TREE STRUCTURED PURSUIT FOR SIMULTANEOUS IMAGE APPROXIMATION

Ariel J. Bernal,

Ryerson University
Department of Electrical Engineering
350 Victoria St., Toronto, Ontario M5B 2K3

Sebastian E. Ferrando *

Ryerson University
Department of Mathematics
350 Victoria St., Toronto, Ontario M5B 2K3

ABSTRACT

The paper introduces an adapted orthonormal system to approximate a collection of given images. The associated construction is performed by a pursuit algorithm constrained to build a tree; the pursuit maximization is performed over a large dictionary of Haar wavelet-like functions. The approximations are given by vector valued discrete martingales that converge to the vector of input images. A natural application for our construction is the case when the set of images is given by a sequence of video frames. We describe the trade off between the size of the input set and the quality of the approximation and provide examples and comparisons.

Index Terms— Image coding. Martingales. Transforms.

1. INTRODUCTION

This note describes a new algorithm for the simultaneous approximation of a given collection of images defined on a common domain Ω. A natural application of the algorithm is the case when the collection of images is given by a sequence of video frames. The algorithm constructs a tree which is associated to a partition of Ω (more precisely: a sequence of partitions). References [1] and [2] provide examples of adaptive trees for image compression. In general, the tree construction is associated to a partition of the base domain which in turn is dependent on a given *single* input image. It follows that it is critical to keep the storage cost of the partition low as it adds to the total storage cost of the compressed image. Therefore, algorithms which partition a given image domain, with the purpose of compressing an image, need to impose strong geometrical constraints on the partition elements (which we will call *atoms*.) In particular, [1] only allows atoms which are polyhedra, further partitions of these atoms can only be performed using line cuts.

As an alternative to the above described situation, the approach introduced in this paper allows for arbitrary partitioning of a given image domain and, hence, we deal with arbitrary atoms. In order to offset the relatively high cost of the resulting adapted partition we consider the case where we have a collection of d images, defined on a common domain Ω. This creates a trade-off as, on the one hand, the relative cost of storing the partition diminishes when we increase d and, on the other hand, the quality of the approximation degrades as d is increased.

We describe a construction (which we will call the Vector Greedy Splitting Algorithm or VGS for short) of an adapted partition of Ω, this partition is common to the given collection of images. Given a certain amount of similarity among the images in this collection, our construction provides associated compression improvements when the common adapted partition is used to compress the collection of images as a single entity.

The rest of this paper is organized as follows, Section 2 describes the VGS algorithm to construct the approximating tree. The description is done under the main setup of the paper, namely the case when the data X has a continuous cumulative distribution. Section 3 provides a general overview on how to bit-code the data structures used to encode the tree approximation. Section 4 provides examples of the proposed compression technique and a brief comparison with current techniques.

2. ADAPTIVE TREE CONSTRUCTION

Consider a collection of d given video frames $X[i]$, $i = 1, \ldots, d$. We will treat them as random variables $X[i] : \Omega \to R$ on a probability space $(\Omega, \mathcal{A}, P)$. We collect the d frames into a vector valued random variable $X : \Omega \to R^d$, and will define the following inner product for $Y, Z \in L^2(\Omega, R^d)$

$$[Y, Z] \equiv \int_\Omega \langle Y(w), Z(w) \rangle \, dP(w), \tag{1}$$

with $\langle Y(w), Z(w) \rangle = \sum_{i=1}^d Y[i](w) \; Z[i](w)$.

We recall a pursuit algorithm in our setup ([3]), let $X \in L^2(\Omega, R^d)$ with inner product $[\ ,\ ]$, also assume a given subset $\mathcal{D} \subseteq L^2(\Omega, R^d)$ is given. Define inductively the $nth. - residue$ by $R^{n+1}X = R^nX - [R^nX, \mu_n] \; \mu_n$, where μ_n satisfies

$$[R^nX, \mu_n] = \sup_{\psi \in \mathcal{D}, ||\psi||=1} [R^nX, \psi], \tag{2}$$

*The research of A.J.Bernal and S.E. Ferrando is supported in part by an NSERC grant.

with $R^0X \equiv X$. Notice that $X = \sum_{k=0}^{n}[R^kX, \mu_k]\mu_k + R^{n+1}X$ and $||R^{n+1}X||^2 = ||R^nX||^2 - |[X, \mu_n]|^2$.

