EXAMPLE-BASED IMAGE COMPRESSION

Jing-Yu Cui, Saurabh Mathur, Michele Covell, Vivek Kwatra, Mei Han*

Google Research, Google Inc., Mountain View CA 94043

ABSTRACT

The current standard image-compression approaches rely on fairly simple predictions, using either block- or wavelet-based methods. While many more sophisticated texture-modeling approaches have been proposed, most do not provide a significant improvement in compression rate over the current standards at a workable encoding complexity level. We re-examine this area, using example-based texture prediction. We find that we can provide consistent and significant improvements over JPEG, reducing the bit rate by more than 20% for many PSNR levels. These improvements require consideration of the differences between residual energy and prediction/residual compressibility when selecting a texture prediction, as well as careful control of the computational complexity in encoding.

***Index Terms*—** Image compression, Texture analysis

1. INTRODUCTION

Current standardized compression approaches can be thought of as being wavelet- or block-based models of image structure. The wavelet-based models, such as JPEG2000 [1], model image correlations as being localized in a simple spatial-frequency dependent manner. While this corresponds well to the most general image statistics, it fails to model changes seen at texture and object boundaries, resulting in "ringing" across quantized frequency bands, with the spatial extent of the ringing dependent on the frequency band.

This frequency-dependent error diffusion is very different from the type of error seen with block-based compression approaches, such as JPEG [2] and H.264 [3]. With block-based prediction, the quantization error associated with texture or object boundaries is typically localized to the block in which it occurs. This difference means that, for block-based methods, many visible artifacts are associated with the blocking structure of the compression approach itself.

A variety of image compression approaches try to combine spatial locality with better across-frequency modeling, using spatially-based texture modeling [4, 5, 6]. In past work, these methods have either suffered from a high overhead cost, describing the parameterized texture models [6], or prohibitively high computational complexity in encoding [4].

In Section 2, we consider example-based texture modeling for compression, but minimize the overhead of the model communication using previously transmitted image structure to predict which examples are most likely to provide good predictions. Instead of simply using mean-square error to select the best prediction, we move to a model that includes the encoding rate for the predictor and the prediction residual. In Section 3, we develop an approach which minimizes the costs associated with communicating the selected prediction. Section 4 explicitly addresses the encoding complexity and provides a heuristic approach that allows us to encode large images efficiently. We conclude with results (Section 5) and future work (Section 6).

*currently at Stanford University, Stanford CA 94305

2. TEXTURE PREDICTION BY DICTIONARY

Example-based texture prediction can be done using a fixed dictionary, as with vector quantizers and texture samplers [7], or using the previously communicated patches of the image. The advantage of using (reconstructed) patches from the same image is the spatial dependencies within textured regions will often provide better matches to the current region of interest. Our results are drawn from this second approach to dictionary formation, but since the overall approach is easily used with a fixed dictionary as well, we discuss our approach according to that terminology. For conceptual purposes, we can consider all previously transmitted image patches as being our dictionary.

Whichever way we obtain the dictionary, for each block within our target image (that is, the to-be-compressed image), we need to pick a prediction that will give the best rate-distortion trade off for that part of the image. To select the dictionary entry that provides the best compression, ideally we would run through the full analysis-synthesis cycle for each predictor in the dictionary. This would involve iterating through the dictionary textures, using each term as the prediction for the current block, noting how many bits will be needed to transmit the (quantized) compressed residual between this prediction and the target block and how many bits will be needed to communicate that choice for the predictor as well as the error that remains in the decoded image after combining the dictionary texture with the (quantized) compressed residual.

To complete this analysis-synthesis process, we need to decide how the predictor selection will be encoded and how the residual will be compressed and encoded. We discuss the predictor-selection encoding in Section 3, which leaves the residual encoding, discussed next.

A simple, but inefficient, method for encoding the residual would be to simply transmit the quantized residual in the spatial domain, with no further compression beyond what is provided by quantization and single-coefficient entropy coding. This approach is very inefficient, taking an average of 2.5 bits per pixel to transmit the residual, but corresponds well with using the quantized residual energy as the predictor selection criteria [3]. Using more efficient compression on the residual, such as described next, requires additional processing to pick the best predictor.

