Chun Jia · Voicu Popescu

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

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 and 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 assumes during a session. Exploiting redundancy between frames with distant time stamps enables aggressive compression rates with high visual and semantic fidelity.

Keywords Image-based modeling and rendering, video compression, vector quantization.

1 Introduction

Computer graphics enables applications such as teleconferencing, distance learning, and distributed virtual environments by providing virtual venues where participants can meet and interact. Such applications imply acquiring, transmitting and rendering believable models of a large number of participants in real time. Graphics algorithms and their hardware implementation have reached a high level of sophistication and performance, which allows rendering complex geometry and color models at interactive rates. However, acquiring and transmitting a 3D model of each participant in real time is challenging due to difficult problems such as depth extraction and limited networking bandwidth. Several alternative approaches have been investigated.

One approach is based on building a 3D model off-line by scanning the participant or by customizing a generic model. The 3D model is pre-uploaded to the remote site. At run-time, motion and texture data are captured and are sent to the remote site where they are applied to the 3D model of the participant. Acquiring a 3D model of the participant provides realism but comes at the cost of great time and scanning equipment expenses. Generic 3D models are readily available but limit the fidelity of the avatar. Other challenges of the approach include the need for robust real-time motion capture—which requires trackers, multiple cameras, and/or markers—and for a considerable bandwidth to transmit the textures in real time.

The modeling challenge is bypassed if the participants are modeled with a live video sprite [SLS*96, LS97] acquired robustly at interactive rates with a single camera. The remaining challenge is the large bandwidth required to transmit the video sprite. Even when the video sprite is compressed with state-of-the-art video codecs [CAVO99,WS03], transmitting multiple high-quality high-resolution sprites exceeds the capabilities of commodity broad band connectivity (e.g. DSL or cable modem).

In this paper we describe a real time method for modeling seated humans that produces only a low volume of data, making it suitable for distributed applications with limited bandwidth between the acquisition and rendering sites. The method is based on the observation that in applications such as teleconferencing and distance learning, a seated participant typically assumes only a few representative body poses, which makes the stream of video sprites highly redundant at the sequence level. Examples of representative body poses include “neutral” (arms along upper body and hands in lap) when listening/watching, 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 or right hand.

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 compress well. However, the pixel-level difference

Fig. 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.

between instances has little semantic significance. Consider for example the case of a participant that raises her left 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 with the first sequence with little negative impact on the application.

Our modeling method proceeds as follows. First, in an off-line phase, a database of video sprite sequences covering representative body poses is constructed. The sprite sequences are organized in a binary tree based on shape. Typically, a database includes 30-50 sprite sequences totaling 1,000-3,000 frames. The database is pre-uploaded to the rendering site. Then, during run-time, the database is searched at the acquisition site for the sprite that best matches the current input sprite, and the index of the sprite found is sent to the rendering site. Since the index can be encoded with as little as 16 bits, this achieves a drastic reduction of the volume of data transferred between the acquisition and rendering sites. Note that the required bandwidth does not depend on the resolution of the video sprites.

In Figure 1, input sprites are matched to the corresponding database sprites. The participant is rendered with a sprite that is not identical but close to the input sprite (see the red silhouette or the different position of the hands for the participant wearing the t-shirt). The rendered sprite has the original 640x480 resolution and is jpeg compressed with a quality factor of 95%, yet the required bandwidth is only 16 bits per frame.

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, 7, 8.

For active (i.e. speaking) participants, the head region of the input sprites is detected, compressed, and sent along with the database index to the rendering site where it is composited with the database sprite. Since the head region is considerably smaller than the entire sprite, and since the head region is tracked which alleviates the pixel differences between consecutive frames, transmitting the head region requires little bandwidth.

The remainder of the paper is organized as follows. The next section reviews prior work. Section 3 gives an overview of the system. Sections 4 and 5 describe the database construction and the database searching algorithms. Section 6 describes the special processing for the head region of active participants. Section 7 presents and discusses results. Section 8 concludes the paper and traces possible directions for future work.

2 Prior work

Many techniques exist for real-time modeling of humans. We group prior work in 3D, 2D, and image-based modeling approaches.

2.1 Three-dimensional modeling

The 3D model based approach acquires 3D models in real time using scanning techniques. However inexpensive and robust solutions are not yet available. In the context of distributed applications an additional challenge is the resulting high data rate. One attempt to reduce the data rate is based on progressively encoding and sending the texture data multiplexed with the geometry data [YLK04], but, even so, handling many participants in the context of commodity connectivity remains challenging.

