Predictive graph construction for image compression

PREDICTIVE GRAPH CONSTRUCTION FOR IMAGE COMPRESSION

Giulia Fracastoro, Enrico Magli

Dept. of Electronics and Telecommunications, Politecnico di Torino, Italy

ABSTRACT

In this work, we propose a new method of graph construction for graph-based image compression. In particular, because of the overhead incurred by graph transmission to the receiver, we focus our attention to develop an efficient method to construct and to code the graph representation of the image. The proposed method employs innovative edge metrics, quantization and prediction techniques, leading to a compact yet high-quality graph, corresponding to a very efficient transform that performs very well on natural as well as piece-wise smooth images. We have tested our method on different images and, compared to the standard DCT, it provides an average quality gain of 1.6 dB.

***Index Terms*—** graph signal processing, graph Fourier transform, image compression

1. INTRODUCTION

The Discrete Cosine Transform (DCT) is the most common transform used for block-based image and video compression ([1]). One of the main drawbacks of the DCT is that it becomes inefficient when a block contains edges that are not horizontal or vertical. In this case, the resulting transform coefficients are not sparse and the high-frequency coefficients can have large magnitude. This leads to higher bitrate or visible artifacts around the edges.

To solve this problem, different solutions have been proposed. Variations of the DCT have been studied, such as directional DCT ([2]), shape-adaptive DCT ([3]) or adaptive block-size transform ([4]), in which the block shape is adapted to the edge location or the transform follows the edge direction. Wavelet approaches have also been proposed. In order to avoid filtering across edges, researchers have developed different wavelet filterbanks based on the image geometry, such as bandelets ([5]), curvelets ([6]) and directionlets ([7]). However, all these methods produce an efficient edge representation only when edges are straight lines and become inefficient with more complex edges.

Recently, the graph-based approach has been proposed, according to which high-dimensional data naturally reside on the vertices of graphs and they can be visualized as a finite collection of samples defined as graph signals, with one sample at each vertex of the graph [8]. In the last years, researchers have studied how to apply classical signal processing techniques in the graph domain. Techniques for filtering, translation, modulation and downsampling in the graph domain have been developed. Several graph transforms have also been proposed, such as the graph Fourier transform ([9]), spectral graph wavelets ([10]) and wavelet filterbanks ([11]).

Graphs have emerged has a useful representation also for processing and analysis of images. Indeed, images can be viewed as a graph, where each pixel represents a node of the graph and arc weights describe connectivity relations. In recent years, researchers have made some attempts for performing image and video compression using graph signal processing techniques. The authors in [12, 13] proposed an efficient compression method for depth map using the graph Fourier transform, but they reported unsatisfactory results on natural images that are not piecewise smooth. Instead, in [14] a method for image compression using graph-based wavelet transform is presented; also in this case satisfactory results are achieved only with piecewise smooth images. Another method for graph-based compression of piecewise smooth images is proposed in [15], where the graph is selected among a large space of possible ones to minimize the total signal representation cost. In [16] a method for image representation using multiscale graph transform is presented, achieving good compression performance, at the expense of using a very complex graph whose coding cost is neglected.

In general, while graph-transforms have been shown to be more efficient than conventional transforms, the overhead of graph transmission may easily outweigh the coding efficiency benefits. Therefore, it is very important to design graph representations and corresponding graph transforms that are efficient also when graph has to be transmitted to a decoder.

In this paper we propose a new method of graph construction for graph-based image compression, obtaining a graph that at the same time gives an efficient image representation and is not too complex. We introduce a new technique for defining the edge weights of the graph and efficiently coding them. One of the main novelty of our work is the development of a technique for edge prediction that nearly halves the cost for graph transmission. With the proposed method, we outperform existing techniques achieving an average gain of 2 dB in PSNR compared to the DCT transform.

This work was supported by Sisvel Technology.

2. GRAPH FOURIER TRANSFORM

A graph can be denoted as $G = (V, E)$, where V is the set of vertices (or nodes) with $|V| = N$ and $E \subset V \times V$ is the set of edges. It is possible to represent a graph by its weighted adjacency matrix $W \in \mathbb{R}^{N \times N}$, where w_{ij} represents the weight of the edge between node i and j and $w_{ij} = 0$ if there is no edge between i and j. The graph Laplacian is defined as $L = D - W$, where D is a diagonal matrix whose ith diagonal element d_i is equal to the sum of the weights of all edges incident to node i.

