Visualization and View Simulation Based on Transputers

Klaus Alvermann*
Institute for Flight Mechanics, Braunschweig 3300, Germany

Introduction

IN the Institute for Flight Mechanics at the German Research Establishment, several different types of simulations of flight vehicles are in use. There are realtime simulations based on transputers (FALKE Space Shuttle model, helicopter rotor model) or based on special simulation computers (ground-based simulators for the flying simulator ATTAS and ATTHeS on AD100). Other simulations are calculated off-line on a mainframe computer (flight vehicles under development, e.g., X-31A). All these simulations need a realtime visualization tool to show a three-dimensional view of the flight vehicles and their movement in space.

This tool was developed based on transputers, i.e., parallel processing. The resulting system is built-up modular and can be implemented on a scalable transputer architecture (i.e., different numbers of transputers). In this way the cost and complexity of the system can be adjusted to the application. The hardware (without the monitor) is contained in a small 19-in. rack and can be transported to applications in the field if necessary.

The visualization system can be used to view the realtime movement of the simulated flight vehicles, to show, e.g., the simulated helicopter rotor or to replay and inspect image-by-image a recorded or off-line simulation. The system can be connected to a telemetry station such that a real flight vehicle can be observed as with a camera. By using enough transputers we can also show the flight vehicle moving in a landscape with texturing, fog simulation etc.; viewing from the cockpit of the flight vehicle results in a view simulation.

In this context a "real-time visualization" must fulfill two demands. The image rate must be high enough such that the human eye perceives a smooth movement (16–24 images/s). The time lag (i.e., the time that passes between the arrival of the data in the system and the completion of the appropriate image) may, for some applications (e.g., simply watching the results of a simulation) be up to half a second. However, if there is interaction with the system (e.g., changing parameters of a simulation or "flying" an aeroplane, "man-in-the-loop") a time lag of less than 80 ms is needed.

Structure of the System

The visualization tool is built-up from modules which result from the standard structure of a three-dimensional-graphics system.[1] The modules are implemented as processes (using the language "occam" for transputers). Depending on the number of transputers used, these processes are mapped on different transputers or several of them are grouped on one processor. The processes communicate by way of logical channels which are mapped onto the hardware links of the transputer or onto software links inside one transputer.[2] The system consists of the following modules.

The movement data and, if an object changes shape, object data is collected from the application in the data acquisition module and forwarded to the system. The transformation pipeline consists of several parts: consulting the object data base, possibly a hidden surface algorithm, the transformation and shading of the faces, and clipping. The scanning modules receive the face data resulting from the transformation pipeline, and decompose the faces into lines and pixels which are stored in an image buffer. When an image is completed, the resulting screen lines are collected from the scanning transputers and are moved into the video memory in the graphics module. A sample process model is shown in Fig. 1.

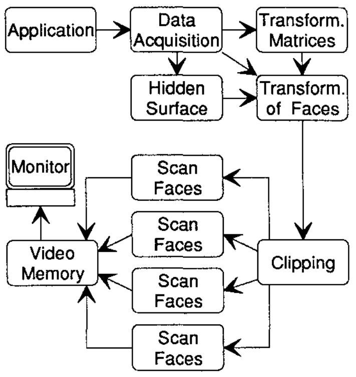

Fig. 1 Processes and communication.

As can be seen, the communication model up to the scanner modules is that of a pipeline. Since most of the calculation power and memory is needed for the scanning process this is done in parallel.

There are several reasons for implementing the visualization on a transputer station instead of on a workstation. The system is small (a 19-in. rack) and may be used with a laptop as the host for applications "in the field." The system is cheap, the hardware for a 20 transputer system costs about $15,000. Most of all the system is scalable: the algorithms and the communication scheme are such that the performance of the system may be increased by simply adding transputers; the system itself is no limit to the performance.

Three-Dimensional Graphics Algorithm

The flight vehicles, or more general, the objects for the system consist of points, lines, and faces. The faces are plane polygons. No curved surface patches are used.

From the multitude of hidden surface algorithms, two were selected which can be implemented easily in a pipeline respectively in parallel.[3] Depending on the visualization task (mostly the number of objects shown) the binary space partitioning (BSP) method or the z-buffer technique is used.

For the BSP method, the relative positions of faces of an object are stored in a binary tree, once off-line.[4] For an image, the faces are painted from "back" to "front" on the screen by traversing the tree dependent on the camera position. In this way a face may be forwarded to the transformation module and processed in the pipeline while the rest of the tree is still traversed, i.e., the transformation process may begin while the ordering process is still working. This makes the BSP method very usable for a pipeline structure. Since the tree is fixed, the relative positions of the faces may not change during the visualization, i.e., the object cannot be deformed. Also, the algorithm gets complicated if two or more objects change their relative positions, i.e., move.