Consider the following dictionary of vector valued functions ($\mathbf{1}_A$ denotes the characteristic function of a set A)

$$\mathcal{C} = \{\psi : \psi = a\mathbf{1}_{A_0} + b\mathbf{1}_{A_1},\ A_0, A_i \in \mathcal{A},\ a, b \in R^d \quad (3)$$

$$\text{and } \int_\Omega \psi\ dP = 0, \int_\Omega ||\psi||^2\ dP = 1\}.$$

Generic elements from $\mathcal{C}$ are too costly to encode. To partly avoid this difficulty we define below a pursuit algorithm constrained by a tree structure. This type of approach can be considered as a constrained non-linear approximation as described in [4]. The reader will check easily that one consequence of the tree structure is that $[\mu_i, \mu_j] = 0$ for $i \neq j$, therefore we will have $[R^nX, \mu_n] = [X, \mu_n]$.

A main contribution of the construction described in this paper is a practical and insightful approach to handle (2) for (a restricted version of) the above dictionary $\mathcal{C}$. In order to incorporate the tree structure into the pursuit algorithm we first need to refine (3) as follows, consider $A \in \mathcal{A}$ and

$$\mathcal{C}_A = \{\psi \in \mathcal{C} : A_0, A_1 \subseteq A,\ A_0 \cup A_1 = A,\ A_0 \cap A_1 = \emptyset\}. \quad (4)$$

This dictionary will be called the *Haar dictionary*. We indicate that we can solve the following key step to define the pursuit algorithm.

Theorem: Assume X has a continuous cumulative distribution, then there exists $\psi_A \in \mathcal{C}_A$ so that

$$[X, \psi_A] = \sup_{\psi \in \mathcal{C}_A} [X, \psi], \quad (5)$$

Remark: For simplicity, on this presentation we will assume P is the uniform measure on $\Omega = [0, 1]^2$. It is possible to extend our results to the general case in which the continuity hypothesis is removed. The functions ψ_A will be called *best functions* (at A) their explicit form will be given shortly.

Our vector approximations are always initialized as follows $\mu_0 = \psi_\emptyset \equiv c\,\mathbf{1}_\Omega$ where c is chosen so that $[X, \psi_\emptyset]\,\psi_\emptyset[i] = \int_\Omega X[i]\ dP$.

For any random variable Y and set A define

$$\mathcal{R}_A(Y) \equiv \{y : \exists w \in A \text{ such that } Y(w) = y\}. \quad (6)$$

In order to completely specify $\psi_A = a\mathbf{1}_{A_0} + b\mathbf{1}_{A_1} \in \mathcal{C}_A$, ψ_A as in the above Proposition, it is enough to provide A_0 explicitly:

$$\mathbf{1}_{A_0}(w) = \mathbf{1}_{\{z : X[b'](z) \leq y\}}(w) \text{ where } X[b'](z) \equiv \langle X(z), b'\rangle \quad (7)$$

$$\text{for some } b' \in S^d \text{ and for some } y \in \mathcal{R}_A(X[b']),$$

moreover $b = ||b||\ b'$. The quantities b' and $y = y(b')$ are obtained by an optimization procedure described in [5].

