1. Department of Mathematics, Ryerson University, Toronto. ferrando@ryerson.ca. Correspondence should be sent to this author at: Department of Mathematics, Ryerson University, 350 Victoria St., Toronto, ON, M5B 2K3, Canada.

Adaptive Vector Valued Martingales. Applications to Image Compression

Sebastian E. Ferrando

Department of Mathematics, Physics and Computer Science, Ryerson Polytechnic University, Toronto, Ontario M5B 2K3, Canada. Email: ferrando@acs.ryerson.ca

Abstract

Given a finite collection of functions defined on a common domain, the paper describes an algorithm that constructs a vector valued approximating martingale sequence. The orthonomal basis functions used to construct the martingale approximation are optimally selected, in each greedy step, from a large dictionary. The resulting approximations are characterized as generalized H-systems and provide scalar and vector valued orthonormal systems which can be employed to perform lossy compression for the given set of input functions. The filtration associated to the martingale allows for a multiresolution analysis/synthesis algorithm to compute the approximating conditional expectation via a Fourier expansion. Convergence of the algorithm as well as several computational properties are established. Numerical examples are also provided for collection of images and video frames in order to study the approximating power of the constructed sequences.

Key words: Vector valued martingales, Simultaneous approximation, Image and video compression, Conditional expectations.

1. Introduction

The reference [4] introduced the formalism of H-systems to perform adaptive approximations. These systems allow computation of conditional expectations via Fourier expansions and are a generalization of the Haar orthonormal system. The present paper is a continuation and extension of [4], this last reference provided an analysis of the scalar greedy splitting algorithm which is adapted to a single input function. Presently, we concentrate on the vector greedy splitting (VGS) algorithm which is our key tool to construct adapted vector valued orthonormal systems which also provide martingale approximating sequences. The VGS was briefly described in [4] but no mathematical properties of the algorithm were established nor was it described in detailed computational terms.

A natural application of our approximations is to lossy image compression, it is apparent that the bit cost of encoding the adapted orthonormal system is relatively high, to offset this cost we work in a vector setting which allows for simultaneous approximations. Moreover, our nonlinear approximations are restricted in such a way that they can be realized by a tree data structure which allows for a more efficient encoding. Our approach can be considered as a constrained non-linear approximation as described in [5] and [6].

A main aspect of the present paper is the detailed description and proof of pointwise convergence for the vector approximation provided by the VGS algorithm. The paper also extends the previous set up to *generalized* H-systems which require n-ary trees as data structures for their realization (as opposed to binary trees in the case of H-systems). Another important aspect of our paper is to provide several set of examples, based on a software implementation, and their bit cost analysis, in order to assess the approximation power of our approach.

The reason for our emphasis on approximations that are martingales is twofold. They are a natural setting when imposing a tree structure as we do, moreover we concentrate in pointwise convergence which, by its very nature, will bring to the fore the underlying geometric structure (actually through the natural σ-algebras) of the input images. Our constructions can also be used to approximate stochastic processes, some details for this setup and more motivations for our approach are described in [4].

In order to motivate our construction we briefly discuss some aspects of the Matching Pursuit algorithm as it will be the basis for our main construction. The reader can consult [10], [11] and [7] for references on the Matching Pursuit algorithm.

Consider $X \in L^2(\Omega, \mathbb{R}^d)$ with an inner product $[\ ,\ ]$, also assume a given subset $\mathcal{D} \subseteq L^2(\Omega, \mathbb{R}^d)$ is given. Define $R^0X \equiv X$ and the $1-residue$ by $R^1X = X - [X, U_0]\ U_0$ where $U_0 \in \mathcal{D}$, $\|U_0\| = 1$, satisfies

$$[X, U_0] = \sup_{\psi \in \mathcal{D}, \|\psi\|=1} [X, \psi].$$

Continue inductively defining the $nth.-residue$ by $R^{n+1}X = R^nX - [R^nX, U_n]\ U_n$, where U_n satisfies

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

Notice that

$$X = \sum_{k=0}^{n} [R^kX, U_k]U_k + R^{n+1}X, \text{ and } \|R^{n+1}X\|^2 = \|R^nX\|^2 - |[X, U_n]|^2. \tag{2}$$

In this setup, lossy transform compression is based on retaining a few terms in the sum of the

right hand side of (2) and dropping the remainder term $R^{n+1}X$. For the dictionaries $\mathcal{D}$ that will be considered in the present paper the cost of storing each element U_k in the pursuit expansion will be too high and for this reason we will impose a tree structure on the pursuit algorithm. Moreover, in order to obtain further storage savings, the construction of the orthonormal system $\{U_k\}$ is based on a common set of scalar orthonormal functions (what we will call later in the paper *generalized H-systems*).

