Object-Based Image Coding for Cooperative 3D Visualization

Jobst Löffler

Institute for Media Communication - Competence Center for Value-added Solutions
GMD-German National Research Center for Information Technology
D-53754 Sankt Augustin - Germany
Jobst.Loeffler@gmd.de

ABSTRACT

This articel presents a new approach to interaction with 3D objects of virtual models in the context of digital libraries, which was developed as part of the author's ongoing PhD research. Collaborative networked environments support users in their work with shared content. One way to provide the visual feedback a user needs to interact with 3D models is to distribute image streams of rendered objects in a client-server-environment, which is described here. Image streams are coded in an object-based way according to the MPEG-4 international standard. As a result, users in a heterogeneous and error-prone network environment can cooperatively visualize complex 3D models. The use of coded video streams for collaborative visualization offers flexible means of user interaction for digital library applications. Therefore, future work should aim at providing a cooperative 3D visual interface for heterogeneous documents in digital library systems.

Keywords: distributed visualization, object-based image coding, model segmentation, digital libraries

1 INTRODUCTION

Interactive visualization of 3D models in distributed applications is a current subject of research and development. It is potentially applicable to many areas of science, education, medicine and industry for purposes of development, planning, communication and learning. In practice, 3D models are made available on a server as geometric data with additional multimedia information such as text, images, video or audio. Problems in this area arise from the combination of visualization techniques for complex data sets and communication techniques in cooperative environments. The possibility of discussing and cooperatively working with 3D models in digital document collections will become increasingly important and interesting for users, as the accessibility of distributed documents in computer networks improves.

A combination of data of different types together with so-called descriptive meta-data is called a heterogeneous digital document [Felln98]. A collection of such documents, which users can access and work with remotely over a network, is called a digital library. This article describes the approach and prototype realization of an interactive visualization environment for heterogeneous digital documents using object-based video coding. The main topic is visualization of 3D models as part of digital documents. The distributed environment described is composed of a server, which provides the digital document collection in a database application, and of several clients interested in interactive visualization (see Fig. 1). The principal steps of this approach are the following: after selecting a 3D model from a database, the geometric data are preprocessed by the server using a looping renderer module and the resulting image data are then coded into elementary video streams, which are sent to the clients. On the client's side the 2D video objects are re-composed again, showing the complete scene on an interactive display. Users can then interact with the server-side objects through a client-server back-channel. The possibility of interactively changing the structure and content of generic documents enables cooperative visualization, which allows cooperative retrieval and work within distributed digital libraries. Further advantages of this system are its ability to preprocess and pre-structure document components before transmission.

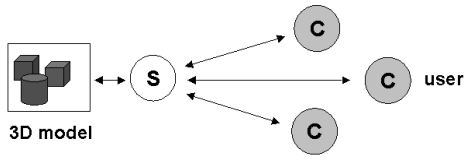

Figure 1: A distributed visualization environment.

The main constraints of distributed visualization in a network environment are the available bandwidth and the local computing power on the client's side in terms of hardware and software [Campa98]. The problem of limited bandwidth can be solved by data compression before transmission. Local computing power in terms of hardware is a minor problem nowadays, whereas a distributed application, demanding the same dedicated client software for visualization in a heterogeneous environment can result in compatibility problems. To meet this requirements, the obvious solution is the use of an international standard for coding of multimedia content such as the MPEG-4 standard.

2 APPROACHES AND RELATED WORK

Various strategies have already been developed to address the mentioned requirements of cooperative visualization applications [Singh99]. To optimize a system in a given network environment, one approach can be more suitable for a certain purpose than another. Depending on network bandwidth, number of clients and the heterogeneity of the environment, a combination of the following concepts can be useful.

2.1 Full Replication and Visual Interaction Approach

Two main concepts for cooperative visualization of 3D models can be distinguished among other approaches. One is the transmission of copies of all model data to every client (Full Replication Approach **FR**), the other is the transmission of preprocessed data, eg. as video streams of the rendered model, together with additional information about the model structure extracted from the scene graph (Model-Vision-Distribution **MVD**).

