Specialized Visualization Systems for Differential Games

V.L.AVERBUKH, S.S.KUMKOV, V.S.PATSKO, O.A.PYKHTEEV, D.A.YURTAEV
Department of System Software, Department of Dynamic Systems
Institute of Mathematics and Mechanics
S.Kovalevskaya str., 16, Ekaterinburg, 620219, Russia
RUSSIA
averbukh@oso.imm.intec.ru 2445@dialup.mplik.ru patsko@cs.imm.intec.ru

Abstract: Visualization environments designed specially for viewing some important objects in differential games are described. The first system is developed for drawing level sets of value function in linear differential games with fixed terminal time. The second one is elaborated for the visualization of the value function graph in time-optimal game problems. Some examples of obtained images are shown.

Key-Words: differential games, numerical simulation, computer graphics, Gouraud shading

1 Introduction

Developing an adequate displaying is very important in visualization of mathematics research results. Here "adequate" means that the image gives both the nature and properties of simulated objects and their internal mental formed in the researcher's mind. Specialized visualization systems have to provide the image conformation so that the user gets the maximally full information on geometry and topology of objects, which are under investigation.

In this paper, a method for working-out such specialized systems is proposed. First of all, the programmer, who develops the system, together with mathematicians (its further customers) considers the gist of the terms to be visualized and their geometrical sense. His duty is to understand how a mathematician sees and apprehends the explored phenomena, how he constructs imaginary the process of solving problem. After that, concrete visualization ways and interactive interface methods are chosen. Often, a number of objects (sometimes multidimensional), also varying in time, must be mapped simultaneously. This demands the usage of photorealistic graphics (for instance, such properties as transparency or illumination) and animation. For apparency of the problem solution process, direct and reverse time regimes are needful. Also sometimes step-by-step mode of the process browsing is useful. So, it is essential to create a dynamic visualization system, where "dynamic" means dynamic imaging of both the object evolving and the process of the mathematics problem solving. Thus, visualization ways depend on concrete problem and vary from one task to another.

On the basis of this methodology, a number of specialized systems of scientific visualization of some optimal control and game problems are realized. They are assigned to describe the results of numerical experiments, to illustrate them, to analyze the calculation model and (sometimes) to debug the computational program. When photorealistic drawing three-dimensional objects is implemented, some modern rendering algorithms (realized, particularly, in OpenGL language) are used.

In this paper, two specialized systems are considered:

1. The system for visualization of value function level sets (maximal stable bridges or Krasovskii bridges) in linear differential games with fixed terminal time.

2. The system for visualization of the graph of the value function in time-optimal game problems.

2 Visualization of Value Function Level Sets

2.1 Visualization Software

In linear differential games with fixed terminal time and terminal payoff function, each level set of the value function is a maximal stable bridge [1]. If the payoff function depends only on two coordinates of the phase vector at the terminal time, then transfer to equivalent game of the second order on phase variable can be applied. So, stable bridges for the equivalent game are built in the three-dimensional space where axes are time and two phase coordinates [2]. Thus, each bridge can be imagine as a "tube", which is determined by a collection of two-dimensional polygonal sections orthogonal to the time axis. Visualization system has to represent individual bridge or a number of them so that the value function structure and its peculiarities could be obvious.

Earlier, usual way to view these tubes was in drawing a number of its section projections in the plane of phase coordinates. But if the quantity of drawn contours grows, the image becomes unclear. Simultaneous view of a number of tubes is much more difficult. So, the problem was to show the object as a surface with its singularities. Here, "singularity" means a smoothness violation, which can born and disappear in the time on the surface of a tube. To get information about the value function, it is needful to view a number of tubes built for distinct meanings of value function. With that, separate tubes and the whole structure in general have to be recognizable.

Developed system allows to view interactively three-dimensional image of a tube or a collection of tubes. The picture can be seen in two regimes. The first is when tubes are represented in the traditional way as a number of contours. But now it is possible to project them not only to phase coordinates plane, but to an arbitrary one. This regime is useful when seeking an appropriate point of view. After it is found, the second view regime can be used. For it, a surface is built by triangulation on base of the collection of separate sections. Then, this surface is drawn using Gouraud shading. The object is illuminated by a dot radiant, which position can be changed by user. Each tube has its own attributes: transparency/opaqueness and color. When a number of tubes are viewed each of them can be set visible/invisible independently.