We can define a signal f on the vertices of the graph and it can be represented as a vector $\mathbf{f} \in \mathbb{R}^N$, where the ith component of $\mathbf{f}$ represents the signal value at the ith vertex in V.

In the graph domain, it is possible to define an equivalent of the Fourier transform, i.e. the graph Fourier transform ([8]). The graph Fourier transform $\hat{\mathbf{f}}$ of any signal $\mathbf{f} \in \mathbb{R}^N$ is defined as

$$\hat{\mathbf{f}} = U\mathbf{f},$$

where U is the matrix whose rows are the eigenvectors of the graph Laplacian L. The inverse graph Fourier transform is then given by

$$\mathbf{f} = U^T\hat{\mathbf{f}}.$$

3. PROPOSED GRAPH CONSTRUCTION TECHNIQUE

We now describe the proposed technique used to construct the weighted graph of an image. One of the main drawbacks of any graph compression technique is that the graph itself has to be transmitted to the decoder. For this reason, the cost of transmitting the graph should be as small as possible. In our work, we pay particular attention to develop techniques that simplify as much as possible the graph structure without a significant decrease in the compression performance.

3.1. Graph weight metric

The graph structure used is a square grid where each pixel is a vertex of the graph and is connected to each of its 4-connected neighbors. We have chosen this structure because it has been proved that, when the graph is a 4-connected grid and all edges have the same weight, the 2D DCT basis functions are eigenvectors of the graph Laplacian, and thus the transform matrix U can be the 2D DCT matrix ([17]).

The edge weights of the graph represent the similarity between the two pixels connected by the edge. Several weighting functions have been used to determine the edge weights in image processing applications, two of the most commonly used are the Cauchy function and the Gaussian function ([18]), that are defined as follows:

$$\text{Cauchy function:} \qquad w_{ij} = \frac{1}{1 + (\frac{d_{ij}}{\alpha})^2},$$

$$\text{Gaussian function:} \qquad w_{ij} = e^{-\frac{d_{ij}^2}{\sigma^2}},$$

where d_{ij} is the Euclidean distance between pixel i and j ($d_{ij} = |f_i - f_j|$) and α and σ are defined as in [19].

Most of the graph-based image compression methods present in the literature use an unweighted graph, even if in some case a weighted graph with Gaussian weights is used, as in [16]. Conversely, we propose to employ a Cauchy function to determine the weights. In Section 4.2 we show that this choice outperforms the Gaussian weights.

To reduce the influence of noise, we do not compute the edge weights using the original image, but first we smooth the image. It is important to use smoothing techniques that do not modify the edges present in the image, in our tests we used anisotropic diffusion [20].

3.2. Quantization

If we use a graph defined as in the previous section, the information we need to encode are only the edge weights, whereas the graph topology is fixed and there is no need to transmit it. Given an image block of n^2 pixels, its grid graph has $2n(n-1)$ edges, i.e. almost twice the number of pixels, which explains why it is extremely important to study techniques for reducing this overhead.

First, we decrease the number of possible edge weight values for each edge in the graph, to do this we quantize the argument of the weight function, i.e the distances d_{ij}. We have seen that their probability distribution can be approximated with a Laplacian, therefore, we use an uniform-threshold quantizer for Laplacian source ([21]) with an overload region.

Using this method, we can arbitrarily reduce the number of edge weight values down to two. In term of compression performance, the case with only two possible values is the most interesting because it has the best ratio between quality gain and cost for the graph transmission.

The graph obtained using only binary weight values is similar to an unweighted graph (i.e. with weights in $\{0,1\}$) such as the one used in [12, 13], with the important difference that, with the quantized weights, the graph cannot be disconnected. Indeed, if the graph is disconnected, the graph transform does not perform well, in particular when there are connected components with a small number of nodes.

In the following sections, we will focus on this binary case, developing an optimized graph compression scheme.

3.3. Graph edge prediction method

When we have only two possible edge weight values, the majority of the edges in the graph will typically have the higher edge weight, meaning that the two pixels that it connects are similar, instead only a small number of edges will have the lower edge weight, indicating that there is a discontinuity between the two pixels connected by the edge. Every node is

connected to four pixels. In order to structure our prediction mechanism, we consider nodes in raster order and, for each node, we consider two edges, namely the one that connects the pixel to the nearest one in the next column, and the one that connects the pixel to the bottom one in the next row. We label each pixel of the image as "edge" or "non-edge" pixel. Specifically, a pixel is labeled as an edge pixel if at least one of the corresponding edges has the lower edge weight, otherwise it is labeled as a non-edge pixel.