To take advantage of the remaining spatial correlation in the residual, we use the standard Discrete Cosine Transform (DCT)

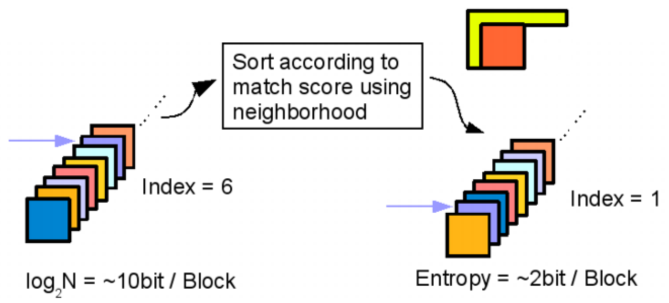

Sort according to match score using neighborhood
Index = 6
Index = 1
log$_2$N = ~10bit / Block
Entropy = ~2bit / Block

Fig. 1: The predictor patch, selected from our dictionary, is encoded indirectly, using a rank ordering based on the match quality of previously transmitted neighbor pixels. The encoder-selected predictor patch based on the choice that leads to the minimum number of bits for a given reconstruction error.

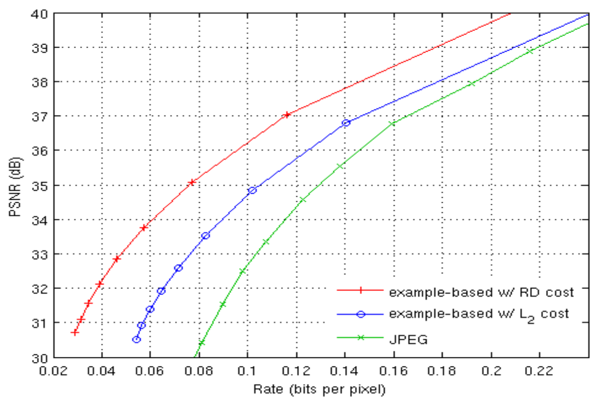


Fig. 2: Rate-distortion curves for example-based coding, using different predictor selections, averaged across 160 test images. (JPEG RD curve included for reference.)

residual compression scheme, which is used in block-based image and video compression. The residual from the block is transformed into the frequency domain, using a DCT. This residual is then quantized according to a quality parameter, which is in turn translated into a quantization step size that increases with the spatial frequency of the DCT coefficient, similar to what is proposed by [8]. The non-zero coefficients are encoded in the standard zig-zag-scan order, with zero-run-length encoding and omission of high-frequency, isolated non-zero components. This type of run-length encoding in the residual frequency domain, with a bias towards the non-zero entries occurring in the lower-frequency ranges, provides the best residual compression. This method of residual encoding is also responsible for the failings of the L_2 residual measure in ranking the predictor patches according to final compression rate. The predictors that result in quantized residuals with a few distinct non-zero coefficients, all at low spatial frequencies, will take far fewer bits to transmit than those with quantized residuals with lower total energy spread over many frequency coefficients, especially if a significant number of those non-zero coefficients are at higher frequencies.

By running the actual analysis-synthesis process for each prediction patch, we can accurately determine the number of bits needed for each of these options as well as the image distortion introduced by the quantization. The distortion is measured at the encoder by comparing the original residual with the transmitted residual. The bit rate is measured by combining the count of the residual-related bits with the predictor-selection bits, which is discussed in the next section.

3. EFFICIENT OVERHEAD ENCODING

The bit stream must include enough information that the decoder can determine what predictor was used by the encoder. One simple approach uses a fixed encoding, where the bit-stream symbol associated with a given dictionary entry does not change but instead is based on some measure of average frequency of use. For dictionaries formed from previously transmitted sections of the image, this fixed-symbol approach could correspond to the offset between the predictor texture and the target texture. For the dictionary sizes and distributions that we tested, this fixed-symbol approach took an average of 10 bits per image block to communicate.

In our experiments, a context-aware approach to symbol assignment provided the best compression. Of the approaches that we tried, using the previously transmitted neighbor pixels to the target block (the yellow “L” in Figure 1) gave the best results. With this approach, the dictionary entries are associated with a similar set of boundary pixels. When the dictionary entries are reconstructions from previously transmitted sections of the image, these extra boundary pixels can be taken directly from those reconstructions.

