Scene understanding by apparent contour extraction

Nancy DANDACHY, Dimitri PLEMENOS, Bachar EL HASSAN*
University of Limoges, XLIM laboratory, 83, rue d'Isle, 87000 Limoges (France)
*Lebanese University, Faculty of Engineering, El Kobbeh, Tripoli, Lebanon
dandachy@msi.unilim.fr, plemenos@unilim.fr, elhassan@ul.edu.lb

Abstract

In this paper, we present techniques allowing visual understanding of scenes which are difficult to understand from a realistic rendering due to reflection and refraction effects. To do this, an apparent contour extraction technique is used, based on scene knowledge and using a ray casting algorithm together with a selective refinement approach. With this technique the real objects of the scene are delimited by their apparent contours and may be overlaid on the realistic rendering of the scene, making the user able to distinguish reality from reflection and refraction effects.

Keywords: *Scene understanding, Contour extraction, Ray casting, Selective refinement.*

1. Introduction

The problem of understanding a scene is currently a more and more pertinent problem because of the development of web applications and possibilities, for a user, to discover new, never seen, scenes on the net. These scenes are sometimes difficult to understand for various reasons. One of these reasons is the use of realistic rendering to get an image of the scene. Even if it seems paradoxical, realistic rendering does not always allow understanding reality. This is the case with scenes containing lights, mirrors and transparent objects. In such cases, shadows, reflections and refractions give the user to see non existing objects and it is difficult for him (her) to understand what is a real object and what is illusion.

In this paper, we propose a new approach to face the problem of understanding scenes containing lights mirrors and transparent objects. The proposed method is based on ray casting and selective refinement improvement, in order to extract apparent contours of the real objects of the scene.

The paper will be organized in the following manner: In section 2 the main current techniques allowing easier understanding of 3D scenes will be presented. A lot of these techniques are based on automatic computing of a good view or of a good path for a virtual camera exploring the scene. As we are going to enhance the visualization by extracting apparent contours we are going to extend our research to study the existing methods in this field which distinguish between image space algorithms, hybrid algorithms and object space algorithms. In section 3 the understanding problem for scenes rendered in realistic manner will be explained and a method to avoid this problem will be presented. In section 4 the first results obtained with the proposed method will be presented and commented. In section 5 a conclusion on the pertinence of our method will be made and possible future work will be considered.

2. Main techniques for scene understanding

The very first works in the area of understanding virtual worlds were published at the end of 80's and the beginning of 90's. There were very few works because the computer graphics community was not convinced that this area was important for computer graphics. The purpose of these works was to offer the user a help to understand simple virtual worlds by computing a good point of view.

2.1 Best view computing for virtual worlds

When the virtual world to understand is simple enough, a single view of it may be enough to understand the virtual world. So, it is important to be able to propose an automatic computation of a "good" viewpoint.
Kamada and Kawai [KK88] consider a position as a good point of view if it minimizes the number of degenerated images of objects when the scene is projected orthogonally. A degenerated image is an

image where more than one edges belong to the same straight line. They have proposed to minimize the angle between a direction of view and the normal of the considered plan for a single face or to minimize the maximum angle deviation for all the faces of a complex scene.
This technique is very interesting for a wire-frame display. However, it is not very useful for a more realistic display. Indeed, this technique does not take into account visibility of the elements of the considered scene and a big element of the scene may hide all the others in the final display.

Plemenos and Benayada [PB96] propose an iterative method of automatic viewpoint calculation and create a heuristic that extends the definition given by Kamada and Kawai. A point is considered as a good point of view if it gives; in addition of the minimization of deviation between a direction of view and normals to the faces, the most important amount of details. The viewpoint quality is based on two main geometric criteria: number of visible polygons and area of the projected visible part of the scene, it is computed using the following formula:

$$I(v) = \frac{\sum_{i=1}^{n} \left\lceil \frac{P_i(v)}{P_i(v)+1} \right\rceil}{n} + \frac{\sum_{i=1}^{n} P_i(v)}{r} \qquad (1)$$

Where: I(V) is the importance of the view point V,
Pi(V) is the projected visible area of the polygon number i obtained from the point of view V,
r is the total projected area,
n is the total number of polygons of the scene.
[a] denotes the smallest integer, greater than or equal to a.
The process used to determine a good point of view works as follows:
The scene is placed on the center of the unit sphere whose surface represents all possible points of view. The sphere is divided into 8 spherical triangle (see Fig.1) and the best one will be whose vertices represent the best quality according to the formula (1). Finally the best point of view is computed by recursive subdivision on the best spherical triangle. (See Fig. 2)