In order to obtain a more compact representation of the graph structure, we have developed a method of edge prediction that reconstructs the graph edges starting from a binary image that specify if each pixel has an edge or non-edge label. Analysing the labels of neighboring pixels, the encoder can predict whether the discontinuity is horizontal, vertical or diagonal, as shown in Figure 1. Then it constructs a new graph, setting the edge weights in accordance with the predicted discontinuities. The graph generation algorithm is explained in detail in Algorithm 1. The binary image containing the labels is compressed using JBIG [22] and transmitted to the decoder. Starting from the received labels, the decoder runs the graph construction algorithm and recovers the same graph used at the encoder.

As will be shown in Section 4.2, the graph provided by Algorithm 1 defines a graph Fourier transform with very high coding efficiency.

Algorithm 1 Prediction-based Graph Construction Algorithm. I_b: binary image, M: higher edge weight, m: lower edge weight

```
Set all edge weights equal to M;
for every edge pixel in I_b do
  N_hor= number of horizontal neighbors that are edge pixel;
  N_ver= number of vertical neighbors that are edge pixel;
  if N_hor > 0 then
    Set the vertical edge to m;
  end if
  if N_ver > 0 then
    Set the horizontal edge to m;
  end if
  if N_hor = 0 and N_ver = 0 then
    Set the vertical edge to m;
    Set the horizontal edge to m;
  end if
end for
```

3.4. Deletion of isolated edges

In order to obtain a smoother binary image and to remove small discontinuities, we delete the connected components present in the binary image whose dimension is smaller than a threshold value.

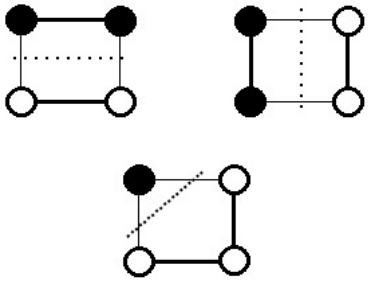

Fig. 1: Prediction-based graph construction method. The nodes represent the binary image transmitted to the decoder: the black ones are the edge pixels, the white ones the non-edge pixels. The dotted lines represent the predicted image discontinuity, the figure represents the three possible situations. The edges are set in accordance with the discontinuities: the ones with a thicker line have the higher weight, the ones with a thinner line have the lower weight.

(a) Clock (b) Cameraman (c) Airplane

Fig. 2: Original images.

4. EXPERIMENTAL RESULTS

To test the proposed method, we have subdivided the images in 32×32 blocks and we constructed the graph of each image block with the techniques discussed in the previous section. After having obtained the graph, we computed the adjacency matrix W and the Laplacian matrix L. We used as transform matrix the matrix U of the eigenvectors of the Laplacian, as defined in Section 2.

To code the transform coefficients we used a bit plane coding on each block and we estimated the bit rate computing the entropy of each bitplane.

We have applied our method to some standard images, three of which are shown in Figure 2. We have chosen different image types, some having very sharp edges, such as airplane, while others are more natural, such as cameraman.

4.1. Edge weight metric evaluation

We have compared the performance of the two weighting functions showed in Section 3.1 by computing the percentage of signal energy in function of the number of retained coefficients. We have found that the Cauchy function has better compression performance than the Gaussian function, as shown in Figure 3. For this reason, in our tests we used the Cauchy weighting function.

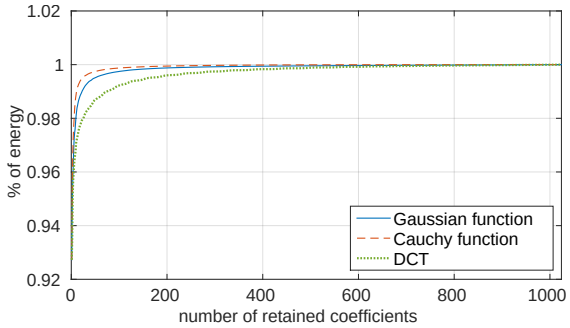

Fig. 3: Percentage of energy in function of the number of retained coefficients.