Define the *best children* of A by

$$A_0 \equiv \{w \in A : \psi_A(w) = a\},\ A_1 \equiv \{w \in A : \psi_A(w) = b\}. \quad (8)$$

Next we define the VGS algorithm, we will indicate how the algorithm constructs recursively a sequence of partitions Π_n indexed by $n = 0, 1, 2, \ldots$. The index n will be referred as the $n-$th. iteration of VGS. Start by setting $\Pi_0 = \{\Omega, \emptyset\}$ (notice that we explicitly include $\emptyset$ in Π_0, this will include $\psi_\emptyset$ in all our approximations) and assume, inductively, that Π_k, $k \leq n$ ($\Pi_k \subseteq \mathcal{A}$) have been constructed and are finite. Now we describe how to generate Π_{n+1}. Consider $A^* \in \Pi_n$ such that it satisfies

$$|[X, \psi_{A^*}]| \geq |[X, \psi_A]| \text{ for all } A \in \Pi_n. \quad (9)$$

If $[X, \psi_{A^*}] = 0$, the algorithm VGS terminates and $\Pi_p \equiv \Pi_n$ for all $p \geq n$. Otherwise, i.e. $[X, \psi_{A^*}] \neq 0$, we set

$$\Pi_{n+1} = \Pi_n \backslash \{A^*\} \cup \{A_0^*, A_1^*\} \quad (10)$$

where, as indicate previously, the sets A_k^* are the best children of A^*. The element ψ_{A^*} selected at iteration n will be denoted μ_n, notice $[\mu_k, \mu_{k'}] = 0$ if $k \neq k'$. Define the tree $\mathcal{T}_n$ as follows

$$\mathcal{T}_n \equiv \cup_{k=0}^{n}\Pi_k. \quad (11)$$

and the associated approximation by

$$X_{\mathcal{T}_n} \equiv \sum_{A \in \mathcal{T}_n} [X, \psi_A]\ \psi_A. \quad (12)$$

$X_{\mathcal{T}_n}$ is a martingale sequence which satisfies:

$$X_{\mathcal{T}_n}(w) = \frac{1}{P(A)} \int_A X\ dP, \text{ for } w \in A \text{ and } A \in \Pi_n. \quad (13)$$

Under general conditions, it can be proven that

$$\lim_{n \to \infty} X_{\mathcal{T}_n}(w) = X(w) \quad \text{for almost all } w \in \Omega. \quad (14)$$

The above limit will actually be finite if X is a simple function (even though the above Theorem does not apply), namely if X takes a finite number of values.

Using a re-ordering function h, we order the expansion (12) in such a way such that $|[X, \mu_{h(k)}]| \geq |[X, \mu_{h(k+1)}]|$. Given a certain error level $\epsilon > 0$, define the *optimized approximation* by

$$X_n = \sum_{k=0}^{n-1} [X, \mu_{h(k)}]\ \mu_{h(k)}, \quad (15)$$

where $n = n(\epsilon)$ is the smallest integer that $||X - X_n|| \leq \epsilon$. The corresponding n nodes A_k which satisfy $\mu_{h(k)} = \psi_{A_{h(k)}}$ are called the *active* nodes for the given ϵ. Figure 1 displays an input set with $d = 9$ and a detail at different levels of approximation.

Fig. 1. Left: Input set. Right: top left: original middle image, top right: PSNR=22, bottom left: PSNR=34, bottom right: PSNR=40.

3. BIT ENCODING OF VECTOR APPROXIMATION

Besides coding the information to identify active nodes, we need to code $[X, \psi_{A_{h(k)}}]$ and the corresponding b'. We also need to keep enough of the tree information in order to evaluate $\psi_{A_{h(k)}}(w)$ at different points $w \in \Omega$. In particular, this information will contain the relevant children-parent relationship. This information will be called the *significance map*. Its actual encoding is technically challenging as our approach only deals with the actives nodes (i.e. we do not complete with the missing nodes in order to obtain a tree). Besides of the information contained in the significance map, we also need to know how active atoms are made up of Ω points. This information will be called the *partition map* and consists on encoding the partition associated to the active nodes only.

When we report bit values of the significance map we will be reporting the bit cost of encoding quantized values $[X, \psi_{A_{h(k)}}]$ and the corresponding quantized values for b' (so, in effect, our significance map also includes a *quantization map*). In some instances we will be reporting the bit cost of encoding a lossless compressed version of the corresponding data structures (we have found that the partition map is very sensitive to quantization and hence the need to use lossless compression).

