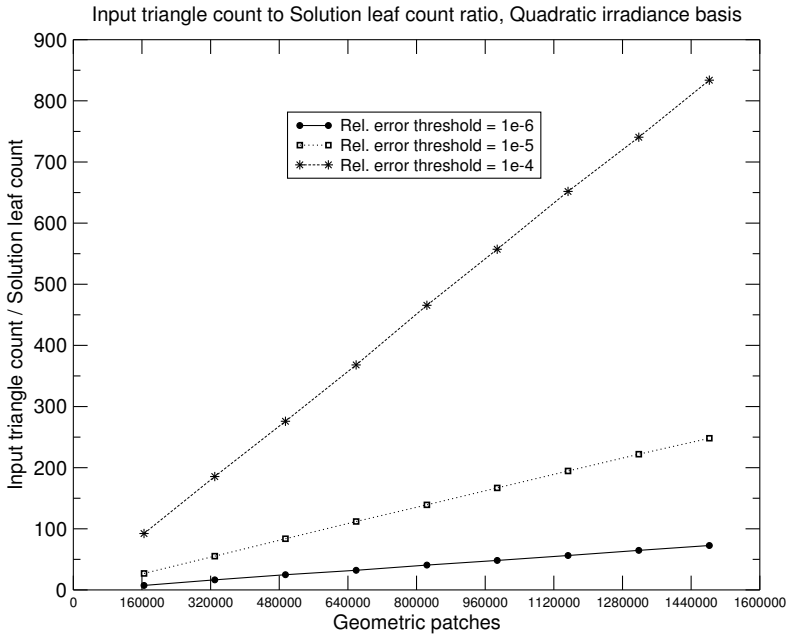

Figure 15: Solution leaf elements. The ratio between input triangle count and solution leaf count is a measure of quality for the refinement oracle.

Leaf elements during a simulation. We expect the number of leaves to grow as solution memory does, during the solution of a given scene. This trend is confirmed by **Figure 16**. As for memory, the number of leaves reaches a steady maximum at the second iteration, stopping to change when refinement ends. Similar behaviours are present in all the test scenes and with different thresholds. The graph also shows that the difference in number of leaves among different bases is less evident than the solution memory variation shown in Figure 12. Again, higher order bases generate less leaves than constant, although the difference is not big.

3.5 Visual evaluation

In this section we show how the overall solution quality depends on the chosen irradiance basis and error threshold and we also want to compare our solutions to those obtained with volume clustering. Furthermore, we show some pictures exhibiting full BRDF (non-diffuse) reflections.
Radiator solutions are displayed using the irradiance distributions computed by the solver. Each solution leaf face cluster has an associated distribution used to shade the input triangles forming cluster. To obtain a correct display there is no need for an expensive final gather because the irradiance values are interpolated and converted to radiosity on a per-vertex basis (using per-vertex shader programs directly on the GPU)[3]. When full BRDF is enabled on the statues, the vertex shader program takes it into account, yielding view dependent effects in real-time.
To grasp the behaviour of our method and explore the computed scenes we have built a renderer program running in parallel with the solver. After loading and preprocessing the MGF scene, various parameters are set (irradiance and radiosity basis, error threshold, residual ratio, etc.) and then the solver is started. The display can be made with or without full BRDF or alternatively just the face clusters are shown using random colors.

[3]Only one normal vector per triangle is used, because of the hypothesis of very small triangles.

