Techniques for off-line scene exploration using a virtual camera

Benoît JAUBERT, Karim TAMINE, Dimitri PLEMENOS
University of Limoges, XLIM laboratory, UMR – CNRS 6172
Limoges – France
jaubert@msi.unilim.fr, tamine@msi.unilim.fr, plemenos@unilim.fr

---

Abstract

A new method for automatic exploration of a three-dimensional scene based on a virtual camera is presented. The proposed method is related to off-line exploration and is made of two steps. In the first step, a "global" view of the scene is computed by determining a minimal set of "good" points of view. The second step uses this set of points of view to compute a camera path around the scene.

Keywords: *scene understanding, automatic virtual camera, good point of view, visibility.*

---

1 Introduction

Virtual worlds exploration techniques become nowadays more and more important. When, more than 10 years ago, we have proposed the very first methods permitting to improve the knowledge of a virtual world [Ple91, PB96], many people thought that it was not an important problem. People begun to understand the importance of this problem and the necessity to have fast and accurate techniques for a good exploration and understanding of various virtual worlds, only during these last years. However, there are very few papers which face this problem from the computer graphics point of view, whereas 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 purpose of techniques used in robotics. In computer graphics, the purpose of the program which guides 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.

In this paper we are mainly concerned by global virtual world exploration, where the camera remains outside the scene. Several papers have been published on *on-line* global virtual world exploration. This time, our purpose is *off-line* exploration. Moreover, the purpose of this paper is visual exploration of fixed unchanging virtual worlds.

The paper is organised in the following manner: In section 2 a study of existing techniques on virtual worlds exploration is presented. Section 3 is the main section of the paper, where new off-line exploration techniques are presented, including viewpoint selection, ordering and camera trajectory computing. Some results obtained with the proposed techniques are presented in section 4 before concluding in section 5.

2 Background

The first works on visual scene understanding were published at the end of the years 1980. Thus, Kamada et al. [KK88] proposed a fast method to compute a point of view minimizing the degenerated edges of a scene.

Colin [Col88] has proposed a method initially developed for scenes modelled by octrees. The purpose 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 direction of view. This principle can be described as follows:

1. Choose the three best directions of view among the 6 directions corresponding to the 3 coordinates axes passing through the centre of the scene.
2. Compute a good direction in the pyramid defined by the 3 chosen directions, taking into account the importance of each one of these directions.

A direction of view is estimated better than another one if this direction of view allows to see more details than the other.

Plemenos [Ple91, PB96] proposed an iterative method of automatic viewpoint calculation. The scene is placed at the centre of a sphere whose surface represents all the possible points of view. The sphere of points of view is divided in 8 spherical triangles and the best one is chosen, according to the quality as points of view of the three vertices of the triangle. Then, the selected spherical triangle is recursively subdivided and the best vertex is chosen as the best point of view at the end of the process.

During the five last years several papers have been published about visibility problems [Dur00, Dur02, Rigau00, Feixas02, Sbert02, PSF04] and selection of good points of view [BDP99, Feixas99, PPV01, PPV02, PPV03a, PPV03b].

A well chosen point of view allows to well understand a simple scene. However, for complex scenes, a single point of view is generally not enough to understand them. Even if the user can see the scene from several good points of view, it is, generally not enough for understand complex scenes because the change from one point of view to another one may be confusing for the user. In order to well understand complex scenes, the best solution is intelligent automatic exploration by a virtual camera. This exploration must avoid brusque changes in camera movement in order to get a smooth exploration of the scene.

When a user discovers a scene on the net, it is possible that the chosen angle of view doesn't allow to well understand the scene. In such a case, the user has two possibilities, according to his (her) disponibility. The first one is to ask immediately a visual exploration of the unknown scene, that is an online exploration. The second possibility is to keep the scene with the intention to discover it later. This possibility implies off-line exploration of the scene.

So, according to the user's decision, two scene exploration modes may be envisaged [Ple03]:

- Real time online exploration, where the scene is visited for the first time and the path of the camera is determined in incremental manner. For this kind of exploration it is important to have fast exploring techniques in order to allow the user to understand in real time the explored world.
- Off-line exploration when the scene exploration program visits the scene before the user. The virtual world is found and analysed by the program guiding the camera's movement in order to determine interesting points to visit and path(s) linking these points. The user will be able to visit the scene later, following an already determined path. For this exploration mode it is less important to use fast techniques to determine the camera's path.

