CGI 2008 Conference Proceedings

Carles Bosch · Gonzalo Besuievsky · Ignacio Martin

A Hardware Accelerated Adaptive Mesh Subdivision for Interactive Daylight Illumination

Abstract Daylight simulation for generic environments involves visibility computations from an infinite area light source (the sky). In order to obtain accurate results, this task tends to be very time consuming and also requires a good model discretization that is not automatically done in general.

This paper presents a daylight simulation method for interactive direct lighting visualization that performs adaptive mesh subdivision. In a first step, the visibility of the scene is computed from the sky by means of hardware parallel projections and occlusion queries. Visibility values at receiver surfaces serve to guide the subdivision of the scene using an iterative process, resulting in a final adaptive mesh where each element stores visibility from a set of hemispherical directions. Lighting visualization is then performed at interactive frame rates for any day-time condition, taking advantage of the precomputed visibility.

The main contribution of our method is that it provides a fast automatic adaptive mesh operation over the model according to visibility values. In this way, no pre-meshed models are required, making the method well suited for generic environments such as indoors or outdoors of buildings and urban models. We show that our mesh subdivision method is also useful as an optimal input model for other lighting visualization techniques like Precomputed Radiance Transfer or a global illumination radiosity solver.

Keywords Hardware Rendering · Interactive Illumination · Daylighting

1 Introduction

Daylight simulation tools have experimented an increase need in the last years for architectural design and environment applications. The importance of sustainable building

Carles Bosch, Gonzalo Besuievsky, Ignacio Martin
Institut d'Informatica i Aplicacions, Universitat de Girona, 17071 Girona, Spain
Tel.: +34-972-418252
Fax: +34-972-418792
E-mail: {cbosch, gonzalo, imartin}@ima.udg.edu

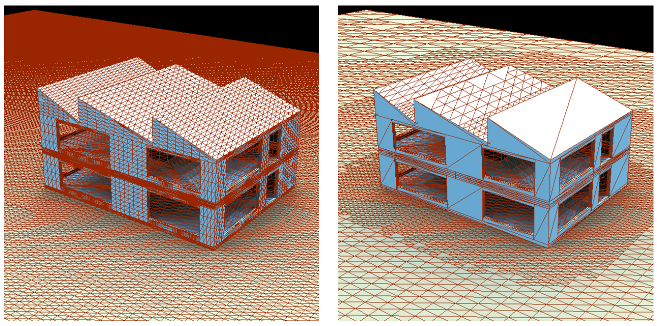

Fig. 1 Left: Architectural scene with a uniform refinement. Right: Same scene adaptively refined according to the sky visibility.

and lighting assessment for tasks like design, visual comfort studies or energy saving calculation requires more integrated and accurate tools. Efficiently solving the daylight illumination problem is still a challenge for reliable and interactive lighting visualization. The complexity of the problem is due to the fact that light energy comes from an infinite extended area source (the whole hemisphere), and the visibility at receiver surfaces must be evaluated for a large set of directions.

Most of the available daylight solutions suffer from several disadvantages that make them not general enough to deal with complex lighting simulations [6, 1, 11]. Input models, for instance, require already pre-meshed geometry or their discretization into texture maps. This usually supposes a restriction when dealing with complex environments, as the whole model must be highly discretized for an accurate visualization (see example of Fig. 1).

In this paper, we propose a daylight illumination method for the accurate simulation of generic environments where the main goal is to provide lighting visualization functionality at interactive frame rates. We focus our effort on efficiently solving the direct daylight illumination problem considering the sun and generic sky models. Our approach is first based on an efficient computation of the visibility coefficients from a set of hemispherical directions, which is achieved using parallel projections and hardware occlusion queries. According to this visibility, we perform an adaptive subdivision of the mesh to properly capture visibility changes. In a second step, light energy transport is computed at run-time using the precomputed visibility and the sun and sky distributions, which allows dynamic day-time conditions at interactive frame rates.

The main contribution of this work is the use of a hardware accelerated procedure in order to provide an automatic subdivision of the scene from an infinite area light source (the sky dome). This procedure is fast, accurate, and allows to deal with more generic environments, such as integrated illuminated spaces for indoors or outdoors of buildings, or complex urban environments. Although here we only deal with the direct lighting problem, we show how our mesh subdivision method is also suitable as a preprocessing step for other lighting visualization approaches. It can be used, for instance, as an optimal pre-meshed input model for precomputed radiance transfer [18], as a fast initial mesh for a global illumination radiosity solver, or for an accurate computation of ambient occlusion [15].

The rest of the paper is organized as follows. Related work is revised in Section 2. Our method is outlined in Section 3 and described in detail in Section 4 and Section 5. Results and discussions are presented in Section 6. Finally, Section 7 presents our conclusions and future work.