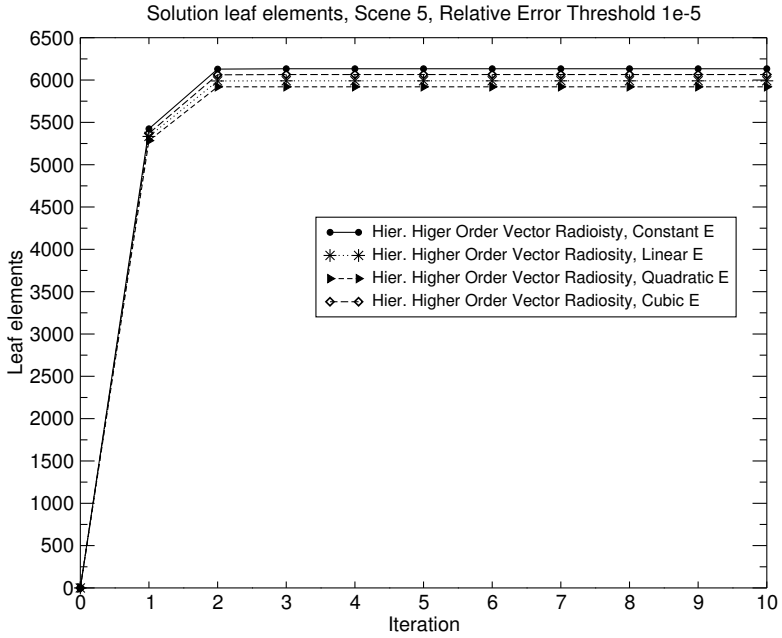

Figure 16: Leaf elements during solution. After the first iterations, the number of leaf elements becomes constant. No more new leaves are created when refinement ends.

All of the pictures were produced without tone mapping.

3.5.1 The museum scene

In **Figure 17** is shown the museum scene solved with error threshold 10^{-5}, constant radiosity and linear irradiance basis. Three views are presented: face clusters without radiosity, diffuse BRDF and full BRDF. We can make some qualitative remarks: the cluster view shows more refinement for higher irradiance gradients; the radiosity view has smooth and realistic shadows together with color bleeding. The full BRDF view adds more realism to the scene, giving a realistic feeling for the shiny materials we chose for the statues. View-dependent color bleeding is highly noticeable on Buddha's pedestal.

3.5.2 Basis comparison

To understand how different bases approximate the same irradiance field, we solved the museum scene keeping the same target precision and using different irradiance bases. The results for constant, linear, quadratic and cubic bases are shown in **Figure 18**. The relative error threshold is 10^{-5} and the radiosity basis is constant for all images. The corresponding clusters are shown in **Figure 19**.

Radiosity. The constant irradiance makes the corresponding picture appearance blocky, because only one vector irradiance sample per cluster was used and for the lack of interpolation. Strong discontinuities are visbile on Isis´ abdomen and legs.
Linear irradiance gives the correct shading on Isis´ abdomen and legs, while the shadow on the wall behind Igea is still blocky.
The quadratic basis smooths Igea´s shadow.
The cubic basis does not show strong differences when compared to quadratic.

