Visual Comput (2006) 22: 506–516
DOI 10.1007/s00371-006-0025-3

ORIGINAL ARTICLE

Dmitry Sokolov
Dimitri Plemenos
Karim Tamine

Methods and data structures for virtual world exploration

Published online: 5 July 2006
© Springer-Verlag 2006

D. Sokolov (✉) · D. Plemenos · K. Tamine
XLIM Laboratory UMR CNRS 6172,
University of Limoges
83, rue d'Isle, 87000 Limoges, France
s@skisa.org, {dimitrios.plemenos,
karim.tamine}@unilim.fr

Abstract This paper is dedicated to virtual world exploration techniques that help humans to understand a 3D scene. The paper presents a technique to calculate the quality of a viewpoint for a scene, and describes how this information can be used. A two-step method for an automatic real-time scene exploration is introduced. In the first step, a minimal set of "good" points of view is determined; in the second step, these viewpoints are used to compute a camera path around the scene. The proposed method enables one to get a good comprehension of a single virtual artifact or of the scene structure.

Keywords Scene understanding · Automatic virtual camera · Good point of view · Visibility

1 Introduction

Virtual world exploration techniques have nowadays become more and more important. When, more than ten years ago, we proposed the very first methods to improve the knowledge of a virtual world [6, 17], many people thought that it was not an important problem. Only during these last years have people begun to understand its importance and the necessity to have fast and accurate techniques for good exploration and clear understanding of various virtual worlds. However, there are very few papers that face this problem from the computer graphics point of view, although several papers have been published on the robotics artificial vision problem.

The purpose of a virtual world exploration in computer graphics is completely different from the objectives of techniques used in robotics. In computer graphics, the purpose of a program guiding a virtual camera is to allow a human being, the user, to understand a new world by using an automatically computed path, depending on the nature of the world. The main interaction is between the camera and the user, a virtual and a human agent, and not between two virtual agents or a virtual agent and his environment.

There are two classes of methods for a virtual world exploration. The first one is the global exploration class, where the camera remains outside the world to be explored. The second one is the class of local exploration. In this class, the camera moves inside a scene and becomes a part of the scene. Local exploration may be useful, and even necessary in some cases, but only global exploration can give the user a general knowledge of a scene.

On the other hand, two different virtual world exploration modes can be considered:

1. *Real-time on-line exploration*, where the virtual world is visited for the first time and the path of the camera is determined in real-time, in an incremental manner. In this mode it is important to apply fast exploration techniques in order to allow the user to understand the explored world in real-time.
2. *Off-line exploration*, where the camera path is computed off-line and an exploration can be undertaken later. In a preliminary step, the virtual world is found and analyzed by the program guiding the camera movement, in order to determine interesting points of

view and paths linking these points. The camera will explore the virtual world later, when necessary, following the determined path. In this mode, it is less important to use fast techniques.

In the present paper, we are mainly concerned with global virtual world exploration, where the camera remains outside the scene. Several papers have already been published on *on-line* global virtual world exploration. The objective of this paper is the *off-line* visual exploration of a fixed unchanging virtual world.

The paper is organized in the following manner. In Sect. 2, a study of existing techniques of virtual world exploration is presented. In Sect. 3, a new data structure, the so-called visibility graph, is introduced. This structure allows fast determination of interesting areas for the camera. A method to compute camera paths, based on the visibility graph, is described in Sect. 4. In Sect. 5 examples of camera paths, obtained with the proposed method, are given, and a first evaluation of the visibility graph-based method is presented. Finally, in Sect. 6, a conclusion and a brief description of future work are presented.

2 Background

2.1 Static explorations

Initial works on visual scene understanding were published at the end of the 80s. Kamada et al. [13] proposed a fast method to compute a viewpoint that minimizes the degenerated edges of a scene.

Colin [6] proposed a method initially developed for scenes modelled by octrees. The aim of the method was to compute a good point of view for an octree. The method uses the principle of "direct approximate computation" to compute a good view direction. This principle can be described as follows:

1. Choose the three best view directions d_1, d_2 and d_3, among the six corresponding to the three coordinate axes passing through the center of the scene.
2. Compute a good direction in the pyramid defined by the three chosen directions, taking into account the importance of each direction (see Fig. 1).

A view direction is considered to be better than another one if it allows one to see more details.

Plemenos [17, 19] proposed an iterative method of automatic viewpoint calculation. The scene is placed at the center of a sphere, whose surface represents all the possible points of view. The sphere is divided into eight spherical triangles (see Fig. 2) and the best one is chosen according to the view qualities of triangle vertices. Then, the selected spherical triangle is recursively subdivided. The best vertex is taken as the best point of view at the end of the process (Fig. 3).

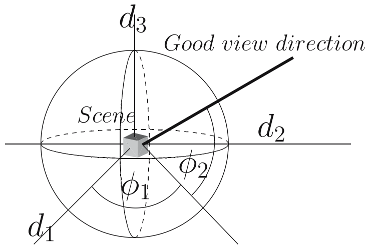

Fig. 1. Direct approximate computation of a good view direction

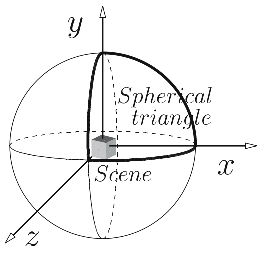

Fig. 2. The sphere of viewpoints divided into eight spherical triangles

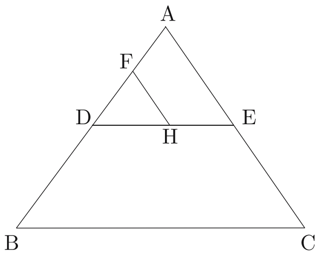

Fig. 3. Recursive subdivision of the "best" spherical triangle

The proposed heuristic considers a viewpoint to be good if it gives a high amount of details in addition to the minimization of the deviation. Plemenos shows that if only the minimization is considered, then resulting views may hide important information of a scene. Therefore, another parameter to the maximizing function is added that counts the observed details. The added parameter is the number of faces that are visible from a viewpoint. According to [19], the viewpoint quality can be computed by the following formula:

$$C(p) = \frac{\sum_{i=1}^{n} \left[\frac{P_i(p)}{P_i(p)+1}\right]}{n} + \frac{\sum_{i=1}^{n} P_i(p)}{r}, \tag{1}$$

where

1. $C(p)$ is the viewpoint quality for the given viewpoint p,
2. $P_i(p)$ is the number of pixels corresponding to the polygon number i in the image obtained from the viewpoint p,
3. r is the total number of pixels in the image (resolution of the image),
4. n is the total number of polygons in the scene.
5. $[a]$ is the ceiling function of a, i.e., the smallest integer number $a_c \in \mathbb{N} : a_c \geq a$.

