Published in the Proceedings of the
World Congress on Advanced IT Tools,IFIP '96 IT, Canberra, Sept 96, pp533-539.

MOTION VECTOR PREDICTION IN INTERACTIVE 3D RENDERED VIDEO STREAM

Javed I. Khan

Laboratories of Intelligent and Parallel Systems
Department of Electrical Engineering, University of Hawaii at Manoa
Holmes# 492, 2540 Dole Street, Honolulu, HI-96848, USA
javed@hawaii.edu

ABSTRACT

In this research we describe a scheme for distributed 3D interactive rendering system, which allows high performance rendering at a remote location and the interactive visualization of the resulting rendered images at a general purpose workstation terminal connected through highspeed network. The key to this approach is a video compression scheme of this system which has been designed for the transmission of *interactive 3D rendered video stream* (i3D-RVS). Most video compression techniques heavily rely on the prediction of block motion. This paper particularly presents a fast numerical technique for motion vector prediction for such block coding compression of rendered animation video. This scheme takes advantage of the model movement information for block motion prediction. Unlike regular video, such information can be extracted in 3D visualization both for the case of animated models and for the interactive model navigation. We demonstrate the application of this new technique in the context of a distributed radiation treatment planning system.

Keywords: Remote Rendering, Video Compression, Highspeed Network based Distributed Systems.

1. Introduction:

Interactive visualization of 3D spatial-temporal models is one of the fastest evolving area of computational sciences in recent years. It has applications in numerous areas ranging from tele-medicine, flight-simulator, automobile crash test, virtual-reality, to multi-billion dollar video-game market.

However, in general 3D visualization is extremely computing resource intensive task. Most of the involved procedures (geometry engine, texture mapping, pixel machine) of 3D spatio-temporal visualizations are extremely expensive both interms of computational horsepower as well as fast access memory. In addition, a typical rendering also involves the additional constraint of being interactive in real-time. A typical rendering of 3D models with any reasonable quality quite frequently tantamounts to computational support at the level of real-time supercomputing. Currently, only highly sophisticated and special purpose hardware [Kauf90] is used to meet such stringent demand.

There are some modest advancements in bringing 3D rendering hardware cost down. However, The size and complexity of the newer and more ambitious 3D models are increasing at much faster pace, far outweighing the advancements at the "affordable" rendering hardware capability. The task of 3D rendering can potentially grow almost limitlessly with the quest for "reality". Also, there are further sophistication of the special-effects, which will remain well beyond the capability of home-bound hardware. In this context, we are particularly interested at a quite different approach of disseminating the sophisticated 3D visualization capability.

In this research we are interested in a scheme where the special purpose resource intensive part of the problem can be carried out remotely and interactive visualization support can be provided to fairly general purpose (local) machines, connected with highspeed networks. The result of most rendering is usually a motion video. In our scheme, instead of local rendering, we are concerned with transmission of the rendered video image over highspeed network.

However, the transmission of high-fidelity digital video, itself is a challenging problem. Various schemes have been devised for motion video compression. In this research we are experimenting with video compression techniques, which are particularly suitable for i3D-RVS transmission. A significant part of video encoding depends on the efficiency and predictability of motion vectors [ArSc96, WaGh96, AlTe96]. In this paper we demonstrate specialized prediction scheme for i3D-RVS.

Our specific application context is remote Radiation Treatment Planning (RTP). RTP requires the radiation planner to design the proper beam setting using a CT-scan based 3D model of the patient's interior anatomy, and subsequent simulation of the radiation dose distribution. The enormous sophistication of the human anatomy needs for specialized visualization capabilities makes this problem extremely resource intensive [SuGr78, FaYZ85]. In this research we will demonstrate how such sophisticated rendering capability can be delivered on a fairly inexpensive radiation planner's terminal through a 3D rendered video stream transmission capability though network connection.