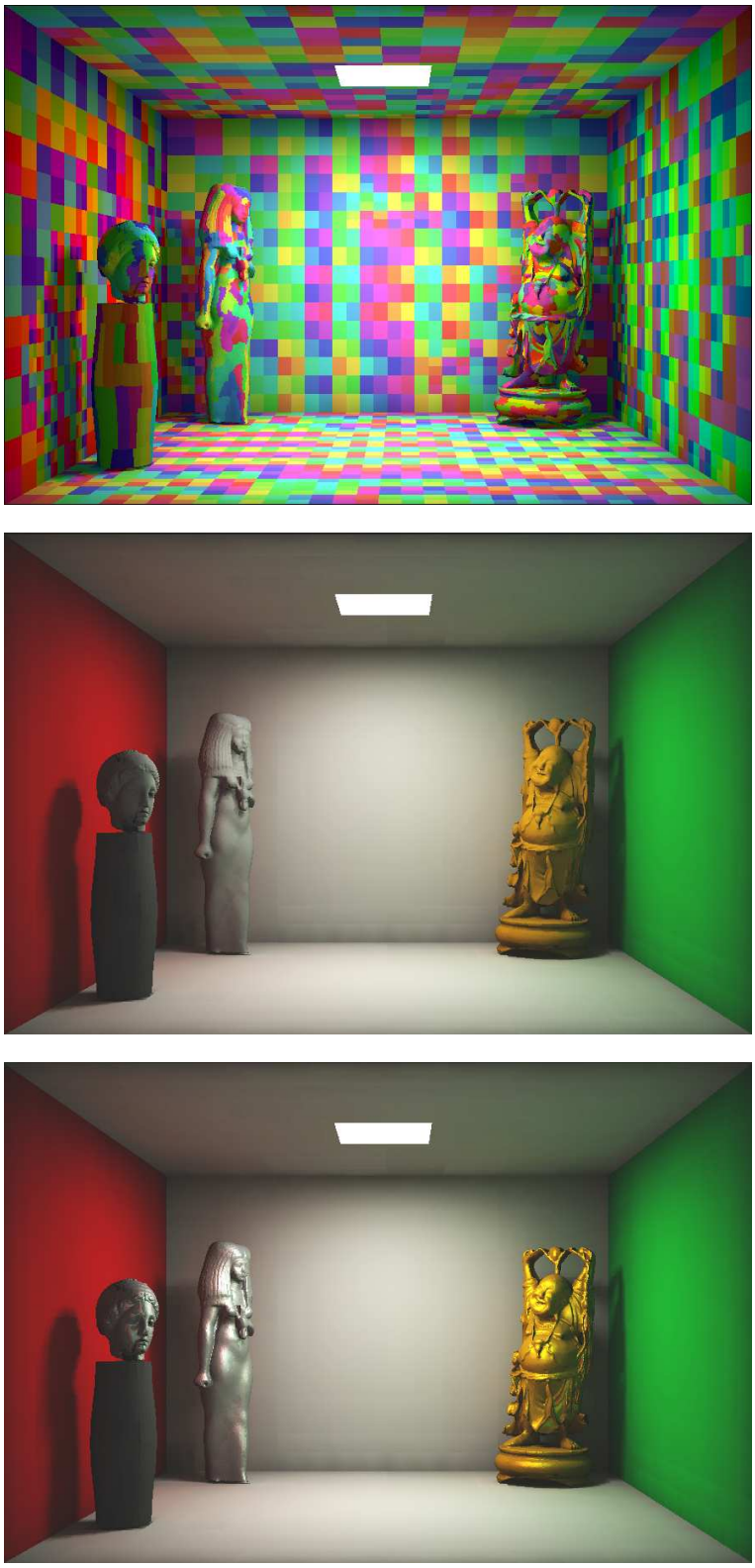

Figure 17: Museum scene. The museum is displayed using random colors face clusters (top), diffuse BRDF only (center) and full BRDF (bottom). (Constant radiosity, linear irradiance, error threshold=10^{-5})