Several systems have been proposed based on a priori 3D models. The 3D avatar is animated in synch with the real person it models. In addition to bypassing the problem of real time 3D modeling, the approach has the advantage of low data rate since only the motion parameter values need to be transmitted. However, the approach suffers from the lack of realism of the rendered avatar and from the difficulty of capturing motion. One possibility is to use specialized motion capture hardware, but such hardware is expensive and inconvenient. Another possibility is to scan the actual user with a laser range finder and to use the surface model in conjunction with a human skeleton model to infer the motion data from video frames. Single camera [RR03] and multi-camera [CTMS03, GD96] systems recover the user motion by matching the contour detected in the video frame(s) to the contour of projections of the 3D model. User motion detection is improved by incorporating pixel-level motion field data into the search [TCM03], but the improvement comes at the cost of a substantial performance decline, making the method unsuitable for real-time applications.

2.2 Two-dimensional modeling

An alternative approach uses a simpler 2D model which bypasses the need for complex 3D modeling and simplifies motion estimation. For many applications, a participant is rendered only from a limited range of viewpoints, case in which a 2D model suffices. The Pfinder system [WADP97] employs a statistical 2D blob model which enables tracking people and their actions in real time. However, the system does not provide rendering support as it does not develop an explicit 2D model to which to apply the estimated motion at the rendering site. A system complete with rendering support [TMS02] tracks body parts (head, hands, and feet) using a generalized Voronoi decomposition and builds a visual-hull-like [Lau94] layered model. The disadvantages of the system include lack of efficiency and 2D model inconsistencies when tracking individual body parts fails.

2.3 Image-based modeling

Image-based techniques abandon geometric modeling in favor of a collection of images. A simple but powerful image-based model is a sprite, which represents the object of interest with an alpha-matted billboard [GBM93, SLS*96, LS97, TK96]. Sprites are constructed from images or video frames by separating the object of interest (foreground) from the rest of the scene (background). The difficulty of sprite construction depends on the complexity of the scene [LCL*97, GBS00]. Handling scenes with

dynamic, low-contrasting backgrounds in real time remains an open problem.

The data rate of a video sprite can be reduced by regular video compression, such as MPEG-4 [CAVO99] and H.264 [WS03], which take advantage of intra- and inter-frame coherence. However, high compression ratios can only be achieved at a significant loss in quality. Moreover, the compressed data rate depends on the scene complexity and increases with resolution. Higher compression ratios are achieved by employing 3D or 2D models within the video encoder [TD00, Pea95, Aiz95]. Improved compression without the use of an explicit 2D or 3D model was achieved by learning an image-based model constructed in an appearance space defined by principal component analysis [HSS05]. In spite of these improvements, delivering high quality video to and from a large number of participants requires bandwidths not presently available.

A solution of compromise is to favor an active participant who is allowed to consume a larger quota of the available bandwidth, while the other participants are modeled with lower-resolution aggressively compressed video, and without audio [SMSP05]. The system employs a text chat window that allows the low resolution participants to be involved using the keyboard. The system has the merit of attempting a priority-based distribution of available resources. However, the low quality video and the distracting chat window reduce the effectiveness of the system.

In some applications, the dynamic scene to be modeled produces only a small set of distinct images, where distinct means that the difference between the two images is relevant in the context of the application. For such applications, it is possible to pre-acquire all distinct images and to query the database in real time. The approach is similar in essence to compression schemes based on quantization, which have been applied to large image database compression such as light fields [LH96], except that the quantization is not based on occurrence frequency but rather on application specific knowledge. One example is the synthesis of speech video using a database of video sequences corresponding to phonemes which is queried based on live audio [BCS97]. The phoneme video database could be used in applications such as distance learning and teleconferencing to model the speaking participant, however, a correctly animated mouth is not sufficient; the correct facial expression is also needed.

A recent system [CG00] models a talking head by pre-recording a short video of a talking person and by constructing a database of layers of different facial parts. In order to extract the facial parts robustly the system requires strictly controlled lighting, restricted head motion, and occasional manual annotation of facial parts. A similar approach [WLC*04] builds a database of layers of facial parts incrementally, at run time. The database is extended when a significant difference to any pre-existing frames is detected. The system is able to achieve low bit rate coding of face video. However, the face decomposition requires a frontal view of the head which limits the range of allowable head motions. Moreover, the error metric used requires a direct match between previous and input frames, which leads to a rapid growth of the database and a decrease in performance.

