Fast Scalable Coding Based on a 3D Low Bit Rate Fractal Video Encoder

Vitor de Lima[1], Thierry Pinheiro Moreira[1], Helio Pedrini[1] and William Robson Schwartz[2]
[1]*Institute of Computing, University of Campinas*
Campinas, SP, Brazil, 13083-852
[2]*Department of Computer Science, Universidade Federal de Minas Gerais*
Belo Horizonte-MG, Brazil, 31270-010

Keywords: Video compression, fractal encoding, adaptive transcoding, scalable coding

Abstract: Video transmissions usually occur at a fixed or at a small number of predefined bit rates. This can lead to several problems in communication channels whose bandwidth can vary along time (e.g. wireless devices). This work proposes a video encoding method for solving such problems through a fine rate control that can be dynamically adjusted with low overhead. The encoder uses fractal compression and a simple rate distortion heuristic to preprocess the content in order to speed up the process of switching between different bit rates. Experimental results show that the proposed approach can accurately transcode a preprocessed video sequence into a large range of bit rates with a small computational overhead.

1 INTRODUCTION

Most video streaming services use a reliable point-to-point channel to transmit videos and usually the bit rate is fixed or can be changed only to a few different possibilities, which might cause visual interruptions during the transmissions if the available bandwidth of the channel is variable (Quinlan et al., 2015; Wien et al., 2007; Zhai et al., 2008). This behavior frequently occurs in wireless communication. Therefore, if the video server could adapt itself to the client's bandwidth, the user would have both the best possible quality given the available bandwidth and the least amount of interruptions.

A proposed solution to this problem is called transcoding (Garrido-Cantos et al., 2013; Joset and Coulombe, 2013; Yeh et al., 2013), which converts the video stream into another one satisfying a given constraint. Most of the proposed methods (Ahmad et al., 2005) are extensions to well-known DCT-based video encoders and are capable of changing the frame rate, bit rate, spatial resolution or the standard used in the transmission. Another approach is scalable coding (Schwarz et al., 2007; Helle et al., 2013; Hinz et al., 2013), based on transmitting a single stream divided into layers that can be acquired separately according to the available bandwidth.

This work proposes an approach that compresses the video at the maximum desired transmission rate and includes some extra data. This data is used to transcode the compressed video to a large range of bit rates.

The target bit rate of this process can be dynamically adjusted with low overhead and the scalable coding algorithm is near optimal in a sense that it must only read the compressed file, execute a binary search in a table to find the correct operating parameters and write the resulting transcoded file.

To the best of our knowledge, the proposed approach is the first scalable coding method based on fractal video encoding. It relies exclusively on changing the resulting bit rate, taking advantage of the spatio-temporal independency of the fractal codes to avoid any changes to the frame rate or the spatial resolution.

In order to reduce the complexity of the algorithm, the encoding is extremely simplified when compared to other fractal-based approaches while still maintaining an acceptable rate-distortion performance. Perceptual quality comparisons with the x264 encoder are presented.

This paper is organized as follows. Section 2 briefly reviews some concepts related to this work. The proposed video encoder based on volumetric and searchless fractal methods is presented in Section 3. Experimental results with well-known video sequences are described and discussed in Section 4 and, finally, the conclusions and future work presented in Section 5.

2 BACKGROUND

This section presents a brief review of fractal image encoding, the searchless method for constructing collages, some related fractal video encoders available in the literature, and a perceptual quality metric used in image comparisons.

2.1 Fractal Image Encoding

Fractal image encoders (Schwartz and Pedrini, 2011) transmit a fractal that approximates the original image. The first of such methods was proposed in the seminal paper by Jacquin (Jacquin, 1992), where the method creates and transmits an operator, called collage, capable of reconstructing an approximation of the original image given a subsampled version of it. During the decoding process, the collage is applied to an arbitrary initial image, the result is subsampled and this process is repeated until the image converges to a fixed point.

The usual process used to construct the collage partitions the image into blocks (called range blocks) and each one of them is matched with a same-sized block in the subsampled image (called domain block) after being transformed by a prespecified function. This match is done either by exhaustive search or through the use of specialized heuristics. In general, the matching process is very time consuming; therefore, most fractal methods have extremely slow and complex encoding processes, but the decoding is usually significantly faster. The collage can rotate, flip or mirror the domain blocks and apply a transform into their gray level values, such as the one used by Tong and Pi (Tong and Pi, 2001)

$$G(D) = \alpha(D - \bar{D}J) + \bar{r}J \quad (1)$$

where G is the gray level transform, D is the downsampled domain block, $\bar{D}$ is the mean value for the domain block, $\bar{r}$ is the mean value for the block in the original scale (the range block), J denotes the unit matrix, and α is a scale parameter.

2.2 Searchless Fractal Encoding

The searchless fractal image coding was introduced by Furao and Hasegawa (Furao and Hasegawa, 2004) as a less complex alternative algorithm for constructing collages. In this approach, each range block has only one candidate domain block to be matched. If this matching does not achieve the desired reconstruction quality, the range block is divided into four blocks (which can be seen as a tree-based decomposition) and the process continues recursively until the range blocks reach a certain minimum size.