One consequence of the tree structure is that $[U_i, U_j] = 0$ for $i \neq j$, this will imply

$$[R^n X, U_n] = [X, U_n].$$

Therefore, in our context, the pursuit algorithm can be described by indicating that it maximizes $|[X, U_n]|^2$, or equivalently, it minimizes $\|R^n X\|^2$, under the constraint of constructing a tree. The maximization in (1) is *greedy* because it is only one look ahead, namely it searches for one function at a time. In general, unless $\mathcal{D}$ has a special structure, it is expected that the maximization of $|[X, U_n]|^2$ requires an impractically large number of computations. A main contribution of the construction described in this paper is a practical and insightful approach to handle (1) for two large dictionaries $\mathcal{D}$.

The paper is organized as follows, Section 2 introduces notation and the setting of generalized H-systems. The brief Section 3 points to the key aspects of the constructions and the algorithm to be discussed in the remaining of the paper. Section 4 presents in detail all the computational preliminaries required to set up the VGS algorithm. In particular, the relationship between the two different dictionaries used and the optimization is made clear. Section 5 formally describes the VGS algorithm and proves its pointwise convergence. Section 6 describes how to encode the approximations and the data structures required, illustration of the different tradeoffs are provided. Section 7 presents a suite of examples used to illustrate an implementation of the algorithm. Section 9 summarizes the paper and draws conclusions. Appendix A states optimizations results needed in the main body of the paper and describes a more general dictionary where our approach could also be realized. Appendix B describes important relationships between vector and scalar approximations. An alternative optimization for one of our main constructions is also sketched.

2. Generalized H-Systems

Let $(\Omega, \mathcal{A}, P)$ denote an arbitrary probability space. The generators of a discrete σ-algebra $\mathcal{B} \subseteq \mathcal{A}$ will be called *atoms.*

Consider the space $L^2(\Omega, \mathbb{R}^d) \equiv L^2((\Omega, \mathcal{A}, P), \mathbb{R}^d)$, where d is a fixed positive integer. This space consists of vector valued random variables $Y = (Y[1], \ldots, Y[d])$ and it is endowed with the following inner product

$$[Z, Y] \equiv \int_\Omega \langle Z(\omega), Y(\omega) \rangle \, dP(\omega),$$

where $\langle \ , \ \rangle$ is the Euclidean inner product in $\mathbb{R}^d$, namely,

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

We will use $\| \ \|^2$ for the squared of the norm for the two different inner products, namely $\|Z\|^2 = [Z, Z]$ and $\|B\|^2 = \langle B, B \rangle$. The reader should be able to distinguish the different meanings from

the context. It will be important to single out the scalar case, namely $d = 1$, and so we need to specialize the above notation. Whenever possible, we denote *vector quantities* with capital letters; we will take, for $Z \in \mathbb{R}^d$, $Z[i] \equiv z[i]$ and use either notation indistinguishably. Whenever it is not clear that a given object is a scalar quantity, we will use a subscript to denote that we are actually dealing with a scalar, for example ψ_s means $\psi_s \in L^2(\Omega, \mathbb{R})$ also $[u, v]_s$ is given by (3) with $d = 1$ and $u, v \in L^2(\Omega, \mathbb{R})$ should be understood from the given context. Dependencies on the point w will be suppressed from the notation whenever possible.

The following definition generalizes the notion of H-systems introduced in [8] (see also [4]). H-systems are a particular case of the definition below by taking $k_n = n$. The classical Haar and Walsh systems, on $L^2([0, 1])$, are examples of H-systems and generalized H-systems respectively.

Definition 1. *An orthonormal system of functions $\{u_k\}_{k\geq 0} \in L^2(\Omega, \mathbb{R})$ defined on Ω is called a generalized H-system if and only if there exists a sequence of integers $\{k_i\}$, $0 = k_0 < k_1 < \ldots < k_n < \ldots$, such that for any $z \in L^2(\Omega, \mathbb{R})$*

$$z_{\mathcal{A}_{k_n}} \equiv \mathbf{E}(z|u_0, u_1, \ldots, u_{k_n}) = \sum_{i=0}^{k_n} [z, u_i]_s \, u_i, \quad \textit{for all } n \geq 0, \tag{3}$$

where $\mathcal{A}_n = \sigma(u_0, \ldots, u_n)$. Also define

$$\mathcal{B}_n \equiv \mathcal{A}_{k_n} \text{ and } |\mathcal{B}_n| \text{ will denote the number of generating atoms in } \mathcal{B}_n.$$