2 Related Work

2.1 Daylight Illumination

First approaches for daylight illumination using finite element methods were presented in [14] and [9]. In these works, the simulation process of indoor scenes is simplified by first identifying important opening structures. In [9], for instance, they preprocess daylight illumination for a set of "window" surfaces that are later used as area light sources. A more general approach is presented by Müller et al. [12] through a progressive refinement radiosity algorithm. Here, the sky dome is subdivided into a fixed number of patches and a daylight shooting step is performed as a preprocessing pass. Although input models are not restricted, no adaptive subdivision is considered. Hierarchical radiosity has been also extended for its use in daylight illumination [5]. In this case, the sky is subdivided into a set of hierarchical patches in order to compute their interaction with the scene as regular patches. This approach, however, is restricted to outdoor scenes. Nimeroff et al. [13] propose a different approach for the fast re-rendering of naturally illuminated scenes. The method is based on rendering the scene for different sun positions and sky conditions and then performing linear combinations of the pre-computed images for new conditions. Although this allows fast simulations over a complete day, the viewpoint must remain static. Another proposed strategy that can be used to compute daylight is by clustering a set of directional light sources using Lightcuts [19].

Works more closer to our approach can be found in [2] and [3]. Ashdown propose the use of hardware parallel projections to compute sun and sky illumination based on an hemi-cube combined with a ray tracing technique [2]. Although interactive results are achieved, no adaptive subdivision is provided. Besuievsky and Martín then propose an adaptive mesh subdivision technique based on visibility values. For each hemispherical direction, a Layered Depth Image (LDI) is computed and later used to perform the mesh subdivision and light energy transfer. Since no hardware acceleration is used, the method is very time consuming. Furthermore, the subdivision takes into account the current sky distribution, thus dynamic day-time changes are not feasible.

2.2 Interactive Lighting Visualization

In recent years, many works have been developed to improve interactive lighting visualization through the use of precomputed radiance transfer [18]. This technique is feasible to perform lighting computations on dynamic low-frequency light environments by means of a two-step approach. First step consists on precomputing a transfer function over the model and representing it using an orthonormal basis (a process that is generally slow). At run time, the lighting is then projected onto this basis and combined with the transfer functions in order to evaluate the illumination equation in real-time. In this case, the use of a properly refined mesh is important to capture illumination details. However, uniformly increasing the mesh density implies more precomputation time and a huge amount of memory in order to store the basis coefficients. Křivánek et al. [8] have faced this problem through an adaptive mesh algorithm where the subdivision is guided by an error measure of the transfer vector. As in our work, all surfaces are assumed as diffuse and only direct illumination is considered. Results are shown only for simple scenes, and although no details are given on how the visibility is computed or the time performance of the mesh operation, no hardware-acceleration is used. The DirectX Software Development Kit [10] also provides an adaptive PRT-based mesh refinement, which is similar to this approach. Although good mesh representations are obtained, it does not provide optimal solutions for a potential infinite light source such as the sky. An excessive undesired mesh subdivision occurs in even completely dark regions or even in completely closed spaces. Our mesh subdivision approach is guided by accurate visibility values computed from the illumination directions of the environment, thus the refinement only happens in potentially illuminated regions.

Our visibility processing algorithm is also related to some works on ambient occlusion evaluation [15]. These methods compute visibility by considering all the sky hemisphere as well. However, their goal is different from ours, as they only compute a unique value representing the overall visibility at each point. In our case, we compute and store visibility values for a set of sky patches for which we then sample its irradiance at run-time, which results in more accurate results. Some works based on ambient occlusion have also used occlusion queries to compute visibility, but using an ”inside-out” (or gathering) approach [7] or a per-vertex solution [17]. Furthermore, none of these methods perform an adaptive refinement of the mesh. They tend to assume that input models are sufficiently refined, which is not the case in most situations.

Finally, adaptive mesh subdivision for illumination has also been treated in the field of radiosity [4]. In this case, an accurate global illumination solution representation is usually desired and they tend to consider fixed lighting conditions, area light sources in general. Our approach focus on the visibility considering the whole hemisphere sky as a virtual source without fixing any particular lighting distribution. However, our resulted mesh could also be useful as a good starting mesh representation for a radiosity solution in a particular day-time condition.

3 Overview

The objective of our method is to perform interactive direct daylight simulations with dynamic lighting conditions over static polygonal scenes. For this, a two-step procedure is used. In the first step, we precompute and store the visibility of each scene triangle and perform the adaptive subdivision according to it, which is the most time-consuming part. This task is summarized in Fig. 2.

```
Uniformly subdivide the sky dome into a set of spherical
patches (see Fig. 3)
for each sky patch
  Generate $s$ random samples within the patch
  Initialize triangle list with the original mesh
  repeat
    for each sample
      Compute the visibility of the triangles using an
      orthographic projection
    end for
    Compute the average visibility of each triangle
    Hierarchically refine triangles according to visibility
    Update list with new triangles
  until no more triangles to refine
end for
```

Fig. 2 Pseudocode of the adaptive subdivision algorithm

For this step, our strategy consists in computing the average visibility of every triangle against each sky patch and to subdivide those triangles that fulfill a certain visibility-based criteria. The hardware-accelerated portion of this iterative process is the computation of the visibility of the triangles, which is done using parallel projections and occlusion queries (see Section 4).

