ALGORITHMS AND SYSTEMS FOR MODELING MOVING SCENES

V. Michael Bove, Jr.
Media Laboratory, Massachusetts Institute of Technology
Room E15-324, 20 Ames Street, Cambridge MA 02139 USA
vmb@media.mit.edu, http://www.media.mit.edu/~vmb/

ABSTRACT

In this paper I describe the application of machine-vision techniques to video coding in order to create what my research group calls *object-oriented television*, where moving scenes are represented in terms of objects (as recovered by analysis methods). Beyond data compactness, such a representation offers the ability to add new degrees of freedom to content creation and display.

I discuss some of the scene analysis problems (particularly 2-D and 3-D model-fitting and object segmentation) and the algorithmic approaches my group has taken to solve them; suggest computational strategies for compact, powerful, programmable decoding hardware (particularly stream-based computing combined with automatic resource management); and demonstrate some of the applications we have developed.

1 INTRODUCTION

Analysis/synthesis, model-based, or object-oriented video coding methods are those in which moving scenes are represented in terms of objects (as recovered by analysis methods) and scripting information that tells how these objects are to be rendered and processed for display.[1][2] These methods are distinguished from typical current digital video representations in which image data are segmented into regular arrays (*e.g.* blocks of pixels) that are defined *a priori* by the algorithm and do not correspond to scene content.

Most interest in analysis/synthesis video coding has been in the context of very low bit rate video compression, such as in videotelephony. My research group has instead been applying these ideas to higher-rate and -quality television, not only to achieve bitrate reduction but also to investigate the additional abilities such a representation will give the content creator, and the new forms of interactivity and personalization it will enable if transmitted to a computationally powerful receiver.

Features of concern to the content creator might include asset management (intelligent search through video databases and reuse of elements), greater editing abilities (computer-graphics-style modification of video from real scenes), and ability to create responsive content (video that adapts to different display sizes or viewing circumstances). The last of these is also a benefit to the viewer, as would be enhanced interactivity, and dynamic assembly of content on the fly from disparate elements.

Success of a system as described above will require the development or improvement of several critical technologies. Scene analysis methods must be able to integrate observations across space and time in order to form a scene model which can then be decomposed into objects. A data syntax, and a storage and distribution system, must be able to handle video in such a form. Finally, a display must be powerful enough to perform real-time re-creation of images from the received model, and flexible enough to decode a variety of possible representations that might have been determined to be optimal by the encoding process.

2 ANALYSIS

Assuming the capture system is one or more ordinary video cameras (and this does not have to be the case; we have also experimented with active rangefinders for this application [3]), the input to our analysis process is a series of 2-D observations taken from one or more viewpoints at regularly spaced temporal intervals.

I propose that two classes of operation are required to move from these image sequences to an object-based representation: *observation integration* and *object segmentation*. I will discuss these individually, but overall they must be considered not as totally independent steps, but rather as cooperating with each other, possibly inseparably as in some iterative algorithms.

2.1 Observation Integration

Compactness and modifiability of the video data are enabled by the finding of correspondences among observations of a moving scene, whether from one camera at different times, spatially separated simultaneous cameras, or current observations and *a priori* knowledge. Many machine-vision techniques (*e.g.* structure-from-motion, stereopsis, depth-from-focus, motion modeling) can be seen as examples of this concept. This integra-

tion process doesn't have to operate on only the input video frames, but might be aided by other cues such as instrumented cameras that can sense position or acceleration. Such a camera constructed at the MIT Media Laboratory is equipped with miniature accelerometers and gyroscopes.[4] As it is necessary to perform a double integration to convert acceleration to position, and as this process is quite sensitive to noise and limited computational precision, we found it very useful to incorporate the result of video egomotion algorithms along with the information from the mechanical sensors.

The goal of the observation integration process is the production of a merged database from which the original observations can be reconstructed, but also from which other observations might be synthesized. A recent research project from my group illustrates this idea, and also provides an example of what we have called human-supervised scene analysis, in which we avoid having to solve the entire "vision problem" by asking a user to supply a small amount of starting information and perhaps a continuing reasonableness check while the encoding system runs. Such methods, of course, are more appropriate for media that undergo a post-production stage than for live video. This system, developed by Becker, interprets a group of uncalibrated two-dimensional images of a static architectural scene (perhaps a movie set) as a single merged texture-mapped 3-D model by using perspective effects to infer vanishing points, focal length, and camera orientation (Figure 2); here a human can indicate in an approximate sense the relative orientations of the 2-D views one to another, greatly simplifying the algorithm's search space for feature correspondences.[5] We are now using such models as the basis for a video coding algorithm in which a 3-D model is transmitted once, and video taken in that space is represented as (automatically extracted) camera position, heading, and optical characteristics, accompanied by an error signal which accounts for noise, model mismatch, and objects (*e.g.* people) not in the model.