$C_{\mathcal{M}_S}[i](d)$ will mean that we have run VGS for d inputs and the component i has a significance cost of $C_{\mathcal{M}_S}[i](d)$ bits. Whenever $d = 1$ we will write $C_{\mathcal{M}_S}[1](1)$ as $C_{\mathcal{M}_S}(1)$. In short, $C_{\mathcal{M}_S}(1)$ represents the (quantized) significance map cost of encoding the output of VGS (*excluding* the partition cost) and VGS was executed on a single image. We use similar notation for the partition map cost but we will assume the partition cost is independent of i. Therefore the notation $C_{\mathcal{M}_\Pi}(d)$ denotes the number of bits needed to store the partition map when VGS was executed on d images.

We expect $C_{\mathcal{M}_S}[i](d)$ to deteriorate as d increases (for any i), and we also expect $C_{\mathcal{M}_S}(1)$ to be of best quality, i.e. $C_{\mathcal{M}_S}(1) << C_{\mathcal{M}_S}[i](d)$ for all i and d. We also note that $C_{\mathcal{M}_\Pi}(d)$ has a uniform upper bound (i.e. the upper bound is independent of d) which depends solely on the size of Ω.

Lets use $C_{FixedBasis}$ to denote the cost of encoding a given image by a certain method with fixed basis (in particular it could be JPEG, JPEG2000, Haar basis, etc.). If there are d images we will denote with $C_{FixedBasis}[i]$ the cost, of the method, for image i. We expect that $C_{\mathcal{M}_S}(1) << C_{FixedBasis}[1]$.

We introduce next a useful quantity to quantify the quality of VGS's approximation

$$\gamma(d) \equiv \frac{C_{\mathcal{M}_\Pi}(d)}{d} + \frac{\sum_{i=1}^{d} C_{\mathcal{M}_S}[i](d)}{d}. \tag{16}$$

Clearly, the optimal $d^\star$ is the one that minimizes $\gamma(d)$. It is clear that there is a tension between how large d has to be so $\frac{C_{\mathcal{M}_\Pi}(d)}{d}$ is small enough and at the same time we want $\frac{\sum_{i=1}^{d} C_{\mathcal{M}_S}[i](d)}{d}$ to remain small but we know that $C_{\mathcal{M}_S}[i](d)$ deteriorates as d growths.

Notice that VGS will outperform the cost of the fixed basis method, namely $C_{FixedBasis}$ if

$$\gamma(d) < \frac{\sum_{i=1}^{d} C_{FixedBasis}[i]}{d}. \tag{17}$$

As an illustration, Figure 2 shows a set of images taken from a video sequence (8 bit monochromatic at 15 fps). We have run VGS on increasingly larger subsets of this collection of images by adding one image at a time to the previous subset. In this way we were able to compute $\gamma(d)$ for $d = 1, \ldots, 20$. The results are plotted in Figure 3, the term in (16) (average cost for Partition Map) $\frac{C_{\mathcal{M}_\Pi}(d)}{d}$ is denoted PM (Partition Map) in the figure, and the term in (16) (average total cost for Quantization Map and Significance Map) $\frac{\sum_{i=1}^{d} C_{\mathcal{M}_S}[i](d)}{d}$ is denoted by $QM + SM$.

4. RESULTS AND COMPARISONS

In this section we will compare the JPEG2000 and our algorithm in a static environment using the video sequence set from Figure 2. This is a rather slow changing video but sampled at a relatively fast rate of 15 fps. When we say "static" we mean that JPEG2000 does not make use of any temporal correlation among frames. Nonetheless, we have run JPEG2000 in the most favorable situation, namely we have collected the input set into a single larger image.

Due to the fact that 9 images is the best number of images for the VGS algorithm, we will use $d = 9$; our approximation is given by (15) and, when reporting results, we label it AVG (or Haar-AVG). There are two versions of our algorithm and both outperform (in terms of the PSNR) the JPEG2000 results in the example provided. The first version is the standard approximation and the second one is the same approximation using the Lempel-Ziv (AVG-LZ) algorithm to encode the partition and the quantization map (lossless compression).