At the end of the adaptive process, each triangle of the refined mesh stores a set of visibility factors that will be used during the illumination step. Given a set of day-time parameters, this second step first obtains the corresponding sky and sun radiance using the analytical model presented in [16]. The radiance contribution of the sky at each element of the mesh is then computed using the visibility values already stored. The contribution of the sun is added after being recomputed for the new position, where its visibility is determined by means of a shadow mapping algorithm.

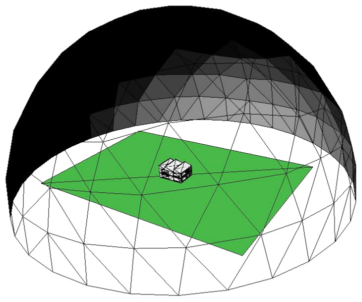

Fig. 3 Sky subdivision for 256 spherical patches.

4 Adaptive Mesh Subdivision

This section describes how visibility is computed and how the scene is subdivided accordingly.

4.1 Visibility Computation

In order to compute the visibility of the scene with respect to the sky dome, we use a shooting or “outside-in” approach. As can be seen in Fig. 2, this approach consists on evaluating the visibility of every triangle from each sample direction of the sky. Such kind of evaluation requires the projection of the whole scene for each sky sample, but for large scenes such as buildings and urban models, it is more efficient than a gathering strategy. In a gathering approach, visibility is computed by evaluating the entire scene over each triangle. Since the number of sky samples s is generally much lower than the number of triangles n in the scene, this supposes a considerable amount of projections: $O(n^2)$ versus $O(sn)$.

Our method first starts by subdividing the sky into a set of uniform spherical patches. Each spherical patch represents the solid angle for which we want to compute the visibility of the mesh. The result of this process will then be a set of visibility factors V_{ij}, defined as the percentage of area of a triangle i visible from a patch j weighted by the cosine term of the angle between the triangle orientation and the solid angle direction.

For the computation of the visibility factors, our method relies in the use of hardware occlusion queries. Occlusion queries is an OpenGL utility that is commonly used to determine the visibility of a primitive or set of primitives, and works by counting the number of fragments drawn into a buffer. If the scene is rendered from a patch’s point of view, we can evaluate the visibility factor of a triangle by counting the number of fragments that are displayed into the fi-

```
reset $V_{ij}$ for all triangles in the list
for each sample $k$ of patch $j$
  set orthographic camera from direction $k$

  enable depth test with less operation
  for each triangle $i$
    render triangle $i$
  end for

  enable depth test with equal operation
  for each triangle $i$
    render triangle $i$ with queries
  end for

  disable depth test
  for each triangle $i$
    $f_v$ = read query $i$
    if $f_v > 0$ then
      render triangle $i$ with queries
    end if
  end for

  for each triangle $i$
    if $f_v > 0$ then
      $f_c$ = read query $i$
      $V_{ij} = V_{ij} + (f_v / f_c) * cos\ \theta_{ik}$
    end if
  end for
end for
get average $V_{ij}$ for all triangles
```

Fig. 4 Pseudocode of the visibility computation for a sky patch

nal image and the number of total fragments covered by the triangle. The ratio of these two numbers will give us an approximation of the visibility factor between the triangle and the current sky patch. Such a process requires a fourth-pass algorithm that is outlined in Fig. 4.

Given a sky patch, the objective is to evaluate an average visibility factor by considering a set of random samples inside the patch. For each sample, an orthographic camera is first set according to the sample direction, adapting its frustum to a precomputed bounding sphere of the scene. First rendering pass simply initializes the depth buffer, while second and third passes use the occlusion queries to count visible and covered fragments of each triangle, f_v and f_c, respectively. Note that completely occluded triangles are directly discarded during the third pass. The fourth pass finally retrieves the last queries and computes the ratios for the visibility factors. This last task is done in a separate pass to avoid waiting for queries immediately after performing them. Visibility factors are then accumulated after being weighted by the cosine of the angle θ between the triangle normal and the sample direction.

The quality of the visibility factors obtained with this process will depend on two parameters: the number of samples per patch and the depth buffer resolution. The second parameter is especially important for small triangles, which may not be visible if the buffer resolution is too low.

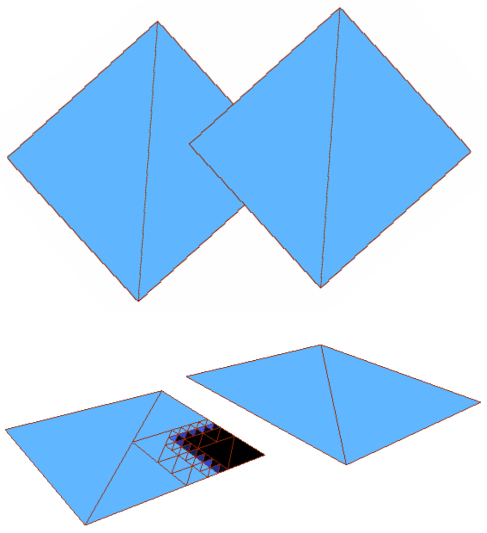

Fig. 5 Top: Orthographic projection of two meshes seen from a sky patch sample. Bottom: Meshes seen from the viewer after the adaptive refinement process. Maximum subdivision level is 4 here.