Using the known “L” boundary pixels from the current target, we can rank the dictionary entries, based on how well the corresponding boundary pixels from each entry match the target-block boundary. We use simple L_2 distance to measure this match quality on the boundary. Since the decoder and the encoder can both rank the dictionary entries in this way, we can use this rank to define the symbol that is used for each dictionary entry, at each new target block encoding. The chosen predictor typically has a small rank, owing to the correlation between the L_2 matching of “L” shapes and the true analysis-synthesis based matching of complete blocks. This fact, along with the use of entropy coding for the ranks, allows us to reduce the number of bits needed to communicate the block predictor from 10 bits per block down to 2 bits per block.

Additionally, since we are pulling predictor blocks from a single image (that is, the previous sections of the to-be-encoded image), we modified the predictor encoding process slightly to reward spatial continuity in the prediction process: the predictor patch that is spatially consistent with the predictors used by the previous neighboring blocks is assigned the top rank in the decoder/encoder predictor rankings for the current block. In this way, we bias our predictor selection and minimize block boundary artifacts.

4. COMPUTATIONAL COST

The full analysis-synthesis approach provides us with the best choice for compressing each image block. Based on our experiments (Figure 2), this allows us to improve our compression rate by 6–7 bits *per* 16×16 *image block* over what we would achieve using a simple L_2 selection. However, the cost of this full evaluation is impractical. Instead, we complete this evaluation for our final selection but only after limiting our range of choices.

We limit our selection process in stages. At the first cut, we limit the selection to the top $N = 1024$ dictionary entries, based on the ranking given by the "L"-boundary match. This first cut also allows us to assign variable length symbols for a known size alphabet ($N = 1024$ symbols), which make the encoding more efficient.

Even with only 1024 predictors to evaluate, the full rate-distortion evaluation is too computationally expensive for efficient encoding: the encoder would be about 1024 times slower than a simple JPEG encoding process, since the forward and reverse DCT evaluations would need to be repeated that often for each block.

To avoid this encoding inefficiency, we want to have a computationally efficient method that will give rankings that are similar to the full rate-distortion evaluation. As is suggested by previous approaches [3], the residual energy is a fairly accurate measure of the expected rate-distortion given by a prediction: it is not as good as the full analysis-synthesis selection process, but it is good enough to further limit the set of candidate predictors for final evaluation.

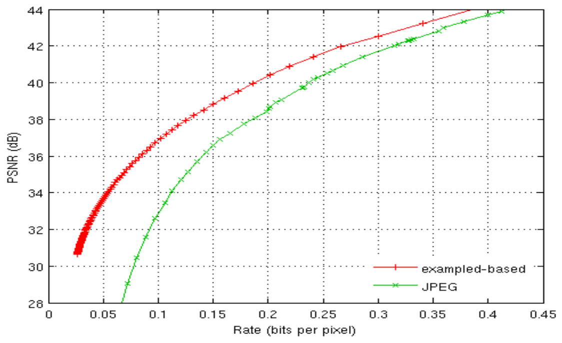
Using this insight, on the encoder side, we re-rank the $N = 1024$ predictors that pass the "L"-boundary criteria according to their residual energy. The M candidates that have the lowest residual energy are then re-evaluated by the analysis-synthesis criteria, for final selection of the encoder predictor that will be used. In our experiments, we have found that this bit-rate improvement curve quickly approaches the lowest rate given by the full-RD-evaluation approach: $M = 2$ and $M = 6$ already recover 25% and 50%, respectively, of the full RD-evaluation savings at a small fraction of the computational cost. Within this $M = 2 - 6$ range, the selected M is clearly a trade-off between encoder compression and encoder speed: generally, the encoder speed will be approximately M times slower than simple JPEG encoding.

5. EXPERIMENTAL RESULTS

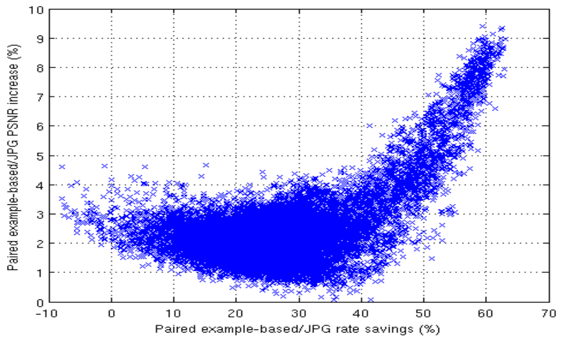
To test our approach, we used 16,473 Google 768×1024 Street View images. This set includes an interesting mix of natural objects (trees, bushes, people) and man-made artifacts (buildings, cars, roads). It was selected as a clean source of compression-artifact-free images as well as a strong test in terms of differences from the small "standard" set that we used for developing our compression approach and settings.