A few papers about online scene exploration have been published since 1999 [PDP99, PDP00a, PDP00b, PPV03c]. Some off-line exploration techniques have

been proposed in a preliminary work [Jau04].

3 New techniques for off-line exploration of three-dimensional scene

This section presents new techniques and methods proposed for automatic off-line exploration of virtual worlds. The exploration process is generally performed in two steps. The goal of the first step is to find a set of points of view allowing to see all the details of the scene. In the second step, the selected points are used in order to compute an optimal trajectory. As it has been explained above, exploration is a global one and the selected points of view are computed on the surface of a sphere surrounding the virtual world.

3.1 Computing an initial set of points of view.

Computing an initial set of points of view is the main part of our off-line exploration techniques. The final set should be minimal and should present as many information of the scene as possible. As an object may be seen from one or more viewpoints, we will have to avoid redundancies in order to keep only useful viewpoints.

The first thing to do is to build a beginning balanced set of viewpoints placed on a sphere which surrounds the scene.

The centre of the surrounding sphere is the centre of the scene and its ray is calculated with the following empiric formula:

$$\rho = 1.5 * \sqrt{\frac{(Xmax-Xmin)^2 + (Ymax-Ymin)^2 + (Zmax-Zmin)^2}{3}}$$

Viewpoints distribution depends on the increment of two loops:

```
For φ from 0 to 2π do
    For θ from 0 to π do
        Point (ρ, φ, θ) is an initial viewpoint
    End
End
```

Smaller increments for ϕ and θ give a better quality of the beginning set. In figure 1, one can see some examples with different increments.

Figure 1 : Starting set of viewpoints

Three criteria have been defined in order to select the most interesting points of view. The first one, as commonly used in exploration methods, is related to the number of visible polygons. The second criterion depends on the number of visible *objects*. The third criterion depends on the above ones and on the angle of each face with the view direction.

Visible polygons

This kind of evaluation considers the number of visible polygons of each point of view. Intuitively, we assume that a point of view is more interesting if it allows to see more polygons. Unfortunately, if this is true in many cases, for some scenes this criterion is not sufficient because the

number of independent parts in the scene is at least as important as the number of polygons. However, the visible polygons criterion will be used, combined with other criteria.

Visible objects

In order to resolve the problem due to some limits of a face-based exploration the concept of object is considered with no consideration of the number of its faces. Indeed, the most interesting point of view allows perceiving the maximum of objects of the scene. This is the natural human perception: in order to describe a room, we first describe the objects it contains. The perception of these objects needs only a partial vision of them. Therefore, it is obvious to consider this criterion as the main one.

Combining evaluation criteria

We add to these criteria a third one which takes into account not only the number of visible polygons but also the quality of view (that is the angle of vision) of a polygon from the point of view. This criterion is resumed in the following formula.

$$mark = (p \times o) + \sum_{polygons} (p_v \times \cos(\alpha))$$

In this formula, p is a real number and represents the number of polygons in the scene divided by the number of objects, o is the number of objects in the scene, p_v is a value between 0 and 1 and shows the visible part of the polygon (0.5 means that the face is 50% visible) and α is the angle between the polygon orientation and the vision direction (Figure 2).

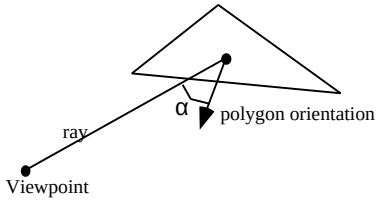

Figure 2: Angle of vision

This criterion offers a additional evaluation mode for a viewpoint using all previous criteria together with quality of view.

3.1.1 Point of view evaluation

The point of view evaluation depends on the values returned by the three criteria which are sorted in the following order, from the more important to the less important*: number of visible objects*, *number of visible faces* and *combination of the above criteria*. For example, let's consider these three criteria. If two different points of view perceive eight objects of the scene, the best one would be the point which has the greatest number of faces. If the number of visible faces for the different objects is the same, the best point of view will be determined by the third criterion. If the values of the three criteria are equal for the two points, one of them will be randomly chosen as the best viewpoint.