4.2 Mesh refinement

The adaptive subdivision of the scene is performed after obtaining the visibility factors for the current sky patch. When a certain triangle fulfills a given subdivision criteria, the refinement step simply subdivides it into four children triangles and selects these new triangles for its processing during the next pass. The entire process is then repeated for all the generated triangles until no more triangles are generated. After processing a given patch, note that the visibility of the next patch is then evaluated starting with the non-subdivided mesh. This is done for efficiency reasons, since the mesh rarely requires the same refinement between different sky patches. Naturally, already subdivided triangles will not be subdivided again. Instead, we will simply select their existing childs.

The criteria used to subdivide the triangles is based on three parameters: the mean visibility factor V_{ij} of the triangle, its mean projected area, and the current subdivision level. According to these, we subdivide a triangle when it is occluded from the current patch ($0 < V_{ij} < 1$), its mean projected area is bigger than a certain threshold A_{min}, and a user specified maximum subdivision level has not been reached. Fig. 5 shows a subdivision example for two planes.

In the previous criteria, the projected area value is based on the number of pixels covered by the triangle (the f_c value computed in Fig. 4), and is used to limit the size of the resulting triangles. Since triangles are subdivided into four sub-triangles, a value lower than 4 may produce triangles of less than 1 pixel of area, which would result in bad visibility values during the next pass. In our case, $A_{min} = 8$ pixels is usually chosen. Including the cosine term in the criteria also improves the refinement algorithm, since it avoids unnecessary subdivisions for large angle incidence directions that result in low illumination contributions.

After the refinement process, each mesh element stores a vector containing the computed visibility factors (one for each sky patch). The amount of memory required to store these factors depends on the number of sky patches and the number of triangles on the resulting mesh. For example, 256 sky patches and a refined mesh of 10^5 triangles represents 100 Mbytes of memory, which is well managed by current system configurations.

5 Interactive Lighting Visualization

Light transport is computed interactively using the mesh subdivision obtained in the previous step. Considering all surfaces as perfect diffuse, the outgoing radiance at a point x due to a distant light source can be expressed as:

$$L(x) = \rho(x) \int_{\Omega} L_{in}(\omega_{in}) V(x, \omega_{in}) \cos\theta \, d\omega$$

where Ω is the whole hemisphere covering point x, $\rho(x)$ is the reflectance at this point, $L_{in}(\omega_{in})$ is the sky incoming radiance from direction ω_{in}, and $V(x, \omega_{in})$ is the visibility function, which is 1 if x is visible from direction ω_{in} and 0 otherwise.

By discretizing the sky dome into a set of m patches, if we assume that each patch has a constant radiance, the previous expression can be rewritten as:

$$L(x) = \rho(x) \sum_{j=1}^{m} \widehat{L}_j V_j(x) \Delta\omega_j \qquad (1)$$

where $\widehat{L}_j$ is the average radiance of the sky patch j, $V_j(x)$ is the visibility factor computed by our algorithm, and $\Delta\omega_j$ is the solid angle subtended by the patch. Remember from Section 4.1 that the visibility factor represents the percentage of area of the triangle that is visible from the sky patch j, and this is already weighted by the $\cos\theta$ term.

As stated in Section 3, the sky illuminance distribution is obtained using the model of Preetham et al. [16]. Note, however, that any other analytical model may be used as well, or even an environment map representing the incoming radiance. Given the geographic coordinates (latitude and longitude), the current day and time, and the atmospheric conditions (turbidity), we first retrieve the radiance of the different sky patches. Then, for each element of the mesh, we compute its final radiance by evaluating Equation (1).

Concerning the sun, its contribution could be determined using the same algorithm of the previous section. Due to the high amount of power to be shoot, it would require a higher buffer resolution than for the sky patches, but a single parallel projection would be sufficient. However, the high frequency illumination changes produced by the sun can be more efficiently handled using a classical shadow mapping algorithm. Thus, it is the approach considered here.

5.1 Changing Lighting Parameters

With our method, several parameters may be changed interactively, such as the day of the year, time of the day, or the different surface materials. For each day-time change, a new sky illuminance distribution must be evaluated and the radiance of all triangles must be recomputed. This is done by reusing the visibility factors computed in the first step and reevaluating Equation 1 with the new sky distribution. The contribution of the sun is then added after determining its position and radiance from the current parameters and computing its visibility from the new position.

6 Results and Discussion

We have implemented and tested our method on a Xeon dual core at 1,6 GHz with 2Gb RAM and a GeForce 8800 graphics card. In Table 1, we include the performance results according to the different scenes used in this section. This includes the size of the meshes before and after the adaptive refinement process, the time that it takes to refine them and the frame rates for the interactive lighting.