For these reasons the z-buffer technique is used as a second hidden surface algorithm.[1] There are no restrictions on the position of the faces or the number of objects. The objects may be deformed and may intersect. This results in a simple way of defining objects. Since depth calculations and comparisons are on pixel level, much calculation power is needed. For storing the depth values in the z-buffer, much memory is used. Both calculation power and memory is available on transputers. Since the calculations may be performed independently, the process is parallelizable to a high degree.

Received Feb. 25, 1992; revision received May 31, 1992; accepted for publication June 8, 1992. Copyright © 1992 by K. Alvermann. Published by the American Institute of Aeronautics and Astronautics, Inc., with permission.

*Mathematician, German Research Establishment.

Depending on the application, a light model is selected. Normally, one light source at infinity (the "sun") is implemented using flat shading (all pixels of a face have the same color) or Gouroud shading (the color of a pixel is interpolated between the edges of the face). Texturing can be implemented if needed. A bitmap is then projected onto the face resulting in arbitrary patterns and shapes (e.g., lettering, symbols, trees).

System Modules

Data Acquisition

The data acquisition module collects data from the application and transfers it to the transformation pipeline. There are three different types of data: 1) movement data describes the position, orientation, and size of an object in a world system; 2) camera data describes the position and orientation of the camera and the light source; and 3) object data is normally sent once to the three-dimensional graphics at the beginning. If the object is deformed during the visualization (e.g., a helicopter rotor) object data may be continuously sent by the application.

Depending on the application, the module is implemented differently. If the three-dimensional graphics should be used with a transputer-based application, this module is implemented on a transputer connecting the two networks. A simple protocol is used to address the graphics system. If recorded simulation data is used, the module is implemented on a host computer. The user interface can be adapted accordingly. Special transputers with interfaces may be used [e.g., a serial link or pulse code modulation (PCM) interface] if the movement data is available in such a way.

These options may be combined such that the movement data is collected by a transputer, but the camera movement and object data are controlled by a host computer.

Transformation Pipeline

This module consists of several parts which may be executed on different transputers. A transformation matrix is calculated from the movement and camera data for each object. If the BSP method is used, a process produces the correct sequence "from back to front" of faces. If the z-buffer technique is used, the faces are processed without ordering. The rest of the pipeline is "face based," i.e., each module receives a face, processes it, and sends it to the next process. The face edges are transformed into a camera coordinate system. There, shading, depth, and texturing information is assigned to each edge. The face is clipped in depth direction and then transformed to screen coordinates. It is then clipped at the screen window and sent to the scanning module.

Depending on the complexity of the scene, these operations may be implemented on a single transputer or on several. If the z-buffer technique is used, the whole pipeline may be used several times in parallel since the sequence of faces is of no concern.

Scanner Module

The face must now be "drawn" on the screen. All image information is collected in a buffer which is sent to the graphics module upon completion of the image. The face is decomposed into horizontal screen lines ("scan lines"). If it is necessary to access parts of the line, it is further decomposed into single pixels. During this process the information attached to the edges (i.e., color, depth, texturing information, etc.) is interpolated such that it is available at line or pixel level. Color is interpolated directly, depth is transformed in previous processes such that it can be interpolated here. Texture information for objects with almost constant depth can also be interpolated directly without visible errors. Fogging is realized by modulating the color in 24 bit color and by dithering in eight bit color systems.

The lines can be processed independently, therefore, scanning can be parallelized on the line level. The calculation can be halved if one transputer processes the even lines and the other one the odd lines. If, e.g., eight transputers are used, every transputer processes every eighth line. In this way the calculation load is distributed almost evenly independent of the composition of the scene. This could be expanded to 16, 32 or more transputers. If the image is distributed in this way, no antialiasing (by producing a higher resolution image and filtering down) is possible.

The drawback in this parallelization is the communication. The face data has to be distributed from the transformation pipeline to the scanning transputers. When the image is completed, the image data (i.e., parts of the screen) have to be sent to the video memory. Currently, this is done by transputer links. Since a transputer has four hardware links and the video memory can be accessed by two transputers at most, this limits the number of parallel scanning transputers to eight. A communication example with four scanning transputers is shown in Fig. 1, the version with eight parallel transputers for scanning would use three additional buffer transputers between the clipper and the scanner processes.