3.1.2 Point of view selection

Evaluation function

The goal of an evaluation function is to estimate the view quality from a point, in

order to choose the most interesting viewpoints. This is very important during the exploration procedure. This function returns a point *value* using various criteria. Viewpoint evaluation uses two main tools.

The first tool uses an intersection determining function. This tool says whether the intersection of the polygon with a ray defined by the point of view and a point on a polygon is visible (Figure 3).

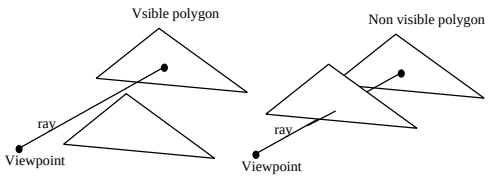

Figure 3: Intersection function

The intersection algorithm is the following:

```
Function Intersection (viewpoint, polygon) : boolean
    result ← true
    ray ← make ray (view point, random polygon point)
    for each other polygon do
      if ray is stopped before the point then
            result ← false
        end
    end
    return result
```

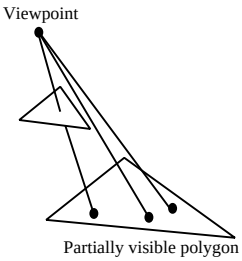

Figure 4: partially visible polygon

The second tool determines the visible part of a polygon from a point of view. To do this, a more or less important number of rays is sent from the point of view to randomly generated points of the polygon (Figure 4). This visible part is given by the following formula.

$$visibility = \frac{number\ of\ non\ stopped\ rays}{number\ of\ rays}$$

The corresponding algorithm is the following:

```
Function visibility (view point, polygon) : real (between 0 and 1)
    Visibility ← 0
    for ray from 1 to Number_of_Rays
        visible ← Intersection (view point, polygon)
        If visible is true then
            visibility← visibility+ (1/ Number_of_Rays)
        end
    end
Return visibility
```

The above tools are used by the global viewpoint evaluation function. This function works in the following manner:

- If a polygon of an object is visible, the object is considered visible.
- At the end of the evaluation process, the function returns three values (seen objects, seen polygons and an evaluation mark of the viewpoint.

The evaluation function algorithm may be schematically presented in the following algorirhm which defines the function *evaluate*.

In this function, a globally visible polygon or object is permanently marked to indicate that it is visible from another viewpoint.

```
function evaluate (scene, viewpoint) : three real
    for each object in the scene do
        for each polygon of the object do
          visible ← visibility (viewpoint, polygon)
          if visible > 0 then
            If the object is not globally visible
              object ← locally visible
            end
            If the polygon is not globally visible
              polygon ← locally visible
              Add 1 to Seen polygons number
              mark ← mark + cos (ray, polygon)
          end
        end
        if the object is locally visible
          Add 1 to Seen objects number
        end
    end
    return (Seen objects number, Seen polygons number,
mark)
```

In this function, a globally visible polygon or object is permanently marked to indicate that it is visible from another viewpoint.

3.1.3 How to mark a polygon or an object as globally visible?

We need a function for marking a polygon or an object permanently to remember which one has already been seen from a selected viewpoint. After the evaluation function finds the best viewpoint, we need to save the set of polygons and objects seen from this viewpoint in order to take them into account. The next iteration of the evaluation function will know which part of the scene was already seen and will choose another viewpoint which fill up the set.

This function works like the evaluation function and marks the globally visible objects and polygons instead of evaluating the viewpoint. The number of rays is much more important than in the evaluation function to assure better results.

```
procedure marking (scene, viewpoint)
    for each object in the scene do
        for each polygon of this object do
            for num from 1 to (number of rays* precision)
              visible ← Intersection (viewpoint, polygon)
              if visible is true then
                object ← globally visible
                polygon ← globally visible
              end
            end
        end
    end
```

3.1.4 First viewpoint selection method

This method uses the initial set of viewpoints and applies the evaluation function on each of them.