This approach was refined by Wu et al. (Wu et al., 2005) by dividing each range block in half either in the vertical or horizontal direction and without imposing any limits to the size of the range blocks. The smaller blocks have their $\bar{r}$ parameter more coarsely quantized than the larger ones, and blocks with only one or two pixels are forced to have their α equal to zero.

A 3D fractal video encoder based on an adaptive spatial subdivision data structure was proposed by Lima et al. (Lima et al., 2011a), which demonstrated to be efficient at very low bit rates.

2.3 Fractal Video Encoding

The first fractal video encoder was proposed by Hurd et al. (Hurd et al., 1992) by creating a collage that transforms the previous frame into the next one. This transform could use either blocks from the original scale or from a subsampled version of the frame. This approach was enhanced by Fisher et al. (Fisher et al., 1994) by varying the size of each used range block through a quadtree.

The fractal video encoding method proposed by Lazar and Bruton (Lazar and Bruton, 1994) and Li et al. (Li et al., 1993) used tridimensional collages that transform a subsampled version of a volume formed by consecutive frames into the original signal by matching volumetric range and domain blocks. Due to the extra dimension, this causes the encoding process to be even more time consuming when compared to the image encoders.

A faster variation of this volumetric approach was proposed by Chabarchine and Creutzburg (Chabarchine and Creutzburg, 2001) that uses a simpler gray scale transform, an extremely restricted domain pool for each range block and a simple spatial subdivision structure. This method was capable of encoding videos in real time, however, its rate-distortion performance was poor. Another refined volumetric fractal encoder was introduced by Yao and Wilson (Yao and Wilson, 2004) by using both vector quantization and domain blocks to approximate the signal. The approach could achieve a fair visual quality at low bitrates while being relatively fast, but its decoder suffered from convergence problems. A video encoder based on the compression of consecutive frame differences using sparse decomposition through matching pursuits was described by Lima and Pedrini (Lima and Pedrini, 2010).

2.4 Structural Dissimilarity

Most comparisons between video and image encoders are based on metrics derived from the sum of squared differences (SSD) or the mean squared error (MSE). The SSD and the MSE between two images A and B with size $W \times H$ are given by

$$\mathrm{SSD}(A,B) = \sum_{x=0}^{W-1}\sum_{y=0}^{H-1}(A_{x,y} - B_{x,y})^2 \qquad (2)$$

where $A_{x,y}$ and $B_{x,y}$ are the intensity of a pixel located at (x,y) in A and B, respectively.

$$\mathrm{MSE}(A,B) = \frac{\mathrm{SSD}(A,B)}{W \times H} \qquad (3)$$

The structural similarity (SSIM) index (Wang et al., 2004) was proposed as a metric for comparing images which is more properly correlated with human perception. It maps two images into an index in the interval $[-1,1]$, where higher values are given to more similar pairs of images, expressed as

$$\mathrm{SSIM}(A,B) = \frac{(2\mu_A\mu_B + c_1)(2\sigma_{AB} + c_2)}{(\mu_A^2 + \mu_B^2 + c_1)(\sigma_A^2 + \sigma_B^2 + c_2)} \qquad (4)$$

where μ_A, μ_B, σ_A^2 and σ_B^2 are the averages and variances of A and B, σ_{AB} is the covariance between A and B, and both c_1 and c_2 are predefined constants.

The structural dissimilarity (DSSIM) index is derived from the structural similarity that results in more distinct values, since a small variation in the original SSIM indicates a large difference in image quality. It is expressed as

$$\mathrm{DSSIM}(x,y) = \frac{1}{1 - \mathrm{SSIM}(x,y)} \qquad (5)$$

3 PROPOSED METHOD

The video encoder described in this section can be separated into two different modules. Section 3.1 describes a heuristic used to decide the number, the position and the volume of the range blocks and Section 3.2 describes how to encode volumetric blocks of pixels using fractal codes and how to split them.

3.1 Rate-Distortion Heuristic

Our method is based on the heuristic created by Saupe et al. (Saupe et al., 1998) for image compression with some adjustments to enable fast scalable coding of the compressed data and replacing the mean squared error (MSE) by the sum of squared differences (SSD), therefore, the total volume of each block contributes to the distortion measure.

Initially, a group of consecutive frames is preprocessed by encoding it at a relative high bit rate by using a predefined i_{upper} value. It is subdivided into a uniform grid of range blocks with $16 \times 16 \times 16$ pixels, which are encoded and inserted into the priority queue.

The rate-distortion heuristic creates a new pair of range blocks at each iteration which replaces the one taken from the queue. Instead of destroying this parent block, the algorithm then keeps all the range blocks created during the entire process marking them with the number of the iteration in which they were created.