Figure 4 shows the total bit cost vs distortion, comparing the JPEG2000 with our algorithms. It is clear from the figure

Fig. 2. Video sequence

that AVG using the Lempel-Ziv encoding algorithm outperforms (in terms of the PSNR) the others for this special video sequence. Table 4 shows numerically the same information as in Figure 4. The last column of the table contains the difference between the JPEG2000 and AVG-LZ, it is possible to see that the AVG-LZ is within $20\% - 35\%$ better (in terms of the PSNR) than JPEG2000 for this case, several more examples are described in [5] and a systematic comparison with JPEG2000 will be described elsewhere. Reasonable comparisons with video codecs are harder to stablish as it may require incorporating additional features to the VGS algorithm; some comparisons with MPEG, for color video, are described in [5].

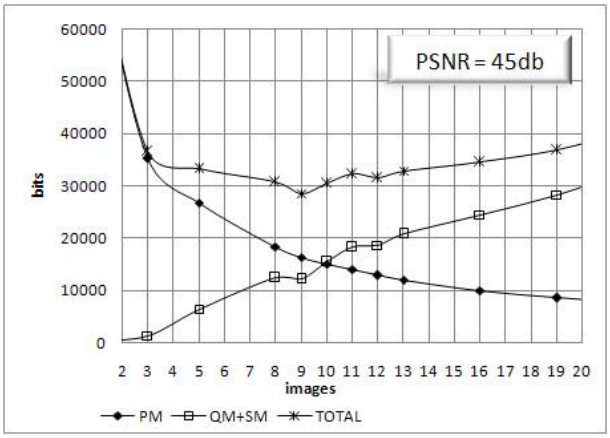

Fig. 3. Average bit cost per image vs number of images.

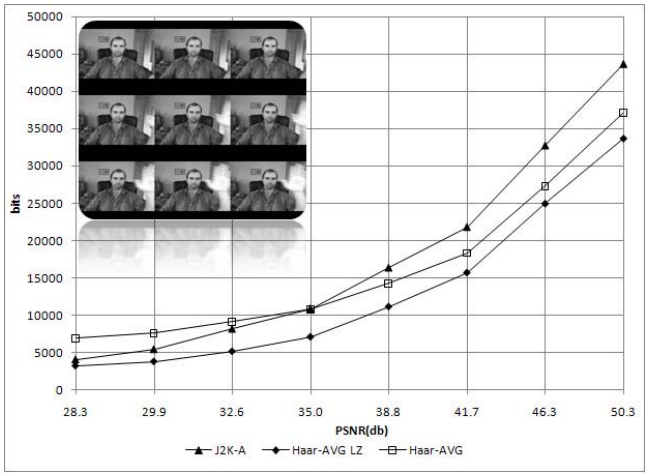

Fig. 4. Total bits vs PSNR for first 9 frames.

PSNR(db)	JPEG2000	AVG	AVG LZ	DIFF
29.9	5461	7650	3779	31%
32.6	8192	9103	5135	37%
35.0	10922	10758	7088	35%
38.8	16384	14260	11143	32%
41.7	21837	18325	15719	28%
46.3	32768	27277	24981	24%

Table 1. Numerical bit cost vs. distortion comparison

5. REFERENCES

[1] D. Alani, A. Averbuch, and S. Dekel, "Image coding with geometric wavelets," *IEEE Transactions on Image Processing*, vol. 16, pp. 69–77, 2007.

[2] Y. Huang, I. Pollak, M.N. Do, and C.A. Bouman, "Fast search for best representations in multitree dictionaries," *IEEE Transactions on Image Processing*, vol. 15, pp. 1779–1793, July 2006.

[3] S. Mallat, "A wavelet tour of signal processing," *Academic Press, second edition*, July 1999.

[4] A. Cohen, I. Daubechies W. Dahmen, and R. De Vore, "Tree approximation and optimal encoding," *Applied and Computational Harmonic Analysis*, vol. 11, pp. 192–226, 2001.

[5] A.J. Bernal, "Simultaneous approximations of images. applications to image and video compression," *MASc Thesis*, vol. Ryerson University, 2007.