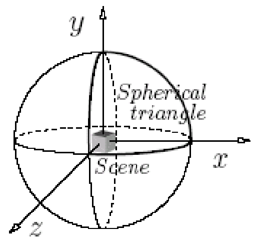


Fig. 1: *The sphere of viewpoint into 8 spherical triangle*

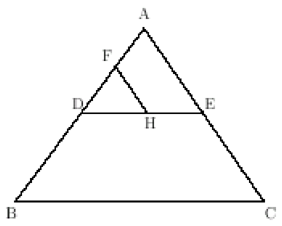


Fig. 2: *Recursive subdivision of the best spherical triangle*

This method gives generally interesting results. However, the number of polygons criterion may produces some drawbacks. To resolve this problem, Sokolov et al. [SP05], [SP06] propose to replace the number of polygon criterion by the criterion of *total curvature* of the scene which is given by the equation:

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

Where:
F(p) is the set of polygons visible from the viewpoint p,
P(f) is the projected area of polygon f,
V(p) is the set of visible vertices of the scene from p,
α(v) is the set of angles adjacent to the vertex v.
Equation 2 uses the curvature in a vertex (see Fig. 3), which is the sum of angles adjacent to the vertex minus 2π.

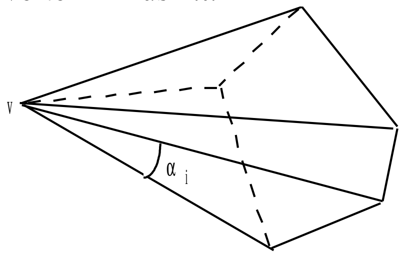


Fig. 3: *Curvature in a vertex*

The best viewpoint is computed by using a data structure, so-called *visibility graph*, which allows, to every discrete potential viewpoint on the surface of the surrounding sphere, the association of the visual pertinence of view from this viewpoint.
Colin [CC88] proposed a method to compute a good view for octree models. The viewpoint is considered to be good if it shows high amount of voxels. This method computes first the "best" initial spherical triangle and then the "best"

viewpoint is approximately estimated on the chosen triangle.
Sbert et al. [SF02] use *viewpoint entropy* criterion to evaluate the amount of information captured from a given point of view which is defined by the following formula:

$$H_p(x) = -\sum_{i=0}^{N_f} \frac{A_i}{A_t} \mathrm{Log} \frac{A_i}{A_t} \qquad (3)$$

Where N_f is the number of faces of the scene, A_i is the projected area of the face i and A_t is the total area covered over the sphere.
The maximum entropy is obtained when a viewpoint can see all the faces with the same relative projected area Ai/At. The best viewpoint is defined as the one that has the maximum entropy.

When we have to understand a complex virtual world, the knowledge of a single point of view is not enough to understand it. Computing more than one point of view is generally not a satisfactory solution in most cases because the transition from a point of view to another one can disconcert the user, especially when the new point of view is far from the current one. The best solution, in the case of complex virtual worlds is to offer an automatic exploration of the virtual world by a camera that chooses good points of view and, at the same time, a path that avoids brusque changes of direction. Several authors have proposed methods for online or offline exploration [PD99, PD00, DG01, VS03, VP03, JP05, JP06, SP05, SP06].

2.2 Apparent contour extraction techniques

In computer graphics, contour extraction and rendering has a central role in a growing number of applications. It is not only used in non photorealistic rendering for artistic styles and cartoons, it's also used for technical illustrations, architectural design and medical atlases [HZ00], in medical robotic [OZ06], and for photo realistic rendering enhancement. It has been used as an efficient means to calculate shadow volumes [HZ00], to rapidly create and render soft shadows on a plane [HE01]. It's also used to facilitate the haptic rendering [JC01]. Some authors, [CP98, JR02] have described the use of silhouettes in CAD/CAM applications. Systems have also been built which use silhouettes to aid in modeling and motion capture tasks [BL01, FP99, and LG00]. More applications and techniques based on line drawings detection are described in the course notes 7 of SIGGRAPH 05 [SG05]

Isenberg distinguish in his paper [IS03] between image space algorithms that only operate on image buffers, hybrid algorithms that perform manipulations in object space but yield the silhouette in an image buffer, and object space algorithms that perform all calculations in object space with the resulting silhouette represented by an analytic description of silhouette edges.

2.2.1 Image space algorithms

In image space, the easiest way to find significant lines would be by detecting discontinuities in the image buffer(s) that result from conventional rendering and extract them using image processing methods. This however, doesn't detect silhouette since changes in shading and texturing can cause erroneous edge detection. Saito and Takahachi [ST90], Decaudin [Dec96], Hertzmann [HA99], Deussen and Strothotte [DS00], Nienhaus and Dellner [ND03], extend this method by using the geometric buffers known as G-Buffers which are the depth buffer (z-buffer), the normal buffer and the Id-buffer.
By detecting the 0 discontinuities of the depth map we can obtain the silhouette. And with the detection of the 1 discontinuities of the normal map we can obtain the crease edge. The advantage of image space algorithms is that they are relatively simple to implement because they operate with buffers which can be generated easily with the existing graphics hardware, can be applied to all kinds of model and give good results in simple cases. However, the main disadvantage of image space algorithms is that they depend on a threshold that has to be adjusted for each scene. A second disadvantage of these techniques is that in the case of complex images they might add undesired contours or miss some lines because of the presence of reflections and refractions.