At every Δi iterations, the group of frames is encoded by the arithmetic encoder several times to generate a table that associates how many iterations must be considered to satisfy each desired maximum rate. This reencoding process has a very low overhead since the rate-distortion heuristic creates, at each step, several versions of the same content with different bit rates and distortions. The transcoder reads this encoded group of frames, uses the table to choose a maximum number of iterations i_{target} that achieves the desired bit rate, and rewrites the file ignoring every block that was created after i_{target} and discarding both the iteration number associated with each block and the encoding parameters of the unused blocks.

Table 1: Uniform quantizers applied according to the volume of the range block.

Volume	Quantization step	Number of used bits
1	16	4
2	16	4
4	16	4
8	8	5
16	8	5
32	4	6
64	4	6
128	2	7
256	2	7
512	1	8
1024	1	8
2048	1	8
4096	1	8

The entire process is applied independently to each group consecutive frames allowing the target rate to be completely different for each part of the video during the scalable coding. Therefore, then the target rate can fluctuate during the transmission of the

video. Some encoders use Rate-Distortion Optimization (Ortega and Ramchandram, 1998) to adjust the rate of each group of frames in order to minimize the total distortion, but since it is impossible to preview the bandwidth of the transmission channel, this approach cannot be used in such a case.

3.2 Fast Fractal Block Video Encoding

During every iteration of the rate distortion heuristic, each range block of the group of frames is encoded by a generalization to three dimensions of the fractal image encoder described by Lima et al. (Lima et al., 2011b). Each range block with dimensions a, b and c at the coordinates (x, y, z) is matched against a domain block with dimensions equal to $2a$, $2b$ and $2c$ located at

$$\begin{aligned} x' &= x - a + p_x \times a/2 \\ y' &= y - b + p_y \times b/2 \\ z' &= z - c + p_z \times c/2 \end{aligned} \tag{6}$$

where parameters p_x, p_y and p_z must be equal to 0, 1 or 2. Range blocks with volumes smaller than 512 pixels have all these parameters set to 1. This relationship between the range and the domain block is illustrated in Figure 1. The domain block is transformed by Equation 1, before the matching, with the best α parameter chosen among the values in the set $\{0.25, 0.5, 0.75, 1.0\}$.

Each parent block is divided by choosing the direction (along horizontal, vertical or temporal axis) and the position of the cutting plane used to split it into two range blocks. This position is chosen in order to minimize the function $\sigma_A \times V_A + \sigma_B \times V_B$, where σ_A, σ_B are the variances of the resulting blocks and V_A, V_B are their volumes. The variances and averages of every block are calculated using integral volumes as described by Glassner (Glassner, 1990).

The resulting range blocks are encoded by transmitting their α and $\bar{r}$ parameters. Range blocks with any dimension smaller than four pixels have their α parameters set to zero.

All the required symbols and parameters are encoded using a context-adaptive arithmetic coder (Said, 2003). Each range block is encoded by its α parameter, which occupies 2 bits in the worst case, along with $\bar{r}$, which is quantized according to the range block volume as shown in Table 1. For range blocks with one or more dimensions smaller than 2 pixels, the only transmitted parameter is $\bar{r}$.

Along with these parameters, the spatial subdivision tree for each block in the initial uniform subdivision is encoded by a sequence of symbols pointing to the decoder, in a depth-first order, whether a certain region was subdivided or not, which direction it was split and the coordinate of the splitting plane. The α parameter and the binary decision symbols in the spatial subdivision tree have their own high order adaptive contexts, one for each possible value of $\lfloor \log V \rfloor$, where V is the total volume of the encoded block. The direction in which each block is split is encoded by another set of 3 high order adaptive contexts chosen according to the direction used to split its parent.

The $\bar{r}$ parameter is encoded as the difference between a quantized prediction and the real quantized value. The prediction is calculated as the average of $\bar{r}$ of the neighboring blocks located at the top, to the left and behind the encoded block weighted by their area of contact. This difference is encoded by the Adaptive Goulomb-Rice code described by Weinberger et al. (Weinberger et al., 1996), using one context for each possible $\lfloor \log V \rfloor$ in the same manner as the other parameters.

The intermediary representation created by the encoder and supplied to the scalable coding process encodes the iteration number in which each block was created using the same Adaptive Goulomb-Rice code as $\bar{r}$ and it stores the $\bar{r}$ and α for every block, including the ones that were split by the rate-distortion heuristic (which are not included in the final stream sent to the decoder).

The decoding process is accelerated by three different and complementary methods. The initial volumetric image that is used in the first iteration of the decoder is composed by filling each range block with its mean (for more details, see the work by Moon et al. (Moon et al., 2000)). The used pixel intensity transform is the one proposed by Øien and Lepsøy (Øien and Lepsøy, 1995) with additional proofs and details given by Pi et al. (Pi et al., 2003). Each iteration is applied according to the Gauss-Seidel inspired method proposed by Hamzaoui (Hamzaoui, 1999), which uses only one image during the iterations to overwrite each range block with its updated contents. The use of these methods assures that the decoding process converges in 4 iterations or less, instead of the usual 8 to 10 iterations used by other fractal decoders.