Our method falls in the image-based modeling and vector quantization category. In contrast to the prior methods discussed above, our method focuses on upper-body modeling as opposed to face modeling, and it does not require a precise direct matching between the live and recorded frame. We are finding the database frame that is semantically similar to the input frame, which reduces the number of reference frames in the database. This enables the use of a compact database without limiting the range of motions of the seated human that is modeled.

3 System overview

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

There is one instance of the acquisition module (Figure 2) for each participant. A database of video sprite sequences is created offline. A 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. The database is created from the raw footage automatically, in 2-3 minutes, as described in Section 4. The database is pre-uploaded to all rendering sites interested in the participant. A new database is created before each application session since the database depends on the color of the clothes the participant wears and on the ambient lighting conditions.

During run time, the current frame is converted into a sprite by background subtraction and then used to find a

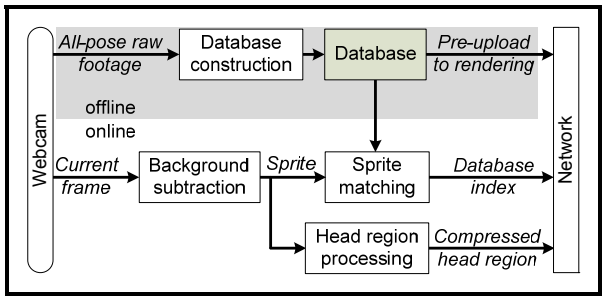

Fig. 2 Acquisition module.

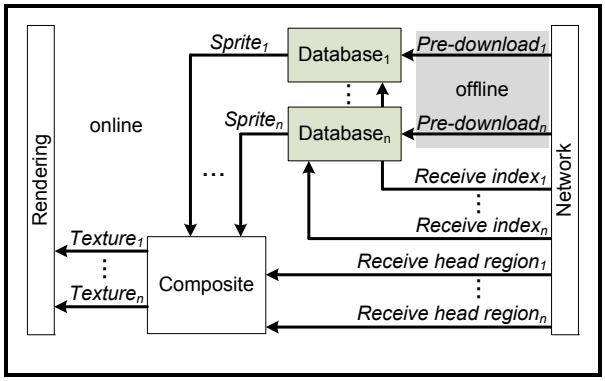

Fig. 3 Rendering module.

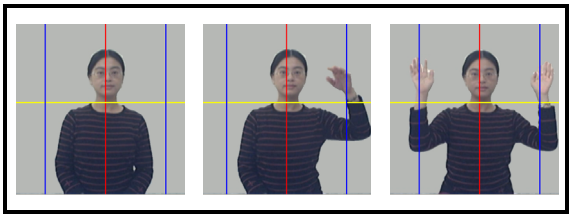

Fig. 4 Shape classification. The 3 sprites are classified as no-left and no-right, no-left and yes-right, and yes-left and yes-right.

matching sprite in the database, as described in Section 5. The index of the matching sprite is sent to all rendering sites. For the active participant(s), the head region of the current sprite is detected, compressed, and sent as described in Section 6.

The architecture of the rendering module is given in Figure 3. The needed databases are obtained before the actual application session. During the session, the database sprite index received from the acquisition sites is used to update the video textures used to render each participant. For active participants, the head region of the database sprite is replaced with the head region received from the acquisition site.

4 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 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 whose pixels are 1 when at least one sprite covers a pixel, and 0 otherwise. The frame is

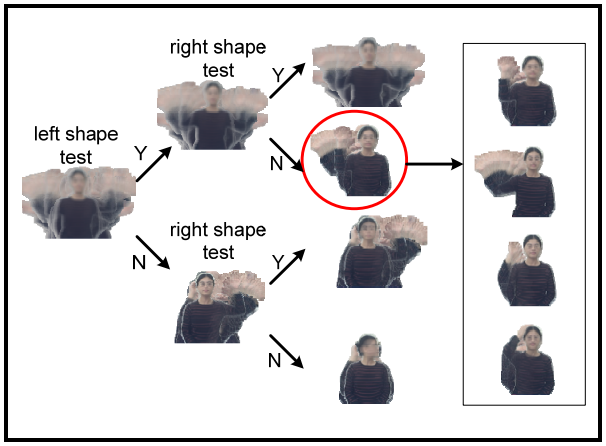

Fig. 5 Hierarchical database structure.

