USING VIEW INTERPOLATION FOR LOW BIT-RATE VIDEO

Richard Radke, Peter Ramadge, Sanjeev Kulkarni

Department of Electrical Engineering
Princeton University
Princeton, NJ 08544
{rjradke, ramadge, kulkarni}@ee.princeton.edu

Tomio Echigo

IBM Research
Tokyo Research Laboratory
1623-14 Shimotsuruma
Yamato-shi, Kanagawa, Japan
echigo@trl.ibm.co.jp

1. INTRODUCTION

The transmission and reconstruction of video in wireless multimedia poses a much more difficult problem than it does in a wired setting. There are three main issues that complicate matters: relatively limited bandwidth (so video data needs to be reduced in both frame size and frame rate), limited power supply at the client (so reconstruction algorithms must be simple), and high bit error rates (so robust error correction is required).

At high bit rates, block-based motion compensation approaches to the lossy video coding problem (e.g. MPEG) have proven difficult to improve upon. However, at very low bit rates, perceptual quality of the reconstructed frames can degrade due to the scarcity of bits available to encode the residuals between each "real" block of pixels and the block used as a predictor. In this paper, we demonstrate that in some situations, perceptual quality can be better maintained using an approach based on synthesizing "virtual" images of a scene that match frames from a source video clip. We use this algorithm for interpolation of video frames in the time domain, using a small amount of information to construct an approximation of the original video. Our algorithm is well-suited for the limitations in bandwidth and complexity characteristic of wireless multimedia channels. Since the approach is based on estimating functions of the underlying camera motion parameters, it can capture relationships between image correspondences that extend across many (perhaps hundreds) of video frames. Each interpolated image can be rendered using only a few tens of bytes of side information, and the rendering process itself has low computational requirements. We present experimental results to demonstrate that for certain types of video, our algorithm can give significant perceptual improvement over MPEG-4 coded video at the same low bit rate (e.g. 47 kpbs). Our approach is particularly amenable to representing computer-generated video, for which the correspondence and camera motion information required for view synthesis is readily available at render time.

The advantage of our method is the use of view morphing instead of block-based motion compensation to synthesize intermediate views between reference frames. As we demonstrate, adjacent reference frames need not be temporally close or visually similar. Consequently, in theory, the reference frames can be taken hundreds of frames apart (much further apart, for example, than typical I frames in an MPEG video). We note that while there has been some work on using "virtual views" for video coding, these approaches are generally aimed at the small-baseline case for compressing video teleconferencing data.

We emphasize that this scheme is meant to augment, not to supplant, standard video coding algorithms. When a segment of video that is suitable for view interpolation is encountered, the server could transmit the low-overhead side information within the framework of a standard video data stream. A "smart" receiver equipped with our algorithm could take advantage of this information to render the video segment at higher quality, while a "normal" receiver would ignore the side information and produce standard-quality video.

2. PRELIMINARIES

The interpolation algorithm depends on both projective transformations and view morphing, which we briefly review here. A projective transformation globally relates the coordinates of an image pair taken by a rotating camera without translation, or images of a planar surface taken by a rotating and translating camera. The form of a projective transformation $M = (A, b, c)$ applied to a point $w \in \mathbb{R}^2$ is:

$$M(w) = \frac{Aw+b}{c^T w+1} \tag{1}$$

where $A \in \mathbb{R}^{2\times 2}, b \in \mathbb{R}^2, c \in \mathbb{R}^2$. The problem of estimating the projective transformation which best relates an image pair has been well-studied [1]. When the optical center of the camera can be treated as stationary, one frame can be approximated from another by projectively warping the

This research was partially supported by grants from the IBM Tokyo Research Laboratory and the New Jersey Center for Multimedia Research.

image plane. The interpolation algorithm we discuss below extends this idea for more general intermediate images, at the cost of slightly more side information.

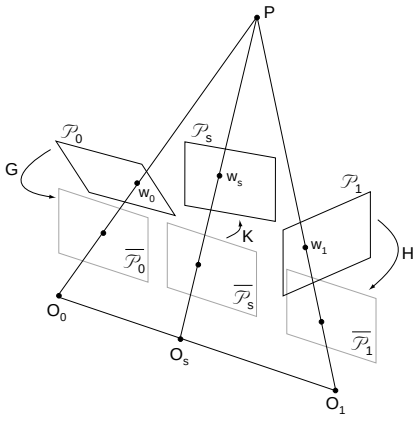


Fig. 1. View morphing.