Sbert et al. [9, 10, 22, 24, 25] introduced an information theory-based approach to estimate the quality of a viewpoint. This quality is computed as a viewpoint entropy function:

$$I(S, p) = \sum_{i=0}^{N_f} \frac{A_i}{A_t} \cdot \ln \frac{A_t}{A_i}, \tag{2}$$

where N_f is the number of faces in the scene, A_i is the projected area of the face number i, A_0 represents the projected area of the background in open scenes and A_t is the total area of the projection.

Recently, Chang Ha Lee et al. [14] introduced the idea of mesh saliency as a measure of regional importance for graphics meshes. They define mesh saliency in a scale-dependent manner using a center-surround operator on Gaussian-weighted mean curvatures. The human-perception-inspired importance measure computed by the mesh saliency operator gives more pleasing results in comparison with purely geometric measures of shape, such as curvature.

2.2 Dynamic explorations

A single good point of view is generally not enough for complex scenes, and even a list of good viewpoints does not allow the user to understand a scene, as frequent changes of viewpoint may confuse him (her). To avoid this problem, virtual world exploration methods were proposed.

Plemenos et al. and Dorme [1–3, 7] have proposed a method, where a virtual camera moves in real-time on the surface of a sphere surrounding the virtual world. The exploration is on-line, the scene is being examined in an incremental manner during the observation. All the polygons of the virtual world are taken into account at each step of the exploration. The method is based on the heuristics $w_c = \frac{v_c}{2} \cdot (1 + \frac{d_c}{p_c})$, where

1. w_c is the weight of the current camera position,
2. v_c is the viewpoint complexity of the scene from the camera's current position,
3. p_c is the path traced by the camera from the starting point to the current position,
4. d_c is the distance from the starting point to the current position.

In order to avoid fast camera returns to the starting position, the importance of a viewpoint is made inversely proportional to the camera path from the starting to the current position. In addition, for a smooth movement of the camera, only three new viewpoints are considered while computing the next position (see Fig. 4). A result of the application of this technique is shown in Fig. 5.

Vázquez et al. [24, 27] proposed an exploration method that is similar to the previous one. The difference is that the next viewpoint is chosen according to the entropy (Eq. 2) and the number of faces not yet visited. To evaluate the qualities of the next three possible positions, the entropy of each viewpoint is multiplied by the number of new faces that appear with respect to the set of already visited faces. In the case where none of the three possible viewpoints show a new face, the one lying furthest from the initial position is chosen.

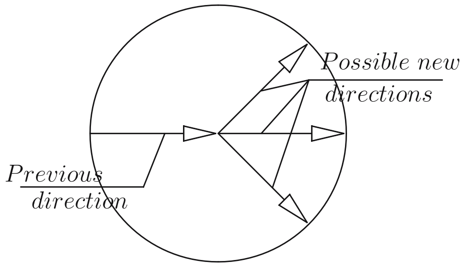

Fig. 4. Only three directions are considered to ensure a smooth movement of the camera

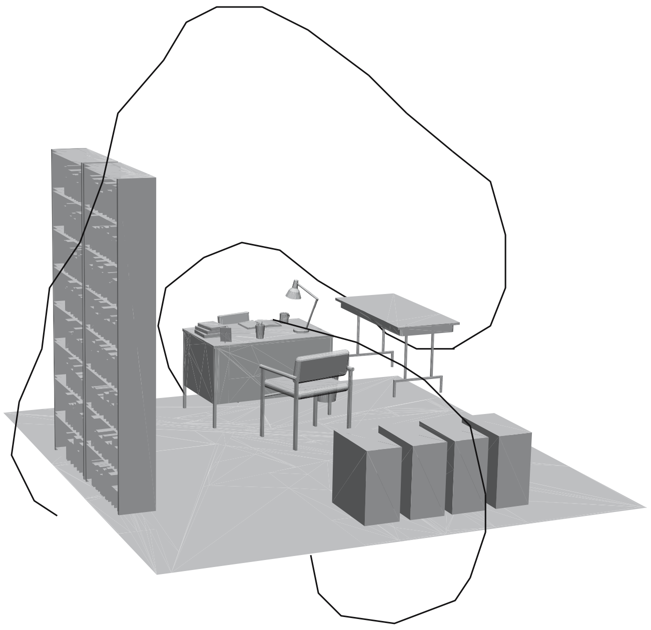

Fig. 5. The virtual office exploration trajectory taken from [21]

In many cases, on-line exploration is not necessary because there is enough time to precompute interesting points of view for a virtual world and even interesting trajectories. Thus, Jaubert [12] proposed an off-line exploration method based on the precomputation of a minimal set of good viewpoints. The computed viewpoints are sorted in importance order and stored together with the virtual world. The stored order is used for each exploration of the virtual world.

In [15, 20, 26], image-based techniques are used to control the camera motions in a changing virtual world.

The problem faced in these papers is the adaptation of the camera behavior to changes of the world.

For more details, a state-of-the-art paper on virtual world global exploration techniques is available [18], whereas viewpoint quality criteria and estimation techniques are presented in [21].

3 Analytic visibility graph

Let us suppose that there is an unknown scene, and the user would like to get a general comprehension of its structure. Since the user would like to explore the exterior of the scene, it is reasonable to restrict the space of possible viewpoints to a surrounding sphere. Moreover, viewpoint quality is quite a smooth function, so the sphere can easily be discretized. Thus, the scene is placed at the center of the sphere, whose discrete surface represents all possible points of view.

It would be very convenient for many quality estimation routines if the undirected bipartite graph $G = (S \bigcup F, E)$ could be obtained, where the first part S corresponds to the set of viewpoints of the discrete sphere and F corresponds to the set of faces in the scene. The set of arcs E represents visibility between objects from S and F, i.e., G is the analytic visibility graph.

Unfortunately, computing such a graph is quite an expensive task, because the proposed methods of viewpoint quality estimation operate with quite expensive point-to-region visibility and approximate calculation of visible parts using a Z-Buffer.

The scene is rendered from a viewpoint, coloring each face with a unique color ID and using flat shading. In the resulting rendered scene, each pixel represents the color code of the face, which is visible in this pixel. OpenGL allows one to obtain a histogram that gives information on the number of displayed colors and the ratio of the image occupied by each color. See [1] and [16] for more detailed descriptions.