4 EXPERIMENTAL RESULTS

The video sequences were encoded on an Intel Core 2 Duo E6750 processor, 2.66 GHz with 8GB of RAM running the Arch Linux operating system. The method was implemented using the C++ programming language without any SIMD optimizations. Each group had at most 32 frames in it because of

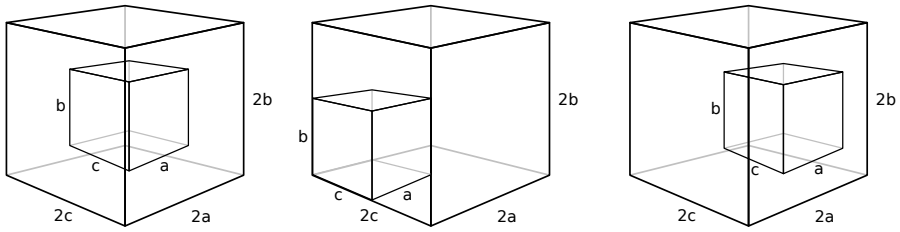

Figure 1: A few possible relationships between a range and a domain block in the proposed video encoder.

precision limits in the construction of the integral volumes and to ensure that every range block in the initial partition had at least one element in its domain pool.

The proposed approach is compared to the H.264 encoder, called x264 (x264, 2016), using the standard grayscale benchmark sequences 'Foreman', 'Car phone', 'Bus', 'Football', 'Akiyo', 'Hall monitor', 'Bowing', and 'Miss America' in the CIF format (CIPR, 2015). Table 2 presents, for each sequence, the number of frames, the average encoding time needed to generate the intermediate file used by the transcoder[1], the resulting file size of this initial encoding, and the final file size at the maximum available bit rate in the preprocessed file (i.e. without any extra data used by the scalable coding process).

The video sequences in the following experiments were encoded at low bit rates, which implies that the results had high distortions when measured by Equation 2. As demonstrated by Wang et al. (Wang et al., 2004), the ambiguity of the metrics derived from the SSD from a perceptual point of view is high and becomes even larger as the distortion increases. It is important to observe that both α and $\bar{r}$ parameters are quantized (i.e. they must assume one of a small set of possible values instead of being continuous), which causes a mean shift and a contrast change in every range block, even though the effect of these quantizations is perceptually negligible.

In order to ensure a proper comparison between both methods, the mean structural dissimilarity is used in the experiments. This metric is widely accepted for its simplicity and reasonably accuracy, being employed in the design of several image encoders, such as (Krause, 2010), and has a built-in implementation in the x264.

The bit rate was varied to closely match the same values in both encoders. As observed in Figure 2, the proposed encoder outperforms the x264 codec at very low bit rates in high motion sequences. The motion compensation algorithm of the H.264 encoder cannot operate properly in these conditions given that an accurate prediction of each frame would require a large amount of bits.

The x264 was configured to closely match the behavior of the proposed encoder by forcing it to insert a keyframe at every 32 frames and compiling it without any CPU specific optimizations. The Rate-Distortion heuristic, described in Section 3.1, was configured to use 125000 iterations for each entire sequence and Δi was set to 1000 iterations. The encoding process shown in Table 2 is executed only once and generates the compressed video combined with extra data to allow the fast scalable coding of the sequence. As shown in the last column of the table, the overhead of the extra data is quite large, almost doubling the size of the compressed video.

Figure 3 shows the speed of the scalable coding process. The required time to encode the sequences is at most 140 ms at the highest bit rates. Regardless the preprocessing step, which is performed only once and can be considered as an offline process, the proposed method is much faster than directly encoding the video with x264.

The scalable coding algorithm is limited only by how fast the data can be read and written since the total transcoding time is linearly correlated with the target rate. Figure 4 shows that the rate control is precise, achieving the desired constraint with a negligible error.

5 CONCLUSIONS

The proposed approach can rapidly transcode a preprocessed video sequence into a large range of different bit rates with extreme fine control of the resulting rate. It employs a fast fractal encoding method using volumetric range and domain blocks matched against each other using a generalization of a fast fractal encoder to three dimensions.

It is important to mention that the proposed rate

[1]It is important to mention that this process is executed only once and each intermediate file is stored and used when necessary.

Table 2: The number of frames for the video sequences, the encoding time of the proposed method and the size of each encoded file with and without the extra data used by the scalable coding process.

Sequence	# Frames	Time (s)	Size (KB)	Size without additional data (KB)
Foreman	300	5.8	390.5	194.7
Car phone	457	6.1	392.6	200.6
Bus	150	5.4	382.6	188.0
Football	260	5.5	387.1	193.5
Akiyo	300	5.0	336.8	170.5
Hall monitor	300	5.2	366.7	187.7
Bowing	300	5.0	343.0	175.6
Miss America	179	5.8	315.9	149.1

control and the transcoding heuristic could be applied to other encoding methods that are not based on fractals but are still adaptive. The near-optimal behavior of the transcoding algorithm, combined with better block encoding methods, could result in viable alternative to the current commonly used video encoders.