2.2 Object Segmentation

Segmentation of the video data into portions that seem to belong to coherent objects in space increases the modifiability and searchability of the resulting representation (it might also increase the quality of compressed video, if the segmentation criterion is itself the basis for a coder, or if coding-algorithm or bit-allocation decisions are made on an object-by-object basis).

Researchers have used higher-level motion models,[6][7] or color and texture,[8] to segment video into regions or layers. In such cases the segmentation criterion is the model used by the decoder to reconstruct the coded regions, and the objects are selected solely on the basis of bitrate minimization. The resulting regions are thus not necessarily those that would be chosen by a human as the elemental objects, and might not be useful for applications like database search, image editing, or "hot buttons." Further, the definition of regions of interest will vary from application to application. In special effects matting, a single region may correspond to many different objects. In an interactive clothing catalog, it might be desirable to segment a man's shirt from his trousers, while in a sports video with hyperlinks to each athlete's statistical information an entire person would be one object.

Chalom has developed a software package that permits a user implicitly to indicate the segmentation for a video sequence by quickly dragging the cursor across representative points for each desired object for only one video frame (Figure 1).[9] The system then finds regions throughout the video sequence that have corresponding color, texture, and motion. The underlying statistical model is capable of identifying regions even if they have multi-modal distributions in one or more of these parameters (*i.e.* the system can correctly infer that a desired region is either red or yellow but not orange).

3 SCRIPTING

We have been considering a number of different decoder architectures, ranging from a "kit of parts" that could be connected in various ways, to a flexible pipeline.[2] In every case the first step performed upon each set of data is decompression, which reverses transform (or other – *e.g.* wavelet, fractal) coding, quantization, run-length coding, and variable-length coding as appropriate. The last step before display is a z-buffer compositing operation, which allows layering of multiple 2-D objects or hidden-surface removal for rendered 3-D objects.

Assembling the objects into a final image requires that they be accompanied by scripting information. The language might be as simple as a bit field in a data packet header, or as complex as a programming language. Our current interpreted language, ISIS, is Scheme-like, but based on arrays rather than lists to simplify memory management, and with a number of special constructs such as a data type called a "timeline" which can have numeric values inserted at arbitrary real-numbered locations but can then be evaluated at any other real-number index by interpolation.[10] In any event, like PostScript, the language will rarely be edited directly, being rather the result of a scene analysis algorithm or a content authoring tool. In interactive or personalized applications, the script control flow and operational parameters for functional blocks might depend on user actions or state variables (such as user identity, history, or display circumstances). We have implemented it as a real-time application program on the Cheops system (to be described below), and also under UNIX, though a typical mid-range workstation manages to produce only one or two frames per second.

4 HARDWARE AND SOFTWARE

The receiver requirements of large amounts of computation in a small, inexpensive, and easily reprogrammable form has led us to the use of *streams* in multimedia processing. In a stream-based system, one doesn't think of a processing element reading or writing memory; instead a specified mapping turns a multidimensional data array into an ordered one-dimensional sequence that flows through a processing element and then (through another mapping) into a stored (or played, or displayed) destination array. An algorithm can therefore easily be described in terms of a graph structure connecting storage buffers with computational elements – thus process parallelism is expressed explicitly, and execution can be parallelized by a resource manager at run-time, supporting multitasking and hardware scalability. In [11] we explain how a suitable description of the dimensional mapping can also describe data parallelism by making explicit the subsections of larger arrays that can be processed independently if multiple suitable processing elements are available. The stream mechanism is particularly well suited to algorithms where the same operations are applied to a large amount of data, such as those used in audio and video processing.

In implementation, a stream system may consist of specialized processors using hardware to implement the stream communications, or general-purpose processors in a shared memory configuration using memory buffers to perform all communications. Or, ideally, it may combine general-purpose and specialized processors into a heterogeneous system using both stream implementations. Even machines with a single general-purpose processor may utilize the stream mechanism to their advantage. Our first implementation of a stream-based system for video processing was Cheops,[12] which combined a general-purpose CPU with a set of specialized processors connected through a full crosspoint switch to banks of memory equipped with multidimensional DMA controllers. Code execution was parallelized at run time by a resource management daemon.

5 APPLICATIONS

Reference [2] describes experiments in which we have used an object-based representation for video programming which displays differently under different circumstances (for example, on a smaller window or narrower-aspect-ratio display the video may contain more cuts, pans, and close-ups than on a large, wide screen).