A quality estimation routine for a single viewpoint, applying these methods, runs in $O(n_f \cdot Z)$ time, where $n_f = |F|$ is the number of faces in the scene and Z is the resolution of the Z-buffer. For $n_s = |S|$ viewpoints, the computation time is $O(n_f \cdot n_s \cdot Z)$. Note that Z should be significantly greater than n_f in order to have at least few pixels to display each face. The complexity of the algorithm forces the user to use adaptive search algorithms, which may lead to inexact results, even for a single good viewpoint selection.

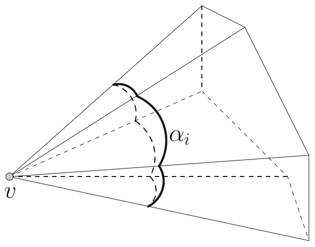

Fig. 6. The curvature in a vertex is equal to the sum of angles adjacent to the vertex minus 2π

Recently, we proposed an improved method of viewpoint quality estimation [23]. The method considers the total curvature of a visible surface as an amount of information appropriate for a viewpoint:

$$I(p) = \sum_{v \in V(p)} \left| 2\pi - \sum_{\alpha_i \in \alpha(v)} \alpha_i \right| \cdot \sum_{f \in F(p)} P(f), \tag{3}$$

where

1. $F(p)$ is the set of faces visible from the viewpoint p,
2. $P(f)$ is the projected area of the face f,
3. $V(p)$ is the set of visible vertices of the scene from p,
4. $\alpha(v)$ is the set of angles adjacent to the vertex v (see Fig. 6).

The proposed heuristic is invariant to any subdivision of a scene maintaining the topology. Indeed, if a flat face is subdivided into several scenes, then all the edges and vertices inside the face are to be discarded due to zero angles. An important property of such a viewpoint quality definition is the possibility to extend it, using the total integral curvature $\int_\Omega |K| \, \mathrm{d}A$, into the class of continuous surfaces, such as NURBS, etc., which nowadays are becoming more and more usable.

A very important advantage of the method is that it replaces the point-to-region by point-to-point visibility computations. Since the visibility of faces no longer plays a main role, the visibility graph is transformed to $G = (S \bigcup V, E)$, where V corresponds to the set of vertices of the scene. The set of arcs E represents visibility between objects from S and V. The graph is computed in $O\left(n_f \cdot \sqrt{n_s} \cdot n_v\right)$ operations in the worst case, where $n_v = |V|$ is the number of vertices in the scene.

Let us suppose that the surrounding sphere is pixelized; each element represents a viewpoint. To evaluate viewpoint qualities, it is necessary to find all the vertices of the scene that are visible from each element of the sphere. In order to do this rapidly, a reverse problem is considered: all the visible viewpoints have to be found for each vertex of the scene. This allows one to use the structure of the rasterized sphere for fast elimination of hidden areas. Without loss of generality, a rasterized plane can be considered instead of the sphere and a triangulated scene (a polygonal mesh could be triangulated in linear time using the algorithm presented in [5]).

The algorithm iterates through each vertex of the scene and finds all pixels of the plane that are visible from the given vertex (see Fig. 7).

The main step of the algorithm is the determination of a set of viewpoints that are visible from a given vertex. The naive way is to project all the triangles on the plane,

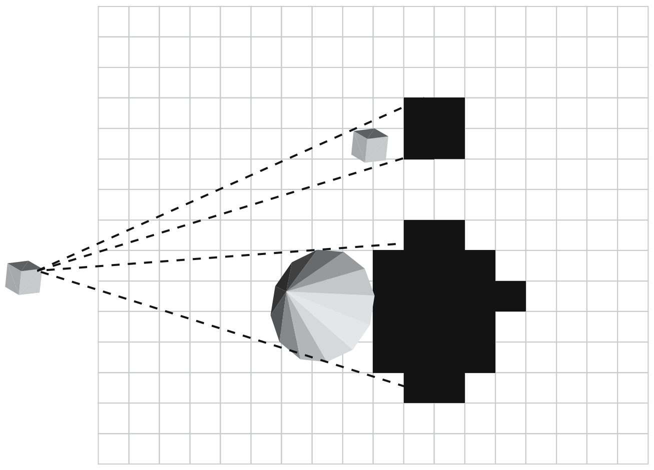

Fig. 7. Scene consisting of two cubes and a cone; the rasterized plane represents a set of viewpoints. The given vertex of the scene is not visible from viewpoints colored in black

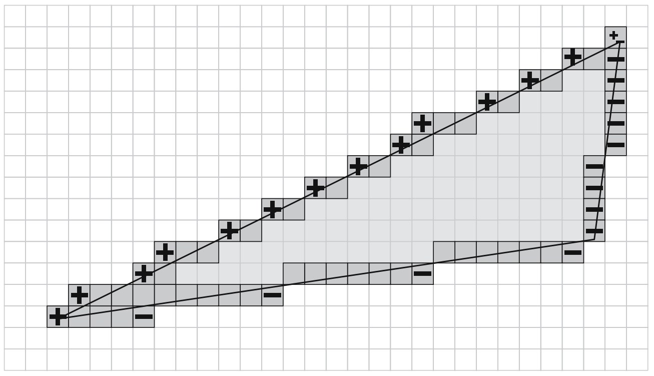

Fig. 8. The triangle is drawn on the rasterized plane, the boundary is shown by the dark gray color, the inner part by the light gray color. The symbols “+” indicate the left part of the boundary, “−” shows the right part

to fill up the projections and then to eliminate colored areas. This way is the simplest, but it is expensive. In such a case, the main step of the algorithm runs in $O(n_f \cdot n_s)$ time in the worst case, because it could color pixels several times.

We propose to keep in memory a matrix of numbers, where each element of the matrix corresponds to a pixel of the plane. The matrix is initially filled up with zeroes. At the main step of the algorithm a projection boundary is to be found for each triangle of the scene. The boundary could be obtained using the Bresenham’s algorithm [4] of digital line drawing. Then the boundary could be divided into two parts: a left part and a right part (see Fig. 8). The matrix elements corresponding to the elements of the left part of the boundary are to be increased, the right ones are to be decreased. Fig. 9 gives a detailed illustration.

