2007

Compact Real-Time Modeling of Seated Humans by Video Sprite Sequence Quantization

Chun Jia

Voicu Popescu
Purdue University, popescu@cs.purdue.edu

Report Number:
07-002

COMPACT REAL-TIME MODELING OF SEATED HUMANS BY VIDEO SPRITE SEQUENCE QUANTIZATION

Chun Jia
Voicu Popescu

Department of Computer Science
Purdue University
West Lafayette, IN 47907

CSD TR #07-002
January 2007

COMPACT REAL-TIME MODELING OF SEATED HUMANS BY VIDEO SPRITE SEQUENCE QUANTIZATION

Chun Jia, Voicu Popescu

Dept. of Computer Science, Purdue University
West Lafayette, IN, USA

ABSTRACT

We propose an image-based method for real-time modeling of seated humans using upper-body video sprites, which is suitable for applications such as teleconferencing and distance learning. A database of representative video sprite sequences is pre-acquired and pre-uploaded to each remote rendering site. At run time, for each input sprite, a closely matching sprite is located in the database. Only the index of the matching sprite is sent to the rendering site which drastically reduces the data rate. Unlike other data compression methods, our method takes advantage of the limited number of significant body positions a participant can assume. Exploiting the redundancy between frames with distant time stamps enables aggressive compression rates with high visual and semantic fidelity.

Index Terms— video sprites, image-based modeling, distributed video communication, compression

1. INTRODUCTION

Distributed applications such as teleconferencing and distance learning imply acquiring, transmitting and rendering believable models of a large number of participants in real time. Current graphics algorithms and their hardware implementation allow rendering complex geometry and color models at interactive rates. However, acquiring and transmitting 3D models of each participant in real time is challenging due to the difficult problems of depth extraction and high bandwidth communication.

An alternative is building 3D models off-line by scanning the participants or by customizing a generic 3D model. At run-time, motion and texture data is captured and sent to the remote site where it is applied to the 3D model [1, 2]. However, the method requires trackers, cameras, and/or markers for robust motion estimation. Moreover, 3D acquisition implies considerable time and equipment expenses, and using generic models implies loss of fidelity. The modeling task is simplified if the participants are modeled in 2D, but the resulting avatars lack realism [3].

The modeling challenge is bypassed by modeling participants with a live video sprite acquired robustly at interactive rates with a single camera [4, 5]. The remaining challenge is the large bandwidth required to transmit the video sprite. The data rate can be reduced by compression techniques that take advantage of intra- and inter-frame coherence. Despite great advances in video codec technology [6, 7], transmitting multiple high-quality video sprites remains challenging. Challenges include the encoding/decoding complexity which requires specialized hardware for real-time performance, and the large bandwidth required in the context of multiple participants, which exceeds the capabilities of commodity connectivity (e.g. DSL, cable modem).

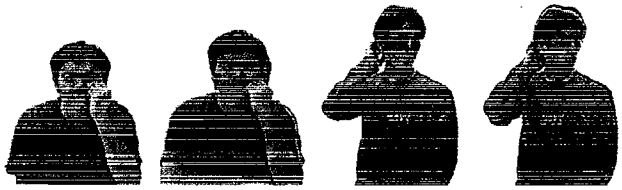

Figure 1. Pairs of input and corresponding database video sprites. The database sprites are superimposed onto a red silhouette of their input sprites to highlight the difference.

In this paper we describe a real time modeling method with a low data rate, making it suitable for distributed applications with limited bandwidth between the acquisition and rendering sites. The method takes advantage of the limited number of significant body positions a participant can assume in applications such as teleconferencing and distance learning. A database of video sprite sequences covering representative body poses is constructed off-line and pre-uploaded to the rendering site. During run time, for each input sprite, a closely matching sprite is located in the database and its index is sent to the rendering site. When a participant has to be rendered in detail, the head region is identified in the input sprite and sent to the rendering site. Since the sprite index can be encoded with as few as 16 bits, the method achieves a drastic reduction in data rate. Note that the required bandwidth does not depend on the resolution of the sprites.

Conventional video compression does not take advantage of redundancy at sequence level since it only detects similarity between consecutive or nearly consecutive frames. Moreover, the various instances of the same body pose have considerable pixel differences which do not

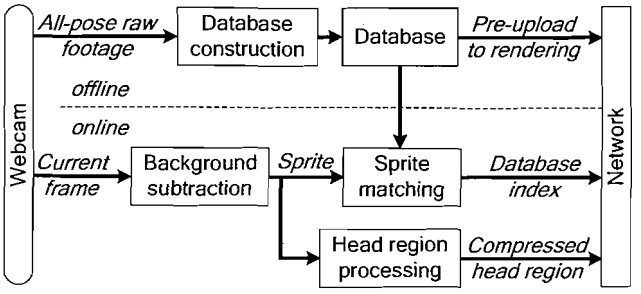

Figure 2: Acquisition module

compress well. However, the pixel-level difference between the two instances has little semantic significance. Consider for example the case of a participant that raises her hand twice during a session. Although the slight difference in motion produces large pixel-level errors between corresponding frames of the two sequences, the second sequence can be replaced using frames from the first sequence with little negative impact on the application.

