EUROGRAPHICS '97 / D. Fellner and L. Szirmay-Kalos
(Guest Editors)

Volume 16, (*1997*), *Number 3*

Efficient Impostor Manipulation for Real-Time Visualization of Urban Scenery

François Sillion, George Drettakis, Benoit Bodelet

iMAGIS † – GRAVIR/IMAG - INRIA. B.P. 53, F-38041 Grenoble Cedex 9.

Abstract

Urban environments present unique challenges to interactive visualization systems, because of the huge complexity of the geometrical data and the widely varying visibility conditions. This paper introduces a new framework for real-time visualisation of such urban scenes. The central concept is that of a dynamic segmentation of the dataset, into a local three-dimensional model and a set of impostors used to represent distant scenery. A segmentation model is presented, based on inherent urban structure. A new impostor structure is introduced, derived from the level-of-detail approach. Impostors combine three-dimensional geometry to correctly model large depth discontinuities and parallax, and textures to rapidly display visual detail. We present the algorithms necessary for the creation of accurate and efficient three-dimensional impostors. The implementation of our algorithms allows interactive navigation in complex urban databases, as required by many applications.

Keywords: *Visualization of large datasets, Image-based rendering, Image caching, Impostors, Urban scenes.*

1. Introduction

Visualisation of urban environments is an exciting domain, with a growing number of important applications. Examples of such use include *city planning*, such as the visualisation of South Central Los Angeles, built and visualised by the UCLA School of Architecture for city planning [1], navigation aid systems for automobiles, driving and flight simulators for civilian and military use, climate and environmental studies, virtual tourism and education etc. The sheer size of the data required to represent a city presents particularly challenging problems for visualisation. In addition, the special structure of cities introduces additional difficulties: the volume of data visible from a given point can change drastically, from densely occluded (e.g., only a few neighbouring buildings visible), to panoramic views (e.g., from a hilltop) where millions of geometric primitives are visible.

Since all of the applications mentioned above require real-time feedback, rendering such large data sets using traditional techniques, such as frustum culling, is problematic. In this sense, the problem of urban visualisation can be seen as a challenge of performing *appropriate* geometric simplification. Two contrasting approaches to simplification have recently become popular, either for visualisation or other applications. The first tendency targets the reduction of the geometric complexity of individual objects. These approaches often implicitly assume the availability of connectivity information, used to maintain a consistent topology [2]. Such information is not readily available for most urban data. Simplification methods which do not preserve topology can make it difficult to control appearance properties such as color[3]. In addition, geometric simplification algorithms are rarely view-dependent, and thus inappropriate for walkthrough-type applications. The second approach assumes that no information is available other than a set of geometric primitives. As a consequence, complex geometry is substituted with an *image* "impostor", or "cache" [4, 5, 6]. The latter approaches have given impressive results, but are restricted in the type of scene they can treat, since they typically fail for densely occluded environments, and suffer from the fact that the number of frames for which these image impostors are valid is very limited due to the problems of parallax.

The new algorithm we present here introduces a solution which combines the strong points of both approaches, and exploits the specific nature of urban landscapes. On the one hand, we use the idea of image impostors, taking advantage

† iMAGIS is a joint research project of CNRS, INRIA, INPG and UJF. Contact E-mail: `Francois.Sillion@imag.fr`.

of the capabilities of texture for rapid display of detail information, and on the other hand we introduce the idea of impostors augmented with *three-dimensional* information, allowing longer cache life, and largely resolving the parallax problem. In addition, we provide a subdivision model of urban scenes, based on the inherent city structure, resulting in a segmentation scheme which is much better adapted than frustum culling. This segmentation provides a full three-dimensional model for the local neighbourhood (or "near field"), and the use of the augmented impostors for more distant landscape. The availability of the full 3D model for the near field can be very important, especially if operations such as collision detection (e.g., for games) or a faithful simulation of the illumination (interreflections for example) are required. The implementation of our new algorithm allows real-time visualisation of very large urban data-bases; the new three-dimensional impostor method results in longer cache life, and the capacity to treat parallax in many cases.

2. Previous work

The focus of most previous related work has been on the development of algorithms for visualisation of large scenes, notably based on database partitioning and culling, on level-of-detail approaches, and image-based rendering.

2.1. Database partitioning and visibility culling

The first algorithms for accelerating visualisation of large architectural models can be traced to the early stages of computer graphics research [7, 8]. Airey *et al.*, Teller and Séquin, Luebke and Georges [9, 10, 11], use spatial subdivision and attempt to precompute visibility relationships within a complex building scene. The central idea of these approaches is to predict the visible part of a scene for the next few frames, thus reducing the number of primitives which must be rendered. This is achieved using intelligent memory management and viewer motion prediction.

A different approach involved the hierarchical Z-buffer algorithm, which uses Z-buffer pyramids to rapidly eliminate occluded parts of the scene [12]. Finally, the issue of constant-frame rate for visualisation of complex environments has been addressed by Funkhouser *et al.*[13]. A computational geometry approach was presented in [14], in which large occluders are dynamically identified as the user moves and used to perform culling using an octree structure. This method successfully eliminates large portions of the model which are invisible but requires large occluders.

2.2. Image-based rendering

The idea of replacing full three-dimensional models by a rendered image can be traced back to the idea of environment maps [15]. Their direct application to rendering was used by Chen and Williams [16], using range-images. This approach allowed limited motion around a viewpoint by using pre-rendered images of the scene, which was then used to substitute three-dimensional rendering on low-end computers. More involved approaches include the use of panoramic images used in Quicktime-VR [17] and plenoptic modelling [18]. These methods allow the choice of different viewpoints, and are noteworthy in the fact that they allow limited three-dimensional navigation of real scenes.