First considered scene is shown in Fig. 6 and consists of a building and a ground plane. The original model contains 718 triangles, while the refined version 17467. This refined mesh is shown in the right image of Fig. 1 and has been obtained using 256 sky patches, 16 samples per patch, and a buffer resolution of 1024x1024 per sample. The image on the left of Fig. 1 corresponds to the same scene but uniformly

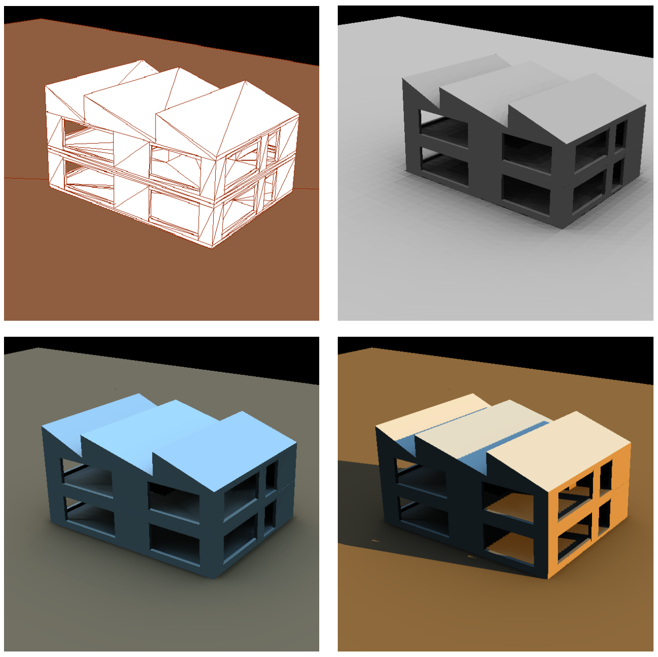
Fig. 6 Building scene. Top left: initial model (see Fig. 1 for the refined version). Top right: mean visibility factor of each triangle. Bottom left: lighting using the sky model. Bottom right: lighting after including the sun.

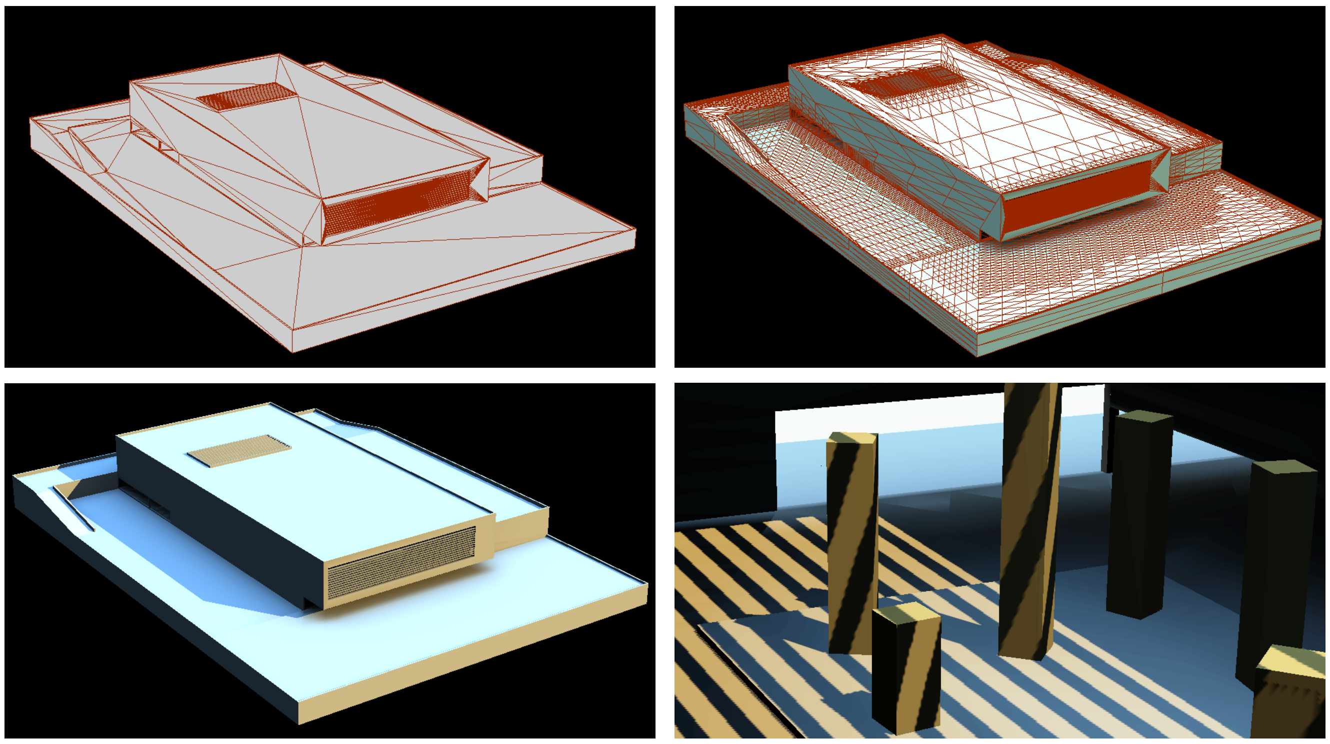

Fig. 7 Museum scene. Top left: input model. Top right: model refined according to sky visibility. Bottom left: daylight illumination. Bottom right: indoor view.

subdivided, in order to approximately capture the same illumination details. In this case, the scene consist of 141646 triangles, which is one order of magnitude bigger than the one obtained with our method. Storing several visibility factors for each triangle (or vertex) of such a scene would suppose a great waste of memory and computations. In the top right image of Fig. 6, we show the mean visibility factors associated to our refined mesh. These have been determined using the different visibility factors computed over each triangle, and can be seen as an equivalent representation of the classical ambient occlusion. The resulting lighting is then shown in the bottom left image for a certain day-time condition. The lighting is obtained by evaluating Equation 1 according to the visibility factors and the corresponding sky distribution. Finally, bottom right image shows the same image after adding the contribution of the sun. The day-time parameters are the 7th April at 12:30PM, and the result of these two bottom images are displayed using Gouraud.

