Generation of Radiosity Texture Atlas for Realistic Real-Time Rendering

Nicolas Ray, Jean-Christophe Ulysse, Xavier Cavin and Bruno Lévy

Project Isa, Inria Lorraine, France

Abstract

Radiosity methods are a both physically correct and efficient way to compute the global illumination giving the visual atmosphere to a 3D scene. In this paper, we present a new approach to optimizing the visualization of meshed models illuminated using these methods. Our approach is based on a texture atlas, which makes it possible to store the global illumination in a set of textures, that can be mapped in real-time onto the model by the graphics hardware. Our contribution is a new robust atlas generation method well adapted to the visualization of illuminated complex meshed models, and based on a tight cooperation between the segmentation and the parameterization algorithms. In our segmentation method, the combinatorial information of the mesh is systematically preferred over the geometry whenever possible, which makes the algorithm both simpler and more robust. Large industrial models with complex topologies and poor mesh qualities can be efficiently processed. The paper concludes with some results, showing our method applied to architectural and car design industry models, and demonstrating that it both speeds up the visualization and is easy to integrate into existing advanced rendering software.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism – Color, shading, shadowing and texture

1. Introduction

Realistic rendering of virtual worlds requires to accurately (*i.e.* physically) compute the direct and indirect effects of light sources on the scene. In real-time rendering, the norm is to rely on a local lighting model, to provide believable realistic apperance. In this local model, only the surface data at the visible point is needed to compute the lighting, allowing very efficient hardware implementations. However, more complex effects, such as soft shadows or indirect illumination, can not be handled by such a coarse approximation.

In a virtual environment with static lighting conditions, the diffuse component on any surface remains the same from any point of view, and can then be pre-computed off-line, using for instance radiosity methods. At the rendering stage, this pre-computed illumination is used to modulate the appearance of each surface. Rendering a scene with such a pre-computed illumination can be achieved through two different approaches: each surface can be subdivided in order to store the amount of light reaching each vertex, or a texture map can be attached to each surface to capture the contribution of light.

The surface refinement approach is the simplest one but presents two main limitations: the number of mesh elements required to capture the illumination may overcome current graphics hardware capabilities, and the geometry modifications applied to the initial surfaces make it harder to combine the illumination information with other features of an existing rendering system (*e.g.* environment mapping, bump mapping, level of details, *etc.*)

To overcome these limitations, this paper investigates the second approach. Our solution automatically generates a texture atlas that will be used as texture coordinates to capture the illumination of a 3D model as a texture map. This way, adding pre-computed illumination to an existing rendering system only requires to rasterize an extra texture.

Since most modern graphics hardware allows multiple textures (currently up to eight) to be applied in a single rendering pass, combining the illumination texture map with environment or bump maps does not affect the overall performance (assuming enough texture memory is available). Moreover, several pre-computed illumination contributions (one per set of light sources) can be combined in real-time to simulate a dynamic illumination. Finally, avoiding the introduction of extra geometry enables multi-resolution structures such as level of details or progressive meshes to share the same texture information between all resolutions.

1.1. Previous Work

The computation of radiosity has found use within the real-time rendering community as a mean to add visual richness to a virtual environment, while pre-computing all diffuse components, thereby allowing faster redisplay than computing these on the fly.

The radiosity - power per unit area $[W/r^2]$ - on a given surface, is governed by an integral equation which can be solved by projecting the unknown radiosity function $B(x)$ onto a set of basis functions ϕ_i with limited supports:

$$B(x) \approx \tilde{B}(x) = \sum_{i=1}^{n} \alpha_i \phi_i(x)$$

Different sets of basis functions and resolution algorithms have been investigated to determine the α_i coefficients (see [4, 15] for more details).

An important problem is then to find a representation of the computed radiosity approximation $\tilde{B}(x)$ that can be efficiently rendered using the graphics hardware. In the radiosity literature, two main approaches have been studied to tackle this problem, based respectively on mesh and texture techniques.