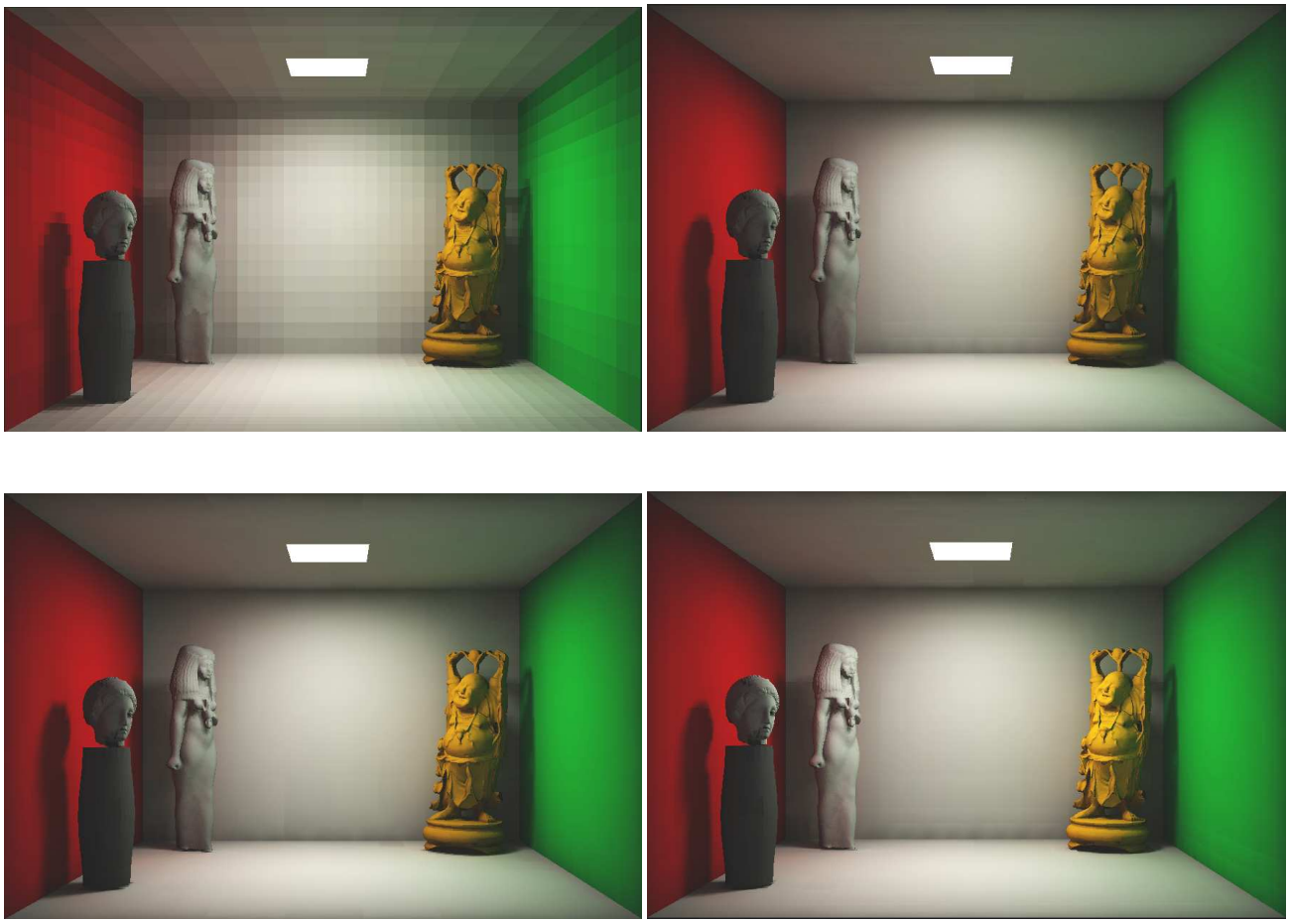

Figure 18: Comparison of different bases. Top left: constant irradiance basis, top right: linear irradiance basis, bottom left: quadratic irradiance basis, bottom right: cubic basis. (Constant radiosity, error threshold=10^{-5}, diffuse BRDF only)

The other visible discontinuities (Isis´ cheek and Igea´s mouth) will disappear with the smallest threshold (see below).

Clusters. The clusters shown in the pictures of Figure 19 visualize how the refiner performs with different bases. Constant irradiance obviously generates many small clusters, particularly visible on the wall facing the camera.

Linear and quadratic bases reduce the number of face clusters but not as noticeably as the cubic basis does.

3.5.3 Error thresholds comparison

In order to understand the behaviour of our method with respect to the desired precision, we compare the results for the same scene with different relative error thresholds. The images (overall scene and Buddha´s detail) corresponding to thresholds $10^{-6}, 10^{-5}, 10^{-4}$ are shown in **Figure 20**. Looking at the overall view obtained with the smallest threshold we do not notice big flaws: for example the Buddha´s shadow and Igea´s face denote high quality appearance.
The medium precision threshold has coarser shadows. Discontinuities appear on Igea´s and Isis´