2. Overview of Encoding and Visualization Schemes

2.1 Video Encoding Scheme:

Most of the current high performance video encoding schemes (such as MPEG) use I, P and B frames for encoding successive frames. Intra (I) frames require all the block to be transmitted, while Predictive (P) frames require the transmission of the difference information (known as motion vector) between the image-blocks which changes significantly between the frames. Bidirection (B) interpolated frames are computed by interpolation of past and future frame block, and thus require even less physical information to be transmitted. An I frame typically demonstrates compression in the range of 1:20, where P frames demonstrate compression in the range of 1:200. B frames can be compressed even further. The extent of net compression depends on (i) the efficiency of motion vector prediction, (ii) the compactness of motion vector representation, and (iii)

longer time interval between the I frames. Therefor it is very important to predict the motion of the blocks from I frame to intermediate P and B frames both accurately and efficiently.

In 3D interactive visualization the motion of the *rendered image blocks* (RIB) is geometrically correlated with the *spatial movement directives* (SMD). The actual mapping of the spatial movement directives and their effect on the RIB motion largely depends on the 3D frustum defined by the location of the viewer, and the viewing volume. The content of the RIBs depends on the lighting and material property (shape, color, texture, etc.). In our scenario, we assume that SMDs are generally interactively controlled by the viewer. Thus, they are available at the receiver end.

In this particular experiment, we demonstrate how the prediction of block movements can be significantly improved for the cased of 3D rendered video by knowledge of the SMDs. SMDs can be transferred from receiver side to transmitter side using almost negligible amount of bandwidth.

2.2 Model Transformation Process:

Each of the frame in the i3D-RVS is obtained from the 3 dimensional geometric model through a series of transformations. The stages involved in obtaining the screen coordinates from the 3D model coordinates are (i) composite model construction, (ii) re-orientation of the composite model with respect to view point, (iii) perspective division, and finally (iv) viewport transformartion. These steps are defined below.

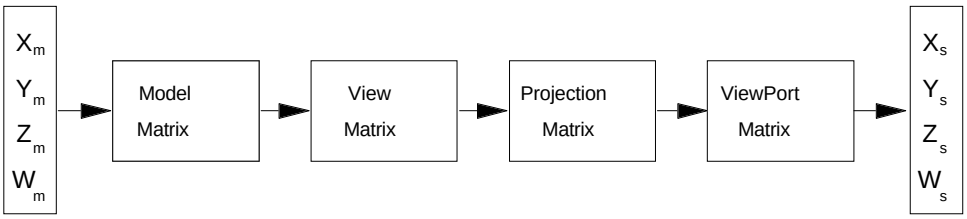

Fig-1 Stages of vertex transformation

The model construction generally involves assembling the component models into the composite coordinate space. Any complex model is composed of many smaller components. Each component is generally defined w.r.t its own co-codinate system. In the model construction stage each of these components are assembled in the composite model space with appropriate translation , scaling, and rotation.

The next step is to choose an appropriate vantage point (or viewing plane) to view the composite model. At this stage the composite model coordinates are re-oriented w.r.t the distance and direction of the viewing plane. Generally, viewing tranformation involves rotation and translation.

The next stage is to generate a pespective view of the objects in the viewing plane, and to clip the viewing volume. A frustrum is defined for this stage. The sides of the frustrum act as cut-planes. Anything outside the Frustrum is clipped off from viewing consideration. The heights and distances of the front and back ends of the frustrum

togather provide the ratio with which the object coordinates should be remapped on the pespective view. The perspective transformation involves scaling the object coordinates with respect to their distance from viewing plane.

The final mapping is performed by transforming the viewplane coordinates onto screen coordinates. A specific screen segment is defined as the viewport where the clipped project plane should be mapped. It generally involves scaling.

As evident from the above analysis, the coordinate transformations in the above stages require only four types of operations namely translation, scaling, rotation and projection. Each of these transformations are actually performed by a sequence of matrix multiplications. The process of obtaining the screen coordinates from model coordinates can be stated by the following equation:

$$X_S^t = L^t \cdot X_M^t \qquad \text{.....(1)}$$

Where, X_M^t is the local model coordinate at time instant T, and X_S^t is the corresponding screen coordinate of this point. individual transforms are represented as a product transform:

$$L^T = (K_n^T \cdot K_{n-1}^T \cdot\cdot K_i^T \cdot\cdot K_3^T K_2^T K_1^T) \qquad \text{.....(2)}$$

Each K_i^T is a 4x4 matrix which represents either a translation, rotation, scaling or projection used during model construction, view adjustment, projection and viewport mapping. generally these matrices are multiplied from right to left to maintain the execution order.