The limit of eight available links reduces the maximal image rate to $8 \cdot \mathit{linkspeed/width} \cdot \mathit{height}$ (if the whole image has to be transported). This results in a (theoretically) possible image rate (without considering any overhead), see Table 1.

The image rate in this case is the rate at which the image in the video memory is updated, the number of images shown by the monitor is independent from this. The three-dimensional graphics system reaches higher image rates since the whole image does not have to be sent for most applications.

This limit may be overcome in two ways. First, a new generation of transputers (H1 or T9000) will be available in 1992 where the link speed is quadrupled to 10 MByte/s. Second, a special high-speed bus is developed by Parsytec allowing a high number of transputers to write 400 MByte/s into the pixel buffer of the display system.

Graphics Module

This module is implemented on a transputer with a special video interface called graphic display system (GDS). The transputer (and one optional add-on transputer) have direct access to the video memory.

The main task of this module is to move the image lines coming from the scanners to the correct location in the video memory. A transputer can receive data along its four links in parallel (each link has its own link engine), therefore, the image parts are moved into the video memory in parallel.

Table 1 Maximal image rate

Resolution	Maximal image rate, 8 bit color	Maximal image rate, 24 bit color
640 × 480	27 images/s	7 images/s
800 × 600	17 images/s	4 images/s
1024 × 768	11 images/s	3 images/s

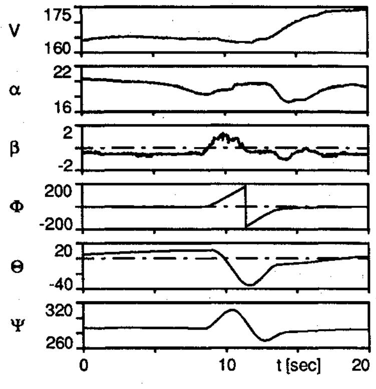

Fig. 2 Flight test data diagrams (V, α, β, Φ, Θ, Ψ).

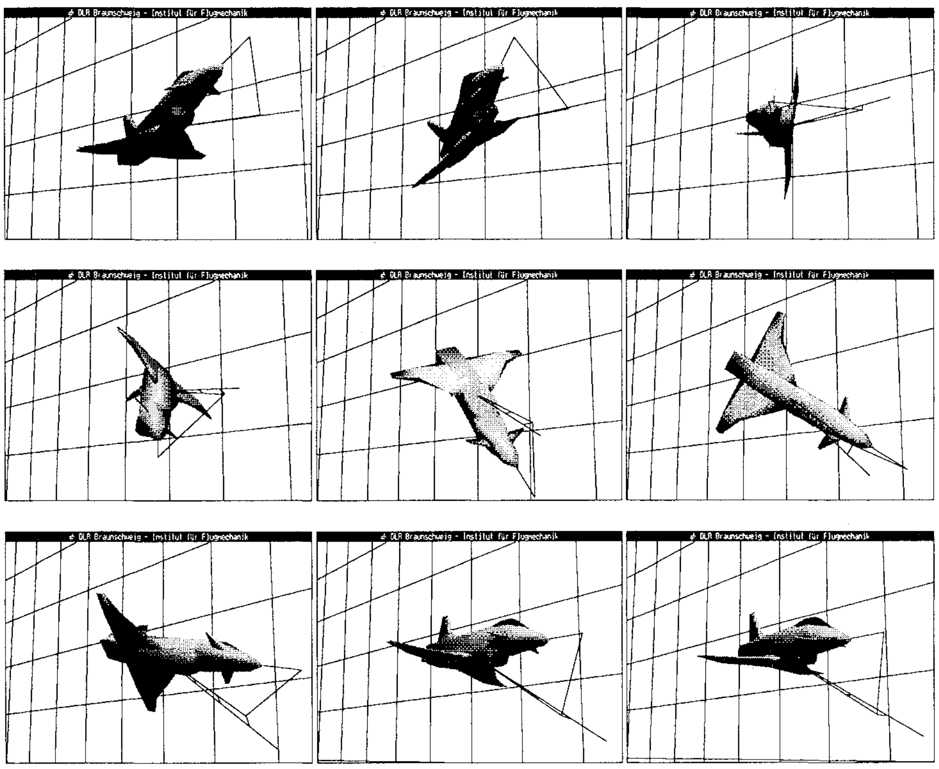
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik
DLR Braunschweig - Institut für Flugmechanik

Fig. 3 X-31A flight test data visualization.

Currently, at most, two transputers (the GDS and an add-on module) can have direct access to the video memory.

Apart from this, the GDS controls the screen flipping, the initial drawing of windows and titles, and displays additional information along with the images.