The first version [3] of the system was written using C language for UNIX-like operational systems with X-Window shell. As the base graphic tool, OpenGL library was used. User interface was implemented with help of Motif. Now, a new version of this system is developed. It is OS Windows 95 oriented. User interface is realized by Delphi interactive system. The latter version was created with participation of students of Ural State University E.A.Shilov and A.I.Zenkov.

Important demand is that algorithms built in such system have to find features interesting for user and to allow him concentrate attention on them. In general, considered system satisfies this demand. So, for more flexible searching non-smoothnesses on the surface of a tube, user can change the threshold angle, separating "smooth" and "non-smooth" vertices, in the range from 0 up to 180 degrees.

The central part of the system discussed is interactive tools for output managing and for changing view regimes. As it was mentioned above, it is possible to change position and orientation of examined object in three-dimensional space, scale of view, location of radiant and cut plane. Also, color and transparency of tubes can be changed.

Here, basic steps of the system work are enumerated:

1. Reading data files;
2. Reconstructing surface of each tube from separate sections;
3. Initializing window system and user interface;
4. Loop of processing window system messages.

Surface reconstruction from separate sections is made using the following algorithm. The aim is to build a system of triangles between each two neighbor sections. Firstly, corner points are detected accordingly to threshold angle set up by user. Starting value of the threshold angle is $\pi \cdot (1 - 1/N)$, where N is the average number of vertices of the tube sections. Further, a connection between these points from considered sections is

established. In other words, corresponding points are connected. After that, points of arcs, laid between these corner points, are also connected so that a system of triangles appears.

If all viewed objects are opaque, then z-buffer algorithm is allowable and the order of drawing primitive elements is not important. Otherwise, if there are some transparent objects, then an additional problem appears. To get an adequate view, it is necessary to draw distant primitives before closer ones using the alpha blending procedure, for which the OpenGL tool permits to define necessary standard functions.

For output of a polygon, the OpenGL language allows to apply two regimes: *flat* and *smooth*. For the latter one, each vertex has its own normal. Then lightness of vertices is calculated on the base of the Lambert reflection law with actual properties of the reflecting surface, location of radiants and direction of the normal. Lightness of all other points of the polygon is computed using bilinear interpolation to two nearest vertices.

For changing the orientation of the object, so-called *arcball controller* algorithm [4] is used. This algorithm allows to rotate the object easily in three-dimensional space with help of two-dimensional coordinate pointer device (for example, by a mouse).

Toolbar of the main window contains the following control elements:

1. for switching the rotation mode;
2. for switching the drag mode when user can move the object;
3. for switching the mode of cutting plane control;
4. for switching the mode of radiant control;
5. for choosing tubes attributes: color, transparency/opaqueness, visibility/invisibility;
6. for choosing the threshold angle;
7. for changing additional marking of corner points;
8. for switching between contour/solid regimes of view;
9. for choosing viewpoint:
    - arbitrary;
    - from the Z axis (as the projection on the XY plane);
    - from the X axis (as the projection on the ZY plane);
    - from the Y axis (as the projection on the XZ plane).

As it was mentioned above, the first version of this software was written using C language for UNIX-like operational systems with X-Window graphic shell. Further, it was modified for OS Windows 95 using Delphi developing environment. A number of new features were added:

- capability to view contour skeleton in solid regime on each tube independently;
- two buttons for quick resetting original viewpoint and location of radiant;
- capability to view coordinate axes.

2.2 Description of Work Session

After loading the program, user has to read data files. Now, data files have no special marks about the problem and values of value function, they were calculated for. So, user must know which files he has to read. When all need data are transferred into computer memory and processed (tube surface are constructed from separate sections), contour view of all loaded surfaces appears (initially all tubes are set up as visible). Then user can switch off tubes, which are not need just at that moment.