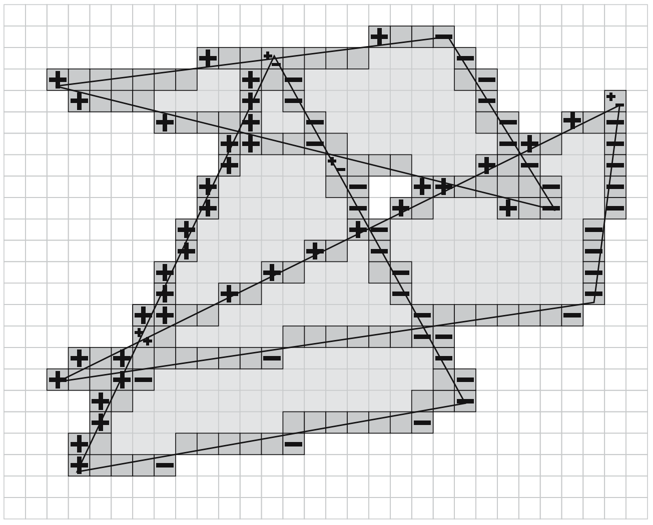

Fig. 9. Three faces are drawn on the plane. The symbol “+” means an increase of the corresponding element of the matrix, the symbol “−” means a decrease

Algorithm 1. The fast visibility computation method

```
Input: The set of faces $F$, the rasterized sphere $S$
Output: The analytic visibility graph $G = (S \bigcup V, E)$
$q_s \leftarrow 0 \; \forall s \in S$
for each vertex $v$ of the scene do
    for each triangle $f \in F$ do
        Find the projection $P_f$ of the triangle $f$ onto the plane
        Increase matrix elements corresponding to the left part of the projection
        Decrease matrix elements corresponding to the right part of the projection
    end for
    Determine the set $A$ of the inner parts according to the calculated matrix (perform the brackets sequence task for each row of the matrix)
    $E \leftarrow E \bigcup (s, v) \; \forall s \in S \setminus A$
end for
```

When all the projections are drawn, the inner parts of the projections are to be eliminated for each row of the matrix. This task is similar to the brackets sequence task, where each row represents a string, each increase of a matrix element means insertion of an opening bracket into the string and each decrease means insertion of a closed bracket. Algorithm 1 shows the scheme.

If the plane consists of n_s pixels, then the maximal boundary drawing time is $O(\sqrt{n_s})$ for a triangle. Having n_f triangles and n_v vertices in a scene, the total running time of the algorithm is $O(n_f \cdot \sqrt{n_s} \cdot n_v)$ operations. Thus, the running time is reduced from $O(n_f \cdot Z \cdot n_s)$ to $O(n_f \cdot n_v \cdot \sqrt{n_s})$ operations, $Z \gg n_f$.

4 Scene visualizations

There are many ways to visualize a scene in order to understand it. They can be separated into two classes of methods: static and dynamic. Static methods give the user a set of photos of a scene and dynamic ones show a "movie".

4.1 Set of images

Let us suppose that we want to find a set of images, by which all the vertices of the scene will be seen. A more strict formulation is as follows: an undirected bipartite graph $G = (S \bigcup V, E)$ is given, the task is to find a set $M \subseteq S : V = \{v | (u, v) \in E, u \in M\}$.

Unfortunately, the minimal set cannot be found in real-time, the minimization of $|M|$ being NP-complete (refer to [11] for a more detailed explanation). Moreover, Feige in [8] proved that for every $\epsilon > 0$ there is no polynomial-time algorithm able to approximate the task within $(1-\epsilon)\ln|V|$ unless $NP \subseteq DTIME\left[n^{O(\ln\ln|V|)}\right]$, which is very strong result. However, even such a simple heuristic as a greedy algorithm (Algorithm 2) has a good bound.

Algorithm 2. The greedy algorithm approximating the set cover problem

```
$G = (S \bigcup V, E)$; $M = \emptyset$
**while** $E \neq \emptyset$ **do**
    Select $s \in S$ with the maximum number of adjacent edges
    $M = M \bigcup \{s\}$
    Remove from $G$ $s$ with all adjacent vertices and edges
**end while**
```

Lemma 1. *The greedy algorithm finds a solution with at most $c_{opt} \cdot \left(\ln \frac{|V|}{c_{opt}} + 1\right)$ vertices, where c_{opt} is the number of vertices in optimal solution. So, it is an $O(\ln|V|)$ approximation – or better if c_{opt} is large.*

Proof. Let us denote the initial number of vertices $|V|$ as n. Since the optimal solution uses c_{opt} vertices, there must be some set that covers at least a $\frac{1}{c_{\text{opt}}}$ fraction of the vertices. Therefore, after the first iteration of the algorithm, there are at most $n \cdot \left(1 - \frac{1}{c_{\text{opt}}}\right)$ vertices left. The optimal solution for the task at the second step can not be greater than c_{opt}, since the initial optimal solution satisfies the new task. After the second step, there are at most $n \cdot \left(1 - \frac{1}{c_{\text{opt}}}\right)^2$ vertices left, etc. After c_{opt} rounds, there are at most $n \cdot \left(1 - \frac{1}{c_{\text{opt}}}\right)^{c_{\text{opt}}} < n \cdot \frac{1}{e}$ vertex nodes left. After $c_{\text{opt}} \ln \frac{n}{c_{\text{opt}}}$ rounds there are at most c_{opt} points left. Thus, the number of iterations the algorithm needs is $It(n) = c_{\text{opt}} + It\left(\frac{n}{e}\right) = O(c_{\text{opt}} \cdot \ln n)$.

4.2 Dynamic understanding – making a movie

Image sets may be sufficient to represent quite simple scenes. However, an image set does not tell us how the camera can pass from one viewpoint (image) to another. Sometimes this is not sufficient to understand a scene well, as the user may lose orientation in the space. In order to improve his (her) knowledge of a scene, a good solution is to give the user the ability to view a film made by the virtual camera.

The goal of this section is to develop a technique for real-time global exploration. Moreover, we would like to develop a method to create films in real-time with the camera remaining outside the virtual world.

First of all, aesthetic criteria of the film quality must be defined.

1. The movie should not be very long, but it must show as much information as possible.
2. The operator (or an algorithm, if the movie is to be created automatically) should avoid fast returns to already visited points.
3. The camera path must be as smooth as possible, in order to avoid transients in direction changes leading to a loss of orientation in the user perception of the explored space.
4. The operator should try to guide the camera via viewpoints as well as possible.

Here we propose an incremental construction method. The main idea is to determine where unexplored areas of a scene lie, then to create "magnets" in these areas. The camera is to be considered as a "magnet" of opposite polarity. Magnetic forces of large unexplored areas will attract the camera. In order to simplify the computations we use gravitational forces instead of magnetic ones.

The method is incremental, thus having a trajectory line from the starting point to the current position of a camera, the camera is to be pushed towards unexplored areas. The aesthetic criteria can be satisfied with the following schema of exploration: at each step a mass is assigned to each point of the discrete sphere and to the current position of the camera. Then the camera point is put under Newton's law of gravity. The superposition of the gravitational forces for the camera point is considered as the vector of movement.