ACKNOWLEDGMENTS

The authors are thankful to the Minas Gerais Research Foundation (FAPEMIG), São Paulo Research Foundation (FAPESP grant #2015/12228-1) and Brazilian National Council for Scientific and Technological Development (CNPq grant #305169/2015-7) for their financial support.

REFERENCES

Ahmad, I., Wei, X., Sun, Y., and Zhang, Y.-Q. (2005). Video Transcoding: An Overview of Various Techniques and Research Issues. *IEEE Transactions on Multimedia*, 7:793–804.

Chabarchine, A. and Creutzburg, R. (2001). 3D Fractal Compression for Real-Time Video. In *2nd International Symposium on Image and Signal Processing and Analysis*, pages 570–573, Pula, Croatia.

CIPR (2015). Sequences. `http://www.cipr.rpi.edu/resource/sequences/`.

Fisher, Y., Rogovin, D., and Shen, T. (1994). Fractal (Self-VQ) Encoding of Video Sequences. *Visual Communications and Image Processing*, 2308(1):1359–1370.

Furao, S. and Hasegawa, O. (2004). A Fast No Search Fractal Image Coding Method. *Signal Processing: Image Communication*, 19(5):393–404.

Garrido-Cantos, R., De Cock, J., Martnez, J., Van Leuven, S., and Garrido, A. (2013). Video Transcoding for Mobile Digital Television. *Telecommunication Systems*, 52(4):2655–2666.

Glassner, A. S. (1990). Graphics Gems. In *Multidimensional Sum Tables*, pages 376–381. Academic Press Professional, Inc., San Diego, CA, USA.

Hamzaoui, R. (1999). Fast Iterative Methods for Fractal Image Compression. *Journal of Mathematical Imaging and Vision*, 11:147–159.

Helle, P., Lakshman, H., Siekmann, M., Stegemann, J., Hinz, T., Schwarz, H., Marpe, D., and Wiegand, T. (2013). A Scalable Video Coding Extension of HEVC. In *Data Compression Conference*, pages 201–210, Snowbird, UT, USA.

Hinz, T., Helle, P., Lakshman, H., Siekmann, M., Stegemann, J., Schwarz, H., Marpe, D., and Wiegand, T. (2013). An HEVC Extension for Spatial and Quality Scalable Video Coding. In *Proc. SPIE*, volume 8666, pages 866605–866605–16.

Hurd, L., Gustavus, M., and Barnsley, M. (1992). Fractal Video Compression. In *Thirty-Seventh IEEE Computer Society International Conference*, pages 41–42.

Jacquin, A. (1992). Image Coding Based on a Fractal Theory of Iterated Contractive Image Transformations. *IEEE Transactions on Image Processing*, 1(1):18–30.

Joset, D. and Coulombe, S. (2013). Visual Quality and File Size Prediction of H.264 Videos and Its Application to Video Transcoding for the Multimedia Messaging Service and Video on Demand. In *IEEE International Symposium on Multimedia*, pages 321–328.

Krause, P. K. (2010). FTC - Floating Precision Texture Compression. *Computers and Graphics*, 34(5):594–601.

Lazar, M. and Bruton, L. (1994). Fractal Block Coding of Digital Video. *IEEE Transactions on Circuits and Systems for Video Technology*, 4(3):297–308.

Li, H., Novak, M., and Forchheimer, R. (1993). Fractal-Based Image Sequence Compression Scheme. *Optical Engineering*, 32(7):1588–1595.

Lima, V. and Pedrini, H. (2010). A Very Low Bit-rate Minimalist Video Encoder based on Matching Pursuits. In *15th Iberoamerican Congress on Pattern Recognition*, pages 176–183, São Paulo, SP, Brazil. Springer-Verlag.

Lima, V., Schwartz, W. R., and Pedrini, H. (2011a). Fast Low Bit-Rate 3D Searchless Fractal Video Encod-