Consider an image pair generated by cameras $(\mathcal{C}_0, \mathcal{C}_1)$ with distinct optical centers (O_0, O_1) and image planes $(\mathcal{P}_0, \mathcal{P}_1)$. If a dense set of correspondences between $\mathcal{P}_0$ and $\mathcal{P}_1$ can be estimated, the view morphing algorithm of [2] can be used to synthesize a virtual image from the perspective of a camera with center on the line segment connecting O_0 to O_1. Specifically, let $\mathcal{I}_0$ and $\mathcal{I}_1$ be images taken by the cameras $\mathcal{C}_0$ and $\mathcal{C}_1$, respectively, and consider an image $\mathcal{I}_s$ taken by a camera with center a fraction s of the distance between O_0 and O_1 (Figure 1) . If w_0 and w_1 are the projections of some scene point P onto the image planes $\mathcal{P}_0$ and $\mathcal{P}_1$, respectively, then the projection w_s of P onto the virtual image plane $\mathcal{P}_s$ is given by:

$$w_s = K^{-1}((1-s)G(w_0) + sH(w_1)) \tag{2}$$

Here, G, H, and K are "rectifying" projective transformations applied to $\mathcal{P}_0$, $\mathcal{P}_1$, and $\mathcal{P}_s$, respectively, chosen to make the image planes of the warped images parallel. The image intensity at w_s can be estimated by a weighted average:

$$\mathcal{I}_s(w_s) = (1-s)\mathcal{I}_0(w_0) + s\mathcal{I}_1(w_1) \tag{3}$$

3. INTERPOLATION

Our approach to frame interpolation is illustrated in Figure 2. Here, dots and arrows represent the positions and orientations of a camera during a video shot of a static scene. A fraction of the frames are selected to be anchor, or "A", frames. These frames are the same as intracoded frames in the standard terminology, and are transmitted with good fidelity using a standard image compression scheme. The rest of the frames are designated as virtual, or "V", frames. These are interpolated between adjacent A frames using the view synthesis and registration algorithm described below.

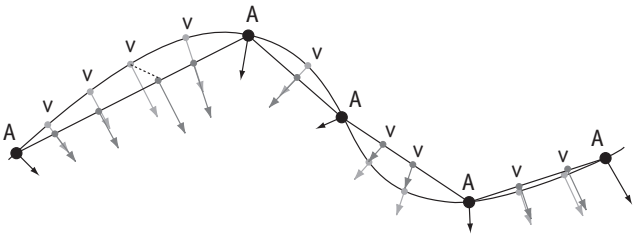


Fig. 2. Interpolating video from a translating camera. Frames are designated as anchor (A) frames or virtual (V) frames.

For piecewise linear paths, we can synthesize virtual frames using view morphing that resemble real views of a scene, and are physically correct perspective views provided the estimate of correspondence between image planes is accurate. This is true even when the perspective difference between the source frames is quite large. Hence, in principle, we may temporally subsample the video at wide intervals to obtain the A frames. The anchor frames need not be equally spaced along the image sequence; ideally they should automatically be chosen with respect to estimated camera dynamics.

In the remainder we assume the A frames have been chosen and consider one line segment of Figure 2b. We will work with a sequence of video frames $\{\mathcal{I}_t\}$, with $t = 0, \Delta t, 2\Delta t, \ldots, 1$, which are generated by a moving camera whose (unknown) parameters at time t are $\{\mathcal{C}_t\}$. We designate $(\mathcal{I}_0, \mathcal{I}_1)$ as the anchor frames, with associated image planes $(\mathcal{P}_0, \mathcal{P}_1)$. Our goal is to synthesize a virtual image $\hat{\mathcal{I}}_t$ using view morphing between $\mathcal{I}_0$ and $\mathcal{I}_1$ to approximate each intermediate frame $\mathcal{I}_t$. The information that comprises the transmitted video is thus:

1. The anchor frames $(\mathcal{I}_0, \mathcal{I}_1)$ (suitably compressed).

   and the following side information:

2. A pair of projective transformations (G, H) that rectify $(\mathcal{I}_0, \mathcal{I}_1)$.

3. The structured correspondence between the image planes $\mathcal{P}_0$ and $\mathcal{P}_1$.

4. The camera position of each virtual frame, described as a fraction s_t of distance along the baseline connecting the optical centers of $\mathcal{C}_0$ and $\mathcal{C}_1$ (1 floating point number per V frame).

5. The projective transformation K_t that aligns each virtual frame with the corresponding actual frame $\mathcal{I}_t$ (8 floating point numbers per V frame).

Then for every estimated corresponding pair (w_0, w_1) in $\mathcal{P}_0 \times \mathcal{P}_1$, a pixel is rendered at position w_t in $\hat{\mathcal{I}}_t$ by the view morphing equations (2) and (3).

4. IMPLEMENTATION

The perceptual fidelity of video frames interpolated using the algorithm above depends crucially on the accuracy of the correspondence at step 3 above, and the estimates of s_t and K_t at steps 4 and 5.