In the first case, object copies are distributed and client replica are locally visualized. All changes of the 3D model are synchronized and transmitted to all clients. The second approach can be described as distribution of rendered viewing sequences of the model using coded image streams. Client actions are sent back to the server, while synchronization of user access and transformations of the model are carried out by the server. In this case, there exists only one copy of the model and users interact with this model during a working conference.

As an extension of the second concept, the new approach presented here uses object-based image coding (MPEG-4). This approach allows the users of a cooperative visualization application to interact with the 3D objects of models on the server over a back-channel. Clients can therefore work cooperatively with the model without the need to download the whole model (Visual Interaction Approach **VI**). This enables selection and pre-structuring of complex 3D models before transmission of the geometric data and is a flexible means of providing a 3D visual interface for digital library collections.

2.2 Related Work

Several systems for cooperative work with 3D models have been presented in the scientific literature [Singh99]. An example of a full replication approach is the DIVE system, where every client holds a copy of a 3D model and changes are distributed over multi-cast communication in a peer-to-peer architecture [Carls93]. A system following the MVD-approach was described by Haulsen et al. [Hauls98]. In this system a virtual environment on a server is controlled using coded image streams and a back-channel from the clients to the server.

3 TECHNICAL BACKGROUND

Besides visualization, methods for object-based coding of images and techniques for distribution of image streams are important for the described cooperative VI-approach. In this section a short introduction to object-based visual coding according to the international standard MPEG-4 is given, and distribution methods in a client-server system are discussed.

3.1 Object-based Visual Coding

The objective of video coding standards is to enable transmission of image sequences over connections with limited bandwidth. To this end lossless or lossy compression algorithms for digital data are used to reach compression ratios of 2:1 (lossless) up to 100:1 (lossy) with good quality. A common video signal in the D1 format requires a data rate of 166 MBit/s uncompressed, while possible bandwidths in heterogeneous networks vary from 28 - 64 kBit/s (analogous telephone and ISDN) to 10 - 100 MBit/s (Ethernet). Higher bit rates are only achieved by ATM techniques, which are not yet widely available. Coding methods, for example those described in the MPEG standards, take advantage of spatial and temporal correlations in image sequences to reduce redundant information. Algorithms for discrete cosine transform and motion compensated prediction are used for these purposes [LeGal92]. The MPEG video standards use lossy compression of image sequences. The MPEG-1 and -2 standards are block based, which means that the algorithms work on blocks of image pixels. An image is seen as a rectangular composition of pixel blocks.

In an object-based coding context such as the MPEG-4 standard frames are seen as compositions consisting of different scene objects. Each object is assigned to a distinct region defined in a segmental mask. The MPEG-4 video verification model (VM) encodes for each of these video objects (VO) a texture map, a binary shape and motion information [Sikor97]. The video objects of a scene are coded into elementary bitstreams and transmitted as multiplexed MPEG-4 streams. During scene recomposition by the decoder, several 2D objects can be composed with qualitative depth layering in front of a background. The decision about the visibility of overlapping pixels belonging to different objects relates to depth knowledge provided by a VRML-like scene description, which is included in the coded MPEG-4 stream.

The structure of the multiplexed bitstream is object-based and its content is arranged in different hierarchical layers: video session (VS), video object (VO), video object layer (VOL) and video object plane (VOP) (see Fig. 2). The highest layer, the video session, is divided into substructures, so-called video objects. A video object is represented in different spatial and temporal resolutions in the next layer, the video object layer. Finally the video objects planes contain the actual coded image data. The whole scene is represented as a video session in the multiplexed MPEG-4 stream, while each video object is coded into an elementary bitstream.

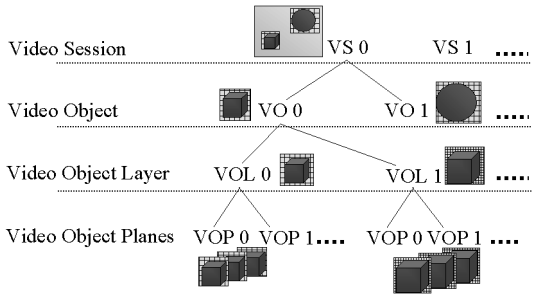

Figure 2: MPEG-4 video bitstream structure.