Maciel and Shirley introduced the idea of image "impostors". In their work, a hierarchy of a three-dimensional complex model is created; on the faces of the bounding boxes images of the cluster contents are created. These images can be used to replace the contents when this representation is judged sufficient. In other approaches [19, 5, 6] images of distant scenery are cached, and replace complex geometry based on an image discrepancy criterion: when an image replacing complex distant geometry is no longer accurate, the cache is invalidated. The Talisman [20] approach moves in the same direction.

Finally, the idea of representing three-dimensional scenes as a light-field was presented by [21, 22]. In such approaches the complexity of the geometry contained in a scene is relatively unimportant, and *slices* of this representation are extracted to generate images.

2.3. Shortcomings of previous approaches

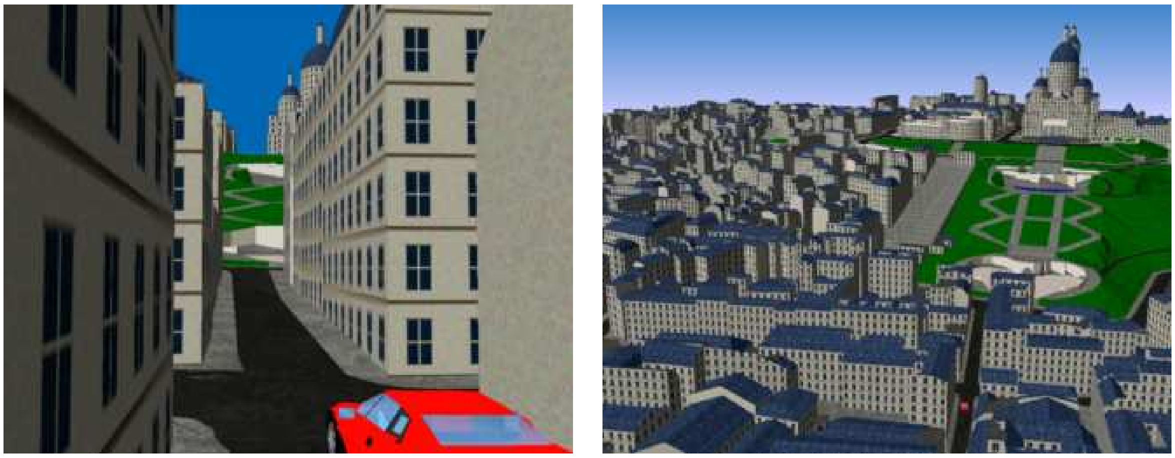
The approaches briefly summarised above have resulted in the efficient visualisation of large data sets, in the case of partitioning and culling for environments which are always densely occluded (such as building interiors), or on the contrary for environments which are never densely occluded in the near-field for the image caching methods.

Inherently, urban environments require the treatment of dense occlusion in the near-field, and that of potentially large data-sets in the distant landscape. The urban data-space segmentation scheme, in conjunction with the new three-dimensionally augmented impostors provides a powerful solution to this challenge.

The dynamic octree culling approach using large occluders [14] could be used as a first stage in our approach, but does not address the case where large amounts of 3D geometry are visible in the distance.

3. An efficient model for urban navigation

Urban models have several distinct properties, unlike other data sets. In general, they have very high geometric complexity, typically requiring very large datasets to represent even the simplest neighbourhood. In addition, even though at the street level there is a high degree of occlusion, it is often the case that the amount of data visible can change abruptly and unpredictably. This occurs for example when turning from a narrow street into a large open square with a view to a hill.



(a) (b)

Figure 1: *Example view of an urban scene. (a) from the street level, (b) from higher up, looking in the same overall direction (See also the color section at the end of this volume).*

Furthermore, urban models are typically quite large, since they extend over several square kilometers.

As a consequence of these properties, traditional methods such as frustum culling are not very well adapted to the needs of urban visualisation. Consider for example the scene depicted in Figure 1: the model is constructed from polyhedral data representing a 2km $\times$3km area of Paris around Montmartre (courtesy of IGN) and consists of 140,800 polygons (note that this rather large number of primitives still only allows a crude geometrical description of the buildings).

The image on the left shows a view taken from street level; clearly only few of the objects are actually visible, but simple culling strategies cannot decide what is relevant for this view and must consider the entire depth of the scene within the frustrum. In this example frustrum culling selects 45,200 polygons, which still far exceeds the real-time capabilities of current mid-range computers. The full depth complexity of the model can be estimated on the right-hand image taken from above.

Note that culling away objects based on the distance to the viewpoint would not be a good idea since it would eliminate landmarks such as the church in the background. Therefore a difficult character of urban scenes becomes apparent with this example: while much of the complexity is typically hidden (for ground views) due to important occlusion, some directions let the user view objects at far distances.

In the new approach presented here we use the idea of a *local neighbourhood*, which has a natural interpretation in the context of urban landscapes. Cities are organised by streets and blocks, and can thus be directly segmented into a local region (for example the blocks adjacent to the street we are moving on) and a distant landscape which contains the rest of the city geometry.

The central principle of our approach is to maintain full three-dimensional geometric information of the local neighborhood, while maintaining an approximate view of the distant landscape, which we call an *impostor*. We will be using inherent urban structure to guide this segmentation and the partitioning, as well as to aid in building and maintaining the accuracy of the impostors.

A straightforward approach implementing these ideas would be the use of textures, as impostors, potentially with the aid of some hierarchical structure [4]. However, such an approach suffers from the constant need to update the image caches which have become invalid due to parallax.

A more promising solution consists in using additional three dimensional information to create augmented impostors, containing for example depth information, contours of the sky-scape etc. These 3D impostors have longer “cache life”, compared to simple texture-based approaches and are more accurate since the additional information can be used to control their validity. Furthermore, the segmentation based on the urban structure potentially allows us to select the number and location of impostors in an optimal fashion.

This approach gives us the additional benefit of having more control over the error committed with respect to depth, while maintaining the advantage of textures which capture fine, but distant, detail very compactly. The principles introduced above are illustrated in Figure 2, where we show the local model associated with a selected viewer configuration. An impostor is constructed to complement the local model for nearby viewpoints. The images on the right clearly show the three-dimensional nature of the impostor.