We compare the results of our approach, using as the dictionary, the previously reconstructed blocks from the same image, with the JPEG implementation in *libjpeg* version 6.2 from the Independent JPEG Group [9]. At the compression rates we examine, JPEG and JPEG2000 have very similar rate-distortion (R-D) curves, using PSNR: this conclusion is based on a comparison of the JPEG and JPEG2000[1] PSNR R-D curves, evaluated over a small subset of our testing images and at compression rates between 0.05 and 0.4 bits per pixel. Similar conclusions have been reported in other studies of non-perceptual comparisons of JPEG and JPEG2000 [11]. By choosing JPEG as our reference instead of JPEG2000, we see the same reference performance levels at significantly lower computation. Using JPEG as our baseline also provides us with directly comparable compression artifacts, since our example-based coding is also block based and most of its artifacts are at block boundaries.

[1]The JPEG2000 encoder we used for this comparison was Kakadu Software version 6.0 [10].



(a)



(b)

Fig. 3: Rate-distortion comparisons of example-based coding and JPEG on 16,473 test images. (a) Rate-distortion rates (b) Paired analysis at 37-dB quality settings.

We compressed the entire set of 16K images using both approaches (JPEG as well as our approach) at 100 different quality levels, and computed average statistics (bit-rate and PSNR) over all images for each quality level individually. We also ran a paired comparison of the two compression methods: on each test image, we picked the quantization levels for a 37-dB quality level and then plotted the percent bandwidth savings versus quality change, per image. As shown in Figure 3, our approach consistently does better than JPEG, with the most pronounced savings at lower PSNR values. Based on our paired analysis (Figure 3b), the example-based approach uses 27% less bandwidth ($\sigma = 11\%$) on the paired compressions, while improving the quality by 2% ($\sigma = 1\%$). Due to the large sample size (along with the paired analysis), these changes are statistically highly significant, with both giving a paired Student's $t > 200$.

Figure 4 shows a qualitative comparison of our results with JPEG. We compressed the image using our technique to 15KB and 34.6dB PSNR. For comparison, we show two JPEG results, one matching our compression rate (Figure 4c) and the other matching our PSNR (Figure 4d). At equal compression rates, we improved 2.8 dB over JPEG (34.6 dB vs. 31.8dB). At equal PSNR, we reduced bandwidth by 32% over JPEG (15KB vs. 22KB). Figure 4(d,e) illustrates the relative spatial error distribution of the two approaches (compared at the same PSNR): in each case, we only plot those pixels where the corresponding approach does better than the other; the brightness of a pixel indicates the amount of gain (difference in residual errors).

Overall, our approach has lower error at almost twice more pix-

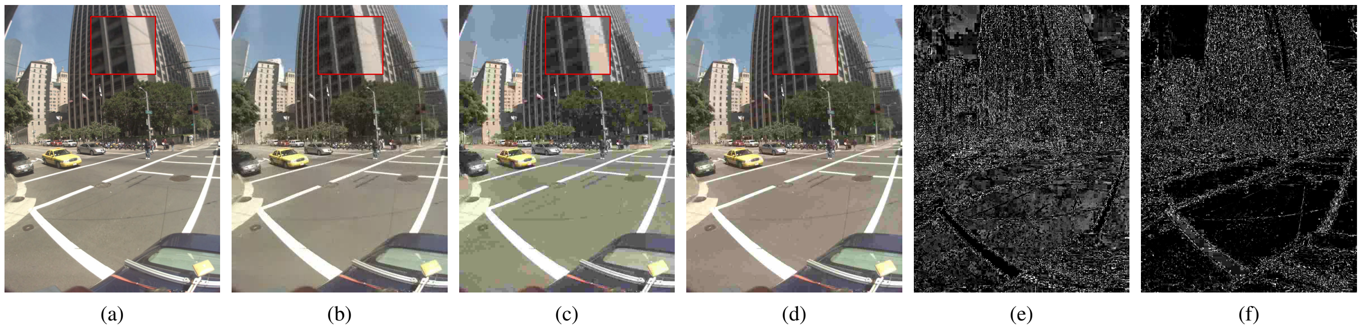


Fig. 4: Comparison with JPEG (zoomed inset view). (a) Original image, (b) Our result (15KB, 34.6dB), (c) JPEG result (15KB, 31.8dB), (d) JPEG result (22KB, 34.6dB), (e,f) Bright pixels have lower error for (e) our method and (f) JPEG at same PSNR.