We have also handled audio in an object-based fashion: rather than channels corresponding to speakers, sound is represented as a set of localized sources and an acoustical environment in which they are placed. These sound sources are then linked to the visual objects with which they are associated. As directed by the script, perhaps in conjunction with user interaction, the audio is "rendered" for the speakers associated with the video display. Thus, the soundscape changes as the visual viewpoint is varied. Synthesis involves simulation of the early reverberation process – for each speaker, a separate finite-impulse-response filter is calculated and applied to each sound source to give the effect of the reflections from walls – then a nondirectional diffuse reverberation signal is calculated and added representing steady-state room noise for all the sources and their echoes.[13] Reference [14] discusses an implementation in software on a mid-range workstation, which proved able to generate sound for two speakers while running UNIX and several other tasks. Until the analysis is better developed, changes in production methods and linkage with the video analysis can assist the process. In our production experiments each actor has carried a separate wireless microphone, and the video analysis methods provided the locations from which their voices emanate.

6 CONCLUSIONS

As of this writing, several standardization efforts are moving in directions that address some of the issues I have outlined above. Although MPEG-4 is more concerned with working on very low bandwidth channels than with image quality or data flexibility, proposals have included support for object segmentation and multiple representation types. VRML is first and foremost a language for describing virtual environments rather than efficiently coding real ones, but some of the proposed extensions echo related concerns.

Much more research remains to be done in audio and video analysis methods for object-based coding, as well as in the design post-production tools that will enable content producers to take full advantage of the potential responsiveness and flexibility of object-based video and audio. Compression concerns will require developing more efficient representations for certain types of objects, particularly 3-D models.

The author wishes to thank numerous co-workers, in particular Stefan Agamanolis, Shawn Becker, Edmond Chalom, Araz Inguilizian, and John Watlington. The research described in this paper has been supported by the Television of Tomorrow Consortium.

7 REFERENCES

[1] H. G. Musmann, *et al.,* "Object-Oriented Analysis-Synthesis Coding of Moving Objects," *Signal Processing: Image Communication 1*, pp. 117-138, 1989.

[2] V. M. Bove, Jr., "Object-Oriented Television," *SMPTE Journal, 104*, pp. 803-807, Dec. 1995.

[3] V. M. Bove, Jr., "Pictorial Applications for Range Sensing Cameras," in *Proc. SPIE Image Processing, Analysis, Measurement, and Quality, 901*, pp. 10-17, 1988.

[4] C. Verplaetse, "Intertial Motion Estimating Camera," SM Thesis, Massachusetts Institute of Technology,

1996.

[5] S. Becker and V. M. Bove, Jr., "Semiautomatic 3-D Model Extraction from Uncalibrated 2-D Camera Views," *Proc. SPIE Image Synthesis, 2410*, pp. 447-461, 1995.

[6] E. A. Adelson and J. Y. A. Wang, "Representing Moving Images with Layers," *IEEE Transactions on Image Processing 3(5),* pp. 625-638, Sept. 1994.

[7] M. Irani, S. Hsu, and P. Anandan, "Mosaic-Based Video Compression," *Proc. SPIE Digital Video Compression: Algorithms and Technologies 1995, 2419*, pp. 242-253, 1995.

[8] M. Kunt, A. Ikonomopoulos and M. Kocher, "Second-Generation Image-Coding Techniques," *Proc. IEEE 73(4),* pp. 549-574, 1985.

[9] E. Chalom and V. M. Bove, Jr., "Segmentation of an Image Sequence Using Multi-Dimensional Image Attributes," *Proc. ICIP 1996*, (in press).

[10] S. Agamanolis, "High-Level Scripting Environments for Interactive Multimedia Systems," SM Thesis, Massachusetts Institute of Technology, 1996.

[11] J. A. Watlington and V. M. Bove, Jr., "Stream-Based Computing and Future Television," *Proc. 137th SMPTE Technical Conference,* pp. 69-79, 1995.

[12] V. M. Bove, Jr. and J. A. Watlington, "Cheops: A Reconfigurable Data-Flow System for Video Processing," *IEEE Transactions on Circuits and Systems for Video Processing, 5,* pp. 140-149, Apr. 1995.

[13] W. G. Gardner, "The Virtual Acoustic Room," SM Thesis, Massachusetts Institute of Technology, 1992.

[14] A. V. Inguilizian, "Building a Better 'Picture': Synchronized Structured Sound," SM Thesis, Massachusetts Institute of Technology, 1995.