Fig. 7 shows the results for a scene of a museum. The refinement due to the sky visibility has been obtained using 256 patches, 8 samples per patch and a resolution of 512x512. The minimum projected area A_{min} is here 25, while for the rest of the scenes was 8. This value is used to limit the size of the subdivided triangles with respect to the current buffer resolution. Top images show the original mesh and the refined mesh. Bottom left image then corresponds to the lighting obtained with the sky and the sun. In the bottom right, an indoor view of the museum is shown, where light mainly comes from the shading device placed at the ceiling.

In Fig. 8, we have used our method to refine and illuminate a urban model based on the city of Ottawa. For its refinement, we used 128 patches, 32 samples per patch, and a resolution of 1024x1024. As can be seen in Table 1, the complexity of the scene requires a considerable subdivision with respect to the original mesh in order to properly capture the visibility changes. Such kind of urban models, because of their complexity and lighting representation changes, would be very difficult to deal without an optimal adaptive mesh refinement.

Table 1 Performance of our method for the different scenes. Mesh sizes represent the amount of triangles of each mesh.

	Building	Museum	Ottawa
Initial mesh size	718	2460	8987
Refined mesh size	17467	12069	128309
Refinement time	36.3 sec.	26.1 sec.	809.6 sec.
Lighting frame rate	42.7 fps	60.7 fps	11.2 fps

6.1 Comparison and Discussion

We next compare our refinement algorithm with previous work on adaptive precomputed radiance transfer (PRT), using the available solution from DirectX SDK [10]. The algorithm presented in [8] is based on a similar approach, thus the following discussion is also valid for this method. The comparison is performed using a simple scene of a room