3.1. Segmentation of the city model

As mentioned above, we propose the segmentation of the model into a nearby local neighbourhood, and more distant landscape, represented by impostors. The segmentation will typically be based on natural urban subdivision ("blocks" divided by streets for example).

We note that the directions along which distant objects can be visible usually correspond to streets or non-built areas, which can be extracted from an analysis of the urban database. Impostors can therefore be associated to these "important" directions, so that they reproduce exact views at well-chosen vantage points in the city, for a given resolution. This segmentation will also permit the combination of distant views in an efficient manner, based on the topological relationships in the arrangement of impostors, further enhancing the "cache-lifetime" of the impostors.

We consider an interactive application in which the user navigates in the urban scene. At any time the viewer location, and direction of sight, are known. Segmenting the model actually means that we partition the space of viewer position and direction of sight into a number of cells, each possessing all the necessary data to quickly render images for viewers in the corresponding neighborhood. Note that the word "neighborhood" here should conceptually refer to a portion of a five-dimensional space (position and direction). However in practice the segmentation operates in spaces of fewer dimenions thanks to the very high structure of urban data.

In the remainder of this paper, we will use the following terminology. For each cell, we define:

- The *local model* as a fraction of the scene 3D model, extracted from the complete model using an appropriate segmentation technique.
- The *distant model* as the remainder of the 3D scene.
- An Impostor as a textured three-dimensional object used to render distant geometry.

Each cell can possess one or more impostors as needed. The essence of the rendering algorithm is then to always draw the combination of the impostors of the current cell and its local model, to obtain a view of the entire model.

The desirable characteristics for a segmentation of the model are therefore primarily linked to its ability to represent distant landscape for cells that are as large as possible (to minimize their number), using the simplest possible impostors (to minimize their cost) and with the smallest possible image error (to avoid annoying artifacts). In particular, the use of a local 3D model and a set of impostors raises the issue of the boundary region between the models. Misalignement of the two boundaries results in "cracking" problems which should be minimized. It is very important to follow the structure of the urban landscape to perform the segmentation: $k-d$ trees, quadtrees or other regular structures are not well suited to this task since they will not produce meaningful chunks of data.

3.2. Definition of three-dimensional impostors

A three-dimensional impostor is initially based on the generation of an image which captures small details of the distant urban landscape in the form of texture information, and exactly complements the view of the current cell's local model for a specific vantage point.

Nonetheless, additional three-dimensional information should be stored with the image from the outset. This is necessary to perform appropriate deformations of the image when the user moves around the initial vantage point, and therefore allow cells to reach a certain size. Instead of deforming, or morphing, the texture image, we can also build from it a collection of three-dimensional elements, which can then be visualised efficiently just like other 3D data.

The main issue for the generation of optimal impostors is the determination of what should be represented as 3D information, and what can be safely left as texture information. 3D information is needed to correctly reproduce parallax effects as the user moves and portions of the distant model occlude other portions. However, small geometric details on facades, for instance, need not be integrated in the 3D information. Thus the addition of three-dimensional information to the impostors can be thought of as a constrained form of geometric simplification, whereby an initial dense mesh (the initial impostor image) is simplified to reduce polygon count while approximating the data within a given tolerance.

Considering that a "simplified mesh" has been created on the impostor image, this mesh can be reprojected in 3D by means of the depth information from a z-buffer, yielding a 3D textured polygonal mesh.

Important desirable properties for impostors therefore center on the ability to reconstruct an accurate view of the distant landscape in a neighborhood around the initial vantage point where the impostor was created. For this we need rich texture information and the relevant 3D structure to recreate important parallax effects, but the 3D complexity should be minimized to avoid excessive costs. The trade-off between geometric and texture detail has been discussed elsewhere [23], in the context of a single object. Simple and accurate validity criteria are also needed to keep reconstruction error in the final image under control. In particular, the "cracking" problem mentioned above should be carefully monitored. In this respect, we see that using simple planar impostors, or image caches [5,6], is not a viable solution because we always render large portions of the model on an impostor.