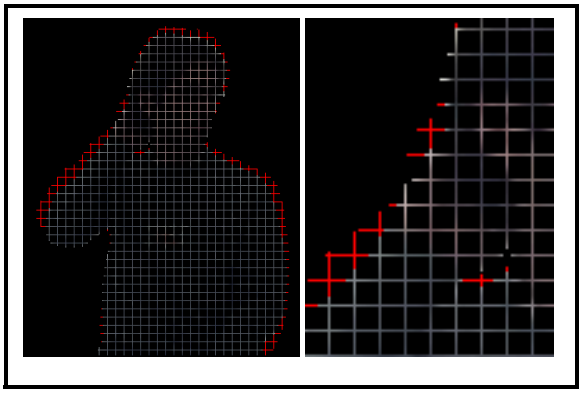

Fig. 6 Visualization of sprite comparison, and magnified fragment.

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 child of an internal node according to whether its shape crosses into the left and/or the 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 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.

The sprites are blurred and down-sampled to expedite the search for the database sprite that matches a given input sprite. The blurred and down-sampled versions of the sprites do not need to be sent to the rendering site since they are used exclusively to find the matching sprite at the acquisition site. Conversely, the acquisition site does not need the full resolution sprites since those are only needed for rendering.

5 Real-Time Sprite Matching

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

1. Down-sample and blur input sprite for efficient and robust color comparison.
2. Find the appropriate database binary tree leaf by descending from the root and classifying the sprite based on shape.
3. At the leaf, find a conservative set of plausible sprite sequences.
4. Linearly traverse each plausible sprite sequence to find the best matching sprite.

At step 3, the sprite sequences that do not match the input sprite are quickly rejected based on the shape bitmaps, which allow comparing a sprite to an entire sequence at once. At step 4, the input sprite is compared to a database sprite by first aligning the 2 sprites. To this effect, a 2D

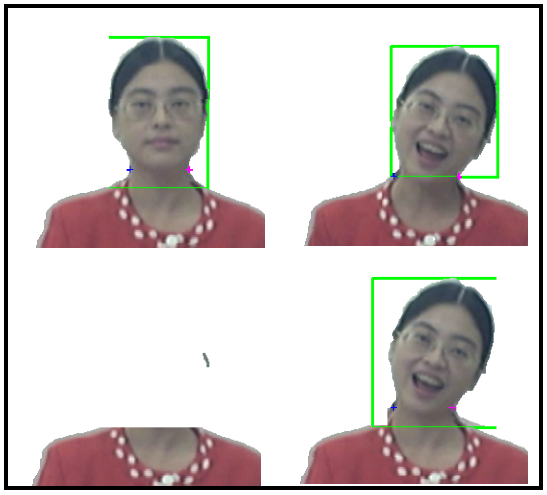

Fig. 7 Head region of the database sprite (top left) is masked off (bottom left) and replaced with the head region of the input sprite (top right), see bottom right.

translation vector is based on the relative position of the top points of the center lines. Then the aligned sprites are compared in shape and in color by comparing pixels inside their bounding boxes. Only a subset of the pixels is used for efficiency. The subset is defined by a regular grid of horizontal and vertical segments, see Figure 6, where the shape difference is highlighted in red.

We take advantage of coherence in the stream of input sprites by starting the search in the same sprite sequence as before. This not only accelerates the search but it also alleviates popping artifacts for input sprites close to the neutral pose for which there are several matching sprites.

6 Head Region Processing

For active participants, detected as those participants for whom the audio level is above a "silence" threshold, 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. The head region is an axis aligned bounding box of the head. The top of the head region is defined by the top foreground point on the central line. The bottom is found by walking down on the central line until the narrowest horizontal foreground segment is found. The left/right extents are defined by the widest foreground horizontal segment.

The head region is mpeg compressed and sent to the renderer, where it is decompressed and composited with the database sprite as illustrated in Figure 7. The pixels of the database head region are looked up into the input head region using a 2D mapping defined by the neck segments, which aligns the necks. 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 position of the head region are avoided by averaging the 2D mapping from the database to the input head region over the k-most recent frames. A typical k value is 10, which corresponds to 1s for a frame rate of 10fps.