One main stage is to find an acceptable viewpoint. For this stage, the first view regime – contour one – is used. All operations with the picture (rotation, zooming, etc.) can be done in the solid regime. But usually, it is carried out too slow when the computer does not equipped by a video card, which has hardware support of OpenGL commands. So, using the contour regime is the optimal way to seek for good viewpoint.

When the desirable aspect is found, the second regime (solid) can be chosen. Now, user has to look for a good placement of the radiant. As the moon craters can be seen only due to cast shadows, some interesting specialties of the tube surface are also visible when an irregularity of lightness exists. At this stage, views from coordinate axes are very convenient for global movements of the radiant. If a number of tubes are viewed, the problem of visibility of their mutual collocation appears. The system suggests two ways for solving

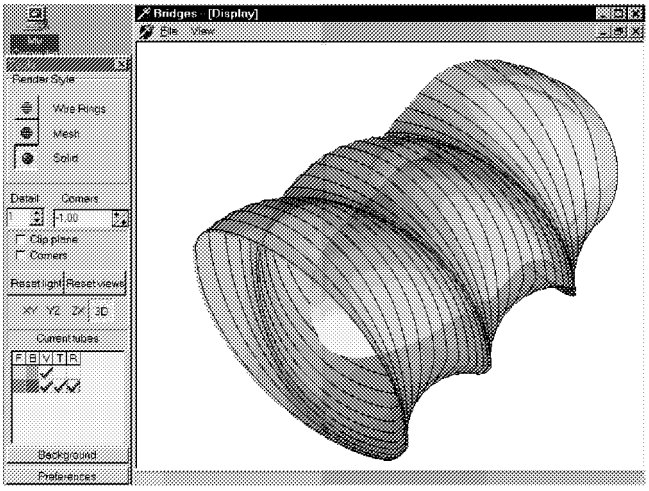

Fig. 1: Two level sets of the value function for the "conflict controlled oscillator" game (screenshot)

this problem. The first one is to make external tubes transparent. This way is acceptable when two or three tubes are drawn; otherwise, internal surfaces become unclear. The second way is to stay all tubes opaque, but to cut them by a plane. In actual version of the software, only a plane parallel to the axes of (X) and (Z) can be used for cutting. However, it is sufficient for majority of pictures.

When a transparent tube is examined, its space configuration is usually lost. It happens due to decreasing of lightness irregularity: the light is coming through the surface, not reflecting back to the observer. To recognize the third dimension of the surface, it is possible to draw the contour skeleton of the tube. The skeleton can be also drawn on opaque tubes if it is necessary for clearness of the view.

Now, development of the system continues accordingly to users' suggestions. For example, it is planned to add such features as storing of good viewpoints, marking some lines (optimal motions of the system, lines of singularity of the value function, and so on) on tube surface, manipulating the second radiant, saving the drawn pictures in some popular graphics format, etc. Development of such software for visualization of value function level sets (stable bridges) in differential games is extremely useful and allows to activate investigations of this subject.