Mesh based techniques

A typical solution is to refine the initial geometry into a sufficient number of triangles, so that the illumination can be stored by color per vertex and rendered using linear interpolation. Unfortunately, even for moderate scale scenes, the number of required triangles is often too high to be rendered in real-time.

Two transverse approaches have been proposed to reduce the refinement needed to represent the illumination function: mesh simplification and discontinuity meshing.

Mesh simplication is the process of reducing the polygon count of a detailed model while preserving its apperance. It is based on a cost function that attemps to maintain given features of the model, such as its volume, crease and boundary edges, or locations presenting color changes. Hoppe [8] has shown that radiosity solutions using color per vertex to record the illumination are excellent candidates for reduction techniques.

The other approach, introduced by Heckbert [7], is to use discontinuity meshing during the radiosity computation. Discontinuity meshing is an approach to meshing that attempts to accurately resolve the most significant discontinuities in the solution by optimal positioning of element boundaries.

Both techniques usually give good results at reducing the number of triangles needed to represent the illumination function, and may be applied for a simple rendering of a simulation result. Their main drawback however lays in the introduction of extra geometry, which makes it harder to combine the illumination information with other features of an existing rendering software. In particular:

- The extra geometry may still be too costly to render, compared to the initial scene.
- If attributes where attached to the initial geometry (like a bump map or an environment map), they must be coherently generated for all the new geometry.
- If the initial scene uses multi-resolution structures (like level of details or progressive meshes), the illumination meshes must be merged into every level.
- Combining several pre-computed illumination simulations leads to merge potentially complex meshes, which must be done by resource-intensive multi-pass rendering.

Texture based techniques

A common alternative used in the real-time rendering community is to decouple lighting from geometry by storing the illumination information as a texture map [1, 6, 12, 3], generally referred to as *light map* or *illumination map*. At the rendering stage, this texture is applied to the surface to modulate its appearance based on the lighting information. Since modern graphics hardware is optimized to render textured surfaces, the illumination information can easily be added to a virtual environment without slowing down the rendering process. Moreover, since the initial geometry of the scene is kept untouched, the light maps can easily be integrated into an existing rendering software.

Basically, a light map allows to store multiple light samples per polygon, as opposed to the mesh-based approach where only one color per vertex is allowed. Applications of light maps have been found for interactive walkthrough of architecural scenes, mostly composed of large polygons (wall, floor, *etc.*) For smaller simple objects (table, chair, staircase, *etc.*), packing methods have been introduced to generate a set of light maps in a single texture for each object (or group of objects) rather than for each polygon, in order to minimize costly texture switching operations.

As shown in Figure 1, representing the illumination on each polygon requires to use all texels that cross the polygon, including the non-completely covered one (on the polygon edges). Moreover, to avoid aliasing effects due to graphics hardware bilinear filtering, extra texels need to be inserted around the polygon. Keeping polygon adjacency in texture space would allow pertinent texels (the ones with illumina-

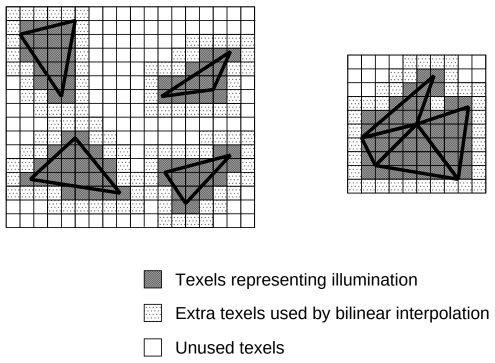
Texels representing illumination
Extra texels used by bilinear interpolation
Unused texels

Figure 1: *Texture space optimization: one lightmap per polygon (on the left) against one lightmap for adjacent polygons (on the right).*

tion information) to be shared between polygons and consequently limit the wasted space. Another benefit is that (illumination) texture coordinates can be shared between the vertices of the adjacent polygons, which is a required property for using optimization structures like triangle strips or vertex arrays.

Few solutions have been proposed that generate a light map for a set of adjacent polygons. Möller [11] described a method to replace radiosity solutions for NURBS (geometry) models with a single texture map. Zhukov *et al.* [16] introduced the "polypack" structure: a "polypack" is a set of adjacent polygons that can be projected onto one of the world coordinate planes without overlapping, and can be mapped with one light map. However, these methods are restricted to specific models that do no exhibit complex topological configurations.