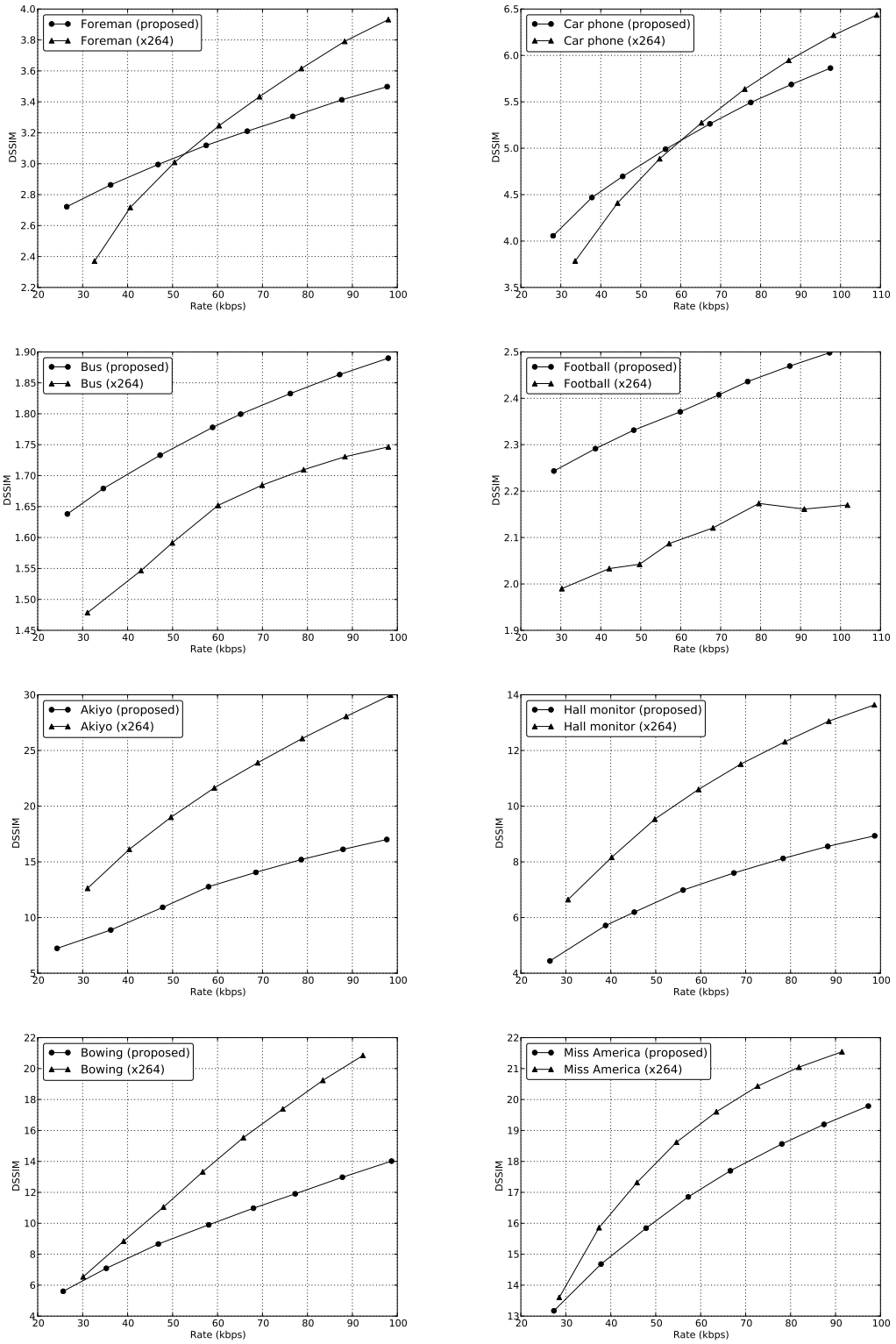

Figure 2: Mean structural dissimilarity at different rates for the proposed video encoder and the x264 encoder.