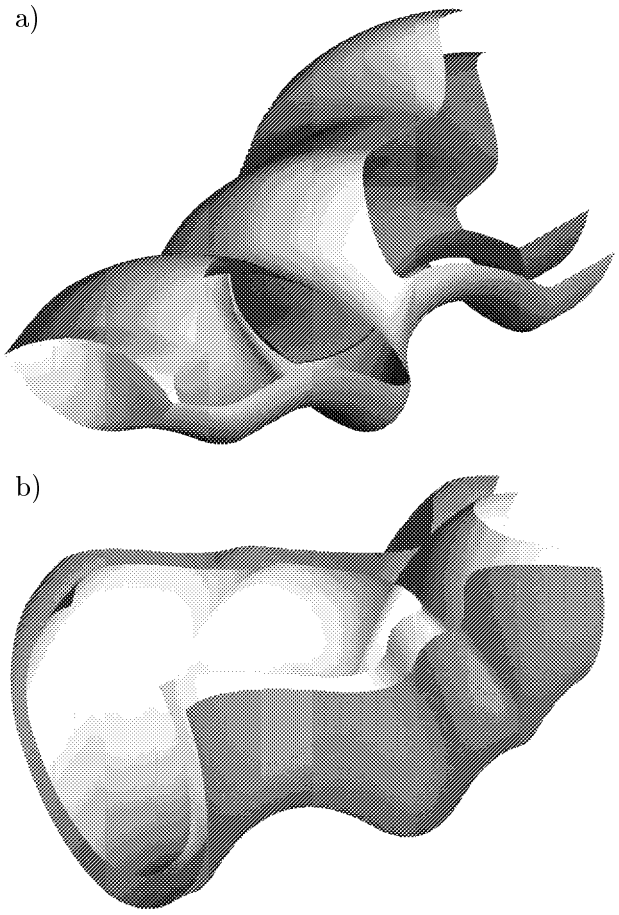

Fig. 2: Two differential games with oscillating dynamics. Two value function level sets cut by a plane are shown for each game

2.3 Examples

In Figure 1, two level sets of value function for the game "conflict controlled oscillator" are drawn. The external tube was made transparent. For reproduction of its space structure, contour skeleton is represented. A screenshot is shown, one can see general view of the visualization environment controls.

Level sets for another variants of parameters in this game are presented in Figures 2 a) and b). The level sets are cut by a plane. The internal structure can be seen.

Figure 3 shows level sets for so-called "drawing-pin" system. These sets are computed for different parameters of the problem. In this way, the dependence of the level set geometry on the problem parameters can be investigated.

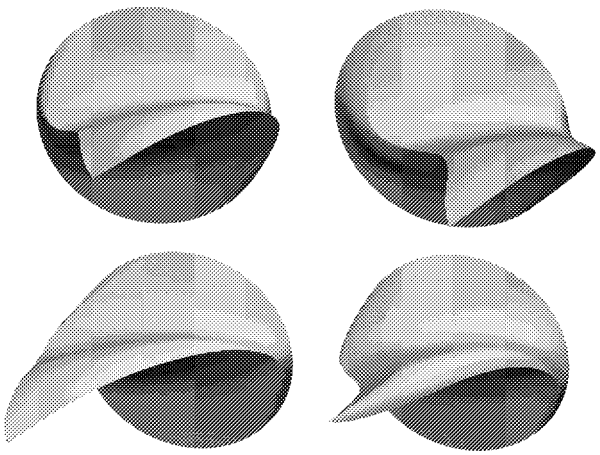

Fig. 3: Evolution of the value function level set depending on the changing of problem parameters

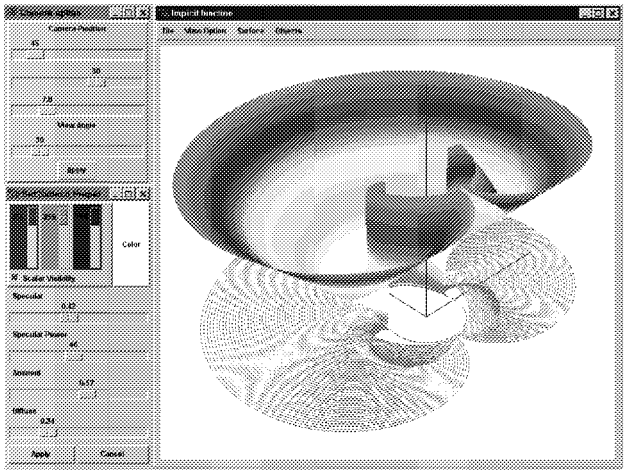

Fig. 4: The graph of the value function for the "homicidal chauffeur" game (screenshot)

3 Drawing of Value Function Graph for Time-Optimal Problems

The second problem is connected to visualization of the graph of the value function in time-optimal games of the second order on the phase variable. The payoff function is the time of approaching the terminal set. In the Institute of Mathematics and Mechanics (Ekaterinburg, Russia), backward algorithms for constructing fronts of level sets of the value function are elaborated. Fronts are computed on a time grid. Each front is a polygonal line defined by its vertices. Neighbor fronts can have different number of connected components. Visualization program using the data on the fronts approximates the graph of the function and shows it. The graph is usually a complicated surface in three-dimensional space.