The best viewpoint (according to chosen criteria) of the set is selected and added to the final set of viewpoints.

The mark function marks the objects and polygons seen from selected viewpoints.

The evaluation function is applied on each point of the initial set (except the viewpoints already selected in the final set) and the best viewpoint is selected and added to the final set. The viewpoint evaluation function does not take into account already seen objects and polygons.

The whole process is repeated until the terminal condition is satisfied.

The following algorithmic scheme describes this viewpoint selection method.

```
function first method (scene, initial set of viewpoints):
 final set of viewpoints
do
      Chosen point ← none
      best note ← ( -∞, -∞, -∞)
      for each viewpoint in the initial set do
            If viewpoint is not in the final set
                  note ← evaluate (scene, viewpoint)
                  if note > best note then
                        Chosen point ← viewpoint
                        best note ← note
                  end
            end
      end
      If Chosen point ≠ none
            mark(scene, chose point)
            add(final set, Chosen point)
      end
until terminal condition or chosen point = none
return (final set)
```

It is easy to see that in this method:

- All the objects and polygons which can be seen are detected and at least one point of view in the final set can see them.
- The final set of viewpoints is minimal.
- The time cost to compute the final set of viewpoints is maximal. The initial set of viewpoints has to be processed n+1 times if n is the number of viewpoints of the final set.

3.1.5 Second viewpoint selection method

In this method all the viewpoints of the initial viewpoint set are evaluated and sorted, according to their values, in decreasing order. The viewpoints of the obtained classified viewpoint set may now be processed by order of importance. The best viewpoints will be processed before the others.

```
function Second Method (scene, initial set of viewpoints)
                              return final set of viewpoints

sorted array ← empty
Number of points of view ← 0
For each point in the initial set
      evaluation ← evaluate(scene, viewpoint)
      add in sorted array (viewpoint, evaluation)
end
marking (scene, sorted array[0]) (the best viewpoint)
Select useful viewpoints (scene, sorted array)
return (sorted array)
```

The *Select useful viewpoints* procedure tries to suppress redundant viewpoints.

- The evaluation function is used on the current point (except the first one which is already marked).
- If this point allows to see new polygons or objects, we mark them and go to the next point.
- Otherwise the viewpoint is erased from the sorted array.

```
procedure Select useful viewpoints (scene, array)
   for sign from 1 to number of viewpoint-1 do
      evaluation ← evaluate (scene, array[sign])
      if evaluation ≠ blank evaluation
            mark (sorted array[sign])
      else
            erase from sorted array (sign)
            Retract 1 from number of viewpoints
      end
   end
return (array)
```

This method has some advantages and some drawbacks, compared to the first method.

- All the objects and polygons which can be seen are detected and at least one point of view in the final set can see them.
- The final set is not minimal and the number of viewpoints in the final set can be much higher than with the first method.

· The method is less time consuming than the first one. The initial set of viewpoints is processed two times.

3.1.6 Improvements

It is possible to improve the proposed methods of computing reduced viewpoint sets, in order to reduce computation time or to improve the quality of selected viewpoints.

Bounding boxes of objects may be used to accelerate the whole process. Thus, if the bounding box of an object is not intersected by the current ray, it is not necessary to test intersection of the ray with the corresponding object or with polygons of this object.

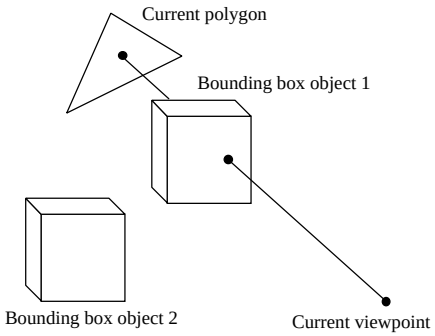

Figure 5: Using bounding boxes to reduce computation cost

In Figure 5 the polygons of object 2 cannot hide the current polygon because te current ray does not intersect its bounding box.

Bounding boxes may also be used instead of objects, in order to accelerate the computation in a first step of the processing where only visible objects will be processed. A second step is then necessary for the non visible objects, in order to verify more precisely some cases difficult to resolve. The use of bounding boxes allows to divide the computation time by a quantity equal to $\frac{average\ of\ faces\ by\ object}{6}$ but the main drawback of this improvement is that embedded objects will never be marked as visible during the first step of the process.