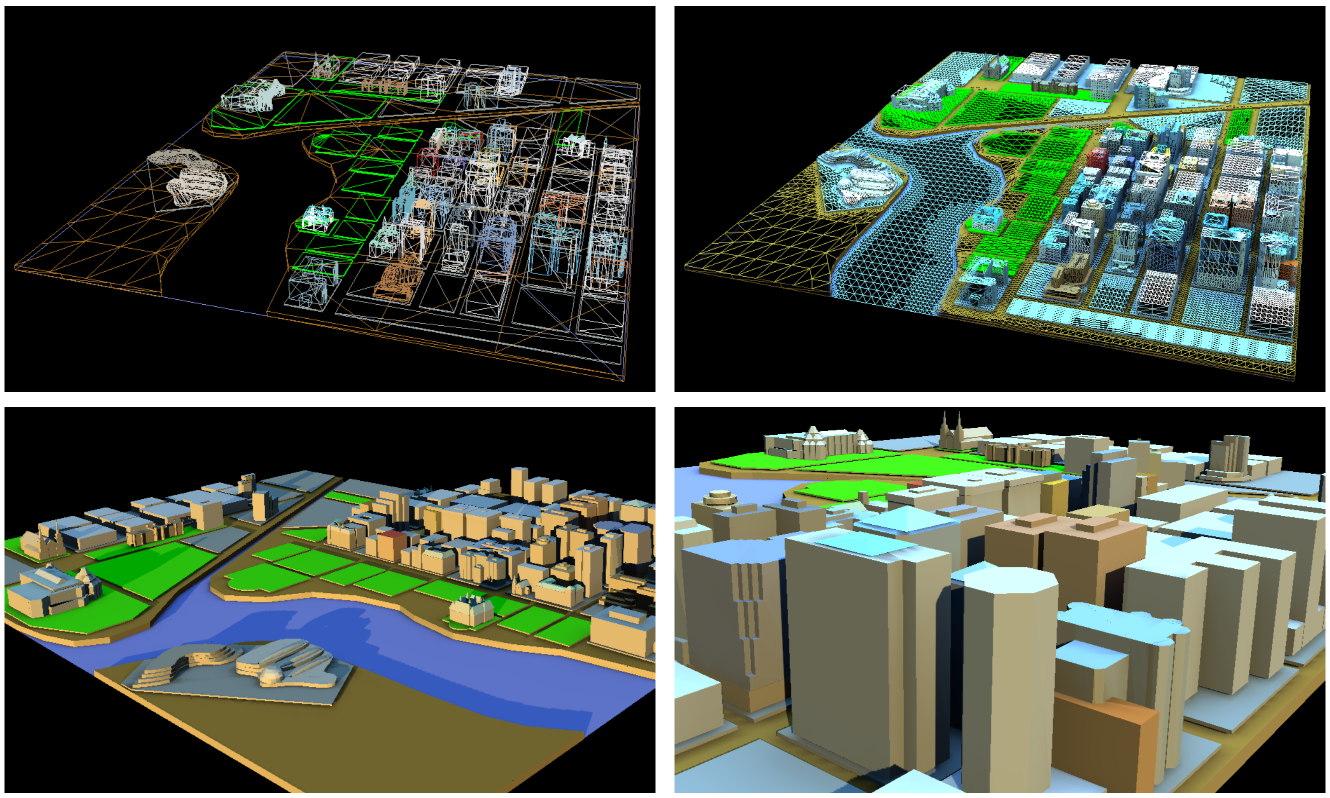

Fig. 8 Urban model of Ottawa city. Top left: original mesh. Top right: refined mesh. Bottom: sky and sun lighting for different day-time conditions and viewpoints.

with a single window and a plane (Fig. 9), consisting of 46 polygons. In order to refine the model, our algorithm takes 2.9 seconds and generates a mesh of 5386 triangles (Fig. 9 bottom). Using an equivalent level of subdivision and a single light bounce (for direct lighting), the PRT-based refinement method, instead, takes 9.4 seconds and results in a mesh of 7933 (Fig. 9 top). Although the refinement procedures are clearly different, it can be observed how our technique presents more ideally results for daylight environments. Since our subdivision is guided by visibility values from the sky dome, only potentially illuminated regions are revised for refinement, whereas the adaptive PRT approach tends to refine the mesh based on the proximity of the surfaces, subdividing even on completely dark regions. In top of Fig. 9, for instance, the ceiling and walls of the room are unnecessarily refined, as can be seen.

Concerning the lighting step, our refinement method is also suitable for being used in combination with other interactive illumination techniques. It can be used, for instance, as an optimal input model for PRT techniques. Fig. 10 shows results of the Ottawa city model refined with our method and illuminated using a PRT technique, where the sky model has been replaced by an environment map. Taking into account that our meshing takes around 13 minutes (see Table 1), the overall precomputation time is 15 minutes with only direct illumination (left) and 24 minutes for the global solution with two bounces (right). Instead, using the adaptive PRT approach available at [10], it took approximately 1 hour only for the direct illumination. The advantage of using our refined model as input for other techniques is that it brings an optimal mesh representation in a very fast way.

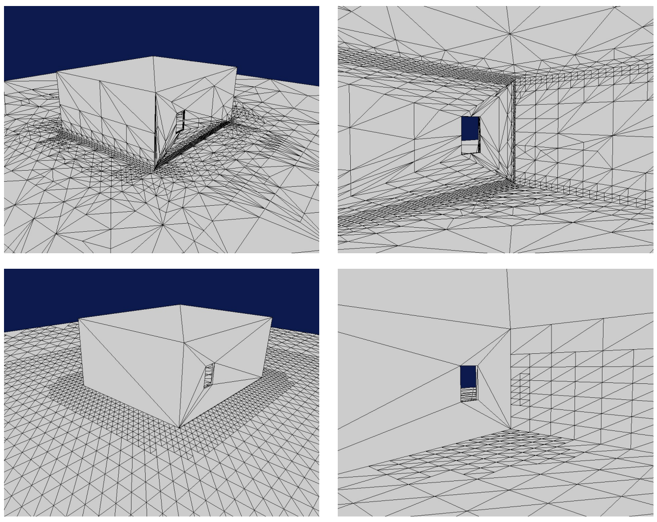

Fig. 9 Comparing refinement results. Top: refinement obtained using the adaptive PRT approach, seen from outside (left) and inside the room (right) (7933 triangles). Bottom: refinement obtained with our algorithm (5386 triangles). Notice how our refinement better adapts to sky visibility changes (see room ceiling and walls).

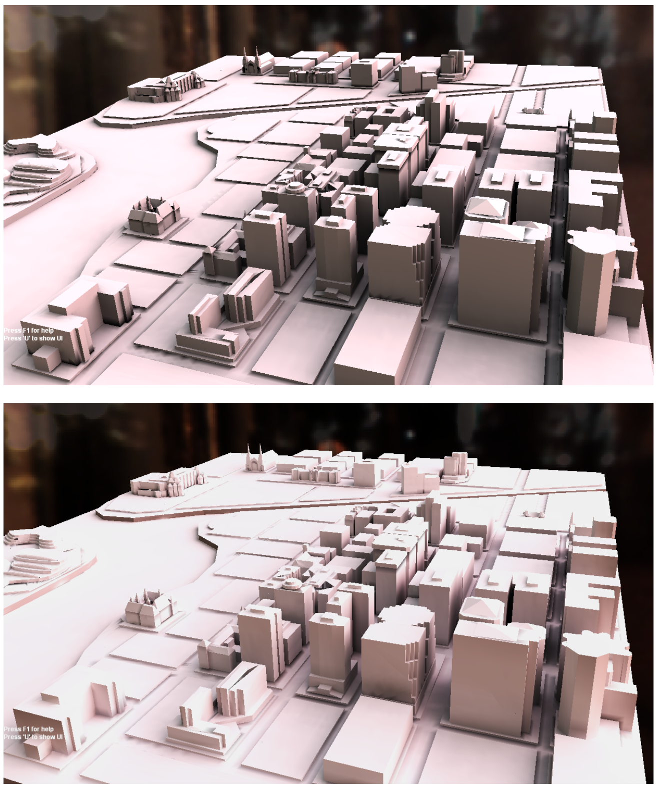

Fig. 10 Combining our adaptive mesh subdivision technique with PRT. The refined model of Ottawa (Fig. 8) is used as input for the PRT computation and then visualized in real-time with direct (left) and indirect (right) illumination. All surface materials were set to gray for this simulation.