In practice, we initialize the correspondence between anchor frames with minimal user input, e.g. a set of corresponding points and line segments. This information seeds an automatic correspondence estimation algorithm which generates a dense correspondence over the entire image pair. When there is a substantial perspective change between anchor frames, there is generally no global parameterization for the correspondence, and traditional methods from stereo or optical flow are unreliable. We use an algorithm based on estimation of the epipolar geometry [3], projective rectification [4], and constrained dynamic programming [5].

When we deal with video of a scene with multiple occluding foreground objects, the important assumption behind many correspondence algorithms, that the ordering of objects along epipolar lines in an image pair is invariant, does not hold. In these cases we use the formalism of *correspondence graphs* [6] to ensure that occlusions are handled correctly.

Given the correspondence between $(\mathcal{P}_0, \mathcal{P}_1)$ and a pair of rectifying projective transformations (G, H), it remains to obtain the approximation $\hat{\mathcal{I}}_{s_t, K_t}$. The problem of estimating (s_t, K_t) can naturally be posed as a minimization problem:

$$\min_{s\in[0,1]} \min_{K\in GL(3)} \sum_{w\in\mathcal{P}_t} (\hat{\mathcal{I}}_{s,K}(w) - \mathcal{I}_t(w))^2$$

Since s is fixed for the interior problem, it is simply the problem of finding the best projective transformation relating $\hat{\mathcal{I}}_{s,I}$ and $\mathcal{I}_t$. We can apply the efficient estimation algorithms discussed in [1]. When $\hat{\mathcal{I}}_{s,I}$ and $\mathcal{I}_t$ are similar, automatic feature extraction techniques (e.g. [7]) can be applied to obtain data points for the optimizations. However, depending on the choice of rectifying projective transformations (G, H), the synthetic image that results from view morphing may have a much different orientation than the frame we wish to predict. For this reason, we apply an initial projective transformation to $\hat{\mathcal{I}}_{s,I}$ to better align the two images for feature matching. The previous estimate of $K_{t-\Delta t}$ is a natural choice (at $t = 0$, we have the zero-error solution $K_0 = G^{-1}$). The same applies to the search neighborhood for $\hat{s}_t$; since we assume the camera motion is continuous and its velocity is bounded, we only need to refine the initial estimate $\hat{s}_t = \hat{s}_{t-\Delta t}$. If the video to be interpolated is in MPEG format, an alternative initial estimate of s_t could be obtained by rapidly estimating the magnitude and direction of camera motion from MPEG motion vectors [8].

5. EXPERIMENTAL RESULTS

We applied our interpolation method to a 180-frame, single-shot, 320 x 240 test sequence captured with a digital video camera. The first and last frames (Figure 3) were designated as anchor frames, and correspondence between them was initialized using 66 user-selected matching points and 28 control line segments. The remaining 178 frames were designated as "V" frames to be interpolated. The camera motion is roughly linear, though the speed is not uniform. From Figure 3 we can see that the perspective difference between the anchor frames is substantial, and that a block from an intermediate frame would probably have a poor (i.e. high MSE) match in either of the anchor frames.

(a) (b)

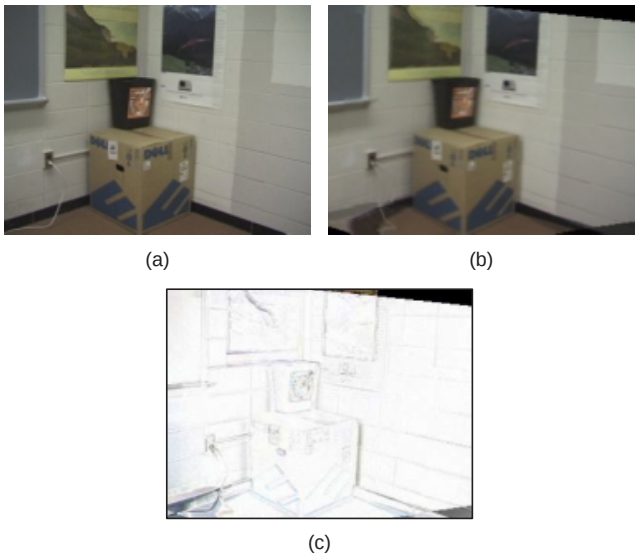
Fig. 3. The two anchor frames: (a) Frame 0, (b) Frame 179.