Our method can also be described as a vector quantization approach where the database is a *codebook* and the sprite sequences are the *codewords*. The codebook enables taking advantage of semantic coherence between frames with distant time stamps, achieving aggressive compression ratios with high visual fidelity as seen in Figures 1, 6, 7, and in our illustrative video [8].

2. SYSTEM OVERVIEW

Our method supports systems with one or more acquisition and one or more rendering modules.

There is one *acquisition module* (Figure 2) for each participant. A database of video sprite sequences is created offline. The participant sits in front of the webcam at the location that will be used during the actual application session and assumes a series of representative body poses under the guidance of pre-recorded audio. Examples of such poses include neutral listening/watching (arms along upper body and hands in lap), raised left or right hand when requesting permission to interject, applause, crossed arms at chest height, crossed arms on desk, and chin resting in left

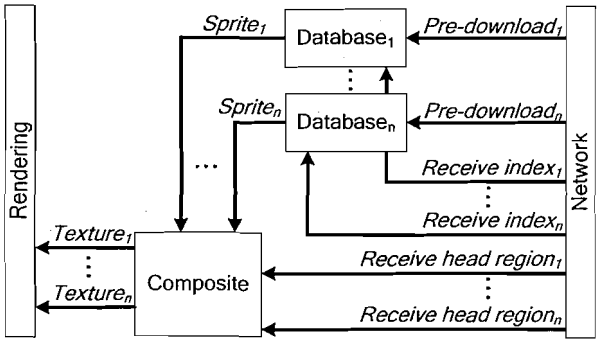

Figure 3: Rendering module

Figure 4: Shape classification. The 3 sprites are classified as no-left and no-right (NL-NR), NL-YR, and YL-YR, respectively.

or right hand. The database is created from the raw footage automatically (Section 3). The database is pre-uploaded to all rendering sites interested in the participant. A database is created in 2-3 minutes. During run time, the current frame is converted into a sprite by background subtraction and then used to find a matching sprite in the database (Section 4). The index of the matching sprite is sent to all interested rendering sites. In the high-fidelity mode, the head region of the current sprite is also transmitted (Section 5).

A *rendering module* (Figure 3) pre-downloads all needed databases. During the application session, the database sprite indices received from the acquisition sites are used to update the video textures that render each participant. In the high fidelity mode, the head region of the database sprite is replaced with the live head region.

3. DATABASE CONSTRUCTION

The raw video footage is processed as follows. In a first step, each frame is transformed into a sprite by separating the foreground (participant) from the background (rest of the scene). Matting is not the focus of this work; we assume that each participant can be placed in front of a favorable background which allows constructing sprites in real time robustly and efficiently using a simple background subtraction algorithm. In a second step, the raw sprite footage is segmented into sequences by taking advantage of the pauses between poses.

In a third step, the segmented 30-50 sprite sequences are arranged at the leaves of a binary tree. The tree is constructed based on sprite sequence shape. The shape of a sprite sequence is a bitmap with a pixel equal to 1 if at least one of the sprites in the sequence covers that pixel, and 0 otherwise. The frame is split into 3 regions using two shoulder lines (blue in Figure 4). The shoulder lines are placed symmetrically at a fixed distance from a central line (red). The central line is found as the column of the tallest uninterrupted foreground vertical segment that starts at the bottom of the frame. The binary tree is constructed recursively from the set of sprite sequences. A sprite sequence is assigned to the left or to the right of the child of an internal node according to whether its shape crosses into the left and/or right regions of the frame. This yields a tree of depth 3 with leaves that store arrays of sprite sequences.

Figure 5 shows a sample tree built from 2,550 raw footage video frames, from which 32 sequences were

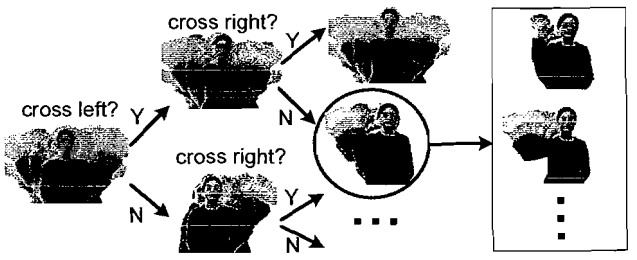

Figure 5: Hierarchical structure of the database.

obtained, totaling 1,592 sprites. The internal nodes do not store data, the aggregate shape images are for illustration purposes. The leaves store 10, 4, 10 and 8 sprite sequences.