2.3 Effect of Spatial Movements:

The video movement in an 3D model is generated from either spontinuous animation or spital movements introduced by user interaction. Example of former case is where the 2D projection varies with time because of some embedded time varying movement (rotation, scaling, or translation) in the model. Example of the later case is where the user invokes some movement by direct interaction, such as dragging a mouse to move the model, zoom the model etc.

These time varying differentials in the 2D rendered frames can be induces (i) in the original model geometry, (ii) in the model view transformations, or even (iii) in the projection or view port resizing stages.

3. Block Reflection in Time:

Our approach of motion vector prediction can be considered as a process of projecting the block grid in time. In this scheme, we first obtain the block grid of the I frame. Let the block grid with bxb grid locations is represented as a $4 \times b^2$ matrix G_S^T of the following form:

$$G = \begin{bmatrix} x_{11} x_{12} \ldots x_{bb} \\ y_{11} y_{12} \ldots y_{bb} \\ z_{11} z_{12} \ldots z_{bb} \\ w_{11} w_{12} \ldots w_{bb} \end{bmatrix}$$

We compute the motion vector in two steps. First, the block grids at time T are applied inverse transform to obtain the corresponding object points. The movement of these object points are then traced at time (T+t) on the screen. Equation (3) explains the transform:

$$X_S^{T+t} = L^{T+t} . X_M^{T+t} = L^{T+t} . L^{-T} . X_S^T$$
$$G_S^{T+t} = L^{T+t} . L^{-T} . G_S^T \qquad \text{.....(3)}$$

Fig-2 explains the process. The left figure shows the position of the box at time T and its screen reflection. The black square represents a screen block. The right figure shows how this block has moved in time sequence (T+t) and correspoding new screen position of the block. In this animation model, we have assumed a simple rotation of the block along its own z axis. Fig-2 dramatises the effect of rotation. However, in actual video stream the motion is tracked atleast once in every 1/30th of an second. Thus the real movement of the screen block is minuscale. And the predicted blocks remain almost close to the block at last instant.

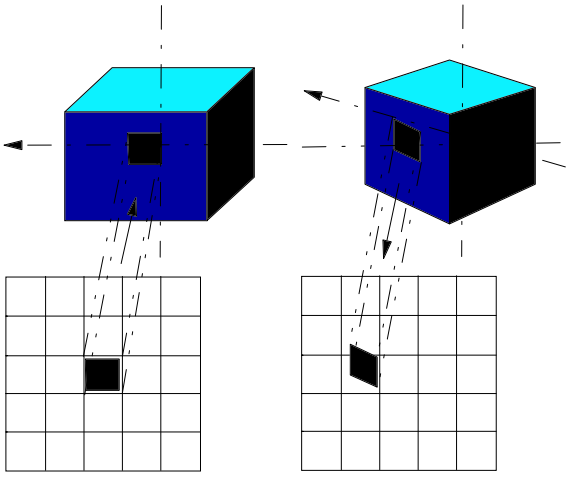

Fig-2 Back Reflection in Time

It is not however required to recompute the entire sequence of L^{T+t} and L^{-T} for Time Reflection. Generally, in the long sequence of transformation in L, only one is involved in the animation. Let, ith matrix be the transformation responsible for animation. If we segment the tranformation sequence L into Animation transform K_i^T, post-animation sequence P^T and pre-animation sequence Q^T, then we can rewrite equation (2) as:

$$L^T = Q^T K_i^T P^T$$

Consequently (3) can be decomposed as:

$$G_S^{T+t} = (Q^{T+t} K_i^{T+t} P^{T+t}) \cdot (Q^T K_i^T P^T)^{-1} G_S^T$$

Since, only the animation matrix changes:

$$G_S^{T+t} = (Q^{T+t} K_i^{T+t} P^{T+t} P^{-T} K_i^{-T} Q^{-T}) \cdot G_S^T$$

$$G_S^{T+t} = (Q^{T+t} K_i^{T+t} K_i^{-T} Q^{-T}) G_S^T \quad .....(4)$$

Thus, only the post animation matrices are required to be pre-computed to support transformation (4). Finally, the motion vector is given by:

$$V = G_S^{T+t} - G_S^T = (Q^{T+t} K_i^{T+t} K_i^{-T} Q^{-T} - 1) G_S^T \quad .....(5)$$