The screen resolution can be chosen dependent on the monitor available. Usually the standard VGA resolution of 640 × 480 is used. The GDS may be programmed to be used with a TV monitor such that the images may be recorded or viewed with a video projector. The GDS usually uses an eight bit color mode. If needed, the true color 24 bit mode can be implemented. This increases the communication load considerably (see Table 1).

Working Implementations

The main tool used at the Institute is a system consisting of 16 transputers and using the z-buffer technique. It normally works with a standard VGA screen resolution of 640 × 480 and eight bit color mode. Used for visualization, the system shows several flight vehicles in an artificial wire box for orientation. The image rate ranges from 20 up to 40 images/s and the time lag is not noticeable. The system is connected to a personal computer as a host where a menu-driven user interface is available. The system is used to replay simulations and/or real flight data, e.g., of the X-31A campaign or during testing of new flight controllers. The flying simulators ATTAS and ATTHeS are observed by way of telemetry during their flights.

By using an interface transputer the system has been connected to a realtime simulation of the Institute's helicopter rotor model. The rotor may be viewed from the simulated rotating camera mounted on the rotor shaft or from the blade tip to examine changes in parameters more closely.

A small system of eight transputers using the BSP method has been permanently connected to the ground-based simulation of the flying simulators ATTAS and ATTHeS. The screen resolution is 640 × 480 or 1024 × 768, the image rate ranging from 16 to more than 25 images/s.

A similar small system has been used during the FALKE campaign. The onboard computer of a Space Shuttle model has been tested with hardware-in-the-loop. The system was used to view the "flights" during the test, resulting in instant recognition of irregularities when changing parameters.[5]

The main system is also used with a landscape implementing texturing and fog simulation, resulting in a simple view simulation. The image rate may come down to 4 to 10 images/s for complicated scenes. This is mainly due to the above mentioned limit of eight scanning transputers, but will be overcome by the use of the special bus system and the T9000 transputer.

As an example, an image series from X-31A flight test data is shown.[6] Figure 2 presents the usual diagrams of the velocity, the angle of attack and sideslip, and the three Euler angles for a 360-deg roll around the velocity vector. The same data is visualized in Fig. 3 by a series of nine images taken from the realtime graphics system. The X-31A is shown in a grid box for reference purposes. The body-fixed coordinate system attached to the plane is also shown by the system. In this way the angle of attack, the angle of sideslip, and the velocity vector can be derived from the image. Changes in these parameters can be seen directly in Fig. 3. Other flight mechanical parameters, i.e., the thrust vector or the flight path, may be visualized in a similar way.

References

[1]Rogers, D. F., *Procedural Elements for Computer Graphics*, McGraw-Hill, New York, 1985.

[2]Alvermann, K., "Dreidimensionale Darstellung von Objekten nach Bewegungsdaten in Echtzeit (Three Dimensional Graphical Representation of Objects According to Movement Data in Realtime)," DLR Forschungsbericht (Research Report), FB 90-51, 1990.

[3]Sutherland, I. E., Sproull, R. F., and Schumacker, R. A., "A Characterization of Ten Hidden Surface Algorithms," *Computing Surveys*, Vol. 6, No. 1, 1974, pp. 1–55.

[4]Fuchs, H., Kedem, Z. M., and Naylor, B. F., "On Visible Surface Generation by a priori Tree Structures," *ACM Computer Graphics*, Vol. 14, No. 3, 1980, pp. 124–133.

[5]Oertel, C.-H., Alvermann, K., Gandert, R., and Gelhaar, B., "A New Approach to Hardware-in-the-Loop Simulation (FALKE Shuttle)," *AGARD Conference Proceedings No. 473*, Computer Aided System Design and Simulation, Paper 44, Turkey, May 1990, pp. 44-1–44-19.

[6]Rohlf, D., Plaetschke, E., and Weiss, S., "X-31A Model Evaluation and Validation via System Identification," AIAA Atmospheric Flight Mechanics Conf., AIAA Paper 91-2875, New Orleans, LA, Aug. 1991.