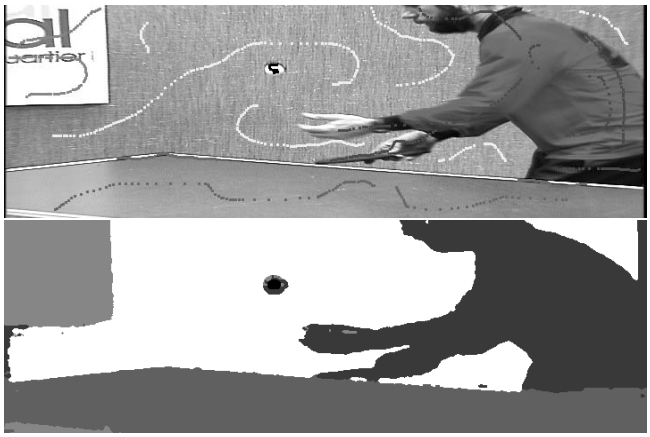

Figure 1: **Object segmentation example:** (Top) Given one frame of a video sequence, a user indicates desired image regions by dragging the cursor across representative points for each. (Bottom) The result is a segmentation of the entire video sequence – in this case into five regions – based on motion, color, texture, and spatial coherence.

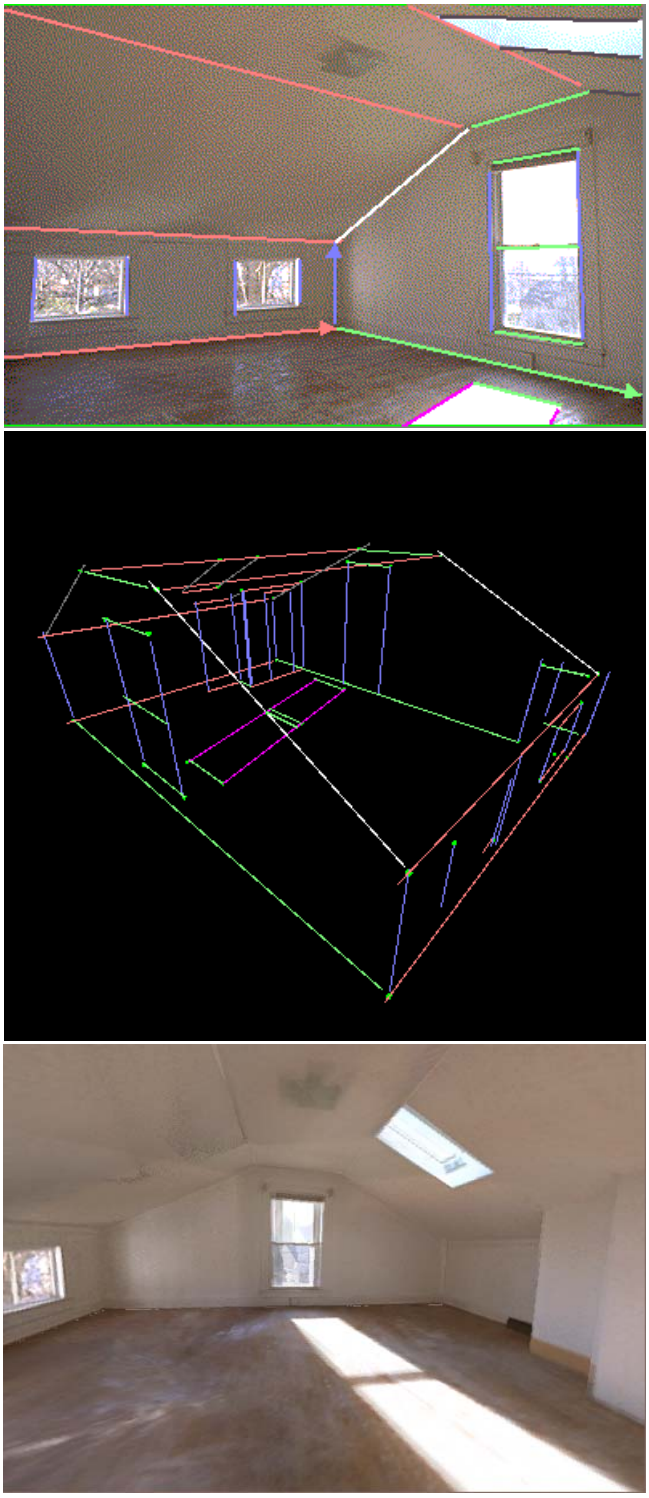

Figure 2: **Observation integration example:** Perspective is used to interpret uncalibrated 2-D images as 3-D, and to merge multiple views into a single 3-D scene model. (Top) One view of a room with extracted edges overlaid. (Center) Wire-frame version of model produced by merging information from five photographs. (Bottom) Rendering of synthetic viewpoint from merged model.