Let us introduce a notation for the set of visible vertices with respect to the viewpoint p: $V(p) = \{v \in V | v$ is visible from $p\}$. $P_0^k = \{p_0, p_1, \ldots, p_k\}$ is the set of viewpoints (the camera trajectory), the set of explored vertices is $V(P_0^k) = \bigcup\limits_{i=0}^{k} V(p_i)$. Exploration is started from the best viewpoint p_0, where the measure of viewpoint quality is expressed by Eq. 3.

The next viewpoint p_{k+1} is to be appended to the trajectory P_0^k. Let us consider the point p_k, where the camera stays. Since the camera cannot leave the surface of the sphere, the movement factors are represented by vectors lying in the tangent plane to the sphere in the point p_k. If a mass m is assigned to the point p, then in the tangent plane a force with the norm $\|\vec{g_p}\| = \frac{m}{|(p_k,p)|^2}$ appears. The direction of $\vec{g_p}$ is the direction from p_k to p. It can be determined by taking the intersection line of the tangent plane and the plane, where the arc (p_k, p) of the sphere lies. Fig. 10 shows an example of gravitational forces.

In order to attract the camera to unexplored areas, the value of the viewpoint mass will be chosen according to new information brought by this viewpoint. Let us introduce a function evaluating the mass of the viewpoint $p \in S$ with respect to the camera trajectory P_0^k (of course, we suppose that $V(p) \neq \emptyset$):

$$m(p) = \frac{\left|N\left(p|P_0^k\right)\right|}{|V(p)|} \cdot I(p),$$

where:

1. $I(p)$ is computed according to Eq. 3
2. $N\left(p|P_0^k\right) = V(p) \setminus \left(V(p) \bigcap V\left(P_0^k\right)\right)$ is the set of new vertices discovered from the viewpoint p.

Thus, $|N(p|P_0^k)|$ is the amount of new information brought by the point p with respect to the trajectory P_0^k. Multiplier $I(p)$ forces the "operator" to guide the trajectory via good viewpoints. Algorithm 3 expresses the scheme.

The proposed algorithm exhibits main properties corresponding to the cognitive constraints previously defined.

– The camera tries to move as quickly as possible to large uncovered areas, due to their higher attractive masses.

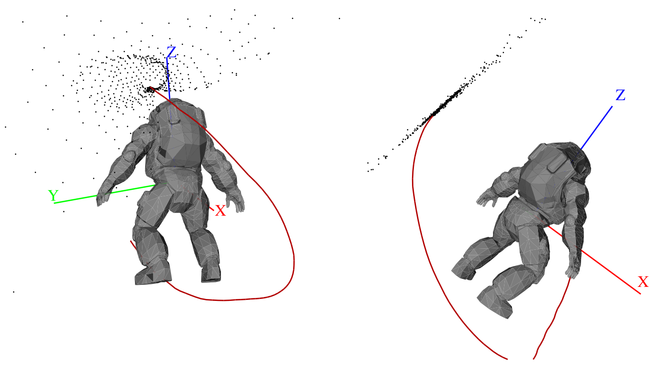

Fig. 10. The black points indicate endpoints of the gravitational forces. They lie on the plane, tangent to the end of the trajectory. Note that the camera will not return to the visited areas since the forces in these directions are equal to zero

Algorithm 3. The incremental algorithm computing the trajectory of the virtual camera.

```
Input: The set of vertices $V$, the discrete sphere $S$,
    the camera step size $c$
Output: The exploration trajectory $P$
$k \leftarrow 0$; $P \leftarrow \{p_0\} : I(p_0) = \max_{s \in S} I(s)$
while $V \setminus V(P_0^k) \neq \emptyset$ do
    $\vec{d} \leftarrow \sum_{s \in S \setminus \{p_k\}} \vec{g_s}$
    $\vec{d} \leftarrow \vec{d} \cdot \frac{c}{\|\vec{d}\|}$
    find $p_{k+1} \in S$ closest to $p_k + \vec{d}$
    $P_0^{k+1} \leftarrow P_0^k \bigcup \{p_{k+1}\}$
    $k \leftarrow k + 1$
end while
```

– There are no comebacks to visited areas, as the mass $m(p)$ of a visited point p is zero.
– The camera does not have any brusque trajectory changes except when it suddenly discovers a new interesting object. An example of such a case is shown in Fig. 11a. The camera suddenly faces the interior of the sphere through the hole, thus discovering new areas. So the previous explored object becomes less attractive than the new uncovered areas and the camera moves rapidly to these new ones. Without a smoothing factor, the trajectory will have high spatial frequencies in its changes. Introducing an inertia factor filters these high spatial changes and the trajectories are smoothed in a very natural way (Fig. 11a). The more influence the inertia has on the trajectory, the smoother a path is obtained.

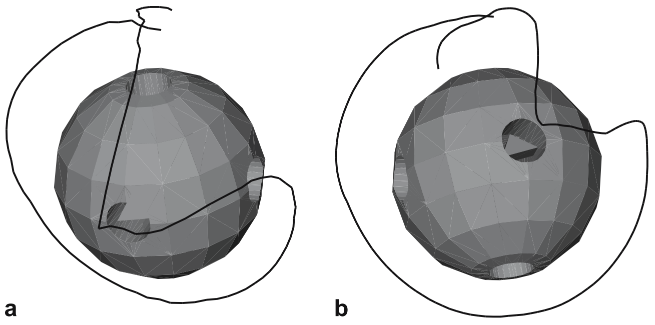

Fig. 11. **a** The interesting object appeared suddenly and the big attractor disappeared. Thus, the camera wants to see another interesting object immediately. **b** By introducing the inertia factor a smooth trajectory is obtained

Sometimes it is not sufficient to view an object once, for example, if the user has passed some parts of the scene quickly, he (she) probably did not understand it properly.

So, it is better to show the parts of the scene several times. The method could be easily adapted to this task. If it is possible to "forget" about parts already seen a long time ago, there will be always some regions of the object to explore.