2.2.2 Hybrid algorithms

Rustagi [RP89], Rossignac and Emmerik [RE92] use hybrid algorithms which are characterized by

operations in object space (translations) that are followed by rendering the modified polygons in a more or less ordinary way using a *z*-buffer. This usually requires two or more rendering passes.
Raskar and Cohen [RC99], Gooch et al. [GB99], Raskar [RR01], they generalize this approach in their work by using traditional *z*-buffering along with back-face or front-face culling, respectively.
With hybrid algorithms, the visual appearance of the generated images tends to be a more stylistic one.
Similarly to image space algorithms, hybrid algorithm might fail to face some numerical problems due to limited *z*-buffer resolution.

2.2.3 Object space algorithms

The computation of silhouettes in this group of algorithms, as the name suggests, takes place entirely in object space. The trivial object space algorithm is based on the definition of a silhouette edge. The algorithm consists of two basic steps. First, it classifies all the mesh's polygons as front or back facing, as seen from the camera. Next, the algorithm examines all model edges and selects only those that share exactly one front- and one back-facing polygon. The algorithm must complete these two steps for every frame.
Buchanan and Sousa [BS00] suggest, to support this process, the use of a data structure called an *edge buffer* where they store two additional bits per edge, F and B for front and back facing. They extend later this idea to support border edges and other feature edges.
This simple algorithm, with or without using the edge buffer data structure, is guaranteed to find all silhouette edges in the model. It is easy to implement and well suited for applications that only use small models. However, it is computationally expensive for common hardware-based rendering systems and the model sizes typically used with them. It must look at every face, determine face orientation (using one dot product per face; for perspective projection it must recompute the view vector for every face), and look at every edge. This is a linear method in terms of the number of edges and faces but too slow for interactive rendering of reasonably sized models

To speed up the algorithm, Card and Mitchell [CM02] suggest employing user-programmable vertex processing hardware (shaders).

Hertzmann and Zorin [HZ00] consider the silhouette of a free-form surface approximated by a polygonal mesh. To find this silhouette, they recompute the normal vectors of the approximated free-form surface in the vertices of the polygonal mesh. Using this normal, they compute its dot product with the respective viewing direction. Then, for every edge where the sign of the dot product is different at both vertices, they use linear interpolation along this edge to find the point where it is zero. These points connect to yield a piecewise linear subpolygon silhouette line. The resulting silhouette line is likely to have far fewer artifacts. Also, the result is much closer to the real silhouette than the result of a traditional polygonal silhouette extraction method.
Nehab and Gattas [NG00] propose a completely different approach for ray traced images. They create an algorithm that divides ray into equivalence categories or classes. They consider a pixel as representing the ray cast through its lower left corner. To determine the pixels that spawn edges, each pixel is compared against its three 4-neighbors (*a*, *b*, *c* for pixel *p* in Fig. 4). If any differences are detected, the pixel is selected as an edge pixel.

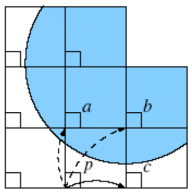


Fig. 4: *Edges are detected when a pixel's category is different from that of one of its neighbors.*

The path followed by a ray is represented by a tree (see Fig. 5) and can be described by:

1. the ray who intersects the object
2. Rays coming from lights visible by the intersected object
3. Reflected & refracted rays

Two pixels are in the same category if their ray trees are equal (have the same word obtained by joining the summit of Knots):

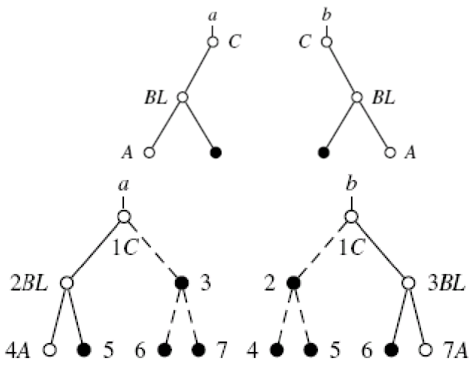


Fig. 5: *With the numbering of the node positions, trees a and b receive strings 4A2BL1C and 7A3BL1, respectively.*

The results are very good for simple scenes. However, for complex scenes, the method can fail unpredictably specially with scenes that represent reflections and refractions; refracted and reflected object's edges are detected (see Fig. 6)