Figure 4 illustrates the original, interpolated, and luminance difference frames for the 90th frame of the test sequence (the difference image has been enhanced to darken the lighter pixels for visibility). From the difference image we can see that the interpolated image aligns quite well with the original frame of video. The errors around edges are largely due to the blurriness of the virtual images introduced by several steps of image resampling. The other major artifacts are the black regions around the borders of the interpolated image that correspond to areas of the virtual frame visible in neither of the anchor frames. In this example, these areas are not too large and could be filled in by the type of error-concealment algorithms devised for other video compression schemes. The total file size of the information required to interpolate is roughly 35.4 KB (18 KB total for the two JPEG-coded anchor frames, 11 KB for the compressed correspondence information, 36 B for the parameters of each virtual frame). Clearly the number of interpolated frames has a negligible effect on the size of the transmitted data. For video in which the camera moves slowly along an approximately piecewise linear path, we therefore expect reasonable performance for a small amount of side information. The total bit rate in this example is 47.22 kbps. The mean PSNR over the entire sequence is

30.2 dB, using the definition

$$\mathrm{PSNR} = 10 \log_{10} \frac{1}{|R|} \sum_{w \in R} \frac{255^2}{(\mathcal{I}_t(w) - \hat{\mathcal{I}}_t(w))^2}$$

where R is the set of rendered pixels. The PSNR could be increased by reducing the blurriness of the virtual images through postprocessing or by reducing the number of explicit image resampling steps.



(a) (b) (c)

Fig. 4. (a) Original frame 90, (b) Interpolated frame 90, (c) Luminance difference.

For comparison, we constructed an MPEG-4 video at 47.22 kbps using the Microsoft Windows Media Encoder [9]. At this bit rate, the MPEG blocking and compression artifacts are severe, especially in high-detail, perceptually significant areas of the image. In contrast, the virtual image is well-defined and relatively sharper in these areas. This can be seen in the close-ups shown in Figure 5.

(a) (b) (c)

Fig. 5. (a) Detail, original frame 90, (b) Detail, interpolated frame 90, (c) Detail, MPEG frame 90

6. CONCLUSIONS

We suggest that methods, like the one introduced here, that exploit the relationship between camera motion and image correspondences can be profitably incorporated into a low bit-rate video scheme. The methods described here are not proposed as a final solution to the problem, or as a substitute for traditional video compression techniques at higher bit rates. However, we hope that these ideas provide a starting point for continuing research on more general video, and that tools such as these will be incorporated into future compression standards.

As demonstrated above, our approach is well-suited to wireless low bit-rate video in special cases of camera motion, and can be used to increase frame rate without much overhead. The computationally expensive (about 5 frames per minute) step of estimating parameters and encoding the video can be done once at the multimedia server, and amortized over a large number of downloads. The low bit-rate (e.g. 45 kbps) data stream can be easily transmitted over a wireless channel. The video can be rendered at the multimedia client for a low computational cost (hence low power consumption), since each rendered pixel is simply a weighted average of two pixels from the source images.

One problem with our current algorithm is the perceptually distracting jitter in the reconstructed video, since the estimation of (s_t, K_t) is essentially independent for each frame. Even though the synthetic frames are individually good approximations to the original frames they represent, the result does not convey a sense of fluid camera motion as well as it could. Of course, since the original camera motion may be jerky, imposing too many smoothness constraints on the estimation could be unwise. This is a topic of current research.

We have not addressed the case when objects are moving in the scene independently of the camera. However, our method generalizes one feature of the MPEG-4 standard, in which foreground objects can be replaced on a projectively warped planar background. In this case, an arbitrary background can be locally warped to the correct position using structured correspondence and view morphing, for little additional overhead.

7. REFERENCES

[1] R. Radke, P. Ramadge, T. Echigo, and S. Iisaku. Efficiently Estimating Projective Transformations. In *Proc. ICIP 2000*, September 2000.

[2] S.M. Seitz and C.R. Dyer. View Morphing. *Computer Graphics (SIGGRAPH '96)*, pp. 21–30, August 1996.

[3] Z. Zhang. Determining the Epipolar Geometry and its Uncertainty: A Review. *International Journal of Computer Vision*, vol. 27, no. 2, pp. 161–195, 1998.

[4] R.I. Hartley. Theory and Practice of Projective Rectification. *International Journal of Computer Vision*, Vol. 35, No. 2, pp. 115–127, November 1999.

[5] Y. Ohta and T. Kanade. Stereo by Intra- and Inter-Scanline Search Using Dynamic Programming. *IEEE PAMI*, Vol. 7, No. 2, pp. 139–154, March 1985.

[6] R. Radke, V. Zagorodnov, S. Kulkarni and P. Ramadge. Estimating Correspondence in Digital Video. In *Proc. ITCC 2001*, Las Vegas, April 2001.

[7] Y.P. Tan, S. Kulkarni, and P. Ramadge. Extracting Good Features for Motion Estimation. *Proc. ICIP 1996*, vol. 1, pp. 117–120, 1996.
[8] Y.P. Tan, D. D. Saur, S. R. Kulkarni, and P. J. Ramadge. Rapid Estimation of Camera Motion from Compressed Video With Application to Video Annotation. *IEEE Trans. on Circuits and Systems for Video Technology*, vol. 10, no. 1, pp. 133–146, February 2000.
[9] Microsoft Windows Media Encoder. `http://www.microsoft.com/windows/windowsmedia/`