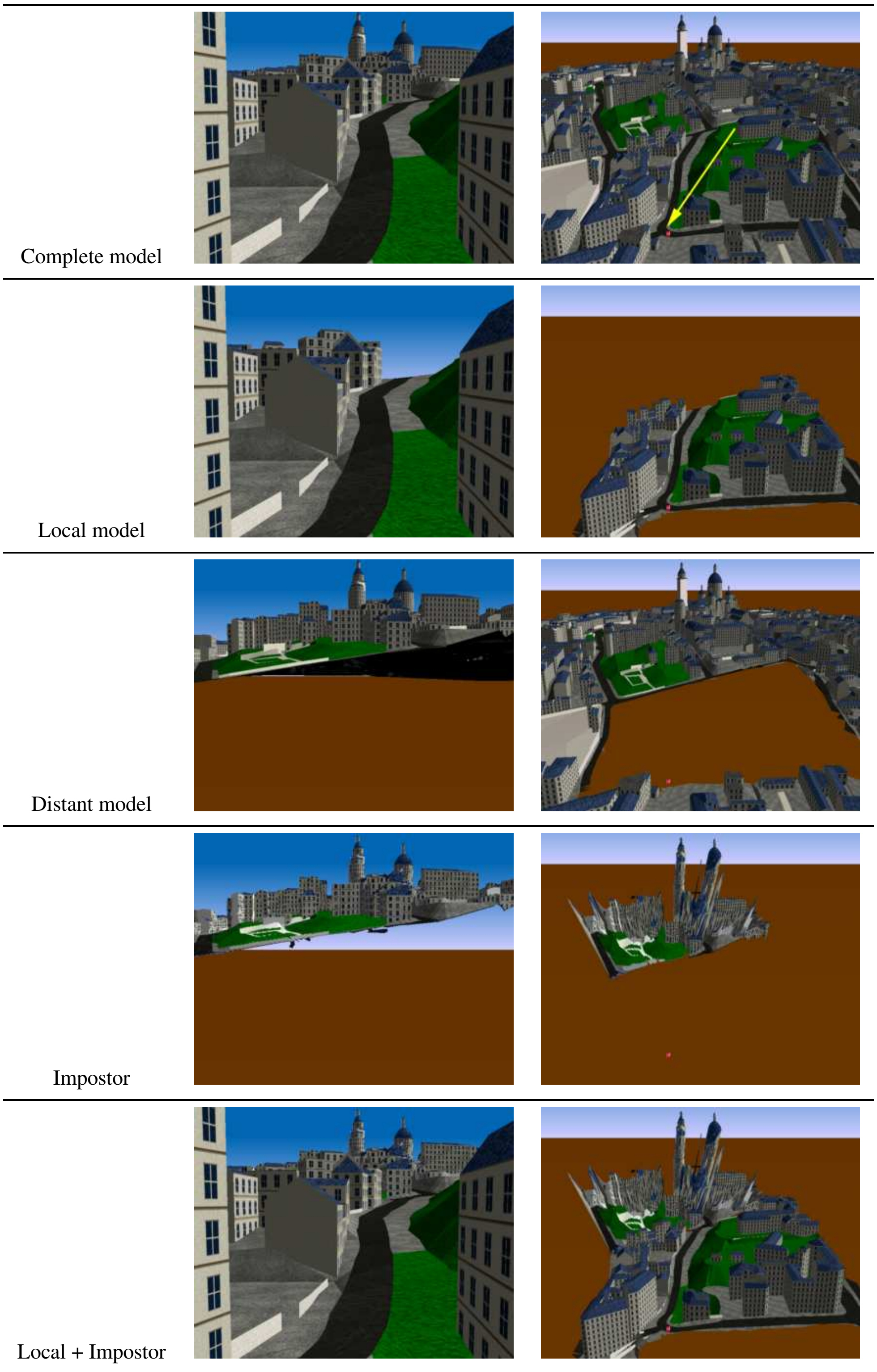


Figure 2: *Principles of model segmentation and use of an impostor. The left column shows views created for a user walking in a street (the viewer location is indicated by the arrow in the top right image). The right-hand column shows a bird's eye view of the scene (See also color section).*

4. Practical algorithms

The principal ideas outlined above have been used to guide the development of several practical algorithms in our urban visualisation system. We present below the algorithm used to segment an urban model based on street connectivity data and a construction method for three-dimensional impostors.

These approaches are not necessarily the most efficient, but provide a first solution to the specific requirements of the visualisation of complex urban environments. We have implemented all the algorithms presented next in our prototype system, with satisfactory results, which will be presented in the following section.

In our current implementation, we have decided to focus on urban navigation for pedestrians or land vehicles. This particularity is important in that it means that there is a lot of occlusion by nearby buildings along the streets, therefore the directions in which distant landscape is visible are well identified. Flying over a city (for games or other simulation applications) would require different segmentations or different sets of impostors. Note that the difficulty would not arise from the fact that distant objects are visible, but rather from the inability to easily predict in which direction this happens.

4.1. Segmentation of the model

A user walking or driving on the ground is typically constrained to a network of streets, therefore this defines a subset of the model where the viewpoint can be. We can safely restrict our algorithms to the creation of impostors that let us recreate views for such possible points.

Visibility in a street is typically blocked sideways by adjacent buildings, but can extend quite far away in the directions of the street endpoints, or above the adjacent buildings/ground ("local" skyline).

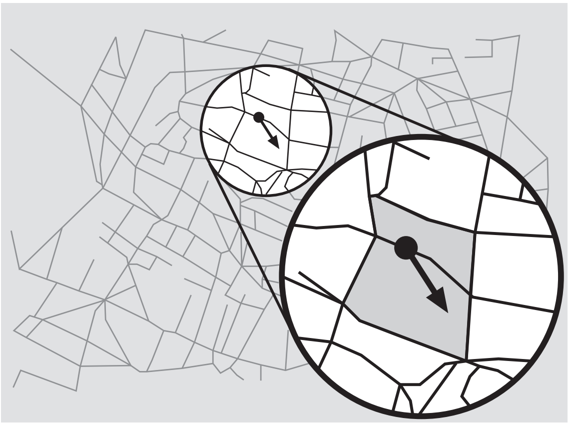


Figure 3: *Blocks adjacent to the current street are selected to compose the local model.*

A graph of the network of streets is first constructed. In our prototype implementation, this graph is created from auxiliary data depicting the geometry of the streets in Paris. While this data allows us to quickly build a topological structure, an equivalent graph can easily be constructed by hand for a given urban dataset. It is indeed probably desirable to build the graph from the geometrical data, since the use of an independent geometry file actually revealed several misalignment problems, with streets running through houses etc.