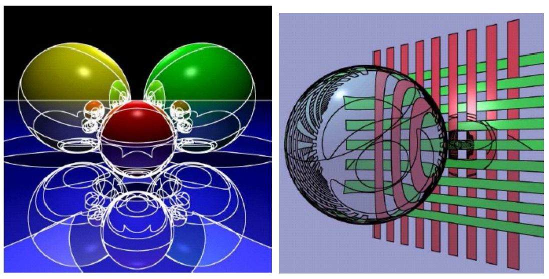


Fig. 6: *(a) A highly reflective scene. (b) Refraction example.*

3. A new approach to understanding visually complex scenes

The term of Non Photorealistic Rendering (NPR) was, a long time ago, only applied for artistic illustrations such as Pen and Ink or watercolor. Many computer graphics researchers are nowadays exploring NPR techniques as an alternative to realistic rendering. More importantly, NPR is now being acknowledged for its ability to communicate the shape and structure of complex scene. These techniques can emphasize specific features, convey material properties and omit extraneous or erroneous information. Therefore, we are going to imitate and take inspiration from these techniques by extracting apparent contours of the real object present in the scene in order to make a better knowledge through realistic rendering.

We define a visually complex scene as a scene containing lights, mirrors and transparent objects. Such a scene is sometimes difficult to understand with a realistic rendering. To face this problem we propose to delimit the objects of the scene by their apparent contour before rendering the image and before adding transparency, reflections and/or refractions. With this technique, the real objects of the scene may be overlaid on the realistic rendering, making the user able to distinguish reality from reflection and refraction effects.

Our approach is divided in two parts:

1. The selective refinement part [PD91] that searches for an initial contour pixel related to each real object presents in the scene.
2. The code direction part that searches for the complete contours.

Both parts use simplified AI heuristic search.

3.1 The selective refinement part

Our goal is to find for each object, one contour point which will be used as a departure point for our code direction method.
First, we divide the image of pixels into a set of macro pixels (8x8 pixels). For each macro pixel, we send rays to the up right (UR), up left (UL) down right (DR) and down left (DL) pixels to detect for each ray the ID of the closest object. We associate each returned ID to its correspondent pixel.

The macro pixels which represent different intersections most contain a contour pixel. They are considered as our useful macro pixels which are subdivided into 4 sub macro pixels. The same process is applied to each sub macro pixel until we obtain a block of 2x2 pixels (see Fig.7). The block of 2x2 pixels that has intersection with different objects, contains certainly at least a one contour point. More we have different intersections in the block, more we have initial contour pixels. To avoid having more than one initial contour pixel for the same object, since we get the first contour pixel of an object, we neglect all other pixels that have the same ID.

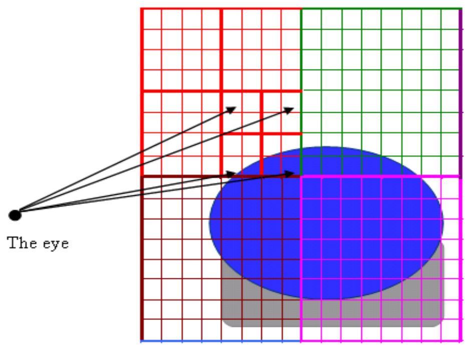


Fig. 7: *The macroPixel that intersects different objects is divided into 4 sub macro pixels.*

3.2 Code direction part

Our code direction algorithm starts with an initial contour pixel and follows, at each time, a certain direction that conducts us to the following contour point. We repeat the same process until we obtain the complete contour of an object.
In other words, this method can be applied by following these 3 steps:

1. Choose the departure point.
2. Choose the initial direction to the second contour point.
3. Choose the following direction to the following contour point

Before talking about the steps of the algorithm, we define first for each pixel, its 8 neighbors. Each pixel in the neighborhood has a previous and a following pixel respecting the order indexed from 0 to 7. For example the neighbor number 1 (coordinate(x-1, y+1)) has the neighbors 0 (coordinate (x, y+1)) and 2 (coordinate(x-1, y)) as its previous and following pixels (see Fig. 8)

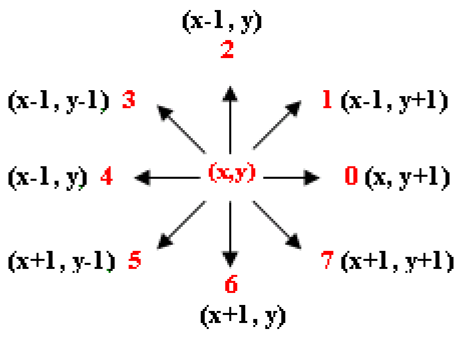


Fig 8: *The coordinates of the 8 neighbors indexed from 0 to 7*

Step 1: Choose the departure point

We start our algorithm with the initial contour pixels obtained by the selective refinement method. More we have initial pixels, obtained by the selective refinement, more we have contours that have to be detected.

Step 2: Choose the initial direction to the second contour point

The second contour pixel should be one of the 8 neighbors of the departure one.
We send a ray to each neighbor. The first one that has the same ID of the departure point will be our current pixel that has to be tested as contour or non contour point.

