Video Codec for Classical Cartoon Animations with Hardware Accelerated Playback

Daniel Sýkora, Jan Buriánek, and Jiří Žára

Czech Technical University in Prague
Digital Media Production

Abstract. We introduce a novel approach to video compression which is suitable for traditional outline-based cartoon animations. In this case the dynamic foreground consists of several homogeneous regions and the background is static textural image. For this drawing style we show how to recover hybrid representation where the background is stored as a single bitmap and the foreground as a sequence of vector images. This allows us to preserve compelling visual quality as well as spatial scalability even for low encoding bit-rates. We also introduce an efficient approach to play back compressed animations in real-time on commodity graphics hardware. Practical results confirm that for the same storage requirements our framework provides better visual quality as compared to standard video compression techniques.

1 Introduction

Generations of children and adults enjoy classical cartoon animations and want to see them again and again in the best possible quality. However, the lifetime of traditional archive formats (such as celluloid negative) is strongly limited. Variety of physical degradations may significantly reduce visual quality or completely destroy the original artwork. When one decides to rescue these amazing works, it is necessary to perform *telecine* [1], i.e. to transfer motion picture from film negative into a digital video format. During this process two important factors leverage the final visual quality and the amount of required storage space: resolution and compression technology.

In the movie industry cartoon animations are typically intended for television broadcasting or `DVD` production, therefore `PAL` or `NTSC` resolution is used for digital conversion. Regarding compression the most popular format for broadcasting is *Digital Betacam* tape [2] that can run up to 124 minutes of high-quality video with nearly lossless `2:1` compression. For `DVD` production `MPEG-2` [3] is used with average bit-rates around 3.5 Mbps which is lower quality but still acceptable for most cases. However these commercial standards produce video stream that requires large storage space and is not simply spatially scalable.

In this paper we adopt well known concept of layered compression which has been standardized in `MPEG-4` video compression scheme [4]. We show how to detach the dynamic foreground layer and how to reconstruct the static background layer in outline-based cartoon animations. We also introduce a novel space-saving vectorization scheme which allows to preserve high visual quality and resolution independency even for low encoding bit-rates. Finally an efficient decompression scheme is addressed to play back compressed animations in real-time on consumer graphics hardware.

The rest of paper is organized as follows. First we present a brief overview of related work on video compression. Next, we describe our method including implementation details and optimization issues. Finally we present several results on real cartoon animations to confirm the efficiency of proposed solution and conclude with plans for future work.

2 Related work

For decades video compression was and still is a very active research area (for survey see [5, 6]). However to our best knowledge there has been only one method published [7] which takes into account visual aspects of cartoon animations. In this approach each region is first represented as a 3D volume by sweeping its 2D shape through the time. Then edge-breaker compression scheme is used to encode volume geometry. Decoding is done by intersecting compressed volume with image plane using GPU-based solid clipping technique. Unfortunately this compression is suitable only for regions that change their shape and/or position slightly. Authors also did not address the problem of region shape extraction for more complicated color and gray-scale image sequences.

Standard approaches to video compression such as MPEG-2 assume that strong discontinuities are not frequent in natural image sequences and thus 2D discrete cosine transform (DCT) [8] can by used to efficiently encode small image blocks. However blocking and ringing artifacts may arise when DCT is applied to cartoon images that contain lots of sharp edges. Even promising discrete wavelet transform DWT [9] does not overcome this problem while quantization errors still produce visible ringing artifacts. To suppress such degradations various post-processing techniques have been developed [10, 11]. However they are usually computationally expensive and thus not tractable for real-time applications. Another possibility is to completely avoid coding along strong edges as in segmentation and object-based compression techniques [12, 13]. Here the aim is to divide the input image into a set of homogeneous regions and then compress each of them separately. Unfortunately these methods require usually much higher encoding bit-rates due to over-segmentation.

3 Framework

Shortcomings of previous approaches lead us to develop our novel framework. It consists of five independent phases: unsupervised segmentation, background reconstruction, foreground vectorization, temporal coherency issues and playback. In this section we describe each step in more details.