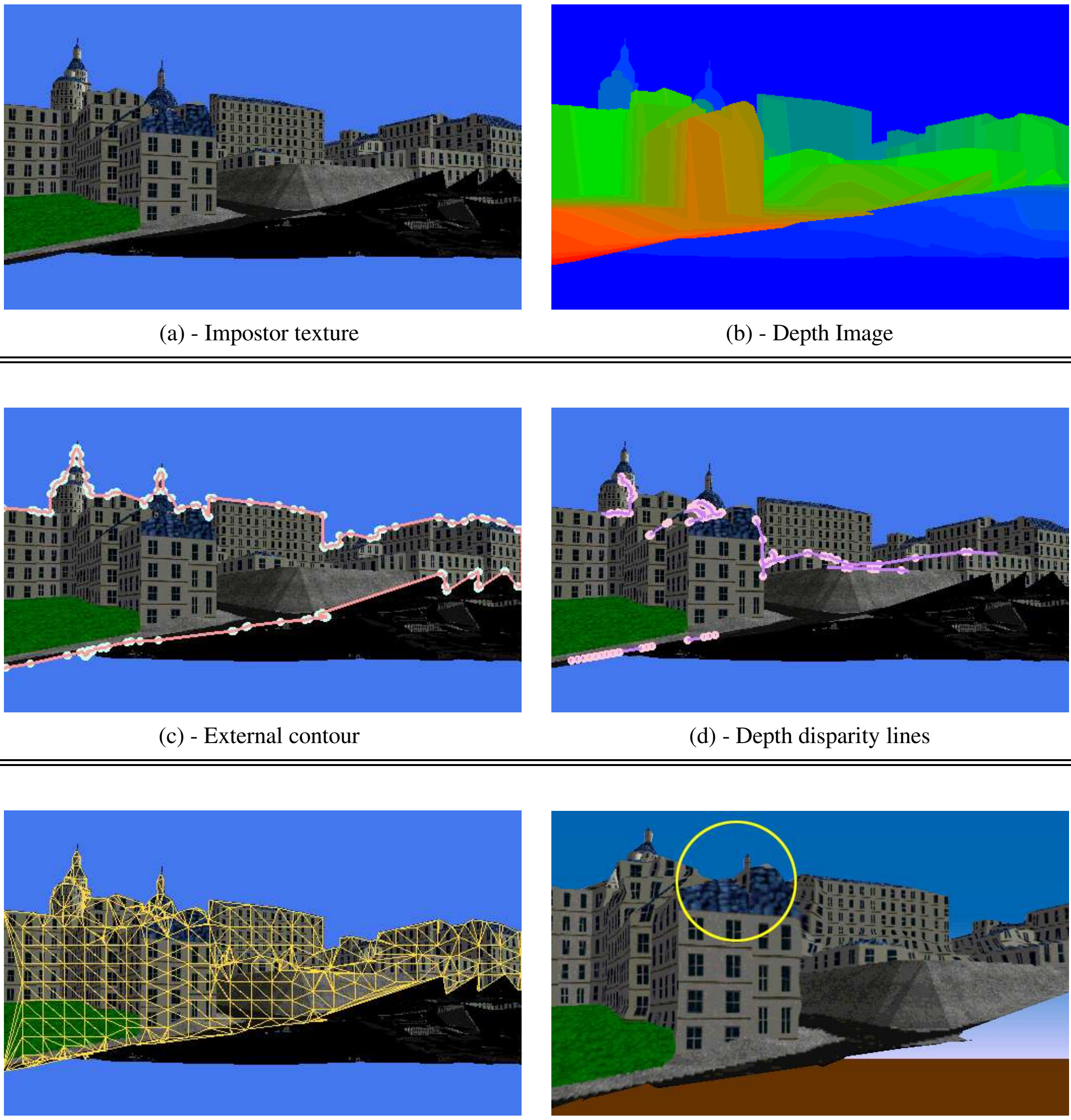
The graph is represented internally using a classical winged-edge data structure, allowing fast navigation in the streets, as well as easy access to "neighboring" geometrical information. The area of interest is therefore treated as a 2D polyhedron, whose faces are city blocks (and point to appropriate buildings and landmarks) and whose edges are portions of streets.

We define a cell of the segmented model as an oriented edge of the street graph. Topological information is then used to extract the local 3D model around a point (a street) as the set of blocks "near" that street. The simplest definition of "near" is that blocks must touch the current street segment (Figure 3). We use this definition in our implementation. Other definitions can be used, in particular to ensure smooth transitions between cells as the viewer moves through the database. In particular, a pointwise notion of connectivity, counting as adjacent all blocks sharing a graph vertex, seems important around the endpoints of a street segment. We are currently investigating better segmentations using this notion. Obviously the chosen selection mechanism should depend on the particular morphology of the city model at hand: short and curved street segments place particular requirements in terms of impostor life span, compared to long and straight streets.

Because visibility of distant objects is in practice mostly limited to the directions of the street ends, we associate two impostors with each street (one with each cell, i.e. a directed edge in the graph). This implicitly assumes a densely built environment, where visibility is blocked by buildings along each side of the street.

4.2. Impostor creation

Recall that the goal of the three-dimensional impostor construction is a representation which maintains the advantages of a texture-based image cache, while at the same time containing richer information related to depth and contouring. This additional data permits the reuse of the impostors for many more frames than in traditional image-caching. In our implementation, impostors are created either off-line as a pre-process, or on demand when the user enters a new area of the model. Since impostors are associated to edges, there are twice as many impostors as streets in the model. The main stages of our construction algorithm are listed below, and illustrated by the images in Figure 4. Please refer to these images while reading this section.



(a) - Impostor texture

(b) - Depth Image

(c) - External contour

(d) - Depth disparity lines

(e) - Impostor triangulation

(f) - Other view

Figure 4: *Illustration of the steps taken to create an impostor. The last image shows the impostor rendered from a different viewpoint and demonstrates the appearance changes due to parallax (See also color section).*

The successive steps of the impostor creation algorithm are:

1. Create an image of the distant scenery (to be used as the impostor texture).
2. Save the corresponding depth image (contents of the *z*-buffer).
3. Extract the external contour of the image.
4. Identify the significant depth disparity contours.
5. Perform a constrained triangulation of the impostor.
6. Store the list of 3D triangles along with the texture image.

Creation of the impostor image

We begin by generating an image of the distant landscape (that is, the entire model of which the "local" model has been removed) from a given point of view (Fig. 4-a). In our case we create the view from the edge origin, looking towards the other end. We also temporarily extract a depth image from the z-buffer (Fig. 4-b).