- The current pixel is considered as a contour point if its previous and following pixels don't have the same ID.
- If the tested pixel wasn't a contour point, we pass to the other neighbors until we get the one that has the same ID of the departure point and apply the same test.
- If all the neighbors were tested and none of them was a contour pixel, we stop the research.

Step 3: Choose the following direction to the following contour point

Since we get the second contour pixel we color it with a special color (red) and apply the same process to find the following direction by considering the obtained contour pixel as a starting one. In order to avoid a return to a chosen contour point, we only test between the 8 neighbors which are not colored yet.
The algorithm will stop when we fall in one of these tow cases:

- We return to the initial departure point (closed contour)
- None of the neighbors of the current pixel is a contour point (opened contour)

4. First results

The method presented in section 3 have been implemented and obtained results allow to conclude that it is possible to visually enhance the understanding of complex scenes which contain reflections refractions and shadows by detecting real object contours.

The rendered image's size is 640 x 480 pixels. The apparent contours are traced in red color in order to overlay the real objects present in the scene. Results are shown in Fig.10 and timing results for the contours computation cost and the rendering cost are given in the Table 1. The images shown in Fig. 10 represent different scene cases.

Scenes 1 and 2 represent the same objects with and without reflections and refractions respectively. We have certainly obtained for both of them the same contour and with the same cost in time. It is du to the fact that our algorithm computes the contour directly from the scene not from its image, apart from the way it will be rendered. Moreover, Scene 1 is not visually good understood because of the presence of the shadow in the floor and its reflection on the sphere. The detection of contours makes the visualization better and helps the observer to distinguish the 3 spheres from, shadows, reflections and refractions.

Scene 3 represents objects which hide others, whereas Scene 4 represents the case of objects in shade where objects intersect each other. In both of them, our algorithm is able to detect visible silhouettes and crease edges.

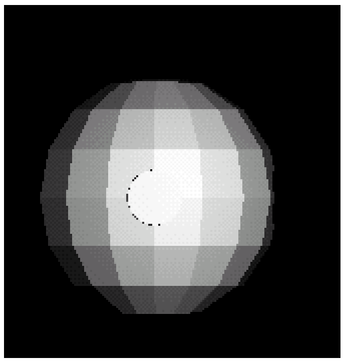
This method can be applied to any scene's model. However, in the case of polygonal mesh, the algorithm detects the contour of each polygon as an object which is not desired for the enhancement of the understanding of the scene (Fig. 9 part (b) and (c)). This problem can be resolved if the polygonal mesh represents only one object or if each object of the scene is modeled separately, by drawing just the silhouette (Fig. 9 part (d) and (e)). It can be done by choosing the contour pixels which are between different objects or between the object and the background of the scene.

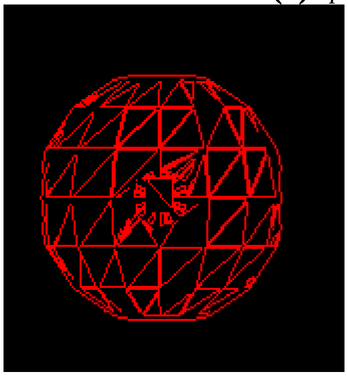
But when the polygonal mesh represents many objects, our algorithm fails to draw the apparent contours of each object because the objects are not defined separately. As it is presented in Fig. 9(a), the scene is modeled by a polygonal mesh with 1056 vertices and 2088 polygons. It is a sphere which contains in the center another object which is a small cube.

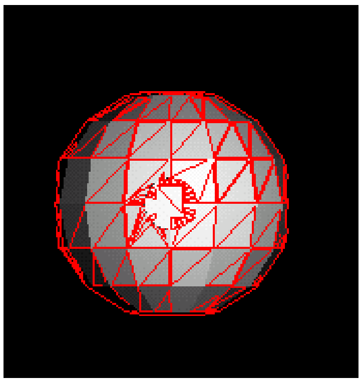
Our algorithm fails to detect the apparent contour of each one separately because we don't know to which polygon belongs each object. It can just detect the silhouette of the sphere.

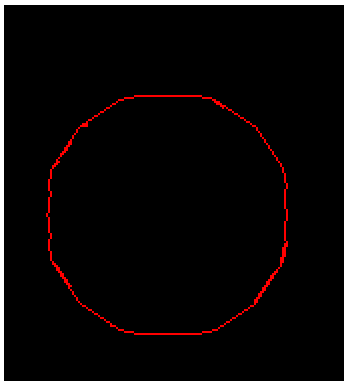
Scene	Time to render the scene (seconds)	Time to detect the contour (seconds)	Scene	Time to render the scene (seconds)	Time to detect the contour (seconds)
Scene 1	16.122	0.15	Scene 3	17.806	0.311
Scene 2	15.172	0.15	Scene 4	17.806	0.1