Another possible contribution of our technique could be an accurate computation of ambient occlusion. This could be done by simply storing the mean visibility factors for all set of sky directions.

7 Conclusions and Future Work

We presented a daylight simulation method for interactive direct lighting visualization of generic environments. The main contribution with respect to current available solutions is that it provides a fast, accurate and adaptive mesh subdivision of the scene according to sky visibility values. With this approach, we can obtain efficient representations of complex models in short time and then perform daylight simulations with dynamic day-time conditions.

We believe that our approach could be used in real-time applications and computer games. It has been shown, for instance, that it can be combined with precomputed radiance transfer techniques in order to provide optimal pre-meshed models. It could be similarly used as a fast way to precompute accurate lightmaps for the composition of virtual scenarios.

Future extensions of our method include a global illumination pass for indirect lighting visualization and lighting analysis functionality for assessment. Another possible improvement would be its extension in order to deal with specular surfaces and dynamic scenes.

Acknowledgements This work was supported by TIN2004-07672-C03, TIN2007-67120 and Ramón y Cajal program of the spanish government. The museum model is a courtesy of Espacio Solar SL.

References

1. AGI32: Lighting software for calculations and visualizations. URL www.agi32.com
2. Ashdown, I.: Fast daylight simulation and analysis. In: Illuminating Engineering Society 2002 Annual Conference (IESNA '02), pp. 177–186 (2002)
3. Besuievsky, G., Martín, I.: A hierarchical algorithm for radiosity daylighting. In: Congreso Español de Informática Gráfica (CEIG'05). Granada, Spain (2005)
4. Cohen, M.F., Wallace, J.R.: Radiosity and Realistic Image Synthesis. Academic Press Professional (1993)
5. Daubert, K., Schirmacher, H., Sillion, F.X., Drettakis, G.: Hierarchical lighting simulation for outdoor scenes. In: Eurographics Rendering Workshop 1997, pp. 229–238 (1997)
6. DIALux: Light software. URL www.dialux.com
7. Franklin, D.: Shader X4, chap. Hardware-Based Ambient Occlusion. Charles River Media (2006)
8. Křivánek, J., Pattanaik, S., Žára, J.: Adaptive mesh subdivision for precomputed radiance transfer. In: Spring Conference on Computer Graphics 2004, pp. 106–111 (2004)
9. Languénou, E., Tellier, P.: Including physical light sources and daylight in global illumination. In: Third Eurographics Workshop on Rendering, pp. 217–225 (1992)
10. Microsoft: The DirectX Software Development Kit, v. 9. URL msdn2.microsoft.com
11. Miguet, F., Grouleau, D.: A daylight simulation tool for urban and architectural spaces-application to transmitted direct and diffuse light through glazing. Building Environments **11**(3), 833–843 (2002)
12. Müller, S., Kresse, W., Schoeffel, F.: A radiosity approach for the simulation of daylight. In: Eurographics Rendering Workshop 1995, pp. 137–146 (1995)
13. Nimeroff, J.S., Simoncelli, E., Dorsey, J.: Efficient re-rendering of naturally illuminated environments. In: Fifth Eurographics Workshop on Rendering, pp. 359–373 (1994)
14. Nishita, T., Nakamae, E.: Continuous tone representation of three-dimensional objects illuminated by sky light. In: Computer Graphics (Proceedings of SIGGRAPH 86), pp. 125–132 (1986)
15. Pharr, M., Green, S.: GPU Gems, chap. Ambient Occlusion, pp. 279–292. Addison Wesley (2004)
16. Preetham, A.J., Shirley, P.S., Smits, B.E.: A practical analytic model for daylight. In: Proceedings of SIGGRAPH 99, Computer Graphics Proceedings, Annual Conference Series, pp. 91–100 (1999)
17. Sattler, M., Sarlette, R., Zachmann, G., Klein, R.: Hardware-accelerated ambient occlusion computation. In: Vision, Modeling, and Visualization (VMV 2004), pp. 119–135 (2004)
18. Sloan, P.P., Kautz, J., Snyder, J.: Precomputed radiance transfer for real-time rendering in dynamic, low-frequency lighting environments. ACM Transactions on Graphics **21**(3), 527–536 (2002)
19. Walter, B., Fernandez, S., Arbree, A., Bala, K., Donikian, M., Greenberg, D.P.: Lightcuts: a scalable approach to illumination. ACM Transactions on Graphics **24**(3), 1098–1107 (2005)