Identification of important features

A standard image-processing contour extraction based on thresholding is then applied to the image to separate actual relevant data from the background [24]. While the separation from an empty background (such as for the skyline) is trivial, care must be taken that some distant portions of the model can be visible *from below*, because we are only considering the distant model, and therefore the ground description of the local model is missing from the rendered data. We use a special color to identify the bottom face of ground elements, and use this information in our contour extraction algorithm.

The resulting contour captures all the details of typical city sky-scapes, including spires, rooftops etc (Fig. 4-c), to the resolution of the computed image, and is referred to as the *external* contour.

The external contour can be considered to be a two-dimensional polygon in the image-plane used for rendering (although there may of course be multiple contours, the structure of urban landscapes, with the presence of the ground, is such that in the vast majority of cases there is a single contour in the image. Therefore we will always speak of "the" contour, but our algorithms apply to all available contours). We want to subdivide this polygon into simple polygons such as triangles in the plane, which can then be reprojected in 3D. Because our goal is to allow the recreation of parallax in the impostor, we next identify lines of maximal depth discrepancy (Fig. 4-d).

This is accomplished by creating a depth discrepancy image from the depth image. Depth discrepancy is defined here as the maximum difference between the depth at a pixel and that of its neighbors. We then extract "interesting" contours (after a thresholding operation to select pixels with an important discrepancy) and fit a set of line segments to the resulting contours.

In practice, we apply the following simple algorithm iteratively, until no more pixels with a depth discrepancy above a given threshold are available:

1. Find pixel with largest discrepancy.
2. Follow chain of pixels until we are blocked by the external contour or the discrepancy falls below the threshold.
3. Compute a polygonal approximation of the chain.

Triangulation and 3D reprojection

The external contour and the discrepancy lines provide the important information needed to recover parallax effects within the impostor. To avoid precision issues as well as extreme triangle aspect ratios, we augment the impostor with a set of points selected on a regular grid within the external contour to impose a minimal sampling density. Finally we perform a constrained Delaunay triangulation of all these points, with the constraint that all external contour segments and all discrepancy line segments be included in the triangulation [25] (Fig. 4-e).

All vertices of the triangulation are then reprojected in 3D using the information of the depth image, and the resulting set of 3D triangles, together with the corresponding texture image, constitutes the impostor. Texture coordinates for each triangle vertex are simply the image coordinates of the corresponding pixel location.

A view of the impostor from a viewpoint other than the point of creation is shown in Fig. 4-e. Note the parallax effects, when the most distant buildings become obscured by nearby ones. This effect is obtained exclusively from the perspective rendering of the 3D triangles composing the impostor.

5. Results

We present in Figure 5 comparative images for different viewer positions in the same area. Notice first the overall quality of the images in the central column, produced by the combination of the local 3D model and the impostor (compare to the right-hand column rendered using the complete 3D model). The images in this central column are produced at a sustained rate of between 11 and 19 frames per second, while a complete 3D rendering only achieves between 1 and 2.5 frames per second (using frustrum culling and the *Performer* high-performance graphics library).

Second, the three-dimensional impostors succeed in providing an accurate view of distant landscape, taking into account some parallax changes. Note in particular how the church towers in the background correctly recede behind the closer buildings.

Finally a small amount of cracking is visible when the user comes very close to the boundary between the local model and the impostor. This suggests that more work is needed to keep all errors under complete control.