Table 1: *Time Results to render each scene and detect its contour*


(a) *Sphere Scene*


(b) *Sphere's Apparent Contour*


(c) *Sphere scene with it's apparent Contour*


(d) *Sphere's silhouette*

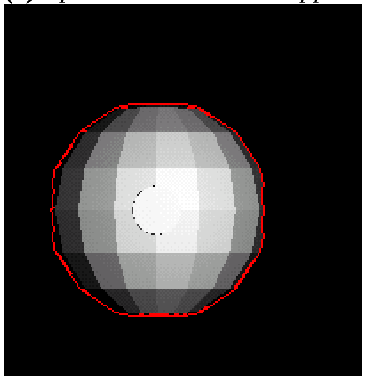

(e) *Sphere scene and it's silhouette*

Fig. 9: *Sphere scene and its apparent contours*

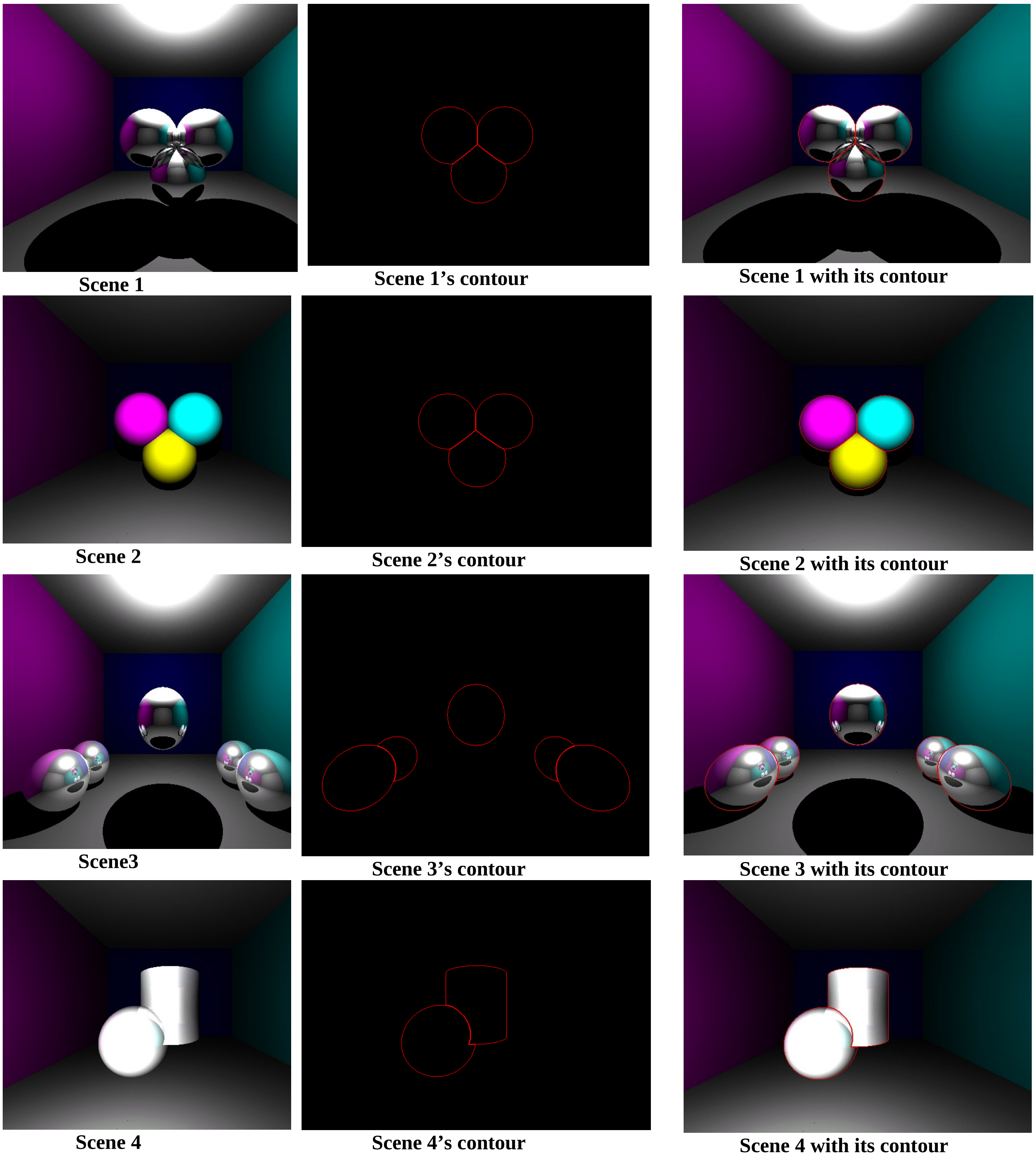


Fig. 10: *Results obtained by using the selective refinement with code direction method*

5. Conclusion and future work

In this paper, after a presentation of the main methods allowing getting a good view of a scene for a better understanding, as well as the main contour extracting methods, we have presented a new method permitting to understand visually complex scenes. This kind of methods can be used to improve computer game programming techniques.