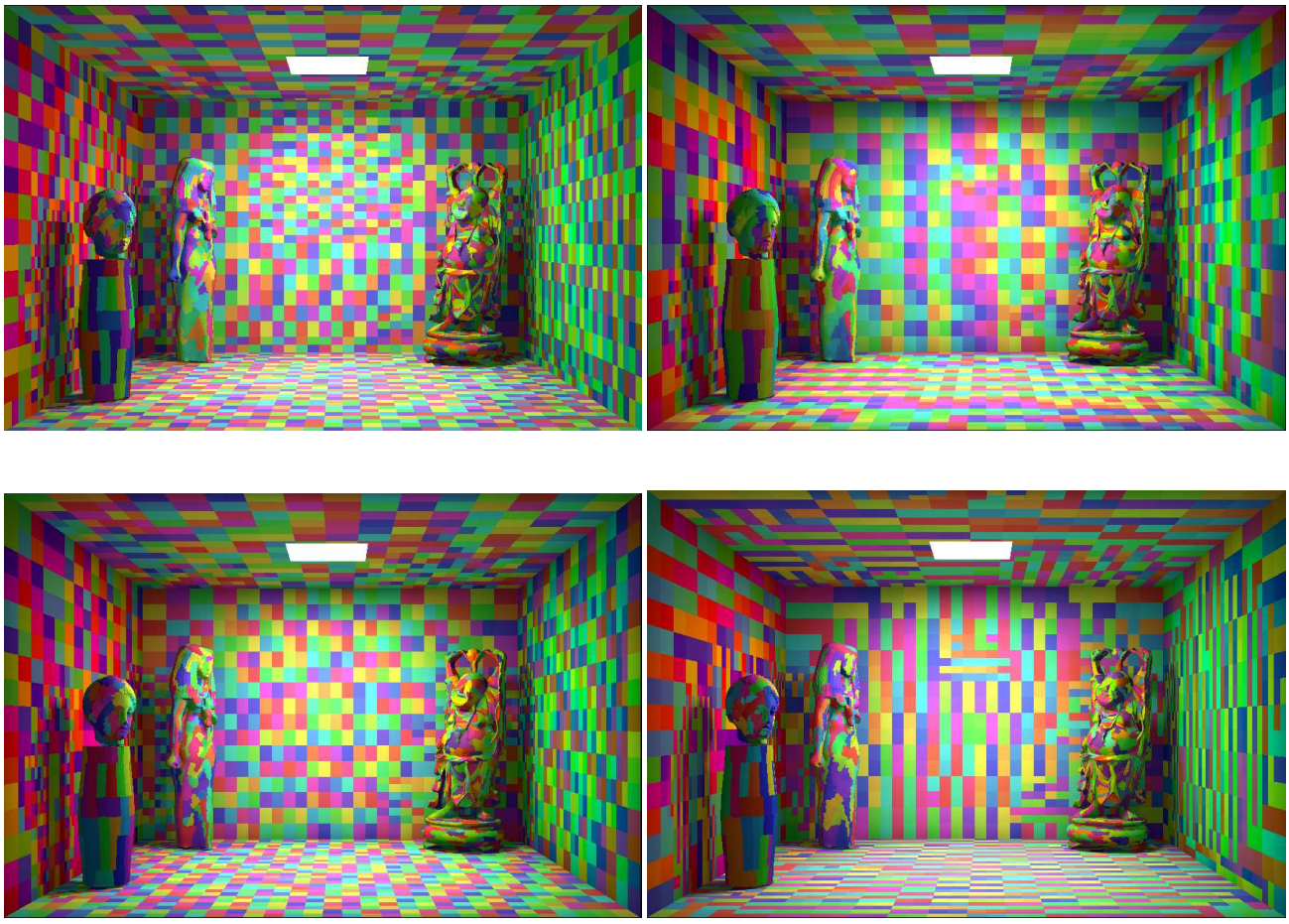

Figure 19: Comparison of different bases (clusters shown). Top left: constant irradiance basis, top right: linear irradiance basis, bottom left: quadratic irradiance basis, bottom right: cubic basis. (Constant radiosity, error threshold=10^{-5})

face.
The lowest precision threshold generates pictures with general coarse appearance with blocky shadows. Another visible symptom of low precision computation is the dark right side of Igea´s face, receiving indirect light only: the large threshold makes the reduced power coming from the walls towards Igea invisible to the refiner.
Looking at the zoomed Buddha in **Figure 21**, we notice an overall good appearance for high precision version, while the medium and low precision reveal some noticeable shading flaws below the neck, on the hands and on the garment covering the right thigh.

3.5.4 Comparison with volume clustering

Besides quantitative differences between our method and volume clustering we wanted to compare their visual difference.
Looking at **Figure 22** (bottom) the blocky nature of volume clustered scalar radiosity solution is evident. Each volume cluster has only one associated radiosity value, leading to the wrong homogeneous shading of the triangles contained in the cluster. The directional nature of the irradiance map computed by face cluster hierarchical vector radiosity enables the correct scene rendering without the need of a final gather, as Figure 22 (top) shows.