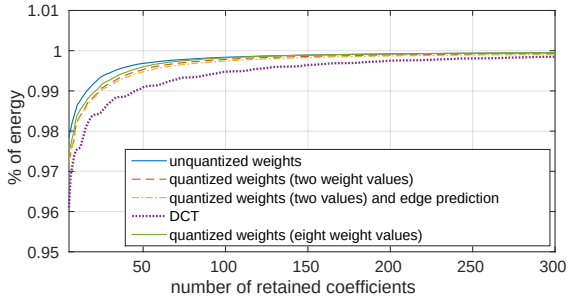

Fig. 4: Percentage of energy in function of the number of retained coefficients.

4.2. Graph compression

We have compared the performance of the graph transform using a graph with unquantized weights, with quantized weights and with the predicted weights. We computed the percentage of signal energy in function of the number of retained coefficients. The results are shown in Figure 4. We can see that using a small number of edge weight values reduces the performance but not in a significant way. Moreover, it is important to note that the edge prediction method produces a graph that is a very good approximation of the original one and the results obtained are nearly the same, but it nearly halves the size of the graph overhead, resulting in a high increase in the performance.

4.3. Image compression performance

To evaluate the performance of the proposed Predictive Graph Transform (PGT), we computed the PSNR of each image as a function of the bitrate. We compared the proposed PGT with the standard DCT and with the edge-adaptive transform (EAT) proposed by Shen et al. in [12]. In order to have a fair comparison, we used the same block dimension and the same method for coding the transform coefficients. For the EAT and our proposed transform, the bitrate also takes into account the cost of transmitting the graph. Figures 5 shows the results obtained. We can see that our method outperforms both the standard DCT and the EAT, with an average gain of approximately 1.6 dB over the DCT and a maximum gain of 3 dB.

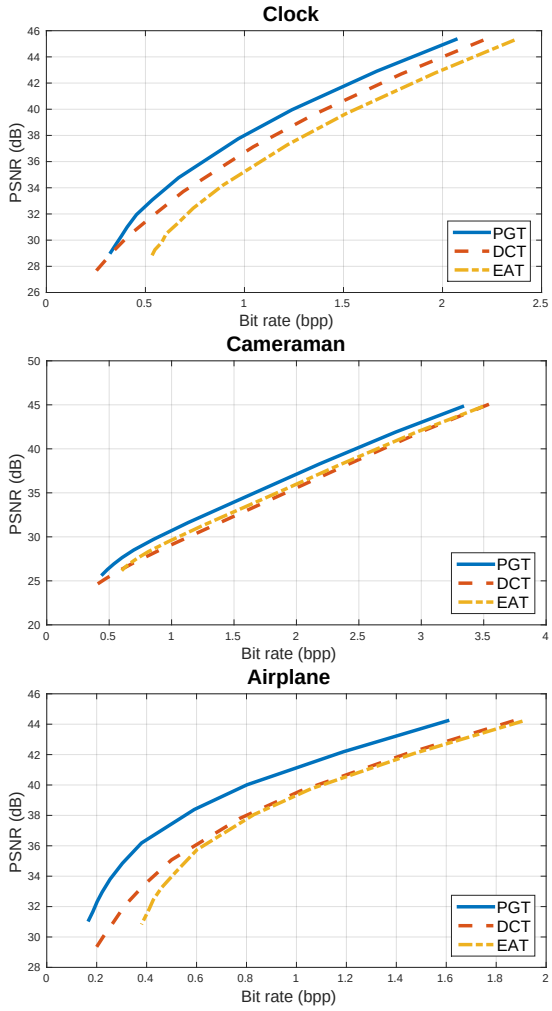

Fig. 5: RD curve comparison between our proposed method, DCT and EAT.

5. CONCLUSION AND FUTURE WORK

We have developed a new method of graph construction for image compression, investigating the best edge weight metric for our purpose, and defining a new technique to reduce the complexity of the graph without a significant decrease in the performance. Moreover, we have introduced an edge predictive technique that enable us to significantly reduce the overhead due to the graph transmission. The proposed method outperforms the standard DCT and other graph transforms present in the literature.

As future work, we will study an efficient way to code a graph with continuous edge weight values, and investigate the application of this method to the prediction error in video coding.

6. REFERENCES

[1] K. Sayood, *Introduction to data compression*, Newnes, 2012.

[2] B. Zeng and J. Fu, "Directional discrete cosine transforms-a new framework for image coding," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 18, no. 3, pp. 305–313, 2008.