These elementary bitstreams are decoded at the receiver end and re-composed with the help of the scene description. If an MPEG-4 scene contains both 2D video objects and 3D objects [Doeng97], the decoder will also perform 3D rendering before composition. (see Fig. 3).

3.2 Distribution Methods

An important performance criterion for distributed visualization environments is *latency*. In this context this means the time needed, before a user action becomes visible on all user displays. Therefore, in order to minimize latency it is crucial to choose a suitable communication structure [Carls93].

A client-server architecture is based on an asymmetric model where many clients connect to a small number of servers, often only to a single server. The number of connections is proportional to the number of clients. The 3D data set

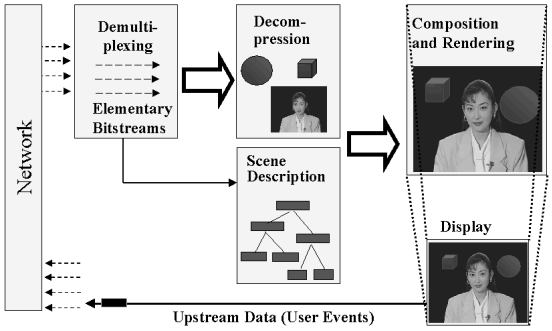

Figure 3: Decoding of MPEG-4 streams: re-composition and rendering.

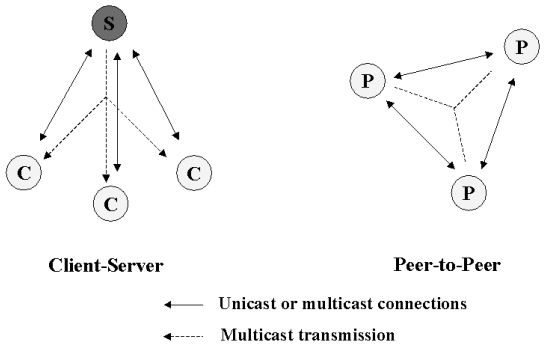

Figure 4: Communication models in a client-server and a peer-to-peer distribution environment.

resides on the server, from which clients request data. Consistency of data can be controlled relatively easily because the 3D model is modified centrally.

On the contrary, communication in a peer-to-peer architecture is distributed equally between all participants. The number of connections increases in a quadratic order compared to the number of peers. When using a peer-to-peer distribution model for full replication, modification of the local model copy is managed according to the messages received from all other participants.

Communication models in a client-server and a peer-to-peer distribution environment are shown in Fig. 4. For collaborative work with virtual 3D models, the following two distribution models fit the requirements best: First, a client-server architecture with multi-cast communication in the server-to-client direction and uni-cast messages in the client-to-server direction. Second, a peer-to-peer model using multi-cast communication between all participants. The first approach will be more successful for distribution of centrally-rendered viewing sequences of 3D models, while the second approach is more suitable for full replication applications.

4 INTERACTION APPROACH USING MPEG-4 IMAGE STREAMS

Interactive visualization of 3D models in a distributed environment can be done by distributing 3D model data (FR), or image streams of the rendered model (VI). Both approaches have advantages and disadvantages under certain conditions, which are determined mainly by the network connectivity and the available computing power either on the server or locally on the client's side. The extended VI-approach will be described in more depth and a conceptual system for cooperative visualization, which was developed by the author, will be introduced.

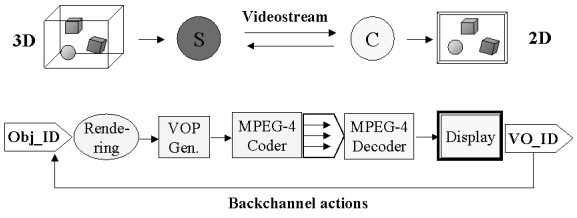

Figure 5: Transmission of object information in the VI-approach.