The "forgetting" can be done in different manners, the simplest one is to determine the "size of memory", the time L, during which the algorithm keeps in memory a visited vertex. Then the amount of new information brought by the point p_{k+1} is

$$N\big(p_{k+1}|P_0^k\big) = V(p_{k+1}) \setminus \Big(V(p_{k+1}) \bigcap V\big(P^k_{\max(k-L,0)}\big)\Big).$$

A more sophisticated method keeps in memory not binary information about whether a vertex was visited or not, but rather the time of the last visit. More precisely, having a trajectory P_k^0, we define the time T of a vertex $v \in V$ visit as follows:

$$T\left(v|P_0^k\right) = \begin{cases} \max\limits_{i=0\dots k} \{i|v \in V(p_i)\} & \text{if } v \in V\left(P_0^k\right) \\ -1 & \text{if } v \notin V\left(P_0^k\right) \end{cases}.$$

Then $m(p)$ is transformed to

$$\hat{m}(p) = \sum_{v\in V} \left(k - T\left(v|P_0^k\right)\right) \cdot I(p),$$

and a vertex not visited during a long time attracts the camera with more and more power until the camera sees it.

5 Results

This section presents several exploration trajectories computed for different 3D scenes.

Fig. 12 shows a trajectory obtained with the "gravitational" method of global scene exploration. The images are snapshots taken consequently from the "movie", the first one corresponds to the start of the movement, i.e., to the best viewpoint. Let us compare it with the trajectory shown in Fig. 5. It is presented in [21] and is computed with the method introduced in [1].

The trajectories are smooth and they have approximately the same length. However, it is easy to see that the new method gives a better path. Figure 5 shows that the camera passes under the floor of the virtual office, and during this time the user does not see the scene. The new trajectory is free of this disadvantage.

The next example of the new method application is shown in Fig. 13. This model is very good for testing exploration techniques; it represents six objects imposed into holes on the sphere, and the explorer should not miss them. None of the previously proposed methods can observe this model properly. All of them, having found an

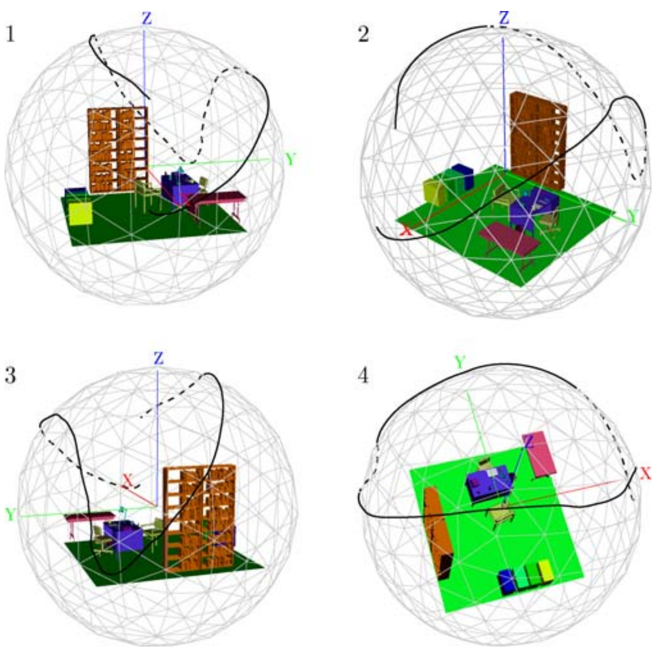

Fig. 12. The exploration trajectory for the virtual office model computed with the new method. The images are snapshots taken from the "movie"; the first one corresponds to the start of the movement (the best viewpoint)

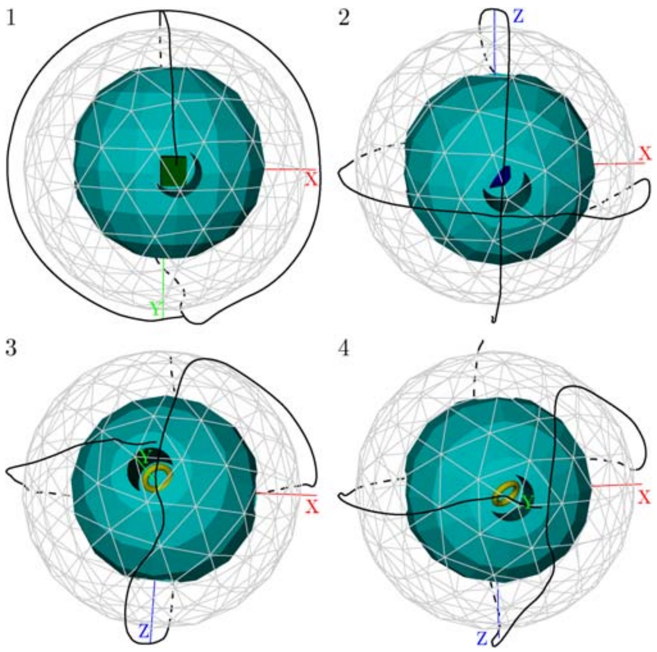

Fig. 13. The exploration trajectory for the sphere with six embedded objects. The images are snapshots taken from the "movie", the first one corresponds to the start of the movement (the best viewpoint)

embedded object, are confused when choosing the next direction of movement. This is due to the missing information about unexplored areas. On the contrary, the new method operates with the analytic visibility graph, which allows determination of where some unexplored areas rest.

Further, we give some more examples of exploration trajectories for different scenes. The exploration trajectory for the Utah teapot is shown in Fig. 14.

Figure 16 shows the best viewpoint a virtual residential quarter scene. The exploration trajectory for the quarter is given in Fig. 15.

Figures 17 and 18 show protracted trajectories made for the Stanford bunny and for the ionic temple model.

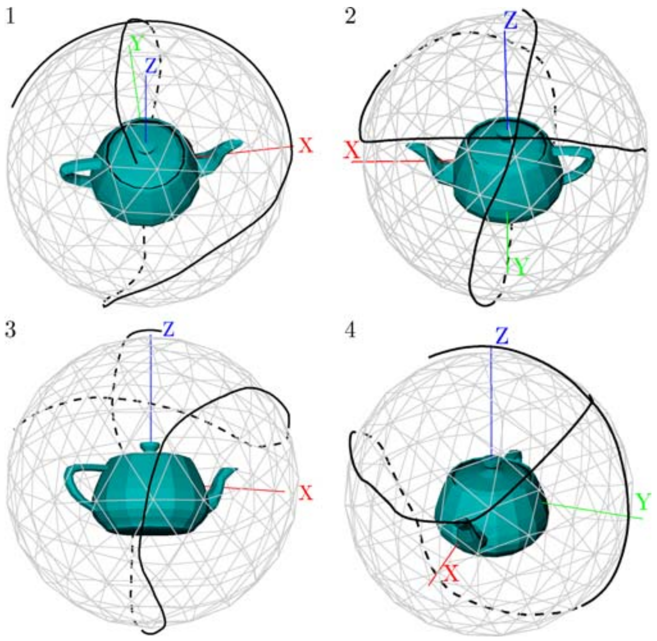

Fig. 14. The exploration trajectory for the Utah teapot model. Images are taken from the "movie", the first one is the best viewpoint