[3] T. Sikora and B. Makai, "Shape-adaptive dct for generic coding of video," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 5, no. 1, pp. 59–62, 1995.

[4] M. Wien, "Variable block-size transforms for h. 264/avc," *Circuits and Systems for Video Technology, IEEE Transactions on*, vol. 13, no. 7, pp. 604–613, 2003.

[5] E. Le Pennec and S. Mallat, "Sparse geometric image representations with bandelets," *Image Processing, IEEE Transactions on*, vol. 14, no. 4, pp. 423–438, 2005.

[6] E.J. Candes and D.L. Donoho, "Curvelets: A surprisingly effective nonadaptive representation for objects with edges," Tech. Rep., DTIC Document, 2000.

[7] V. Velisavljevic, B. Beferull-Lozano, M. Vetterli, and P.L. Dragotti, "Directionlets: anisotropic multidirectional representation with separable filtering," *Image Processing, IEEE Transactions on*, vol. 15, no. 7, pp. 1916–1933, 2006.

[8] D.I. Shuman, S.K. Narang, P. Frossard, A. Ortega, and P. Vandergheynst, "The emerging field of signal processing on graphs: Extending high-dimensional data analysis to networks and other irregular domains," *Signal Processing Magazine, IEEE*, vol. 30, no. 3, pp. 83–98, 2013.

[9] G. Taubin, "A signal processing approach to fair surface design," in *Proceedings of the 22nd annual conference on Computer graphics and interactive techniques*. ACM, 1995, pp. 351–358.

[10] D.K. Hammond, P. Vandergheynst, and R. Gribonval, "Wavelets on graphs via spectral graph theory," *Applied and Computational Harmonic Analysis*, vol. 30, no. 2, pp. 129–150, 2011.

[11] S.K. Narang and A. Ortega, "Perfect reconstruction two-channel wavelet filter banks for graph structured data," *Signal Processing, IEEE Transactions on*, vol. 60, no. 6, pp. 2786–2799, 2012.

[12] G. Shen, W.S. Kim, S.K. Narang, A. Ortega, J. Lee, and H. Wey, "Edge-adaptive transforms for efficient depth map coding," IEEE, 2010, pp. 2808–2811.

[13] W.S. Kim, S.K. Narang, and A. Ortega, "Graph based transforms for depth video coding," in *Acoustics, Speech and Signal Processing (ICASSP), 2012 IEEE International Conference on*. IEEE, 2012, pp. 813–816.

[14] S.K. Narang, Y.H. Chao, and A. Ortega, "Critically sampled graph-based wavelet transforms for image coding," in *Signal and Information Processing Association Annual Summit and Conference (APSIPA), 2013 Asia-Pacific*. IEEE, 2013, pp. 1–4.

[15] W. Hu, G. Cheung, A. Ortega, and O.C. Au, "Multi-resolution graph fourier transform for compression of piecewise smooth images," *Image Processing, IEEE Transactions on*, vol. 24, no. 1, pp. 419–433, 2015.

[16] M. Hidane, O. Lezoray, and A. Elmoataz, "Lifting scheme on graphs with application to image representation," in *Signal and Information Processing (GlobalSIP), 2013 IEEE Global Conference on*. IEEE, 2013, pp. 431–434.

[17] C. Zhang and D. Florêncio, "Analyzing the optimality of predictive transform coding using graph-based models," *Signal Processing Letters, IEEE*, vol. 20, no. 1, pp. 106–109, 2013.

[18] L.J. Grady and J.R. Polimeni, *Discrete calculus: Applied analysis on graphs for computational science*, Springer, 2010.

[19] M.J. Black, G. Sapiro, D.H. Marimont, and D. Heeger, "Robust anisotropic diffusion," *Image Processing, IEEE Transactions on*, vol. 7, no. 3, pp. 421–432, 1998.

[20] P. Perona and J. Malik, "Scale-space and edge detection using anisotropic diffusion," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 12, no. 7, pp. 629–639, 1990.

[21] G.J. Sullivan, "Decoder inference of optimal reconstruction values for dz+utq quantization of laplacian source random variables," in *16 th JVT meeting, JVT-P111, Poznan, Poland*, 2005.

[22] ITUT CCITT, "Rec. t. 82 & iso/iec 11544: 1993," *Information Technology–Coded Representation of Picture and Audio Information–Progressive Bi-Level Image Comp*, 1993.