4 Conclusions

We have shown evidence that our hierarchical higher order radiosity method using vector irradiance together with face clusters gives correct results within good time bounds when processing highly tessellated scenes.

We can summarize the advantages of our method:

- Hierarchical vector radiosity better exploits the directional nature of the global illumination problem, associating a distribution of vector irradiance to the objects while refining them until convergence. The number of energy transfers, the running time and the memory requirements are independent on the geometric complexity of the input scene, when the scene is sufficiently tessellated.
- The objects in the scene and the irradiance distribution are represented hierarchically, making possible during the solution process the choice of the most convenient level of detail for the energy exchanges. The solution level of detail strongly depends on the error threshold and slightly depends on the chosen basis, leading to good figures: for instance, in our simplest test scene, the ratio of input triangles to solution leaves is about 26 to 1 with medium precision error threshold and quadratic irradiance basis.
- A good range of glossy BRDFs can be simulated.
- No final gather is needed to display correctly shaded environments, beacause of the directional nature of vector irradiance.
- The vector irradiance maps output by our algorithm are easily implemented on recent programmable GPUs allowing a fast rendering of glossy environments.

In the following we detail the above advantages.

4.1 Time

Given a scene represented as a hierarchy of face clusters, the time performance of our solver is independent on input scene complexity, when the geometric details are finer than the illumination details. The target precision has a strong influence on the running time: a simulation with

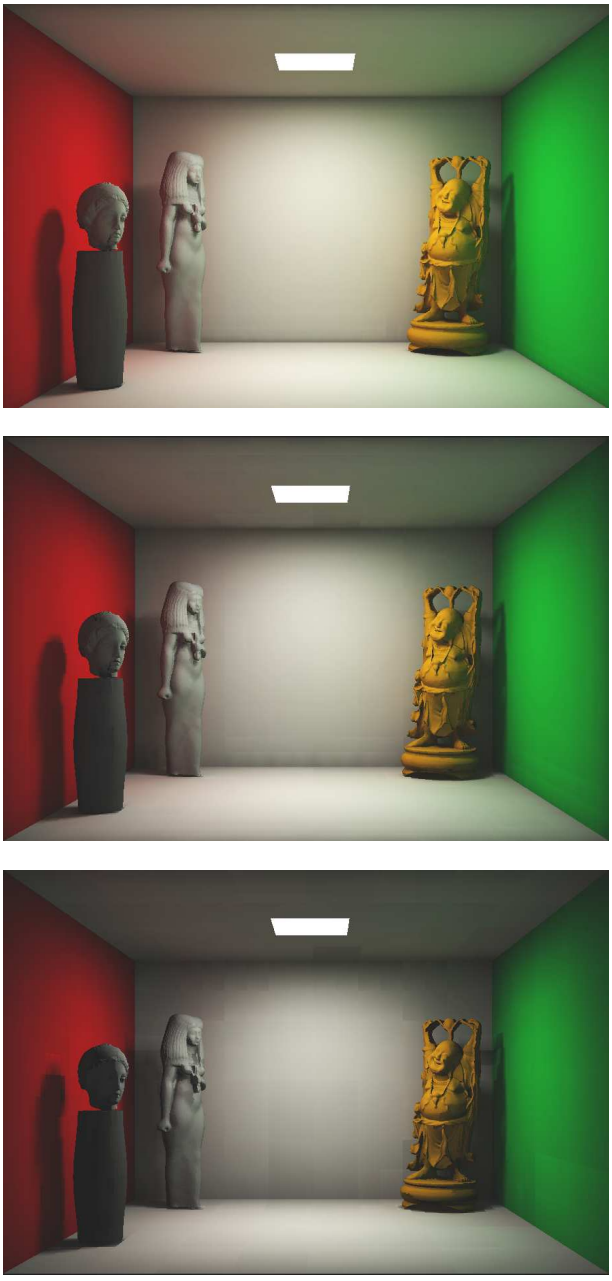

Figure 20: Comparison of different error thresholds (Overall scene). Error thresholds: 10^{-6} (top), 10^{-5} (centre), 10^{-4} (bottom). Linear irradiance was used for all the pictures.