The proposed method combines ray casting and selective refinement and allows extracting the apparent contours of the "real" objects of a scene. These contours, overlaid on the realistic rendering of the scene allow the user to distinguish between parts of the image that correspond to real objects of the scene and the reflections, refractions and

shadows parts. The first obtained results seem convincing.
This method, could be combined with techniques computing a good point of view and highly improve visually complex scene understanding.

6. Acknowledgements

This work has been supported and financed in part with funds of the European project GameTools. It has also been partly supported and financed by the Limousin Region (France). The authors would like to thank all people and organisations which have supported in any manner this project.

7. References

[BL01] BOTTINO A. and LAURENTINI A., “Experimenting with non instructive motion capture in a virtual environment”. *The visual Computer*, Volume 17, Number 1, 2001, pp. 14-29, ISSN 0178-2789.

[BS00] BUCHANAN J.W. and SOUSA M.C., “The Edge Buffer: A Data Structure for Easy Silhouette Rendering,” *Proceedings 1st Int’l Symp. Non-Photorealistic Animation and Rendering*, ACM Press, 2000, pp. 39-42.

[CC88] COLIN C., “A System for Exploring the Universe of Polyhedral Shapes”, *Eurographics’88*, Nice (France), September 1988.

[CM02] CARD D. and MITCHELL J.L., “Non-Photorealistic Rendering with Pixel and Vertex Shaders,” *Vertex and Pixel Shaders Tips and Trick*s, W. Engel, ed., Wordware, 2002.

[CP98] CHUNG Y. C., PARK J. W., SHIN H., and CHOI B.K., “Modeling the surface swept by generalized cutter for NC verification”. *Computer-aided Design,* Volume 30, Number 8, July 1998, pp. 587-594.

[DC06] DECAUDIN P., “Cartoon looking rendering of 3D scenes”, *Research Report INIRIA 2919,* June 1996.

[DG01] DORME G., “Study and implementation of 3D scene understanding techniques”. PhD thesis, University of Limoges (France), June 2001. In French.

[DS00] DEUSSEN O. and STROTHOTTE T., “Computer-Generated Pen and Ink Illustration of Trees”, *Proceedingss Siggraaph 2000, Computer Graphics,* (Proceedings Ann. Conf. Series), Volume 34, ACM Press, 2000, pp. 13-18.

[FP99] FUA P., PLANKERS R., and THALMANN D., “From synthesis to analysis: Fitting human animation models to image data”. *In computer Graphics internationnal’ 99,* June 1999, pp.4, IEEE CS Press. ISBN 0-7695-0185-0.

[GB99] GOOCH B. et al., “Interactive Technical Illustration,” *Proceedings 1999 ACM Symp. Interactive 3D Graphic*s, ACM Press, 1999,pp. 31-38.

[HA90] HERTZMANN A., “Introduction to 3D Non-Photorealistic Rendering: Silhouettes and outlines”, *Non-Photorealistic rendering(Siggraph 99 Course Notes),*S.Green, ed., ACM Press 1999.

[HE01] HAINES E., “Soft planar shadows using plateaus”. *Journal of graphics Tools,* Volume 6, Issue 1, 2001, pp. 19-27.

[HZ00] HERTZMANN A. and ZORIN D., “Illustrating Smooth Surfaces,”*Proceedings Siggraph 00*, *Computer Graphics* (Proceedings Ann. Conf. Series), S.N. Spencer, ed., ACM Press, 2000, pp. 517-526.

[IS03] ISENBERG T. et al., « A Developper’s Guide to silhouette Algorithms for Polygonal Models », *IEEE Computer Graphics and Applications,* Volume 23, Number 4, pp. 28-37, July/August 2003.

[JC01] JOHNSON D. E., COHEN E., “Spatialized normal cone hierarchies”, *Symposium on interactive 3D Graphics,* March 01, pp.129-134. In 2001 ACM, ISBN 1-58113-292-1

[JP05] PLEMENOS D., GRASSET J., JAUBERT B., TAMINE K., “Intelligent visibility-based 3D scene processing techniques for computer games”, *GraphiCon’2005,* Novosibirsk (Russia), June 2005.

[JP06] JAUBERT B., TAMINE K., PLEMENOS D., “Techniques for off-line exploration using a virtual camera”, *International Conference 3IA’2006*, Limoges (France), May 23 – 24, 2006.

[JR02] JENSEN C. G., RED W. E., and PI J., “Tool selection for five axis curvature matched machining”. *Computer-aided Design*, Volume 34, Number 3, March 2002, pp. 251-266, ISSN 0010-4485.

[KK88] KAMADA T., KAWAI S., “A Simple Method for Computing General Position in Displaying Three-dimensional Objects”, *Computer Vision, Graphics and Image Processing*, 41 (1988).