7 Results and Discussion

We tested our method by modeling 4 different subjects, seen in figures of the paper and in the accompanying video. We first discuss database construction, then run-time, and then overall system performance.

Database construction 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 [WS03] 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 creation 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).

Run time performance
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 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 display about 10 participants 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 display 120 participants. Figure 8 shows the output image of a rendering module with 30 participants. The 30 databases of 160x120 sprites total 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.

Fig. 8 Rendering module output image.

Codec	Frame rate [fps]	Resolution 640x480 Quality Low	Resolution 640x480 Quality High	Resolution 160x120 Quality Low	Resolution 160x120 Quality High	Our method 640x480 Head region No	Our method 640x480 Head region 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.

Overall system performance

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 [WS03] and DivX 6.4.0 implementation of MPEG-4 [CAVO99]), 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 9).

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 on-campus classroom at 10fps, 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 (1,500 bytes for TCP), the index should be added to audio packets. The bandwidth at the classroom is also very small: 30 x 0.16 = 4.8kbps.

When the 320x240 head region is transmitted, it is encoded using the same MPEG-4 codec. In this case, our method outperforms the codecs by a factor of 3 when using similar settings, leveraging the smaller size of the head region compared to the entire sprite.

8 Conclusion

We have presented a method for compact real-time modeling of seated humans suitable for applications such as teleconferencing and distance learning. The method takes advantage of the limited number of significant body positions a participant can assume in such applications. At the cost of a pre-upload of a few minutes, the method drastically reduces the amount of data that needs to be sent to the rendering site by replacing the bulk of the input sprite with a two-byte index.

For active participants the head region of the video sprite is segmented and transmitted in real time. In these first experiments an active participant is a participant who speaks. This classification can be relaxed to capture facial expressions of participants who do not speak, at a proportionate bandwidth cost. One option is to send the head regions of all participants other than those in the neutral pose, which will convey, for example, the facial expression of students who raise their hand confused, hesitant, or confident. Another option is to develop algorithms that detect facial expressions that deviate from the neutral expression (and not pose), which will capture all facial expressions.

The present work is orthogonal to research aimed at improving the robustness of background subtraction in scenarios with dynamic, low contrasting backgrounds. For now we require the participant to provide a favorable background, restriction that will be lifted as background subtraction state-of-the-art progresses.

As future work we will use the method in the context of an actual distance learning system, which we are currently developing in parallel. Our method will enable students to attend class remotely relying solely on commodity level hardware and connectivity. Moreover our method will facilitate the assessment of the distance learning system by automating the analysis of the video footage recorded during lecture sessions. The sprite databases and the traces of matched sprite indices can be seen as the output of powerful video abstraction and summarization tools.

9 Acknowledgments

We are grateful to Radu Dondera, to Mihai Mudure, to Yi Xu, and to all members of the Distance Learning Group of Purdue University for their help. This work was supported by the United States National Science Foundation through grant SCI-0417458.

References

[Aiz95] AIZAWA K.: Model-Based Image Coding: Advanced Video Coding Techniques for Very Low Bit-Rate Applications. *In Proceedings of the IEEE* (Feb. 1995), vol. 83, issue2, pp. 259-271.

[BCS97] BREGLER C., COVELL M., SLANEY M.: Video Rewrite: Driving Visual Speech with Audio. *In Proceedings of SIGGRAPH 97* (1997), 353-360.

[CAVO99] Coding of Audio-Visual Objects. *ISO/IEC 14496-2* (1999).

[CG00] COSATTO E., GRAF H. P.: Photo-Realistic Talking-Heads from Image Samples. *In IEEE Transaction on Multimedia* (September 2000), vol. 2, no. 3, pp. 152-163.

[CTMS03] CARRANZA J., THEOBALT C., MAGNOR M.

Fig. 9 High and low quality codec settings used in the Table 1 experiments.

SEIDE H.: Free-Viewpoint Video of Human Actors. *In Proceedings of SIGGRAPH 2003* (2003), 569-577.

[GBS00] GRAMMALIDIS N., BELETSIOTIS D., STRINTZIS M. G.: Sprite generation and Coding in Multiview Image Sequences. *IEEE Transaction on Circuits and Systems for Video Technology* (March 2000), vol. 10, no. 2, pp. 302-311.