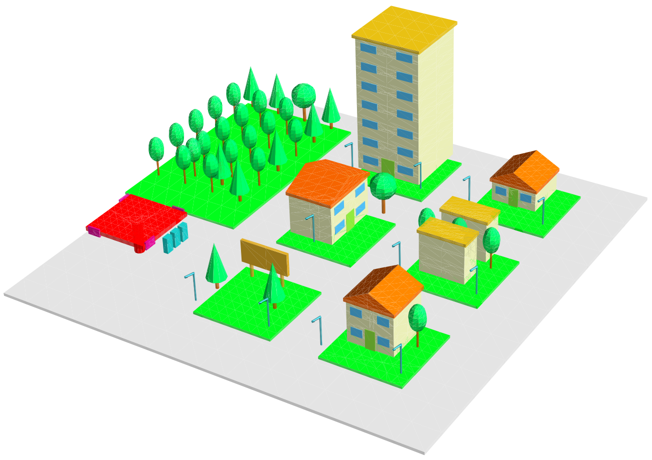

Fig. 16. The first frame from the exploration movie – the best viewpoint

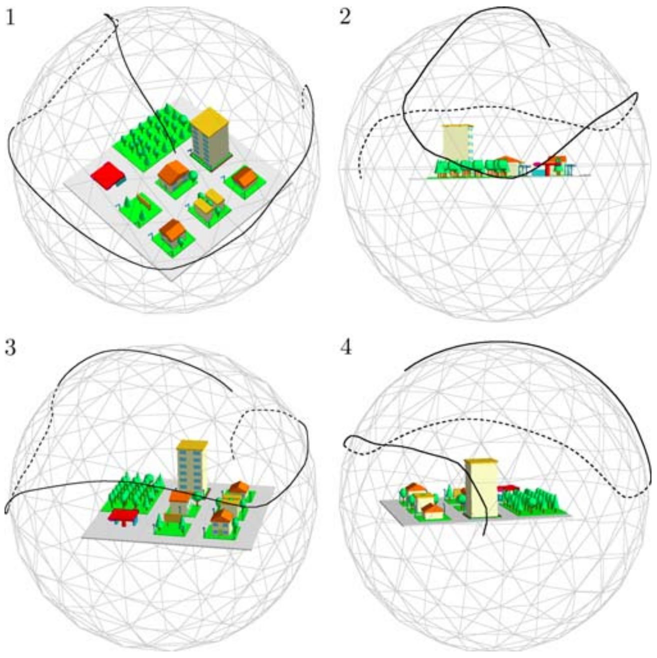

Fig. 15. Residential quarter exploration. The reader is referred to Fig. 16 for a more detailed picture

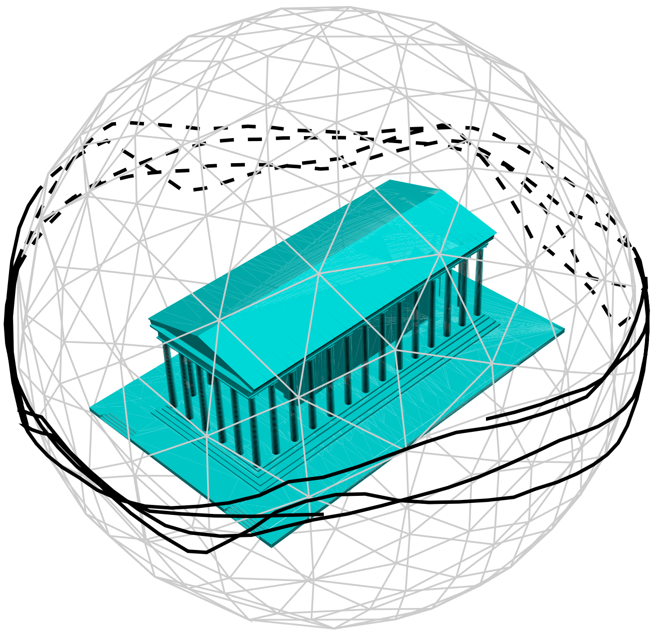

Fig. 17. The protracted exploration trajectory for the ionic temple model

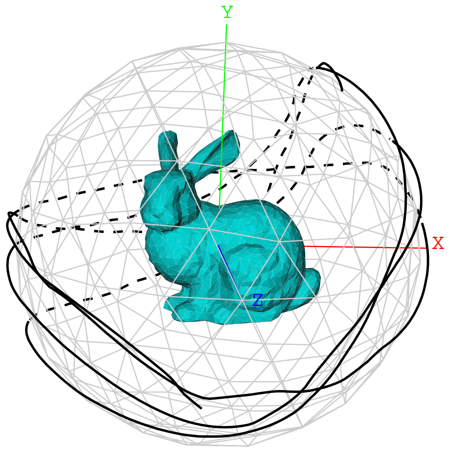

Fig. 18. The protracted exploration trajectory for the Stanford bunny

6 Conclusion and future work

In this paper, several approaches of virtual worlds exploration are described and discussed. An approximate method that finds a minimal set of viewpoints sufficient to see the entire scene is presented. A method for virtual "film making" (automatic scene exploration) is introduced. Finally, a very rapid method of viewpoint quality estimation is presented. Note that previously we fixed an *offline* exploration of a virtual world as our purpose, but in the majority of cases the proposed techniques may work in *real-time*.

This work leaves several interesting problems, such as the inner exploration of an object, exploration with surface ripping and so on. Certainly, the considered methods are far from ideal and can be improved. Different people have different tastes, and we suppose that artificial intelligence techniques can help to handle some uncertainties.

The proposed techniques allow one to get a good comprehension of a single virtual artifact or a general comprehension of a scene. In the future it would be interesting to extend our definitions. For example, we can consider not only curvature or the number of faces, but also the number of visible objects. An example is shown in Fig. 19. The display is almost completely hidden by the case, but it can be clearly recognized. If we can properly split (in human perception) a scene into a set of objects, then the number of visible objects becomes a very important criterion of viewpoint quality estimation.