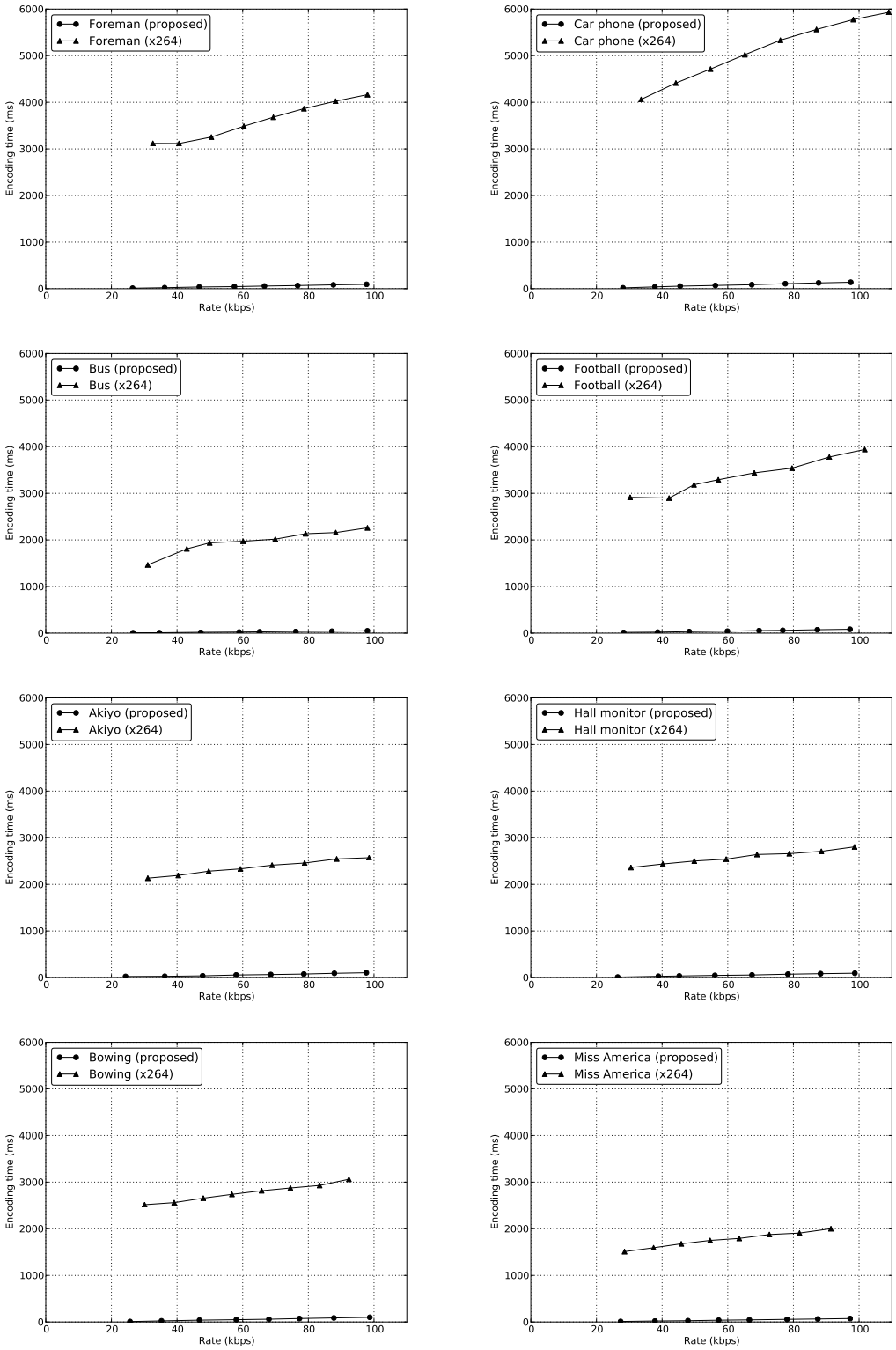

Figure 3: Encoding time at different rates for the video transcoder and the x264 encoder.