The quality of selected points of view may be improved in the following manner: For each point of view of the final set, some points in its neighborhood are tested and, if they allow to see new objects or polygons, they are included in the set. This improvement allows to find new potentially interesting viewpoints but the computation time and the number of selected viewpoints increases.

3.2 Computing an optimized trajectory

After the reduced set of viewpoints is computed, we have to compute an optimized trajectory using the points of this set. The proposed method works in two steps. The first step performs an ordering of the viewpoints of the set while the second step computes the final trajectory.

3.2.1 Ordering of viewpoints

As the camera will have to visit all the viewpoints of the reduced set of viewpoints, an order must be defined for this visit. This order may depend on various criteria.

3.2.1.1 Proximity-based ordering

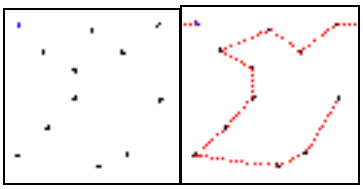

Figure 6 : Proximity-based ordering

In this method (figure 6), proximity of viewpoints is considered as the first ordering criterion:

- The best viewpoint of the reduced viewpoint set is chosen as starting point for the camera trajectory.
- The next trajectory point is the closest to the current point viewpoint. This point becomes the current trajectory point.
- The previous step is repeated for each remaining viewpoint of the reduced viewpoint set.

This ordering is not always satisfactory, especially if the camera uses a straight line trajectory to go from a position to the next one and several trajectory points are close to each other.

3.2.1.2 Minimal path-based ordering

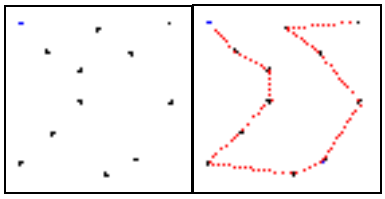

Figure 7 : Minimal path-based ordering

In this method (figure 7), the main ordering criterion is the total length of the whole trajectory, which has to be minimal. This ordering criterion gives better results than the proximity based one but the ordering process is more time consuming. Ordering is achieved by testing all the solutions and retaining the best one. Optimization is not necessary for the moment because the number of points of view is small and exploration is made off-line.

3.2.2 Computing the camera trajectory

At the end of the ordering process, applied to the reduced set of viewpoint, it is possible to compute a camera trajectory.

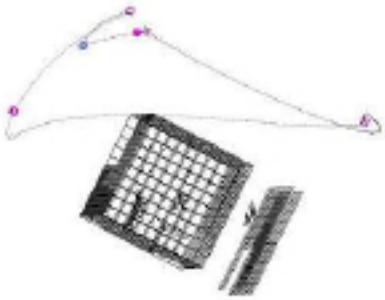

Figure 8 : smooth trajectory for scene exploration

The method has to avoid abrupt changes of direction in the camera movement. To do this, the following technique is used.

Let us suppose (figure 10) that the current camera position on the surface of the sphere is A and the viewpoint to reach is B.

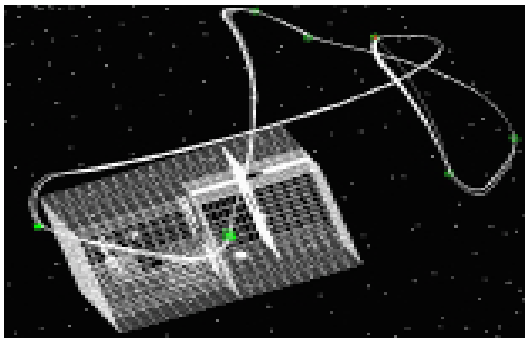

Figure 9 : another example of smooth scene exploration

If the current direction of the camera movement at point A is d, a new direction AA1, between d and AB is chosen and the next position A1 of the camera is defined on the vector AA1. The same process is repeated from each new step. In figure 10, A, A1, A2, … are successive positions of the camera.