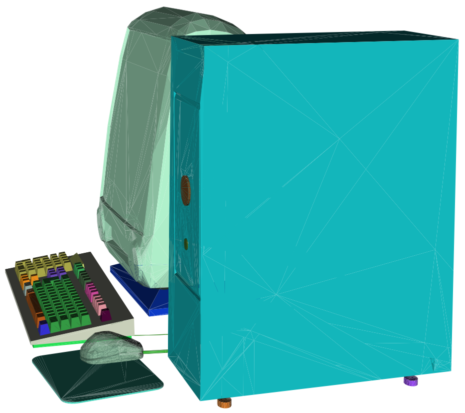

Fig. 19. The display is almost completely hidden by the case, but it can be clearly recognized

For example, in a virtual museum different objects should have different importance. Obviously, the artefacts should have significantly greater importance than walls, chairs and so on. With a proper division of a virtual museum model into a set of objects, we obtain good heuristics for an automatic (or guided manual) exploration.

It would be interesting to develop new automatic methods of scene exploration allowing interaction with the user, where the user can point out which parts of a scene he would like to explore in detail. Probably, machine learning techniques can be used in order to improve interaction with the user.

Acknowledgement The authors would like to express their gratitude towards the European Community and the Limousin Region for the financial support of the present work.

References

1. Barral, P., Dorme, G., Plemenos, D.: Visual understanding of a scene by automatic movement of a camera. In: Y. Bayakovsky (ed.) Graphicon'99. Moscow (Russia) (1999)
2. Barral, P., Dorme, G., Plemenos, D.: Intelligent scene exploration with a camera. In: D. Plemenos (ed.) International Conference 3IA'00. Limoges (France) (2000)
3. Barral, P., Dorme, G., Plemenos, D.: Scene understanding techniques using a virtual camera. In: Eurographics'2000. Interlaken (Switzerland) (2000)
4. Bresenham, J.E.: Algorithm for computer control of a digital plotter. IBM Syst. J. **4**(1), 25–30 (1965)
5. Chazelle, B.: Triangulating a simple polygon in linear time. Discrete Comput. Geom. **6**(5), 485–524 (1991)
6. Colin, C.: A system for exploring the universe of polyhedral shapes. In: Eurographics'88. Nice (France) (1988)
7. Dorme, G.: Study and implementation of 3D scene understanding techniques. Ph.D. Thesis, University of Limoges (France) (2001) (in French)
8. Feige, U.: A threshold of ln n for approximating set cover. J. ACM **45**(4), 634–652 (1998)
9. Feixas, M.: An information theory framework for the study of the complexity of visibility and radiosity in a scene. PhD Thesis, University of Catalonia (2002)

10. Feixas, M., del Acebo, E., Bekaert, P., Sbert, M.: An information theory framework for the analysis of scene complexity. Comput. Graph. Forum **18**(3), 95–106 (1999)
11. Garey, M.R., Johnson, D.S.: Computers and Intractability – A Guide to the Theory of NP-Completeness. Freeman and Company, New York (1979)
12. Jaubert, B., Tamine, K., Plemenos, D.: Techniques for off-line scene exploration using a virtual camera. In: D. Plemenos (ed.) International Conference 3IA'06. Limoges (France) (2006)
13. Kamada, T., Kawai, S.: A simple method for computing general position in displaying three-dimensional objects. Comput. Vision Graph. Image Process. **41**(1), 43–56 (1988)
14. Lee, C.H., Varshney, A., Jacobs, D.W.: Mesh saliency. ACM Trans. Graph. **24**(3), 659–666 (2005). DOI http://doi.acm.org/10.1145/1073204.1073244
15. Marchand, E., Courty, N.: Image-based virtual camera motion strategies. In: S. Fels, P. Poulin (eds.) Graphics Interface Conference, GI'00, pp. 69–76. Morgan Kaufmann, Montreal, Quebec (2000)
16. Noser, H., Renault, O., Thalmann, D., Thalmann, N.M.: Navigation for digital actors based on synthetic vision, memory, and learning. Comput. Graph. **19**(1), 7–19 (1995)
17. Plemenos, D.: A contribution to the study and development of scene modelling, generation and visualisation techniques. The Multiformes project. Professorial dissertation, University of Nantes, France (1991)
18. Plemenos, D.: Exploring virtual worlds: Current techniques and future issues. In: E. Kuzmin, D. Ivanov (eds.) International Conference Graphicon'03. Moscow (Russia) (2003)
19. Plemenos, D., Benayada, M.: Intelligent display in scene modelling. New techniques to automatically compute good views. In: T. Paltashev (ed.) Graphicon'96. Saint Petersburg (Russia) (1996)
20. Plemenos, D., Grasset, J., Jaubert, B., Tamine, K.: Intelligent visibility-based 3D scene processing techniques for computer games. In: V. Debelov (ed.) Graphicon'05. Novosibirsk (Russia) (2005)
21. Plemenos, D., Sbert, M., Feixas, M.: On viewpoint complexity of 3D scenes. In: E. Kuzmin, D. Ivanov (eds.) International Conference Graphicon'04. Moscow (Russia) (2004)
22. Sbert, M., Feixas, M., Rigau, J., Castro, F., Vázquez, P.P.: Applications of the information theory to computer graphics. In: International Conference 3IA'02. Limoges (France) (2002)
23. Sokolov, D., Plemenos, D.: Viewpoint quality and scene understanding. In: M. Mudge, N. Ryan, R. Scopigno (eds.) VAST 2005: Eurographics Symposium Proceedings, pp. 67–73. Eurographics Association, ISTI-CNR Pisa, Italy (2005)
24. Vázquez, P.P.: On the selection of good views and its application to computer graphics. Ph.D. Thesis, Barcelona (Spain) (2003)
25. Vázquez, P.P., Feixas, M., Sbert, M., Heidrich, W.: Viewpoint selection using viewpoint entropy. In: VMV '01: Proceedings of the Vision Modeling and Visualization Conference 2001, pp. 273–280. Aka GmbH (2001)
26. Vázquez, P.P., Feixas, M., Sbert, M., Heidrich, W.: Image-based modeling using viewpoint entropy. In: Computer Graphics International, pp. 267–279 (2002)
27. Vázquez, P.P., Sbert, M.: Automatic indoor scene exploration. In: International Conference 3IA'03. Limoges (France) (2003)

DMITRY SOKOLOV has been a PhD student at MSI laboratory at the University of Limoges (France) since 2004. His research area is automatic virtual world exploration.

DIMITRI PLEMENOS is a full professor at the University of Limoges (France). His research areas include intelligent techniques in computer graphics, as well as declarative modelling, intelligent rendering and intelligent virtual world exploration. He is the author or co-author of several papers and a member of the IPC of many international conferences and journals. Dimitri Plemenos is the organizer and general chair of the 3IA International Conference on Computer Graphics and Artificial Intelligence.

KARIM TAMINE is an associate professor in Computer Science at the University of Limoges (France). His research areas are computer graphics and network security.