3.1 Segmentation

First we need to separate the original image into a set of regions. For this task we adopt image segmentation technique first presented in [14] and later refined in [15, 16]. This approach has been designed directly for outline-based cartoons. The main idea is to exploit negative response of Laplacian-of-Gaussian ($\mathbf{L} \circ \mathbf{G}$) to locate outlines. Similarly to [16] we additionally perform interpolation in the $\mathbf{L} \circ \mathbf{G}$ domain to reach fourfold sub-pixel precision [17]. In this case unsupervised σ-fitting [15] helps us to suppress spurious regions that arise inside outlines when extrapolating $\mathbf{L} \circ \mathbf{G}$ response to the higher resolution.

Next we want to detach foreground regions and reconstruct one single image of the whole background layer. To do that we first roughly estimate whether a given region belongs to the background layer using region area size thresholding as in [15]. Then a hierarchical motion estimation [18] helps us to register parts of the background layer using consecutive animation frames. After the registration fragments of background are stitched together and stored as a single image using standard JPEG compression [19]. We also store estimated camera motion vectors. Later this information allows us to extract currently visible portion of the background layer.

In the second phase we refine classification of smaller regions. We accumulate normalized sum of absolute differences between region pixels and corresponding pixels in the reconstructed background. When this sum falls under a specified limit then the region is classified as a background. This approach may fail to classify very small regions while anti-aliased outlines can significantly bias the sum of absolute differences. However such regions are usually homogenous and thus from the visual point of view it does not matter when they remain in the foreground layer.

Finally, it is necessary to assign visually dominant color to each region in the foreground layer. To avoid flickering due to biased mean color estimation we apply mean-shift clustering [20] to obtain static palette of most used colors.

3.2 Vectorization

After the segmentation we need to convert shapes of homogeneous regions into a scalable vector representation. There are various techniques suitable for this task. In our framework we experiment with cubic B-splines [21] and 1D DCT [22]. B-splines are optimal in the sense of rate distortion and DCT representation is more suitable for scalable compression.

In order to combine advantages of both approaches we first apply standard contour tracing algorithm [23] to fit a set of cubic B-splines to the raster representation of the original shape. Then we perform curvature sensitive subdivision using central differencing [24] to obtain adaptive sampling of the original shape that is much more suitable for compression in the frequency domain than uniform sampling (see Figure 1). Moreover in our experiments we found that there is a high correlation between the optimal number of DCT coefficients and the optimal number of control points in the original B-spline representation. We use

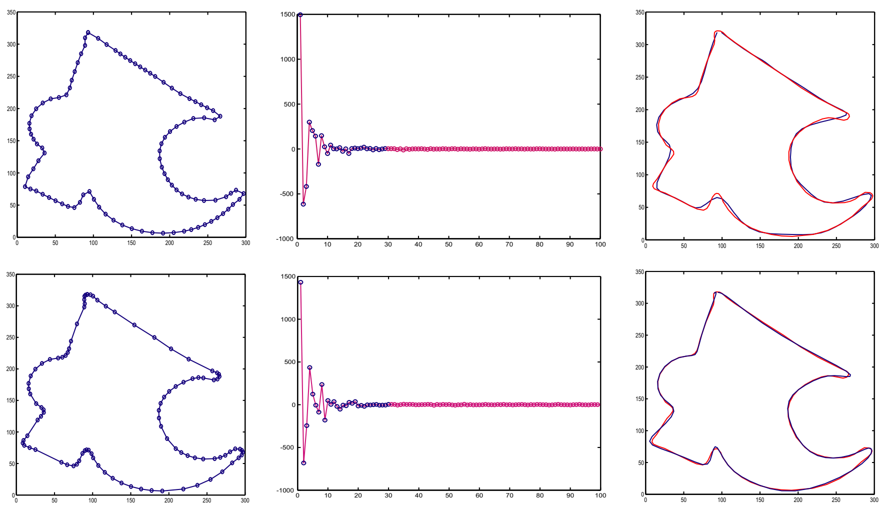

Fig. 1. Compare the efficiency of `1D DCT`-based vector compression for uniformly (top) and adaptively (bottom) sampled shape. For the same number of `DCT` coefficients adaptive sampling produces better approximation.

this feature to estimate the number of `DCT` coefficients needed to store vector representation without significant loss of precision.

Finally, when `DCT` coefficients are extracted, we normalize them and quantize uniformly to 16-bit representation. Burrows-Wheeler block transform [25] together with Huffman coding [26] is used to store them into the output stream. In contrast to the original B-spline representation `DCT`-based compression can save up to 50% of storage space with negligible loss of precision.