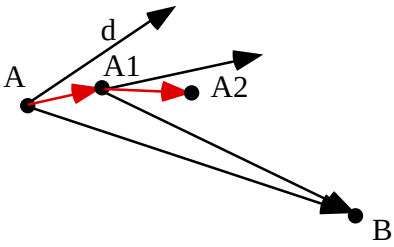

Figure 10 : Computing a smooth camera trajectory

In figure 8 and 9 one can see illustration of smooth camera trajectories obtained with the above technique.

4 Results

This section shows some results obtained with the above techniques, applied to various scenes. Performances of the two viewpoint selection methods, presented in section 3 are compared for the same set of 4 scenes.

Comparison of the proposed methods is made with an initial set of 17 viewpoints. The purpose of these methods is to compute a reduced set of viewpoints. Of course, the minimal set of viewpoints needed to understand a scene is strongly dependent on the nature of the scene. For the scene of figure 10 a single point of view allows to well understand it whereas for the scene of figure 11 at least 4 points of view are needed.

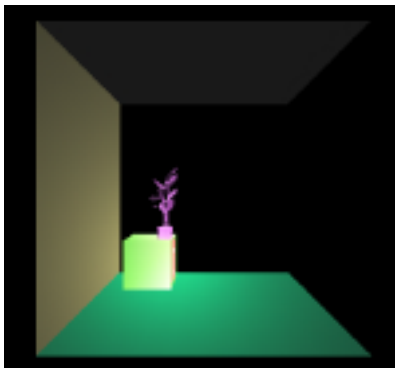

Figure 10: test scene for a single point of view

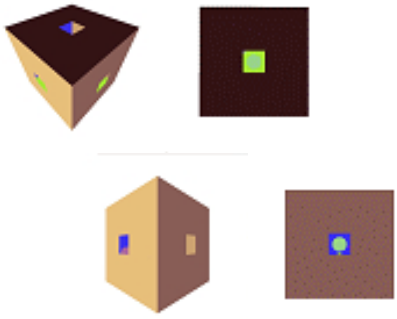

Figure 11: scene with multiple points of view needed

Four different scenes, A, B, C and D, are used for comparisons and two criteria were

chosen to evaluate the two methods: number of points of view of the reduced set of viewpoints computed by each method and time cost of each method.

Results of evaluation using the number of points of view criterion are shown in Table 1 where one can see that the two methods give very often the same results but the first method is generally better than the second one because the computed set of viewpoints is really minimal.

Table 2 shows results of evaluation of the two methods using the time cost criterion. All tests were done with an AMD 2800+ computer with 1Gbyte of RAM.

	Sc A	Sc B	Sc C	Sc D
Method I	4	3	4	3
Method II	4	3	4	6

Table 1: number of points of view by method

	Sc A	Sc B	Sc C	Sc D
Method I	1m57s	3m49s	4m18s	5m49s
Method II	1m23s	3m07s	3m19s	3m30s
Gain	29%	18%	22%	39%

Table 2: computation time by method

The second method is always faster than the first one. The obtained gain grows with the complexity of the scene, even if the notion of scene complexity is not formally defined. This complexity is approximated by the number of needed viewpoints for understanding the scene.

Both, proximity-based and minimal path-based viewpoint ordering, methods give satisfactory results.

5. Conclusion

New techniques for off-line exploration of 3D scenes have been presented in this paper. Unlike in on-line incremental exploration, off-line exploration needs a first step to prepare data, in order to improve the quality and the speed of exploration. In this first step a reduced set of viewpoints has to be computed from an initial set and then the viewpoints of the reduced set have to be ordered for improved exploration of the scene.

The techniques presented in this paper, especially the ones used to compute a minimum set of points of view, may also be used to improve image-based modelling and rendering because they allow selection of pertinent and non redundant positions for the camera which will capture the images.

6. Acknowledgments

This project has been supported and financed in part with funds of the European project GameTools. The authors would like to thank all people and organisations which have supported in any manner this project.

7. References

[BDP00a] P. Barral, G. Dorme, D. Plemenos. Intelligent scene exploration with a camera. International Conference

3IA’2002, Limoges (France), May 3-4, 2000.