1.2. Our approach

We propose in this paper to use a *texture atlas* to store the illumination on a polygonized 3D model. The texture atlas, introduced by Maillot *et al.* [10], is a set of maps packed in a 2D square. The 3D model to be textured is partitionned into a set of parts, referred to as *charts*, and each of them is provided with a parameterization. Finally, the unfolded charts are packed in a 2D texture. In our case, each map represents the illumination of a chart of the 3D model. This representation is the most efficient way to benefit from modern graphics hardware.

Lévy *et al.* [9] described a solution to automatically generate a texture atlas from a triangulated surface, based on the following contributions:

- Segmentation methods to decompose the model into charts with natural shapes.
- A new optimization-based parameterization method: Least Squares Conformal Maps (LSCMs).
- Two criteria to validate the unfolded charts, and associated subdivision rules for the invalid ones.
- A new packing algorithm to gather the unfolded charts in texture space.

We propose in this paper to adapt this solution to generate radiosity texture atlases for industrial models, such as the ones found in architecture or computer aided design (CAD). These models are typically composed of a huge number of triangles, including a non negligible amount of T-vertices, non-manifold edges, sharp edges and bad shaped triangles. In those cases, their segmentation method, based on differential geometry and Morse function analysis, would generate too many charts.

In Section 2, we present a more robust texture atlas generation method. Then, Section 3 presents the radiosity texture generation. The paper concludes in Section 4 with a presentation of first results obtained with our method and a discussion of limitations and future work in Section 5.

2. Texture Atlas Generation

We introduce in this Section a new pipeline, depicted on Figure 2, to generate a texture atlas. We use a recursive approach in which the segmentation is only used to split the chart when its map is invalid (*i.e.* not one-to-one or too distorted). This approach allows to generate a map for non topological discs, like surfaces including T-vertices or inner borders.

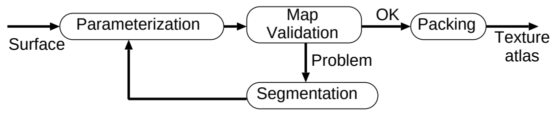

Figure 2: *The texture atlas generation pipeline.*

The choice of the LSCM parameterization method is motivated by the properties proved by its authors:

- Their conformal criterion minimizes angle deformations, which allows to often find a valid parameterization.
- The parameterization is fast enough to be applied recursively (in cooperation with the segmentation) until a valid and low distorted map is found.
- The natural border extrapolation avoids the distortions implied by the fixed border of other methods.
- The result is independent of the resolution of the mesh. This property reduces texture swimming when applied on a multiresolution structure such as progessive mesh.

In the remainder of this Section, we present a new condition added to the map validation, that checks a filling ratio to minimize the waste of texture space, and we introduce a more robust segmentation method to split charts without any assumption on the mesh quality. The parameterization and the packing are performed as in [9].

2.1. Map validation

As explained in [9], some surfaces are too complex to be correctly parameterized. In such cases, triangles may overlap in the parameteric space or the map could present important variations of area. In order to avoid these problems, the authors proposed to add a validation step to ensure the validity of the parameterization: overlappings are managed in 2D space and high area distortions are removed thanks to an extra segmentation.

In CAD models, waste of memory also occurs due to complex borders or holes in surfaces. To solve this problem, we add a third test in the validation step to control the efficiency of the parameterization in terms of occupancy quality.

To do so, we count the number of texels effectively rendered during the rendering process of the triangles needed for calculating the overlaps, and compare it to the bounding box of the chart. Thus, the ratio $\frac{nb\_texels\_used}{nb\_texels\_of\_chart's\_bounding\_box}$ gives a *filling ratio* that has to be as near as possible to 1.0 in order to validate the parameterization of the current chart.