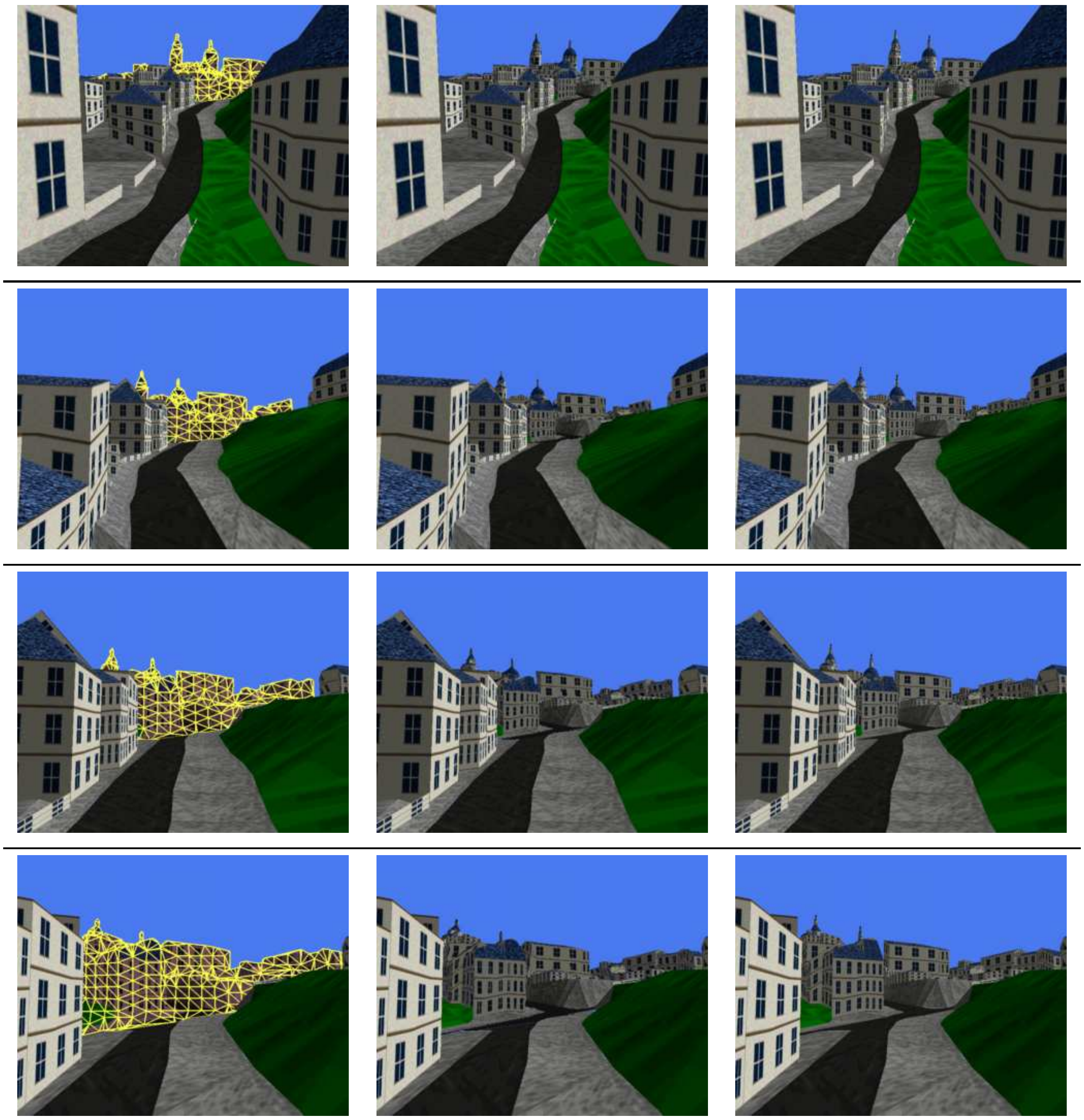


Figure 5: *Images extracted from an interactive tour of the Montmartre area. An sustained frame rate of about 15 fps was obtained. In the left column the impostor was colored to make it visible. The center column shows the view displayed during interactive navigation. The right-hand column is a reference view rendered using the entire model (1 to 2 fps). Note the self-masking effects within the impostor, due to the evolution of parallax: in particular the church in the background disappears behind the closest buildings as the viewer moves forward (See also color section).*

Note that since the life time of impostors corresponds to the time the user spends in a given street, impostors remain active in areas whose size exceeds several tens of meters.

5.1. Performance

The results of our implementation are quite satisfactory. The algorithm achieves over 11 frames per second for the Montmartre model (recall that this is a model of around 140,000 polygons). All timings were computed on an SGI Indigo^2 200 MHz R4400 computer. We observe that the speedup value is only an indication, very much dependent on the complexity of the data. A more complex model containing more detail information on the facades in the form of geometry, for instance, would not affect the complexity of the impostors, and may produce even greater acceleration factors.

In terms of space, a typical impostor, starting with a texture image of 512x512, results in about 1,000 polygons. This is much less than the typical *visible* geometric complexity of the distant model. The average size of the local model is approximately 4,000 polygons, therefore the total amount of 3D geometry being drawn at any given time is reduced to about 5k polygons.

The creation of an impostor takes about 12 seconds on the above computer, of which the initial offscreen rendering takes approximately 10. These figures could therefore be greatly reduced using hardware rendering (for instance with the pixel buffer mechanism available in recent SGI software). All of these operations can be seen as preprocessing, and can be stored with the model.

The above model has 1168 edges. Precomputing and storing all the impostor images clearly represents a significant expense. While the calculation time can be largely amortized by later usage, memory requirements are potentially large. However we observe that very little of this memory is needed at any given time, since very few impostors are active. A concurrent process can therefore be used to pre-fetch impostors in a neighborhood of the viewer.

6. Conclusions and Future Directions

This paper introduces a new framework for the efficient segmentation and visualisation of urban landscapes. The central idea is the efficient segmentation of the urban model based on inherent city information (such as streets and blocks), resulting in two distinct sub-sets at each frame: the local neighbourhood and distant scenery. We use the full three-dimensional model to render the local neighbourhood, thus providing detailed geometric information to the viewer. For distant scenery, we introduce "impostors" of the complex data set, augmented with appropriate three-dimensional information. These impostors are initially based on an image-view of the distant scenery; the contour(s) of the building sky- and ground-line are extracted, as well as lines of depth discrepancy. The result is polygonalised and reprojected into three-dimensional space using the depth-buffer information. As a consequence, the impostor can be used for a much larger number of frames than previous image-caching techniques. In addition, parallax distortions can be captured to a certain extent.

We have implemented our new approach, achieving near real-time visualisation of an real complex urban model. We have shown how important sky-line features (rooftops, spires etc.) are preserved using our 3D impostors, and that parallax effects such as mutual occlusion of two buildings contained in the impostor, are sucessfully rendered.

Much research remains to be done. One important issue is the improvement of transitions between cells of segmented models. In the current approach, the impostors are completely correct at the view points for which they were generated. One approach would consist in morphing between impostors. We are currently investigating solutions to this problem, based on storing different views at edge endpoints, and developing appropriate combination algorithms.

Segmentation models should take advantage of the characteristics of urban structure. Different cities, built in different parts of the world in different cultures, can have distinct properties. For example, the sky-scraper sky-lines of North-American cities differ from the large boulevards and monuments of Paris. Such characteristics have an important influence on occlusion and the regularity of visibility changes.

An important issue is the error measure used to determine the validity range for impostors. The addition of three-dimensional information adds different types of distortion, but due to its richer nature can provide improved discrepancy measure compared to simple pixel differences of image caches.

Although the problem of cracking is greatly diminished using 3D impostors, it has not been completely eliminated. Different possibilities exist, such as sharing boundaries or vertices between the model and the impostor. Better heuristics can be used to extract relevant data in impostors, e.g., using decimation techniques.

Finally, we hope that the insight gained by studying urban visualization, and designing algorithms tailored to this application, will prove useful in developping more general solutions to the issue of impostor usage for the visualization of very large databases.

7. Acknowledgements

The authors wish to thank the French "Institut Géographique National" for the TRAPU urban data set, as well as Mathieu Desbrun and Frédo Durand for their help in preparing the images. Computer vision contouring code was courteously provided by Luc Robert and by Radu Horaud. Dani Lischinski provided crucial help with the Delaunay triangulation code.

References

1. W. Jepson, R. Liggett, and S. Friedman, "An environment for real-time urban simulation", in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 165–166, ACM SIGGRAPH, (Apr. 1995).