els than JPEG. For our approach, the gains are strongly aligned with repetitive structures, such as window edges in buildings, and lightly textured regions such as the sky and the road. Intuitively, our example-based prediction does a better job at predicting such repetitive structures. We also observe that in lightly-textured regions, our approach has fewer blocking and color artifacts compared to JPEG. On the other hand JPEG's gains are more spatially diffuse: for example, it performs well on the wide, white stripes on the road, where the stripe width covers several full JPEG blocks and has a flat internal texture.

6. CONCLUSION AND FUTURE WORK

We have developed a general approach to image compression based on a dictionary of predictor patches. The quality of the compression is improved by selecting the predictor using repeated analysis-synthesis evaluation of the available choices. The transmission overhead of the predictor is minimized using dynamic symbol assignment, where the assignment is based on the match quality of previously transmitted neighbor pixels for the current location and the predictor. Finally, the encoding complexity is controlled by limiting the number of analysis-synthesis evaluations that are done for each compression block to only those predictor patches that are most likely to provide good results. In this way, we benefit from the more complex texture models that can be included in an example-based dictionary, without losing its compression benefits in predictor overhead and without creating an unusably complex encoder.

Our overall improvement in compression quality was fairly constant at 0.05–0.06 bits per pixel, up to around 40 dB PSNR, at which point it drops to merely even with JPEG, just above 45 dB PSNR. At 37 dB PSNR, this is a 27% bandwidth savings. This compression savings was achieved by reducing the bits used to encode the residual by about 30%, through texture prediction with a predictor communication overhead of 2-3 bits per 16×16 block.

This approach is general to many other contexts, including audio and video encoding. It can also be used with dictionaries taken from other images, such as previous video frames or from standardized dictionaries. It can be extended to use separate predictions on different frequency bands, similar to the approaches used with wavelet-based methods.

7. REFERENCES

[1] M. Boliek, J. S. Houchin, and G. Wu, "JPEG 2000 next generation image compression system features and syntax," in *Proc. International Conference on Image Processing*, 10–13 Sept. 2000, vol. 2, pp. 45–48.

[2] Gregory K. Wallace, "The JPEG still picture compression standard," *Commun. ACM*, vol. 34, no. 4, pp. 30–44, 1991.

[3] T. Wiegand, G. J. Sullivan, G. Bjontegaard, and A. Luthra, "Overview of the H.264/AVC video coding standard," vol. 13, no. 7, pp. 560–576, July 2003.

[4] Michael F. Barnsley and Lyman P. Hurd, *Fractal image compression*, A. K. Peters, Ltd., Natick, MA, USA, 1993.

[5] Vivek Kwatra, Irfan Essa, Aaron Bobick, and Nipun Kwatra, "Texture optimization for example-based synthesis," in *SIGGRAPH '05: ACM SIGGRAPH 2005 Papers*, New York, NY, USA, 2005, pp. 795–802, ACM.

[6] Huamin Wang, Yonatan Wexler, Eyal Ofek, and Hugues Hoppe, "Factoring repeated content within and among images," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, New York, NY, USA, 2008, pp. 1–10, ACM.

[7] Li-Yi Wei, Jianwei Han, Kun Zhou, Hujun Bao, Baining Guo, and Heung-Yeung Shum, "Inverse texture synthesis," in *SIGGRAPH '08: ACM SIGGRAPH 2008 papers*, New York, NY, USA, 2008, ACM.

[8] H. A. Peterson, H. Peng, J. H. Morgan, and W. B. Pennebaker, "Quantization of color image components in the DCT domain," in *Proc. SPIE Human Vision, Visual Processing, and Digital Display*, 1991, pp. 210–222.

[9] Thomas G. Lane, "Independent JPEG group's libjpeg version 6.2," in *http://www.ijg.org*, 2008.

[10] Kakadu Software, "Kakadu JPEG2000 version 6.0," in *http://www.kakdusoftware.com*, 2009.

[11] D. Santa-Cruz and T. Ebrahimi, "A study of JPEG2000 still image coding versus other standards," in *Proc. European Signal Processing Conference*, 5–8 Sep. 2000, pp. 673–676.