2.2. Segmentation

The segmentation is used when the map is considered invalid. It relaxes the continuity constraint of the mapping function along some edges in order to ensure the validity and the quality of the generated maps. The approaches based on differential geometry and Morse theory used in [9] are not able to efficiently deal with industrial models including bad shaped triangles, like those shown in Figure 3. A simpler and more robust solution is to recursively split the model until each chart is correctly parameterized.

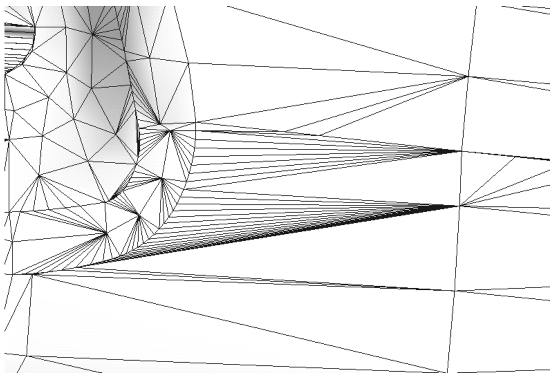

Figure 3: *Example of bad shaped triangles.*

The chart split is simply performed by a greedy algorithm that grows two charts. In order to find far enough seeds, a random vertex v is chosen, the farest vertex A from v is set as the first seed, and the farest vertex B from A is set as the second one.

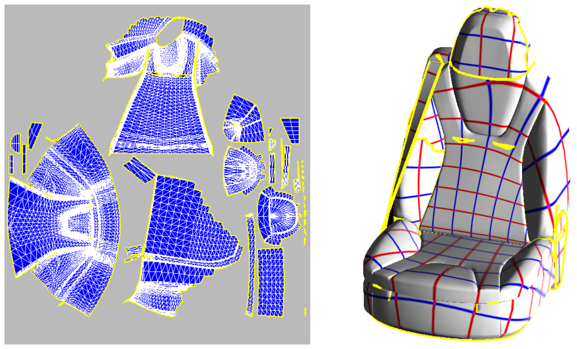

Figure 4: *Generation of a texture atlas for the seat model.*

Combined with the validation step, our segmentation method ensures to find a valid solution with a controlled distortion. As shown in Figure 4, all kinds of surfaces can be segmented into a few charts. This algorithm is brute force, but it has proven to be robust and suitable for our purpose. However, as discussed in the future work of Section 4, taking the surface topology and geometry into account should be used to improve the segmentation.

3. Radiosity Texture Generation

We briefly present in this Section how the radiosity texture is generated based on the texture atlas computed in Section 2, *i.e.* how the texture atlas of the geometry is rasterized with the illumination function computed by the global illumination simulation.

The result of a wavelet radiosity computation is an illumination function $E(u,v) \rightarrow (R,G,B)$ defined on a local normalized mapping of each mesh element. The first step of our lightmap generation method approximates the function with a triangulation in the parametric domain with colors stored at the triangle corners.

The second step of our method uses theses triangles to rasterize the illumination function on each mesh element. As show by Figure 5, the texture atlas defines how to transform their local (mesh element) coordinates (u,v) to global (texture atlas) coordinates.

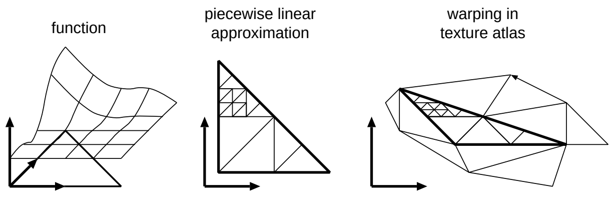

Figure 5: *Rasterization of the illumination function in the texture atlas.*

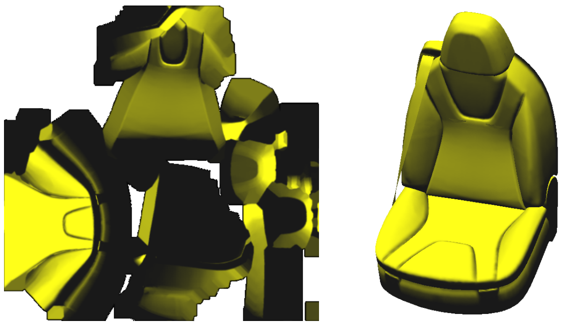

Figure 6: *The radiosity texture atlas (left) mapped to the seat model (right).*

The radiosity texture atlas can then be mapped to the object geometry like other existing textures, as shown by Figure 6. However during the rendering phase, the unused texels (see Figure 1) will be melted by the graphics hardware with the ones with illumination information. This generates two kinds of artifact near chart boundaries:

- Aliasing due to hardware bilinear interpolation of texture colors.
- Bad mipmap colors due to undefined colors averaging.