[BDP00b] P. Barral, G. Dorme, D. Plemenos. Scene understanding techniques using a virtual camera. Eurographics 2000, Interlagen (Switzerland), August 20-25, 2000, Short papers proceedings.

[BDP99] P. Barral, G. Dorme, D. Plemenos. Visual understanding of a scene by automatic movement of a camera. International Conference GraphiCon'99, Moscow (Russia), August 26 – September 3, 1999.

[Col88] C. Colin. A System for Exploring the Universe of Polyhedral Shapes. Eurographics’88, Nice (France), September 1988.

[Dur00] F. Durand. A multidisciplinary survey of visibility. ACM Siggraph course notes Visibility, Problems, Techniques, and Applications 2000

[Dur02] F. Durand, G. Drettakis, C. Puech. The 3D visibility complex. ACM Trans. Graph. 2002, 21, 176-206.

[Feixas02] Miquel Feixas, An Information Theory Framework for the Study of the Complexity of Visibility and Radiosity in a Scene. PhD thesis, Technical University of Catalonia, 2002.

[Feixas99] M.Feixas, E.Acebo, Philippe Bekaert and M.Sbert. An information theory framework for the analysis of scene complexity, Eurographics'99.

[Jau04] B.Jaubert. Off-line automatic exploration of virtual worlds. MSc report (in French), Limoges (France), July 2004.

[KK88] T. Kamada, S. Kawai. A Simple Method for Computing General Position in Displaying Three-Dimensional Objects. Computer Vision, Graphics and Image Processing, vol. 41, 1988.

[PB96] D. Plemenos, M. Benayada. Intelligent Display Techniques in Scene Modelling. New Techniques to Automatically Compute Good Views. International Conference GraphiCon'96, St Petersburg (Russia), 1-5 of July 1996.

[Ple91] D. Plemenos. A contribution to the study and development of scene modelling, generation and display techniques: The MultiFormes project. Professorial dissertation. Nantes (France), November 18, 1991. In French.

[Ple03] D. Plemenos. Exploring Virtual Worlds: Current Techniques and Future Issues. International Conference GraphiCon'2003, Moscow (Russia), September 5-10, 2003.

[PPV01] P.P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Viewpoint Selection Using Viewpoint Entropy. Vision, Modeling, and Visualization 2001 (Stuttgart, Germany), pp. 273-280, 2001.

[PPV02] P.P. Vázquez, M. Feixas, M. Sbert, and A. Llobet. Viewpoint Entropy: A New Tool for Obtaining Good Views for Molecules. VisSym '02 (Eurographics - IEEE TCVG Symposium on Visualization) (Barcelona, Spain), 2002.

[PPV03] Pere Pau Vázquez, PhD thesis. On the Selection of Good Views and its Application to Computer Graphics. Technical University of Catalonia, 2003.

[PPV03b] Pere-Pau Vázquez and Mateu Sbert. Fast adaptive selection of best views. Lecture Notes in Computer Science, 2003 (Proc. of ICCSA'2003).

[PPV03d] P.P. Vázquez, M. Feixas, M. Sbert, and W. Heidrich. Automatic View Selection Using Viewpoint Entropy and its Application to Image-Based Modeling. Computer Graphics Forum, desember-2003.

[PPV03e] Pere-Pau Vázquez and Mateu Sbert. Automatic indoor scene exploration. In International Conference on Artificial Intelligence and Computer Graphics, 3IA'2003, Limoges, May 2003.

[PSF04] D. Plemenos, M. Sbert, M. Feixas. On viewpoint complexity of 3D scenes. STAR report. International Conference GraphiCon'2004, Moscow (Russia). September 5-10, 2004.

[Rigau00] J. Rigau, M. Feixas, and M. Sbert. Information Theory Point Measures in a Scene. IIiA-00-08-RR, Institut d'Informàtica i Aplicacions, Universitat de Girona (Girona, Spain), 2000.

[Sbert02] M. Sbert, M. Feixas, J. Rigau, F. Castro, and P.P. Vázquez. Applications of Information Theory to Computer Graphics. Proceedings of 5th International Conference on Computer Graphics and Artificial Intelligence, 3IA'2002 (Limoges, France), pp. 21-36, May 2002.