4.1 Accessing 3D objects over visual object information

Using the VI-approach, image streams of the rendered 3D model are distributed to several clients. The minimum local client configuration consists of a decoder and an interactive display. Using an object-based video codec allows coding of single video objects into separate bitstreams by generating video object planes. These VOPs are referred to the respective object identifier (Obj_ID = VO_ID) of the model. Elementary bitstreams are then multiplexed and transmitted as one MPEG-4 stream (see Fig. 5).

After de-multiplexing and decoding, the video objects planes are superimposed as layers by α-blending at the receiving end. It is possible to locate and reference single objects of the 3D model by comparing the pixel position of the client display with the object shapes during composition. The user can now define 3D actions, which contain an identifier for the respective object and a 3D transformation matrix. These action messages are then send back to the server. Messages containing object identifiers and frame numbers allow synchronization of user actions on the server. In a networked environment, it would not be sufficient to send back client display positions without this additional information, because latency and network errors would prevent synchronization.

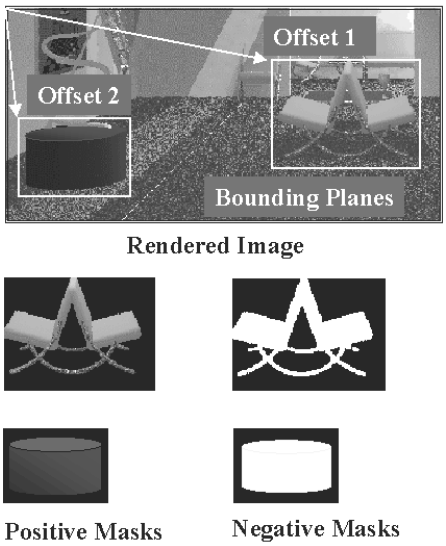

Figure 6: Segmentation of objects.

4.2 Generation of Video Object Streams for 3D Models

Without restriction to a special illumination model and rendering method, the VI-approach described here requires an image generation algorithm which is able to generate shape images for single objects. While the image is rendered, view-dependent texture, contour and bounding plane data for all visible objects are generated. This information can be obtained by segmentation of a positive mask, which contains the texture, and a negative mask, which provides the shape as an α-plane (see Fig. 6). The masks are bounded by a rectangle, the bounding plane, which is estimated during rendering, together with its offset to a reference point in the image. If an object is partly covered, the shape is deformed accordingly. The masks are processed in a suitable format, e.g. rgb format, and attached to the main image.

A video object is described in the MPEG-4 standard through texture, shape and motion information. To generate video object planes of the output of a renderer, geometric rendering data are used, which are defined as the number of the intersected object, the color and the z-buffer value for every pixel [Yun97]. Using the positive mask for texture information, the negative mask for shape information and the bounding plane for motion information, it is a straight forward task to convert between rendering output and video coding input. The generation of video object planes for a virtual model is carried out by this conversion. The MPEG-4 VM demands the YUVS format (Y = luminance, U = blue color component-Y, V = red color component-Y, S = α -plane) as VOP input format. Here, the color components are down-sampled to a quarter of the resolution of the luminance and the shape, because the human eye is more sensitive to differences in brightness than to differences in color.

4.3 Distributed Application Concept

When handling a complex model, users will in most cases work with several objects for a given time period, leaving the other objects unchanged. Taking this into consideration, a useful application concept is one where the user selects an image area for work and then gets the information needed for object-based interaction. This idea is illustrated in Fig. 7A.

Joining a working conference, a user gets a video stream without any additional object information. The view point obtained results from a standard setting belonging to the model or was already selected by a previous user. Then an image area has to be marked with a rectangle which contains the objects the user wants to interact with. Two vertex points of the area are sent back to the server, where VOPs for all objects are generated, whose bounding plane intersects the working area. These VOPs are attached to the main background images of the video stream before coding and are transmitted to the clients. Even if the model is changing dynamically or the transmission is delayed, the user can now pick objects with an input device, since the object identifiers are coded into the bitstream. After sending back the identifier of an object together with a 3D transformation matrix, providing parameters for translation, scaling, rotation etc., the model on the server side will be changed accordingly and visual feedback given back to the clients (see Fig. 7B).