Figure 21: Comparison of different error thresholds (Buddha's detail). Error thresholds: 10^{-6} (left), 10^{-5} (centre), 10^{-4} (right). Linear irradiance was used for all the pictures.

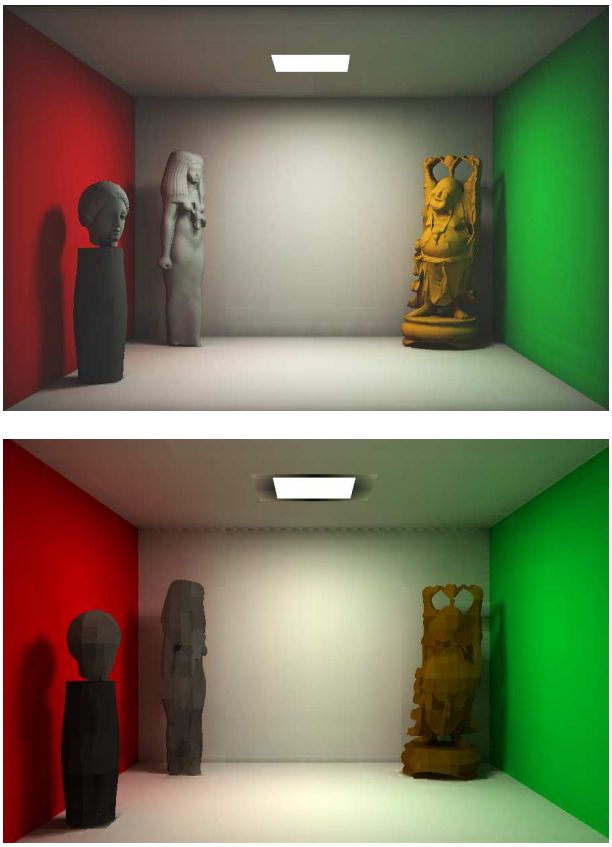

Figure 22: Visual comparison. Top: hierarchical higher order vector radiosity (constant radiosity, linear irradiance, error threshold=10^{-5}). Bottom: volume clustering (same threshold).

precision ten times bigger runs about 10 times longer, with any basis. Our approach is faster than volume clustering for the decoupling between light and geometry and for the reduced refinement due to the greater descriptive power of higher order bases. The best performing basis is constant, followed by linear and quadratic. The cubic basis is the slowest. Using medium precision, the muesum scene used in our tests is solved in about 150s with small difference among the bases.
When we disable visibility queries (i.e. the shadows are fully ignored) the solution is about 10 times faster, with quadratic basis performing best and with a large (relative) difference among the bases. This is due to the current implementation of the solver, where visibility sampling is coupled to kernel integration.

4.2 Energy transfers

As the timings suggest, the number of energy transfer is independent on the input scene geometric complexity. For our scene and thresholds, quadratic basis always performs best, followed by linear, cubic and constant. The constant basis produces about 20% more transfers than quadratic basis.

The dependence on error threshold is strong. A precision ten times higher requires ten times the transfers needed by the coarser approximation. With respect to the error threshold, quadratic basis is again the best choice, particularly with small thresholds.
With respect to volume clustering, our method saves about 90% of the transfers.

4.3 Memory

The memory requirements are independent on the geometric scene complexity. The required memory is always a small fraction of the input memory footprint. The use of memory is proportional to the order of the chosen basis, thus suggesting to use linear or quadratic basis for precise simulations to achieve a good quality-storage tradeoff. The memory required with the medium error threshold varies from 0.87 MB using constant basis to 7.8 MB using cubic basis.

4.4 Solution leaves

The number of solution leaves does not depend on the input scene. The test scene at any resolution is solved using about 6000 solution leaves. The irradiance bases we tested all generate a similar number of leaves, probably because the irradiance patterns are quite smooth. Quadratic basis saves a small amount with respect to linear, cubic and constant. The ratio of input triangles to solution leaves seems large, but we do not have yet quantitative means to judge that.