[LG00] LEE W., GU J., and MAGNENAT-THALMANN N., “Generating animable 3D virtual humans from photographs. *Computer Graphics Forum*, August 2000, Volume 19, Number 3, ISSN 1067-7055.

[NG00] NEHAB D. and GATTAS M., “Ray Path Categorization”, *Proceedingss of the Brazilian Symposium on Computer Graphics and Image Processing -SIBGRAPI*, Gramado, Brazil, 2000, pp. 227-234.

[ND03] NIENHAUS M. and DOELLNER J., “Edge Enhancement- An algorithm for real time Non-Photorealistic Rendering”, *Journal* of *WSCG 2003*, Plzen Czech Republic, 2003, Volume 11, Number 1, ISSN 1213-6972.

[OZ06] OLSON M. and ZHANG H., “Silhouette Extraction in Hough Space”, *Computer Graphics Forum (*special issue on *Eurographics 2006)*, Volume 25, Number 3, 2006, pp.273-282.

[PB96] PLEMENOS D., BENAYADA M., “Intelligent display in scene modeling. New techniques to automatically compute good views”, *GraphiCon’96*, Saint Petersburg, July 1996.

[PD91] PLEMENOS D., “A contribution to the study and development of scene modelling, generation and visualisation techniques”. The MultiFormes project, Professorial dissertation, Nantes (France), November 1991.

[PD99] BARRAL P., DORME G., PLEMENOS D., “Visual understanding of a scene by automatic movement of a camera”, *GraphiCon’99*, Moscow (Russia), August 26 - September 3, 1999.

[PD00] BARRAL P., DORME G. PLEMENOS D., “Scene understanding techniques using a virtual camera”, Short paper, *Eurographics’2000*, Interlagen (Switzerland), August 20 - 25, 2000.

[RC99] RASKAR R. and COHEN M., “Image Precision Silhouette Edges,” *Proceedings 1999 ACM Symp. Interactive 3D Graphic*s, S.N. Spencer, ed., ACM Press, 1999, pp. 135-140.11.

[RE92] ROSSIGNAC J.R. and VAN EMMERIK M., “Hidden Contours on a Frame-Buffer,” *Proceedings 7th Eurographics Workshop Computer Graphics Hardware*, Eurographics, 1992, pp. 188-204.

[RP89] RUSTAGI P., “Silhouette Line Display from Shaded Models”,*Iris Univers*e, Fall 1989, p. 42-44.

[RR01] RASKAR R., “Hardware Support for Non-Photorealistic Rendering,”*Proceedings 2001 Siggraph/Eurographics Workshop onGraphics Hardware*, ACM Press, 2001, pp. 41-46

[SF02] SBERT M., FEIXAS M., RIGAU J., CASTRO F., VAZQUEZ P.-P., “Applications of the information theory to computer graphics”, *International Conference 3IA’2002*, Limoges (France), May 14-15, 2002.

[SG05] RUSINKIEWICS S. et al., « Line Drawings from 3D Models”, *International Conference on Computer Graphics and Interactive Techniques, ACM Siggraph 2005 Course 7*, Los Angelos, California, Number 1, July 2005

[SP05] SOKOLOV D., PLEMENOS D., “Viewpoint quality and scene understanding”, *VAST 2005 Eurographics Symposium Proceedingss*, pp. 67-73, Pisa, Italy (2005).

[SP06] SOKOLOV D., PLEMENOS D.,, TAMINE K., “Methods and data structures for virtual world exploration”, *The Visual Computer*, 2006.

[ST90] SAITO T. and TAKAHASHI.T. “Comprehensible Rendering of 3-D Shapes”, *Computer Graphics (SIGGRAPH ’90 Proceedings)*, Volume 24, pages 197–206, August 1990.

[VP03] VAZQUEZ P.-P., “On the selection of good views and its application to computer graphics”. PhD Thesis, Barcelona (Spain), May 26, 2003.

[VS03] VAZQUEZ P.-P., SBERT M., “Automatic indoor scene exploration”, *Proceedingss of the International Conference 3IA’2003*, Limoges (France), May 14-15, 2003.

About the authors

Nancy DANDACHY is a PhD student at the XLIM laboratory of the University of Limoges (France). Her research area is scene understanding using alternative rendering methods.
dandachy@msi.unilim.fr

Dimitri PLEMENOS is an emeritus professor at the XLIM laboratory of the University of Limoges (France). His research area is intelligent techniques in computer graphics, including declarative modeling, intelligent rendering and intelligent virtual world exploration. He is author or co-author of several papers and member of the IPC of many international conferences and journals. Dimitri Plemenos is the organizer and general chair of the 3IA international annual conference on Computer Graphics and Artificial Intelligence.
plemenos@unilim.fr

Bachar EL HASSAN is an assistant professor at the Lebanese University, Faculty of Engineering, 1st branch (Lebanon). His research area is Computer vision, image processing, and wireless networking.
elhassan@ul.edu.lb