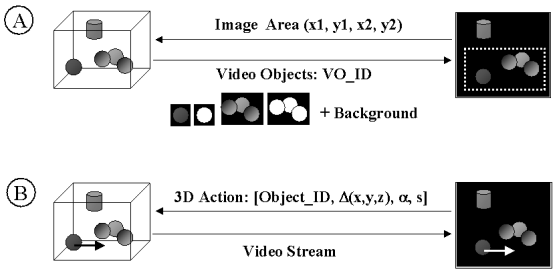

Figure 7: A: Indicating an image area B: Picking and manipulating an object.

On the server side, a list of objects and working areas of all users is generated during rendering and maintained in the following working session. This object list contains information about object numbers, geometric object parameters, object anchors, which consist of the bounding plane plus offset, user identifiers and a status flag for all visible objects. This information is evaluated continuously and is used for generation of needed VOPs and for release of objects.

5 PROTOTYPE REALIZATION

A client-server environment was created to test this new approach. Implementation and performance measurements are presented in this sections.

5.1 Implementation

The main parts of the environment included a distribution server for image streams and action messages, a rendering/coding client for image generation and coding, and finally, decoding/display clients, which represented the user applications and were composed of a decoder and an interactive display (see Fig. 8). The server functionality of the visual interaction approach was provided by combining a distribution server together with a rendering/coding client. The application was controlled via a WWW-Interface, which was put on top of the main modules using a WWW-Server and a HTML-embedded scripting language.

All these components were connected via TCP/IP sockets and run on UNIX machines. For rendering either a ray-tracer (RADIANCE [Ward94]) or a simple OpenGL implementation were used. The object segmentation functionality was tested with

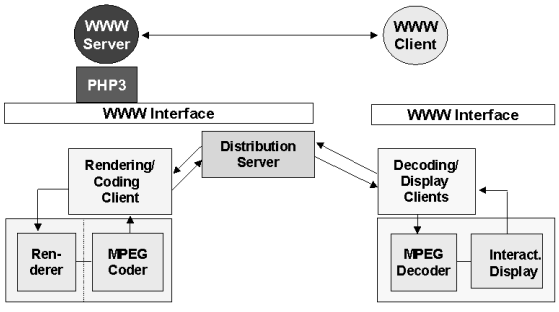

Figure 8: Client-Server test implementation.

a modified version of RADIANCE, since rays offer a clear concept for mask generation with the help of geometric rendering data. The video codec was an implementation of the MPEG-4 Verification Model (VM), which provides the functionality for object-based coding. The interactive display was implemented using OSF/Motif functionality in the modified decoder software. This allowed the input of display positions and simple actions during image composition, and therefore, the identification of single objects in the 3D model. The back-channel was also connected using TCP/IP sockets. So-called action strings, which use a certain syntax for action types and transformation parameters, contained the information about user actions. The transformation of 3D objects was done by changing the internal representation of the scene graph of the renderer according to the information read from the action strings. Testing was performed on a local Ethernet and bit rates varied from 64 kBit/s up to 2MBit/s.

5.2 Results

Tests were carried out in the previously described prototypical environment. The concept of the object-based VI-approach was tested successfully, and client interaction with 3D models on a server was demonstrated. Experimental results were gathered by measuring the execution times of single operations. For the tests, synthetically modeled scenes were used: scene 1, which contained only a few objects and a light source, and scene 2, which was a 3D model of a room containing around 20 objects and 4 light sources. Additional tests using the model of a marble bust scanned at our institute (3D-mesh: 100000 Triangles, 10MB with texture) were performed, which is part of a digital library document about composers of classical music. Image resolution was either QCIF (176x144 pixels) or CIF (352x288 pixels), which are common formats in video coding. Two video objects were coded and transmitted for each model and two clients were connected. Table 1 shows the execution times for each of the test scenes with a transmission bandwidth of 128 kBit/s. Because the system components were executed on several machines simultaneously, total time was not the sum of all time components, but approximately the duration of the most time-consuming process.