The idea of specialized algorithm of constructing the graph surface is the following. Some points are detected on the fronts where the line has fracture greater then a given threshold. So, these points divide the fronts to a number of components. Each of the components is quite "smooth". After that, a correspondence between components from two neighbor fronts is established. When all correspondences are determined, the triangulation between related components is made. So, the algorithm allows to detect the places where smoothness or continuity of the value function are violated. The surface constructed is lighted by a radiant and visualized using the Gouraud shading. The program has controls for changing the viewpoint, the direction of sight, the properties of the surface, the location of the radiant, etc. There is a painting regime when the surface color changes with the meaning of the drawing function.

As a test example for the program, the well-known "homicidal chauffeur" game was taken [5]. Numerical algorithm for obtaining fronts for this problem is described in [6].

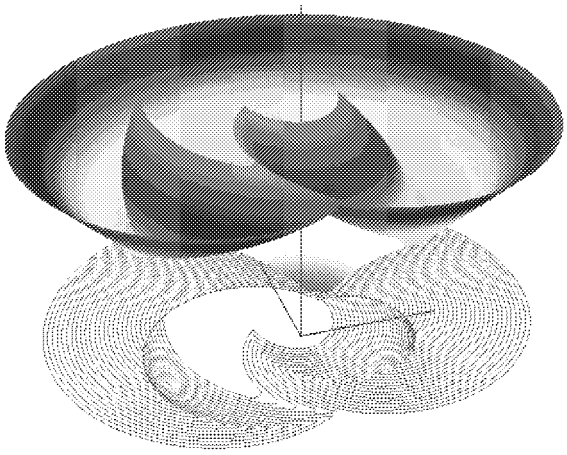

Fig. 5: The graph of the value function for the "homicidal chauffeur" game with non-classic terminal set

Figure 4 shows a screenshot with a picture of the value function graph for the "homicidal chauffeur" game. The terminal set is close to a circle. Places, where the value function grows very quickly (staying continuous), are well seen. Fronts are drawn in the horizontal plane.

In the Figure 5, a graph of the value function for the same game, but with the terminal set taken as a very small circle moved away the origin, is presented.

The collections of the fronts for these two pictures were calculated by V.L.Turova.

4 Conclusion

The main goal of the research is to develop convenient systems of computer graphics for representation of the value function in some classes of differential games. Elaborated systems can be used also for visualizing objects, appearing in other branches of mathematics. So, the first program allows to view any three-dimensional objects if they are tube-like, that is if they are defined by a collection of their parallel sections. The second one can show a graph of the time of front propagation. And it is unimportant which scientific problem generates these fronts.

References:

[1] N.N.Krasovskii and A.I.Subbotin, *Game-Theoretical Control Problems*. Springer-Verlag, New York, 1988.

[2] E.A.Isakova, G.V.Logunova and V.S.Patsko, Computation of stable bridges for linear differential games with fixed time of termination. In: *Algorithms and Programs for Solving Linear Differential Games* (A.I.Subbotin and V.S.Patsko (Eds)), Inst. of Math. and Mech., Sverdlovsk, 1984, pp. 125 – 158 (in Russian).

[3] V.L.Averbukh, D.A.Yurtaev, Approach to elaboration of specialized visualization systems taking the problem of stable bridge construction in linear differential games as example, *Algorithms and software for parallel computations, Vol. 2*, Inst. of Math. and Mech, Ekaterinburg, 1998, pp. 3 – 9 (in Russian).

[4] K.Shoemake, ARCBALL: A user interface for specifying three-dimensional orientation using a mouse. In: *Proceedings of Graphics Interface'92*, Canadian Information Processing Society, Toronto, 1992, pp. 151 – 156.

[5] R.Isaacs, *Differential games*, John Wiley and Sons, New York, 1965.

[6] V.S.Patsko and V.L.Turova, Numerical Study of the Homicidal Chauffeur Game, *Proceedings of the Eighth International Colloquium on Differential Equations, Plovdiv, Bulgaria, 18–23 August, 1997*, D.Bainov (Ed.), VSP, Utrecht, the Netherlands, 1998, pp. 363 – 371.