Our solution is to use a dilatation algorithm to avoid aliasing and a push-pull algorithm to enable mipmaps to be correctly generated, as done in [14].

4. First results

We have applied our radiosity texture atlas generation method to different industrial models from architectural and car design industries. Our experiments show that our radiosity textures can easily be integrated into an existing rendering software, and dramatically increase the realism of a virtual scene without slowing down the rendering speed (assuming enough texture memory is available on the graphics hardware).

Our method can handle complex illumination effects that have been computed on an arbitrary huge amount of mesh elements, and could not have been rendered by traditional approaches. Figure 7 shows an architectural scene representing an immersive visualization system with colorful effects on the floor. This model can be visualized at full speed on a standard graphics PC.

Our method has also been used to improve realism in an existing rendering software of a car design department. Our method can generate radiosity texture atlas even for highly complex meshes, as shown by Figure 8. We have been able to integrate this new feature into their rendering software, without conflicting with other existing features (including: material texture, reflection map, bump map, triangle strips, levels of details, *etc.*) designers are used to and do not want

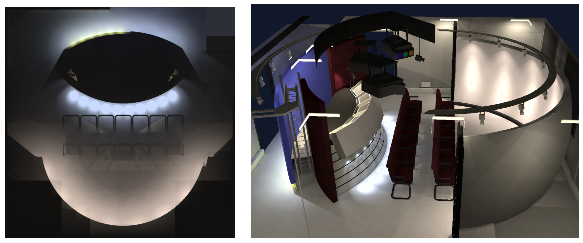

Figure 7: *All complex shadows in this architectural model (right) are captured by radiosity textures (left) and do not slow down the rendering.*

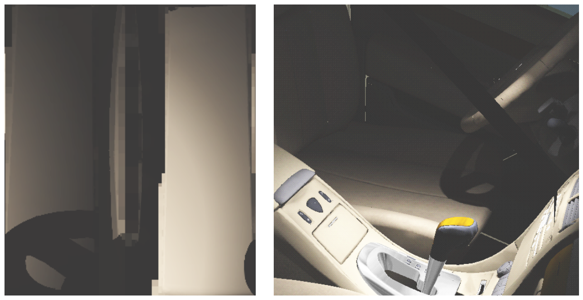

Figure 8: *All complex meshes (right) are mapped with a radiosity texture atlas (left) combined with existing material textures.*

to loose. Furthemore, we have added the possibility to store multiple radiosity textures per object in order to simulate dynamic lighting conditions.

5. Conclusion and Future Work

The solution presented in this paper allows to add precomputed global illumination in real-time rendering of virtual scenes. Storing radiosity in textures has proven to be both fast to render and easy to integrate into an existing rendering system. However, our actual solution do not use the available texture space in an optimal way. Two complementary approaches should be used to improve this drawback.

First, the texture atlas generation can be optimized at all steps (parameterization, segmentation and packing):

- The LSCM method suffers from instabilities with very bad shaped triangles (*i.e.* very narrow triangles) that should be removed from the conformal energy to minimize. This upgrade should limit the number of segmentations needed to find a valid and low distorted map.
- The high level of cooperation between the segmentation and the parameterization makes the speed of the whole process strongly dependent from the LSCM method. Using a hierarchical version of the LSCM method would both improve speed and enable to deal with extremely large datasets (see Figure 11).