Use of Eigenvectors in the Solution of the Flutter Equation

Louw H van Zyl*
AERQTEK, CSIR, Pretoria 0001, South Africa

Nomenclature

$[A]$	=	aerodynamic matrix
c	=	reference chord
g	=	structural damping
$[K]$	=	stiffness matrix
k	=	$\omega c/V$, reduced frequency
$[M]$	=	inertia matrix
p	=	$\gamma k \pm ik$, eigenvalue of the flutter equation
q	=	vector of degrees of freedom
V	=	true air speed
X, Y	=	complex vectors of length n
γ	=	damping coefficient
ρ	=	air density
ω	=	angular frequency

Introduction

THE flutter equation and its solution is central to almost all aeroelasticity problems. Various formulations of the flutter equation exist, each involving some approximations. The *p-k* formulation of Hassig[1] is generally accepted as the one giving the most realistic damping values. However, the determinant iteration solution procedure proposed by him may fail under certain conditions, and some trial and error is usually required to obtain a satisfactory solution. An alternative solution procedure using eigenvectors to assign eigenvalues to modes is suggested which will in most cases substantially reduce the chance of failure.

Although the use of eigenvectors is presented for the *p-k* formulation only, the method can be applied equally well to the *V-g* formulation. The method was originally developed to assign eigenvalues to modes in the solution of the latter formulation, "the fundamental and troublesome problem," according the Desmarais and Bennett.[2] The description of the method given here is directly applicable to the *V-g* formulation.

p-k Method

A simplified form of the *p-k* formulation of Hassig is

$$\left[\frac{V^2}{c^2}[M]p^2 + (1 + ig)[K] - \frac{1}{2}\rho V^2[A(k)]\right] q = \mathbf{0} \qquad (1)$$

Aerodynamic matrices are calculated for a relatively small number of reduced frequencies and are interpolated to other reduced frequencies as required during the solution process. The solution consists of a frequency and damping coefficient, calculated from the eigenvalues of the flutter equation, for each mode at a given set of true air speeds. The eigenvalues determined at previous speeds are used to calculate initial values for a determinant iteration procedure to determine the eigenvalues at the next speed. The solution is started at very low speed where the eigenvalues are close to the structural values determined by either ground vibration testing or finite element analysis.

Two commonly encountered conditions may cause the procedure to fail. If an eigenvalue changes rapidly with increasing speed, the determinant iteration procedure may not converge or converge to the wrong eigenvalue due to too large an error in the initial value of p. Decreasing the speed increment usually solves this problem at the cost of increased computer time. When two eigenvalues are close to each other or even identical, it is sometimes impossible to prevent the procedure from converging to the wrong eigenvalue, even with very small speed increments.

Alternative Procedure

One possible way to avoid the problems of failure to converge or converging to the wrong eigenvalue, is to use an eigenvalue routine which can handle repeated eigenvalues. Such a routine will return a number of eigenvalues equal to the number of modes, and it must still be determined which is the desired eigenvalue. Eigenvectors can be used effectively to select the eigenvalue even if the eigenvalues have changed substantially from the previous speed, justifying the use of a more expensive (in computer time) eigenvalue routine by the possibility of using larger speed increments and having fewer failures.

Implementation

In the present implementation, the solution for one mode at all speeds is found before the solution for the next mode is started. The explanation which follows is for the solution of mode i only. Although only the solution for mode i is valid (because the aerodynamic matrix is a function of reduced frequency), the eigenvalues and eigenvectors corresponding to all the other modes are calculated.

The natural frequency of mode i is used in the first calculation of the reduced frequency at the first speed. The aerodynamic matrix is interpolated and the eigenvalue routine is called to solve the eigenvalues and eigenvectors. The eigenvectors are compared to unit vectors, corresponding to the natural modes, in order to determine which eigenvalue corresponds to mode i. This eigenvalue is used in the next calculation of the reduced frequency and the eigenvalue routine is called again. The process is repeated until the reduced frequency converges. At each successive speed, the converged eigenvalue of the previous speed is used in the first calculation of the reduced frequency. The aerodynamic matrix is interpolated and the eigenvalue routine is called to solve the eigenvalues and eigenvectors. The eigenvectors are compared to the converged eigenvectors of the previous speed to determine which eigenvalue corresponds to mode i. This eigenvalue is used in the next calculation of the reduced frequency.

Eigenvectors are compared as follows: A matrix of scalar products of the converged eigenvectors of the previous speed and the new eigenvectors is calculated. Each column of the matrix corresponds to a new eigenvector and each row to an old eigenvector. The element in row i and column j of the matrix is the scalar product of old eigenvector i and new eigenvector j. The matrix is then searched for the largest element. The corresponding old and new eigenvectors are taken to belong to the same mode. The corresponding row and column is zeroed and the process is repeated until the whole matrix is zero.

Received Feb. 3, 1992; revision received July 6, 1992; accepted for publication Sept. 15, 1992. Copyright © 1992 by the American Institute of Aeronautics and Astronautics, Inc. All rights reserved.

*Engineer, Aeroelasticity Facility, P.O. Box 395.