4.5 Glossy BRDFs

Computing. The results of our benchmarks prove that a radiosity simulation is computed in minutes for scenes containing complex objects and for a good range of glossy BRDFs. While the displayed solution is not the result of a full global illumination simulation, since glossy reflections are limited to the final stage of any illumination path, its visual quality is not very far from what provided by ray-tracing post-processing algorithms commonly in use in state-of-the art architectural lighting systems. We thus believe that our approach has great promise, since it can be used to generate low to moderate quality solutions for glossy environments, that are suitable for interactive viewing.

Rendering. The computed scenes can be examined interactively exploitng the programming capabilities of modern commodity graphics to render illuminated scenes with complex models directly from the vector irradiance map, using hardware acceleration for computing view dependent illumination during the interactive walkthrough. At the moment the rendering is implemented with hardware vertex programs, giving excellent rendering speed also with full BRDF but with some artifacts visible when zooming on coarse models.

4.6 Visual quality

The appearance of the museum scene is generally good, presenting color bleeding and smooth shadows. Striking effects appear using full BRDF. Medium and low precision thresholds reveal some shading discontinuities due to the underestimated energy exchanges. A good tradeoff between time and quality is linear or quadratic basis with medium precision error threshold.

4.7 Future work

We are going to decouple visibility from kernel integration, with an approach similar to shadow masks and adaptive shadow masking.

A detailed analysis of the approximation error introduced by our method is an important area for future research. The error analysis should lead to the design of a better refiner: for instance we need to understand how good the oracle is at refining, a measure could be the solution leaves to input triangles ratio.

Different scenes with different reflection and lighting properties need to be studied, it is very likely that scenes with different irradiance patterns would emphasize the efficiency of higher order bases even more.

We will use per-pixel harware programs to obtain smooth shading with any zoom factor and size of triangles.

References

[1] F. Cuny, L. Alonso, and N. Holzschuch. A novel approach makes higher order wavelets really efficient for radiosity. In M. Gross and F. R. A. Hopgood, editors, *Computer Graphics Forum (Eurographics 2000)*, volume 19(3), pages 99–108, 2000.

[2] Cyberware free 3d samples. http://www.cyberware.com/samples.

[3] Leonardo Spanò Enrico Gobbetti and Marco Agus. Hierarchical higher order face cluster radiosity. Technical report, CRS4, Center for Advanced Studies, Research, and Development in Sardinia, March 2002. http://www.crs4.it/vvr/bib/papers/crs4-report-hhofcr-2002.pdf.

[4] Michael Garland. *Quadric-Based polygonal simplification.* PhD thesis, Carnegie Mellon University, May 1999.

[5] Frank Suykens de Laet Philippe Bekaert and Philippe Dutre. Renderpark, a photorealistic rendering tool. http://www.cs.kuleuven.ac.be/cwis/research/graphics/RENDERPARK.

[6] François Sillion. Clustering and Volume Scattering for Hierarchical Radiosity Calculations. In *Fifth Eurographics Workshop on Rendering*, pages 105–117, Darmstadt, Germany, June 1994.

[7] Brian Smits, James Arvo, and Donald Greenberg. A clustering algorithm for radiosity in complex environments. In Andrew Glassner, editor, *SIGGRAPH 94 Conference Proceedings*, Computer Graphics Proceedings, Annual Conference Series, pages 435–442. ACM SIGGRAPH, ACM Press, July 1994.

[8] Leonardo Spanò and Enrico Gobbetti. Radiosity for highly tessellated models. In *SIMAI 2002 Symposium on Adaptive Techniques in Numerical Simulation and Data Processing*, Conference held in Chia, CA, Italy, May 27–31, 2002, May 2002.

[9] Andrew J. Willmott and Paul S. Heckbert. An empirical comparison of radiosity algorithms. Technical Report CMU-CS-97-115, School of Computer Science, Carnegie Mellon University, April 1997.

[10] Andrew J. Willmott, Paul S. Heckbert, and Michael Garland. Face cluster radiosity. In Dani Lischinski and Greg Ward Larson, editors, *Rendering Techniques '99*, Eurographics, pages 293–304. Springer-Verlag Wien New York, 1999.