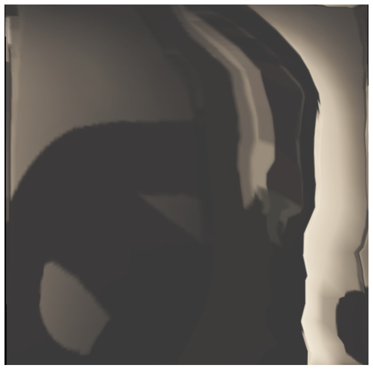

Figure 9: *Fish-eye effect when improving frequencies repartition using* [13] *on the radiosity texture atlas of Figure 8.*

- The segmentation step is currently a brute force approach. An analysis of the topological and geometrical properties of the surface should help in finding more adapted segmentation strategies for each kind of mesh. The result of the parameterization algorithm (invalid or too distorted map) also provides useful informations that can be used during the segmentation step, like in [5].
- Since sharp edges lead to illumination discontinuities, they should also create discontinuities in the texture atlas. An efficient way to detect such edges should improve our lightmap generation. Unfortunately, simple solutions like SOD (Second Order Difference) fail in high curvatures zones, and more complex curvature estimations assume that no sharp edge exists in a given neighborhood.

The second way to minimize the amount of texture required to store the illumination is to optimize the texture atlas in order to have a better frequency repartition in the radiosity texture. Indeed, the illumination signal represents both very high frequencies around shadow boundaries (like the wheel's shadow in Figure 8) and very low frequencies due to light reflection (everywhere else in Figure 8). The texture resolution required to efficiently represent high frequencies leads to an oversampling of the illumination for the rest of the chart.

Different approaches to avoid this waste of texture space will be investigated. The chart parameterization can take the signal into account, like in [13]. This method improves frequencies repartition, but leads to high distortions of the map with a fish-eye like effect, as shown in Figure 9. A simpler approach will be to optimize the texture atlas with an extra segmentation to separate charts with high frequencies in order to scale them in texture space, like in Figure 10. A last optimisation step could also be used to directly optimize the texture atlas like in [2].

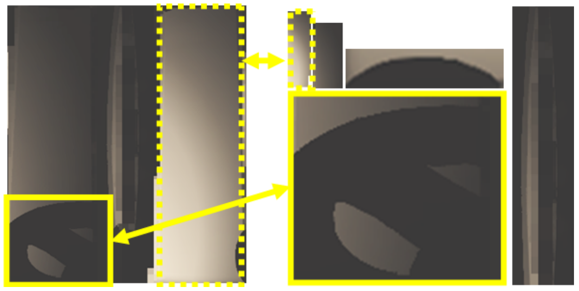

Figure 10: *Optimizing space with an extra segmentation of the radiosity texture atlas of Figure 8.*

Acknowledgements

This work has been supported by the CRVHP branch of the PRST “Intelligence Logicielle”. We would like to thank the research and development team of the Candelux software at Inria Lorraine and VSP Technology. Thanks also to the Graphite research team for providing the parameterization software. CAD car model courtesy of Renault. CAD Reality Center model courtesy of SGI. All illuminated models courtesy of VSP Technology.

References

1. James R. Arvo. Backward Ray Tracing. In *ACM SIGGRAPH '86 Course Notes - Developments in Ray Tracing*, volume 12, August 1986. 2
2. Laurent Balmelli, Fausto Bernardini, and Gabriel Taubin. Space-optimized texture maps. *Computer Graphics Forum (Proceeding of Eurographics)*, 21(3):411–420, 2002. 6
3. Rui Bastos, Michael Goslin, and Norman I. Badler. Efficient rendering of radiosity using texture and bicubic interpolation. In *1997 Symposium on Interactive 3D Graphics*, pages 71–74. ACM SIGGRAPH, April 1997. 2
4. Michael F. Cohen and John R. Wallace. *Radiosity and Realistic Image Synthesis*. Academic Press Professional, Boston, MA, 1993. 2
5. Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361, 2002. 6
6. Paul Heckbert. Adaptive Radiosity Textures for Bidirectional Ray Tracing. *Computer Graphics (ACM SIGGRAPH '90 Proceedings)*, 24(4):145–154, August 1990. 2
7. Paul Heckbert. Discontinuity Meshing for Radiosity. In *Third Eurographics Workshop on Rendering*, pages 203–226, Bristol, UK, May 1992. 2