2. M. Eck, T. DeRose, T. Duchamp, H. Hoppe, M. Lounsbery, and W. Stuetzle, "Multiresolution analysis of arbitrary meshes", in *SIGGRAPH 95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 173–182, ACM SIGGRAPH, Addison Wesley, (Aug. 1995). held in Los Angeles, California, 06-11 August 1995.

3. J. Rossignac and P. Borrel, "Multi-resolution 3D approximation for rendering complex scenes", in *Second Conference on Geometric Modelling in Computer Graphics*, pp. 453–465, (June 1993). Genova, Italy.

4. P. W. C. Maciel and P. Shirley, "Visual navigation of large environments using textured clusters", in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 95–102, ACM SIGGRAPH, (Apr. 1995).

5. J. Shade, D. Lischinski, D. Salesin, T. DeRose, and J. Snyder, "Hierarchical image caching for accelerated walkthroughs of complex environments", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 75–82, ACM SIGGRAPH, Addison Wesley, (Aug. 1996).

6. G. Schaufler and W. Stürzlinger, "A three dimensional image cache for virtual reality", in *Computer Graphics Forum, 15(3). Proceedings Eurographics '96* (J. Rossignac and F. Sillion, eds.), pp. 227–236, Blackwell, (Sept. 1996).

7. J. H. Clark, "Hierarchical geometric models for visible surface algorithms", *Communications of the ACM*, **19**(10), pp. 547–554 (1976).

8. C. B. Jones, "A new approach to the 'hidden line' problem", *Computer Journal*, **14**(3), pp. 232–237 (1971).

9. J. M. Airey, J. H. Rohlf, and F. P. Brooks, Jr., "Towards image realism with interactive update rates in complex virtual building environments", in *Computer Graphics (1990 Symposium on Interactive 3D Graphics)* (R. Riesenfeld and C. Sequin, eds.), vol. 24, pp. 41–50, (Mar. 1990).

10. S. J. Teller and C. H. Séquin, "Visibility preprocessing for interactive walkthroughs", in *Computer Graphics (SIGGRAPH '91 Proceedings)* (T. W. Sederberg, ed.), vol. 25, pp. 61–69, (July 1991).

11. D. Luebke and C. Georges, "Portals and mirrors: Simple, fast evaluation of potentially visible sets", in *1995 Symposium on Interactive 3D Graphics* (P. Hanrahan and J. Winget, eds.), pp. 105–106, ACM SIGGRAPH, (Apr. 1995).

12. N. Greene and M. Kass, "Hierarchical Z-buffer visibility", in *Computer Graphics Proceedings, Annual Conference Series, 1993*, pp. 231–240, (1993).

13. T. A. Funkhouser and C. H. Séquin, "Adaptive display algorithm for interactive frame rates during visualization of complex virtual environments", in *Computer Graphics (SIGGRAPH '93 Proceedings)* (J. T. Kajiya, ed.), vol. 27, pp. 247–254, (Aug. 1993).

14. S. Coorg and S. Teller, "Temporally coherent conservative visibility", in *Proc. 12^{th} Annual ACM Symposium on Computational Geometry*, (1996).

15. J. F. Blinn and M. E. Newell, "Texture and reflection in computer generated images", *Communications of the ACM*, **19**, pp. 542–546 (1976).

16. S. E. Chen and L. Williams, "View interpolation for image synthesis", in *Computer Graphics (SIGGRAPH '93 Proceedings)* (J. T. Kajiya, ed.), vol. 27, pp. 279–288, (Aug. 1993).

17. S. E. Chen, "Quicktime VR - an image-based approach to virtual environment navigation", in *SIGGRAPH 95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 29–38, ACM SIGGRAPH, Addison Wesley, (Aug. 1995). held in Los Angeles, California, 06-11 August 1995.

18. L. McMillan and G. Bishop, "Plenoptic modeling: An image-based rendering system", in *SIGGRAPH 95 Conference Proceedings* (R. Cook, ed.), Annual Conference Series, pp. 39–46, ACM SIGGRAPH, Addison Wesley, (Aug. 1995). held in Los Angeles, California, 06-11 August 1995.

19. M. Regan and R. Pose, "Priority rendering with a virtual reality address recalculation pipeline", in *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)* (A. Glassner, ed.), Computer Graphics Proceedings, Annual Conference Series, pp. 155–162, ACM SIGGRAPH, ACM Press, (July 1994).

20. J. Torborg and J. Kajiya, "Talisman: Commodity Realtime 3D graphics for the PC", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 353–364, ACM SIGGRAPH, Addison Wesley, (Aug. 1996). held in New Orleans, Louisiana, 04-09 August 1996.

21. M. Levoy and P. Hanrahan, "Light field rendering", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 31–42, ACM SIGGRAPH, Addison Wesley, (Aug. 1996).

22. S. J. Gortler, R. Grzeszczuk, R. Szeliski, and M. F. Cohen, "The lumigraph", in *SIGGRAPH 96 Conference*

Proceedings (H. Rushmeier, ed.), Annual Conference Series, pp. 43–54, ACM SIGGRAPH, Addison Wesley, (Aug. 1996).

23. A. Certain, J. Popović, T. DeRose, T. Duchamp, D. Salesin, and W. Stuetzle, "Interactive multiresolution surface viewing", in *SIGGRAPH 96 Conference Proceedings* (H. Rushmeier, ed.), Annual Conference Series, pp. 91–98, ACM SIGGRAPH, Addison Wesley, (Aug. 1996). held in New Orleans, Louisiana, 04-09 August 1996.

24. B. Jähne, *Digital Image Processing*. Springer Verlag, (1995). Third edition.

25. D. Lischinski, "Incremental delaunay triangulation", in *Graphics Gems IV* (P. S. Heckbert, ed.), pp. 47–59, San Diego, CA: Academic Press Professional, (1994).