4. Conclusions

In this paper we have presented a geometrical technique for predicting the motion vectors for i3D-RVS. From (5) it is evident that each block perdition requires only about three 4x4 matrix multiplications. If the movement in the model is more complicated requiring lets say *m* moving transformation then *3m* such multiplications will be needed. In actuality, the transform matrices are also sparse. Clearly, this new method computes the prediction with almost negligible cost, compared to current search based prediction methods. This method demonstrates that how motion information of a 3D model movement can be utilized for improving video transmission.

The above scheme has been developed for an experimental distributed treatment planning system for radiation therapy, which requires 3D visualization of human anatomy from an ordinary workstation connected through high performance communication link.

A part of this research has been funded by ACTS and Supercomputing in Remote, Co-operative Medical Triage Support and Radiation Treatment Planning Project of DARPA under research grant DABT 63-93-C-0056.

5. REFERENCES

[ArSc96] Armitano, R. A., & R. W. Schafer, "Motion vector Estimation Using Spatio-Temporal Prediction and Its Application to Video Coding", IS&T/SPIE's Conf. on Digital Video Compression: Algorithms and Technilogies, 2668-32, Jan 28, 1996, San Jose, pp217.

[AlTe96] Altunbasak Y, A. M. Tekalp, "Very Low bit-rate Video Coding Using Object-based Mesh Design and Tracking", IS&T/SPIE's Conf. on Digital Video Compression: Algorithms and Technilogies, 2668-05, Jan 28, 1996, San Jose, pp209.

[WaGh96] Wang, Q., & M Ghanbari, "Motion Compensation for Super High Definition Videos", IS&T/SPIE's Symposium on Electrinic Imaging, 2660-20, Jan 28, 1996, San Jose, pp162.

[SuGr78] Sunguroff A, & Greenberg D., "Computer Generated Images for Medical Applications", Computer Graphics, V.12, no.3, August 1978 p196-202.

[FaYZ85] Farrell, E. J., Yang W. C., & Zappulla R.A, "Animated 3D CT Imaging", IEEE Computer Graphics & Applications, V.5, no.12, December 1985 p26-32.

[Kauf90] Arie Kaufman, "Volume Visualization", IEEE Comp. Society Press, 1990.

Published in the Proceedings of the
World Congress on Advanced IT Tools,IFIP '96 IT, Canberra, Sept 96, pp533-539.

$$X_S^t = L^t \cdot X_M^t$$

$$L^T = (K_n^T \cdot K_{n-1}^T \cdot\cdot K_i^T \cdot\cdot K_3^T K_2^T K_1^T)$$

$$L^T = Q^T K_i^T P^T$$

$$L^{T+t} = K_n^{T+t} \cdot K_{n-1}^{T+t} \cdot\cdot K_i^{T+t} \cdot\cdot K_3^{T+t} K_2^{T+t} K_1^{T+t}$$

$$X_S^{T+t} = L^{T+t}.X_M^{T+t} = L^{T+t}.L^{-T}.X_S^T$$

$$G_S^{T+t} = L^{T+t}.L^{-T}.G_S^T$$

$$G_S^{T+t} = (Q^{T+t} K_i^{T+t} P^{T+t}) \cdot (Q^T K_i^T P^T)^{-1} G_S^T$$

$$G_S^{T+t} = (Q^{T+t} K_i^{T+t} P^{T+t} P^{-T} K_i^{-T} Q^{-T}) \cdot G_S^T$$

$$G_S^{T+t} = (Q^{T+t} K_i^{T+t} K_i^{-T} Q^{-T}) G_S^T$$

$$G = \begin{bmatrix} x_{11} x_{12} \ldots x_{bb} \\ y_{11} y_{12} \ldots y_{bb} \\ z_{11} z_{12} \ldots z_{bb} \\ w_{11} w_{12} \ldots w_{bb} \end{bmatrix}$$

$$4 \times b^2$$

$$V$$

$$V = G_S^{T+t} - G_S^T = (Q^{T+t} K_i^{T+t} K_i^{-T} Q^{-T} - 1) G_S^T$$