3.3 Temporal coherency

Usually it is not necessary to store foreground layer for each frame since in cartoon animations lots of frames are identical. To alleviate tedious manual inbetweening artists usually decide to significantly reduce frame rate. Typically only every second frame is different. For long animations they also use repetitive sequences of same frames. In our framework we exploit this redundancy by maintaining pool of already stored frames in which we search for duplicities.

The search itself is hierarchical and the similarity is computed only for outline-extracted images. We use phase correlation [27] to align and preselect similar down-sampled thumbnails of past frames. Then we apply bitwise `XOR` and distance transform [28] to obtain weighted difference bitmap which emphasizes large changes against small global shifts. Finally we sum up weights in this bitmap and if the sum falls under the user-specified threshold we store only the identification number of the corresponding frame. By tuning this threshold we can further lower encoding bit-rate by omitting frames containing insignificant changes.

3.4 Playback

In order to reach compelling visual quality as well as real-time performance during the playback, we implement a simple `OpenGL`-based player that utilizes `GPU` to render anti-aliased background and foreground layer for each animation frame. See our web-site[1] where this player is available together with short testing sequence.

During the playback we first activate hardware accelerated full screen anti-aliasing (`FSAA`) to preserve smooth polygon boundaries. Then the background layer is rendered as a textured rectangle with proper shift and scale according to the estimated camera motion vectors. In front of this background we put the foreground layer. For each region we first apply zero-padding in the frequency domain to increase the resulting number of control points. Then we use `IDCT` together with proper normalization to obtain coordinates of control points. Thanks to prior adaptive sampling this process yields a good piecewise linear approximation of the region shape that can be directly used for polygon rendering. Extra processing is required only for non-convex polygons that have to be tesselated into convex triangle strips.

Rendering phase itself is time critical since we need to generate all control points and tessellation of non-convex polygons in real-time (typically tens of polygons with hundreds of vertices 25 times per second). To achieve this performance it is necessary to utilize fast `FFT`-based implementation of the `IDCT` [29]. For tessellation of non-convex polygons we exploit standard framework `gluTess*` [30]. However for larger polygons we recommend to use `FIST` [31].

4 Results

In this section we present several results obtained on real cartoon animations scanned in `PAL` resolution (`720x576`) from the original celluloid negative of Czech cartoon *O loupežníku Rumcajsovi*. We encode five different sequences using our codec and using `DivX` with comparable storage requirements. We set the parameters to reach average encoding bit-rate of 256 kbps. Using `AMD Athlon A64 2800+` with `ATI Radeon 9700` and `6x FSAA` it takes in average 10 seconds to compress and 30 milliseconds to decompress and render one frame. See our web-site[1] and compare visual quality with accompanied `AVI` encoded using `DivX` codec.

In Figure 2 three still images from different sequences are presented. To render them we use the original `PAL` resolution. However the resolution itself can be much higher (to see this try to maximize rendering window in our player). The limiting factor is bilinear interpolation of the background texture which is fortunately not so disturbing while the background layer usually does not contain sharp edges. In Figure 2 no blocking or ringing artifacts are visible in the output of our codec, only several shape details are omitted. There are also a few examples of small misclassified regions that are not disturbing during the playback while the background layer is almost homogeneous at the same position.

[1] `http://www.cgg.cvut.cz/~sykorad`

5 Conclusions and Future work

A novel video compression scheme for traditional outline-based cartoon animations has been presented. Practical experiments performed on real cartoon animations confirm that for comparable encoding bit-rates our approach achieve better visual quality as compared to standard video compression techniques.

As a future work, we plan to significantly speed up compression phase and further lower encoding bit-rates by estimating warping transformation between corresponding regions. Next we also contemplate to adopt our approach for other drawing styles e.g. cartoon animations that contain only outlines and homogeneous regions. Another issue is an automatic sequence annotation which is desirable for consistent background reconstruction.

6 Acknowledgements

This work has been supported by the Ministry of Education, Youth and Sports of the Czech Republic under research program No. `MSM-6840770014` (Research in the area of information technologies and communications) and under the student research program: FRVŠ-2005-1170.