[GD96] GAVRILA D. M., DAVIS L. S.: 3-D Model-based Tracking of Human Upper Body Movement: a Multi-view Approach. *In Proceedings CVPR '96* (1996), 73-80.

[GBM93] GIBBS S., BREITENEDER C., DE MEY V.: Video Widgets and Video Actors. *In Proceedings UIST'93* (1993), 179-184.

[HSS05] HAKEEM A., SHAFIQUE K., SHAH M.: An Object-based Video Coding Framwork for Video Sequences Obtained from Static Cameras. *In Proceedings of the 13th annual ACM International Conference on Multimedia* (2005), 608-617.

[Lau94] LAURENTINI A.: The Visual Hull Concept for Silhouette-Based Image Understanding. *In IEEE Transaction on Pattern Analysis and Machine Intelligence* (February 1994), vol. 16, no. 2, pp150-162.

[LCL*97] LEE M. C., CHEN W. G., LIN C. B., GU C., MARKOC T. ZABINSKY S. I., SZELISKI R.: A Layered Video Object Coding System Using Sprite and Affine Motion Model. *IEEE Transaction on Curcuits and Systems for Video Technology* (1997), vol. 7, no. 1, pp. 130-145.

[LH96] LEVOY M., HANRAHAN P.: Light Field Rendering. *In Proceedings of SIGGRAPH 1996* (1996), 31-42.

[LS97] LENGYEL J., SNYDER J.: Rendering with Coherent Layers. *Proceedings of SIGGRAPH 1997* (1997), 233-242.

[Pea95] PEARSON D. E.: Developments in Model-Based video Coding. *In Proceedings of the IEEE* (June 1995) vol. 83, issue 6, pp. 892-906.

[RR03] REMONDINO F., RODITAKIS A.: Human Figure Reconstruction and Modeling from Single Image or Monocular Video Sequence. *In Proceedings of 4th International Conference on 3D Digital Imaging and Modeling* (3DIM 2003), 116-123.

[SLS*96] SHADE J., LISCHINSKI D., SALESIN D.H., DEROSE T., SNYDER J.: Hierarchical Image Caching for Accelerated Walkthroughs of Complex Environments. *Proceedings of SIGGRAPH 1996 (*1996), 75-82.

[SMSP05] SCHOLL J., MCCARTHY J. D., SASSE A. PARNES P.: Designing a Large-Scale Video Chat Application. *In Proceedings of the 13th annual ACM International Conference on Multimedia* (2005), 71-80.

[TCM03] THEOBALT C., CARRANZA J., MAGNOR M.: Enhancing Silhouette-based Human Motion Capture with 3D Motion Fields. *In Proceedings of 11th Pacific Conference on Computer Graphics and Applications* (2003), 185-193.

[TD00] TORRES L., DELP E. J.: New Trends in Image and Video Compression. *In European Signal Processing Conference* (September 2000).

[TK96] TORBORG J., KAJIYA J.: TALISMAN: Commodity Realtime 3D graphics for the PC. *In Proceedings of SIGGRAPH 96*, 353-364.

[TMS02] THEOBALT C., MAGNOR M. SCHULER P.: Combining 2D Feature Tracking and Volume Reconstruction for Online Video-Based Human Motion Capture. *In Proceedings of 10th Pacific Conference* on *Computer Graphics and Applications* (2002), 96-103.

[WADP97] WREN C., AZARBAYEJANI A., DARRELL T. PENTLAND A.: Pfinder: Real-Time Tracking of the Human Body. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (July 1997), vol. 19, no. 7, pp. 780-785.

[WLC*04] WEN Z., LIU Z. C., COHEN M., LI J., ZHENG K., HUANG T.: Low Bit-Rate Video Streaming for Face-to-Face Teleconference. *In Proc. IEEE Int. Conf. Multimedia and Expo* (2004), vol. 3, pp. 1631-1634.

[WS03] WIEGAND T., SULLIVAN G.: 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).

[YLK04] YANG S., LEE C. H., KUO C.-C. J.: Optimized Mesh and Texture Multiplexing for Progressive Textured Model Transmission. *In Proceedings of the 13th annual ACM International Conference on Multimedia* (2004), 676-683.

Chun Jia
Computer Science Department
Purdue University
E-mail: chun.jia@gmail.com

Voicu Popescu (contact author)
Computer Science Department
Purdue University
E-mail: popescu@cs.purdue.edu