4. REAL-TIME SEARCH FOR MATHCHING SPRITE

Given an input sprite, the best matching database sprite is found in 4 steps.

- Down-sample and blur input sprite for efficient and robust color comparisons.
- Find the appropriate tree leaf by descending from the root and classifying the sprite based on shape.
- Trivially reject leaf sprite sequences that do not match the input sprite using sprite sequence shape bitmaps.
- Linearly traverse the remaining leaf sprite sequences to find the best matching sprite.

Two sprites are compared by first aligning them using a translation vector defined by the topmost central line pixels. The aligned sprites are compared in shape and color using a subset of their bounding boxes. The subset is defined by a regular grid of horizontal and vertical pixel segments. We take advantage of coherence in the stream of input sprites by starting the search in the previous sprite sequence. This also alleviates popping artifacts for input sprites close to the neutral pose for which there are several matching sprites.

5. HEAD REGION PROCESSING

In the high fidelity mode, the head region of the input sprite is detected, compressed and sent to the rendering sites where it is composited with the matching database sprite (Figure 6). The head region is an axis aligned bounding box of the head. The top of the head region is defined by the topmost central line pixel. The bottom of the head region is found by walking on the central line downwards until the narrowest horizontal foreground segment is found. The left and right extents of the head region are defined by the widest horizontal segment intersecting the center line.

Figure 6: Database, input, and composited sprite.

The head region is MPEG compressed and sent to the renderer, where it is decompressed and composited with the database sprite. The compositing uses a 2D mapping that matches the database sprite head region to the input sprite head region. The transition from the input head region to the database sprite is performed gradually over a blending area below the head region. Small, erratic frame to frame changes of the head region position are avoided using an average 2D mapping computed over k frames. For 10 fps, a typical k value is 10, which corresponds to 1s.

6. RESULTS AND DISCUSSIONS

We tested our method by modeling 4 different subjects, seen in the images throughout this paper video and in the video [8]. We first discuss database construction, then run-time, and then overall system performance.

A database is constructed at each acquisition site from 30fps 640x480 raw video footage. In preparation for searching, the sprites are down-sampled to 160x120 and blurred, which produces an acquisition site database of 100MB on average. A 2,000 sprite database is constructed in 2-3 minutes (Pentium 4, 3GHz, 2GB). The 640x480 sprite sequences are H.264 [7] encoded into video sequences with an average total size of 6MB. The video sequences are uploaded to the interested rendering sites in about 2 minutes total time (DSL upload speed of 360kbps), for a total database construction time of 5 minutes. At the rendering site, the sprites are recovered from the video sequences and stored as JPEGs that total on average 60MB (2,000 640x480 sprites, 95% compression quality factor).

At run-time, for each input sprite, the acquisition site module searches for the matching database sprite. A brute force linear search on all sprites takes on average 500ms. The binary tree reduces the search time to 250ms, with the speedup being limited by the tree imbalance. Coherence further reduces the average search time to 120ms. The rendering module receives the sprite index, decompresses the JPEG sprite, and updates the texture of the participant. The dominant factor is decompression time, which for our 640x480 sprites is 12ms. Therefore, a rendering site can keep up with about 10 acquisition sites, at 8fps.

The required sprite resolution depends on rendering resolution. Ten 640x480 sprites that are seen at full resolution cover 3Mpixels, which exceeds HDTV resolution. For lower rendering resolution and a large number of participants, the sprite resolution can be reduced. A sprite of 160x120 is decompressed in only 1ms, which means that the rendering module could keep up with 120 acquisition modules. Figure 7 shows the output image of a rendering module with 30 160x120 database sprites (total of

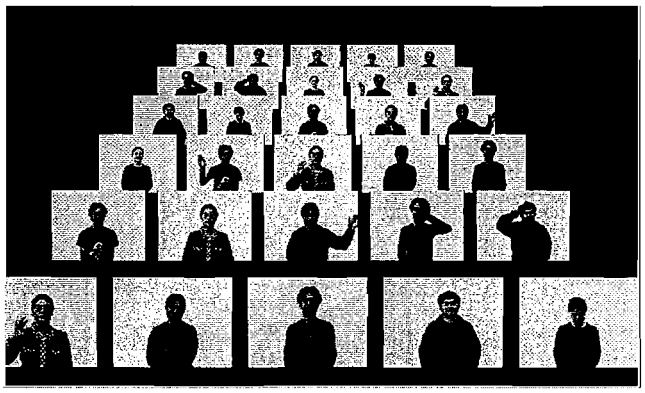

Figure 7: Rendering module output image.