References

1. Bancroft, D.J.: Advanced and economical telecine technology for global DTV production. In: Proceedings of Broadcast Engineering Conference. (2000)
2. Sykes, P.J.: Digital Betacam: A new approach to broadcast digital recording. In: Proceedings of International Conference on Storage and Recording Systems. (1994) 9–14
3. Wong, A.H., Chen, C.: Comparison of ISO MPEG1 and MPEG2 video-coding standards. In: Proceedings of SPIE Visual Communications and Image Processing. Volume 2094. (1993) 1436–1448
4. Ebrahimi, T., Horne, C.: MPEG-4 natural video coding – An overview. Signal Processing: Image Communication **15** (2000) 365–385
5. Reid, M.M., Millar, R.J., Black, N.D.: Second-generation image coding: An overview. ACM Computing Surveys **29** (1997) 3–29
6. Clarke, R.J.: Image and video compression: A survey. International Journal of Imaging Systems and Technology **10** (1999) 20–32
7. Kwatra, V., Rossignac, J.: Space-time surface simplification and edgebreaker compression for 2D cel animations. International Journal on Shape Modeling **8** (2002) 119–137
8. Ahmed, N., Natarajan, T., Rao, K.R.: Discrete cosine transform. IEEE Transactions on Computers **C** (1974) 90–93
9. Shapiro, J.M.: Embedded image coding using zerotrees of wavelet coefficients. IEEE Transactions on Signal Processing **41** (1993) 3445–3463
10. Yang, Y., Galatsanos, N.P., Katsaggelos, A.K.: Projection-based spatially adaptive reconstruction of block-transform compressed images. IEEE Transactions on Image Processing **4** (1995) 896–908

11. Fan, G., Cham, W.K.: Model-based edge reconstruction for low bit-rate wavelet-compressed images. IEEE Transactions on Circuits and Systems for Video Technology **10** (2000) 120–132
12. Kwon, O., Chellappa, R.: Segmentation-based image compression. Optical Engeneering **7** (1993) 1581–1587
13. van Beek, P.J.L., Tekalp, A.M.: Object-based video coding using forward tracking 2-D mesh layers. In: Proceedings of SPIE Visual Communications and Image Processing. (1997) 699–710
14. Sýkora, D., Buriánek, J., Žára, J.: Segmentation of black and white cartoons. In: Proceedings of Spring Conference on Computer Graphics. (2003) 245–254
15. Sýkora, D., Buriánek, J., Žára, J.: Colorization of black-and-white cartoons. Image and Vision Computing **23** (2005) 767–852
16. Sýkora, D., Buriánek, J., Žára, J.: Sketching cartoons by example. In: Proceedings of Eurographics Workshop on Sketch-Based Interfaces and Modeling. (2005) 27–34
17. Huertas, A., Medioni, G.: Detection of intensity changes with subpixel accuracy using Laplacian-Gaussian masks. IEEE Transactions on Pattern Analysis and Machine Intelligence **8** (1986) 651–664
18. Odobez, J.M., Bouthemy, P.: Robust multiresolution estimation of parametric motion models. Journal of Visual Communication and Image Representation **6** (1995) 348–365
19. Wallace, G.K.: The JPEG still picture compression standard. Communications of the ACM **34** (1991) 30–44
20. Comaniciu, D., Meer, P.: Mean Shift: A robust approach toward feature space analysis. IEEE Transactions on Pattern Analysis and Machine Intelligence **24** (2002) 603–619
21. Spaan, F., Lagendijk, R.L., Biermond, J.: Shape coding using polar coordinates and the discrete cosine transform. In: Proceedings of International Conference on Image Processing. (1997) 516–519
22. Zaletelj, J., Pecci, R., Spaan, F., Hanjalic, A., Lagendijk, R.L.: Rate distortion optimal contour compression using cubic B-splines. In: Proceedings of European Signal Processing Conference. (1998) 1497–1500
23. Weber, M.: AutoTrace: A utility for converting bitmap into vector graphics (2004) `http://autotrace.sourceforge.net`.
24. Clark, J.H.: A fast scan-line algorithm for rendering parametric surfaces. IEEE Transactions on Image Processing **13** (1979) 289–299
25. Burrows, M., Wheeler, D.J.: Block-sorting lossless data compression algorithm. Technical Report 124, SRC, Palo Alto, USA (1994)
26. Salomon, D.: Data compression: The complete reference. Springer Verlag (1998)
27. Kuglin, C.D., Hines, D.C.: The phase correlation image alignment method. In: Proceedings of IEEE International Conference on Cybernetics and Society. (1975) 163–165
28. Borgefors, G.: Distance transformations in digital images. Computer Vision, Graphics, and Image Processing **34** (1986) 344–371
29. Frigo, M., Johnson, S.G.: FFTW: Library for computing the discrete Fourier transform (2005) `http://www.fftw.org`.
30. Woo, M., Davis, T., Sheridan, M.B.: OpenGL Programming Guide: The Official Guide to Learning OpenGL. Addison-Wesley (1999)
31. Held, M.: FIST: Fast industrial-strength triangulation of polygons. Algorithmica **30** (2001) 563–596