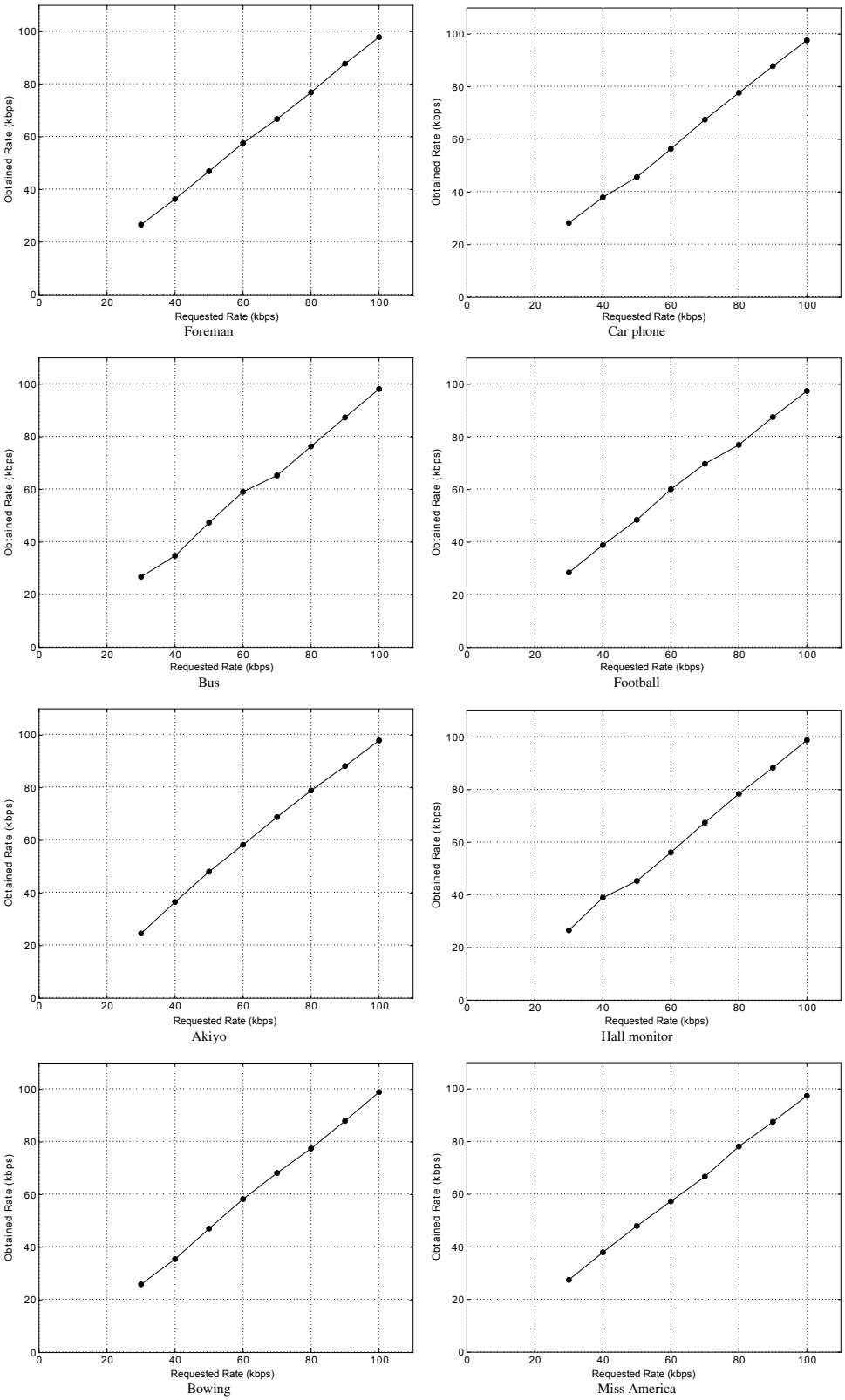

Figure 4: Requested and obtained rates for the proposed method.

ing. In *24th SIBGRAPI Conference on Graphics, Patterns and Images*, pages 189–196, Maceio, AL, Brazil. IEEE.

Lima, V., Schwartz, W. R., and Pedrini, H. (2011b). Fractal Image Encoding Using a Constant Size Domain Pool. In *VII Workshop of Computer Vision*, pages 137–142, Curitiba, PR, Brazil.

Moon, Y. H., Kim, H. S., and Kim, J. H. (2000). A Fast Fractal Decoding Algorithm based on the Selection of an Initial Image. *IEEE Transactions on Image Processing*, 9(5):941–945.

Øien, G. and Lepsøy, S. (1995). *A Class of Fractal Image Coders with Fast Decoder Convergence*, chapter Fractal Image Compression, pages 153–175. Springer-Verlag, London, UK.

Ortega, A. and Ramchandram, K. (1998). Rate-Distortion Methods for Image and Video Compression. *IEEE Signal Processing Magazine*, 15(6):23–50.

Pi, M., Basu, A., and Mandal, M. (2003). A New Decoding Algorithm based on Range Block Mean and Contrast Scaling. In *International Conference on Image Processing*, volume 3, pages II – 271–4, Barcelona, Spain.

Quinlan, J. J., Zahran, A. H., Ramakrishnan, K. K., and Sreenan, C. J. (2015). Delivery of Adaptive Bit Rate Video: Balancing Fairness, Efficiency and Quality. In *IEEE International Workshop on Local and Metropolitan Area Networks*, pages 1–6.

Said, A. (2003). *Lossless Compression Handbook*, chapter Arithmetic Coding. Communications, Networking, and Multimedia. Academic Press.

Saupe, D., Ruhl, M., Hamzaoui, R., Grandi, L., and Marini, D. (1998). Optimal Hierarchical Partitions for Fractal Image Compression. In *IEEE International Conference on Image Processing*, pages 737–741, Chicago, IL, USA.

Schwartz, W. R. and Pedrini, H. (2011). Improved Fractal Image Compression based on Robust Feature Descriptors. *International Journal of Image and Graphics*, 11(04):571–587.

Schwarz, H., Marpe, D., and Wiegand, T. (2007). Overview of the Scalable Video Coding Extension of the H.264/AVC Standard. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1103–1120.

Tong, C. and Pi, M. (2001). Fast Fractal Image Encoding based on Adaptive Search. *IEEE Transactions on Image Processing*, 10(9):1269–1277.

Wang, Z., Bovik, A., Sheikh, H., and Simoncelli, E. (2004). Image Quality Assessment: From Error Visibility to Structural Similarity. *IEEE Transactions on Image Processing*, 13(4):600 –612.

Weinberger, M., Seroussi, G., and Sapiro, G. (1996). LOCO-I: A Low Complexity, Context-based, Lossless Image Compression Algorithm. In *Data Compression Conference*, pages 140–149, Snowbird, UT, USA. IEEE Computer Society.

Wien, M., Cazoulat, R., Graffunder, A., Hutter, A., and Amon, P. (2007). Real-Time System for Adaptive Video Streaming Based on SVC. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(9):1227–1237.

Wu, X., Jackson, D., and Chen, H. (2005). Novel Fractal Image-Encoding Algorithm Based on a Full-Binary-Tree Searchless Iterated Function System. *Optical Engineering*, 44(10):107002–107014.

x264 (2016). Video Encoder. `http://www.videolan.org/developers/x264.html`.

Yao, Z. and Wilson, R. (2004). Hybrid 3D Fractal Coding with Neighbourhood Vector Quantisation. *EURASIP Journal on Applied Signal Processing*, 2004:2571–2579.

Yeh, C.-H., Jiang, S.-J. F., Lin, C.-Y., and Chen, M.-J. (2013). Temporal Video Transcoding Based on Frame Complexity Analysis for Mobile Video Communication. *IEEE Transactions on Broadcasting*, 59(1):38–46.

Zhai, G., Cai, J., Lin, W., Yang, X., and Zhang, W. (2008). Three Dimensional Scalable Video Adaptation via User-End Perceptual Quality Assessment. *IEEE Transactions on Broadcasting*, 54(3):719–727.