The most time-consuming processes were rendering, coding and decoding. Frame rates of 1.3 and 0.3 frames per second were attained respectively with QCIF and CIF resolution for scene 1. Frame rates for scene 2 were 0.3 and 0.1 frames per

Format	Rendering	VOP-Conversion	Coding	Decoding/ Display	Total Time
scene1:					
QCIF	0.64	0.02	0.73	0.42	0.79
CIF	2.20	0.07	2.90	1.30	3.12
scene2:					
QCIF	3.10	0.02	0.94	0.45	3.35
CIF	10.20	0.07	3.82	1.36	10.90
marble bust:					
QCIF	1.20	0.02	0.81	0.44	1.34

Table 1: Time per frame [s], 128 kBit/s

second, respectively. For the sculpture model, a frame rate of 0.75 frames/s was reached in QCIF resolution. Most of the execution time was spent on rendering, while coding did not last considerably longer for the more complex scene. Time needed for VOP conversion was negligible compared to the duration of other operations.

The disproportionately increasing rendering time for more complex scenes causes frame rates to become insufficient. Coding and decoding times depend mainly on the image resolution and the number of coded video objects. Therefore, the coding time does not increase proportionately to the complexity of the 3D model.

6 CONCLUSIONS AND FUTURE WORK

Based on the VI-approach, a system for visual interaction with 3D models in a client-server environment was presented. Using object-based video coding together with this approach, it was possible to interactively manipulate 3D objects using image streams and a back-channel. The components of such a system for collaborative work were defined and suggestions for implementation of the components' functionality were made. The basic concept was tested using a prototypical environment, which was implemented using the MPEG-4 Verification Model.

The next steps will be tests with a real-time rendering engine on the server side and the use of a hardware coder instead of a software implementation to enable interactive frame rates. The integration of the visual interaction approach in a digital library system will be the focus of future work to provide a 3D visual interface for cooperative navigation and interaction with heterogeneous digital documents. The extension of this concept for use with documents which contain text, video, audio and 3D models will be implemented using the standardized features and tools of MPEG-4 and the upcoming standard for description of audio-visual content MPEG-7. A combination of standards for visualization, audio-visual coding and retrieval, and network transmission will lead to a highly flexible and platform-independent toolkit for 3D visualization of heterogeneous distributed documents.

REFERENCES

[Campa98] Campagna,S. et al.: *Enhancing Digital Documents by Including 3D-Models*, Computer & Graphics, Vol.22(6): 655-666, 1998

[Carls93] Carlsson,C., Hagsand,O.: *DIVE- A Platform for Multi-User Virtual Environments*, Computer & Graphics, Vol.7(6): 663-669, 1993

[Doeng97] Doenges,P., Capin,T., Lavagetto,F., Ostermann,J. et al.: *MPEG-4: Audio/video and synthetic graphics/audio for mixed media*, Signal Processing: Image Communication, Vol.9(4): 433-463, 1997

[Felln98] Fellner,D.W., Havemann,S., Müller,G.: *Modelling of and Navigation in complex 3D Documents*, Computer & Graphics, Vol.22(6): 647-653, 1998

[Hauls98] Haulsen,I., Jung,T., Tuchtenhagen,D.: *Remote Control of Virtual Environments Using Image Streams*, To appear in Proc. of Second IMA Conference on Image Processing, Leicester, United Kingdom, 1998

[LeGal92] Le Gall,D.J.: *The MPEG video compression algorithm*, Signal Processing: Image Communication, Vol.4(4): 129-140, 1992

[Sikor97] Sikora,T.: *The MPEG-4 Video Standard Verification Model*, IEEE Transactions on circuits and systems for video technology, Vol.7(1): 19-31, 1997

[Singh99] Singhal,S., Zyda,M.: *Networked virtual environments: design and implementation*, Addison-Wesley, ISBN 0-201-32557-8, 1999

[Ward94] Ward,G.J.: *The RADIANCE lighting simulation and rendering system*, Computer Graphics, SIGGRAPH'94 Proceedings: 459-472, 1994

[Yun97] Yun,H.C., Guenter,B.K., Mersereau,R.M.: *Lossless Compression of Computer- Generated Animation Frames*, ACM Transactions on Graphics, Vol.16(4): 359-396, 1997