540MB). The image is refreshed at 25fps, which enables posting with little delay individual sprite updates that arrive from various acquisition sites at an average 8fps.

Table 1 gives a comparison between the average data rate achieved by our method and that of state of the art codecs (QuickTime Pro 7 implementation of H.264 [7] and DivX 6.4.0 implementation of MPEG-4 [6]), on an input sequence of 1,000 frames. The high quality setting produced an image quality comparable to our method. The low quality setting produced a far inferior image (Figure 8).

When no head region is transmitted, our method achieves a data rate that is approximately two orders of magnitude smaller than the low quality, low resolution setting of the video codecs. In a distance learning application with 30 remote students that have to be sent to the classroom at 10fps to be seen by the instructor, the required bandwidth for each student is a negligible 0.16 kbps, which leaves ample room for receiving classroom video, and for sending and receiving audio. Since there is no benefit in reducing the communication packet below the Maximum Transmissible Unit which (1,500 bytes for TCP), the index should be added to audio packets. The classroom bandwidth is also insignificant: 30 x 0.16 = 4.8kbps.

When the 320x240 head region is transmitted, it is encoded using the same codecs. Our method outperforms the codecs for the comparable setting on average by a factor

Codec	Frame rate [fps]	Resolution				Our method 640x480	
		640x480		160x120			
		Quality		Quality		Head region	
		Low	High	Low	High	No	Yes
H.264	10	27	338	6	65	0.16	113
	30	62	853	16	152	0.47	267
MPG-4	10	59	330	12	66	0.16	113
	30	131	842	27	151	0.47	269

Table 1: Video data rates in kbps for various settings.

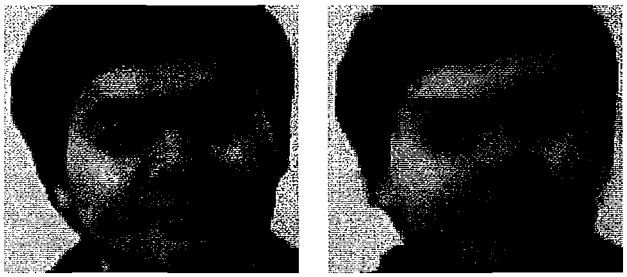

Figure 8: Output image fragments: high & low MPEG-4 quality.

of 3. Although current sprite searching performance at the acquisition module cannot operate at 30fps, the more important head region which is simply cropped out of the input frame can and should be sent to the rendering site. In the context of a teleconferencing application with one active (speaking) participant modeled with head region and several passive participants, an active/passive participant can upload/download the head region using DSL connectivity.

7. CONCLUSIONS

We have presented a method for compact real-time modeling of seated humans that drastically reduces the data rate by replacing the bulk of the input sprite with a two-byte index. As future work we will use the method in the context of an actual distance learning system. In addition to enabling students to attend class remotely with commodity level hardware and connectivity, our method promises to be useful during the validation phase as an analysis tool for the vast amount of video footage that will be collected. The databases and the traces of matched sprite indices are equivalent to output of powerful video abstraction and summarization tools.

8. REFERENCES

[1] F. Remondino and A. Roditakis, "Human Figure Reconstruction and Modeling from Single Image or Monocular Video Sequence," Proceedings of 3DIM, pp. 116-123, 2003.

[2] J. Carranza, C. Theobalt, M. Magnor and H. Seidel, "Free-Viewpoint Video of Human Actors," Proceedings of SIGGRAPH 2003, pp. 569-577, 2003.

[3] C. Wren et al. "Pfinder: Real-Time Tracking of the Human Body," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp 780-785, July 1997.

[4] J. Shade et al. "Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments," Proceedings of SIGGRAPH 1996, pp. 75-82, 1996.

[5] J. Lengyel and J. Snyder, "Rendering with Coherent Layers," Proceedings of SIGGRAPH 1997, pp. 233-242, 1997.

[6] "Coding of Audio-Visual Objects", ISO/IEC 14496-2, 1999.

[7] "Draft ITU-T Recommendation and Final Draft International Standard of Joint Video Specification (ITU-T Rec. H.264/ISO/IEC 14 496-10 AVC)," Joint Video Team (JVC), JVT-G050, 2003.

[8] Paper video URL: http://www.cs.purdue.edu/cgvlab/ICIP07.