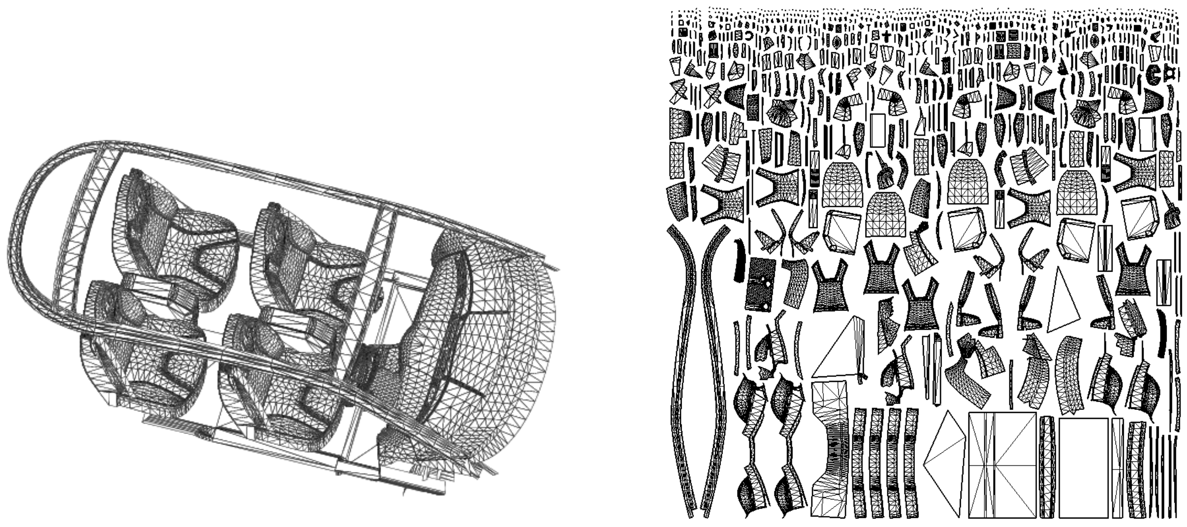

Figure 11: *Atlas generation for an extremely large model with a hierarchical LSCM method (early result).*

8. Hugues Hoppe. Progressive meshes. *Computer Graphics (SIGGRAPH'96 Proceedings)*, 30:99–108, 1996. 2

9. Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. *acm Transactions on Graphics (Proceedings of ACM SIGGRAPH 2002)*, 21(3):362–371, July 2002. 3, 4

10. Jérôme Maillot, Hussein Yahia, and Anne Verroust. Interactive texture mapping. In *Proceedings of the 20th annual conference on Computer graphics and interactive techniques*, pages 27–34. ACM Press, 1993. 3

11. T. Moller. Radiosity techniques for virtual reality - faster reconstruction and support for levels of detail. In *Proceedings of WSCG 96*, pages 209–216, 1996. 3

12. Karol Myszkowski and Tosiyasu L. Kunii. Texture Mapping as an Alternative for Meshing During Walkthrough Animation. In *Fifth Eurographics Workshop on Rendering*, pages 375–388, Darmstadt, Germany, June 1994. 2

13. P. Sander, S. Gortler, J. Snyder, and H. Hoppe. Signal-specialized parametrization. Technical report, Microsoft Research, 2002. 6

14. Pedro V. Sander, John Snyder, Steven J. Gortler, and Hugues Hoppe. Texture mapping progressive meshes. In *SIGGRAPH 2001 Conference Proceedings, August 12–17, 2001, Los Angeles, CA*, pages 409–416, 2001. 5

15. Francois Sillion and Claude Puech. *Radiosity and Global Illumination*. Morgan Kaufmann, San Francisco, CA, 1994. 2

16. S. Zhukov, A. Iones, and G. Kronin. Using light maps to create realistic lighting in real-time applications. In *Proceedings of WSCG '98*, pages 464–471, 1998. 3