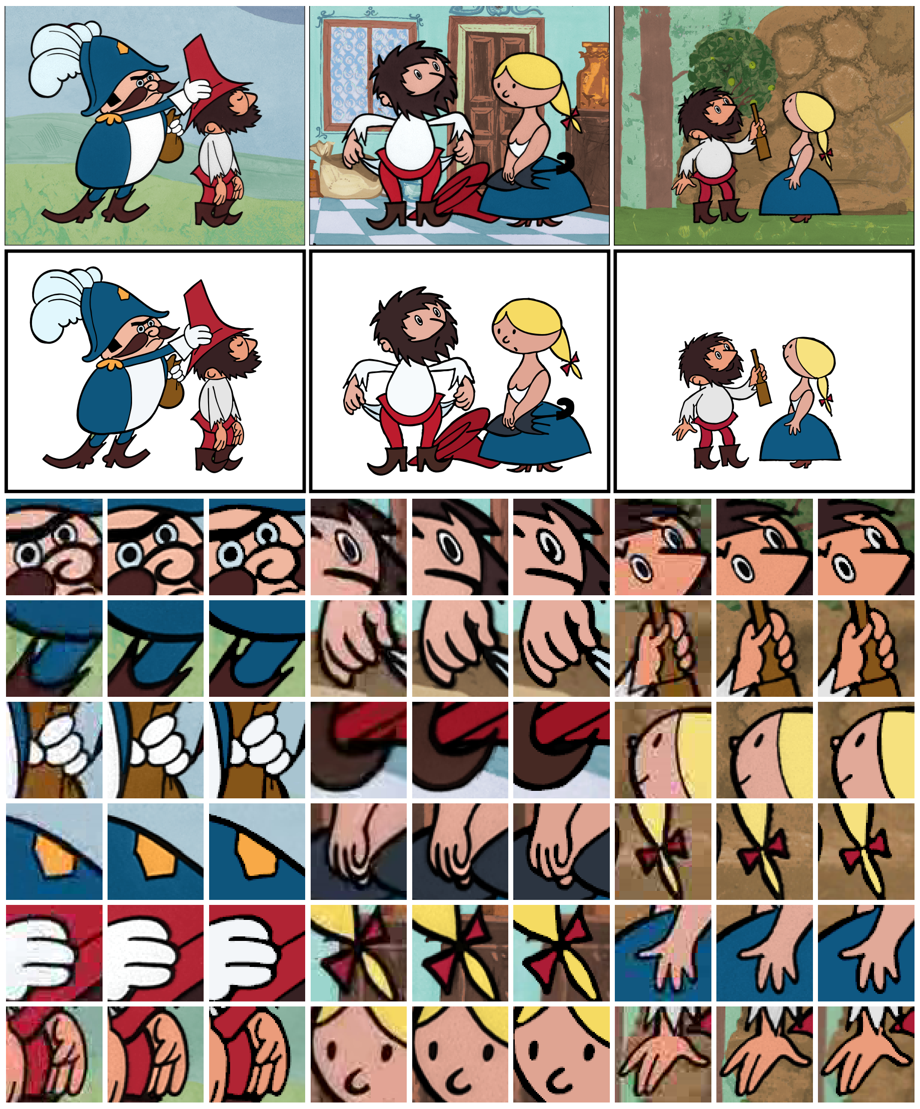

Fig. 2. Still image results – in each column see: the original image (top), vectorization of the foreground layer (middle), and detail views of different compression techniques (bottom). Small nested columns: detail views of `DivX` with comparable storage requirements (left), the original image (middle), and the output of our codec (right). In the vectorization of the foreground layer see misclassified foreground regions (e.g. small region between general's boot and leg). They are not disturbing during the playback while the background layer is homogeneous at the same position (images in this figure are published with permission of © *Vít Komrzí*, *Universal Production Partners* and *Lubomír Celar*, *Digital Media Production*).