The intended meaning of $n \geq 0$ in the above definition is to allow the system $\{u_k\}_{k\geq 0}$ to be finite or infinite. We also use the notation $\mathcal{A}_\infty = \sigma(\cup_{n\geq 0}\mathcal{A}_n)$; also, for a given $A \in \mathcal{A}$, with $P(A) > 0$, $\sigma_A(z_1, \ldots, z_N)$ denotes the sigma algebra generated by given random variables z_i relative to the measurable space $(A, \mathcal{A}_A, P_A)$. Notice that $\mathcal{A}_n$, for finite n, is necessarily finite.
The following proposition gives an alternative characterization of generalized H-systems equivalent to Definition 1. It is convenient to set $k_{-1} = -1$.

Proposition 1. *An orthonormal system $\{u_k\}_{k\geq 0} \subseteq L^2(\Omega, \mathbb{R})$ is a generalized H-system if and only if the following three conditions hold for all $n \geq 0$:*

i) For each atom $A \in \mathcal{B}_n$, $|\sigma_A(u_k : k_{n-1} < k \leq k_n)| \leq k_n - k_{n-1} + 1$.

ii) The σ-algebras $\mathcal{B}_n$ satisfy $|\mathcal{B}_n| = k_n + 1$.

iii) $\mathbf{E}(u_k|u_0, u_1, \ldots, u_{k_n}) = 0$; for $k_n < k \leq k_{n+1}$.

Proof. Suppose first that $\{u_k\}$ is an orthonormal system that satisfies the three properties above, consider $y \in L^2(\Omega, \mathbb{R})$, then $y_{\mathcal{B}_n}$ can be written as a linear combination of the $k_n + 1$ characteristic functions of the atoms of $\mathcal{B}_n$. At the same time, the functions $u_0, \ldots, u_{k_n}$ are an orthonormal basis for the subspace given by the range of the projection operator $\mathbf{E}(.|u_0, u_1, \ldots, u_{k_n})$. These facts imply (3).

Conversely, assume now that the given orthonormal system $\{u_k\}$ satisfies (3). We will first prove iii) above. Consider the martingale difference $d_{n+1} \equiv z_{\mathcal{B}_{n+1}} - z_{\mathcal{B}_n}$, so $\sum_{j=k_n+1}^{k_{n+1}} [z, u_j]_s \mathbf{E}(u_j|\mathcal{B}_n) = 0$. Take now $z = u_k$, $k_n < k \leq k_{n+1}$, then $\mathbf{E}(u_k|\mathcal{B}_n) = 0$. To prove i) and ii) above we proceed by induction. Consider $z = 1 \in \mathbb{R}^d$, then it follows that $u_0 = 1_\Omega$, this indicates that ii) is true for $n = 0$, notice that i) is also true for $n = 0$ by our convention. Assume both statements hold for $n = N$. From (3) the range of the projection operator $\mathbf{E}(.|u_0, u_1, \ldots, u_{k_{N+1}})$ is an $k_{N+1} + 1$

dimensional subspace, therefore $\mathcal{B}_{N+1}$ is generated by $k_{N+1}+1$ atoms, i.e. $|\mathcal{B}_{N+1}| = k_{N+1}+1$. To establish i) for $N+1$ we proceed by contradiction, consider there is an atom $A \in \mathcal{B}_N$ and u_k, $k_N < k \leq k_{N+1}$, such that u_k takes more than $k_{N+1} - k_N + 1$ distinct values on A. Then $|\mathcal{B}_{N+1}| > |\mathcal{B}_N| - 1 + k_{N+1} - k_N + 1 = k_N + 1 - 1 + k_{N+1} - k_N + 1 = k_{N+1} + 1$. □

Notice that in the case $k_n = n$, the inequality in part *i*) of Proposition 1 becomes for $n \geq 1$ and $A \in \mathcal{B}_n$, $|\sigma_A(u_k : n-1 < k \leq n)| = 2$. This last equality forces the increasing sequence of partitions generating the sigma algebras $\mathcal{A}_n$ to form a binary tree and the support of the functions u_k to be localized to nodes of this binary tree.

From now on we restrict our attention to generalized H-systems that satisfy a stronger version of *i*) in Proposition 1, namely, for each n there exists $A_n \in \mathcal{B}_n$ such that

$$|\sigma_{A_n}(u_k : k_{n-1} < k \leq k_n)| = k_n - k_{n-1} + 1$$

and for all other $A \in \mathcal{B}_n$

$$|\sigma_A(u_k : k_{n-1} < k \leq k_n)| = 1.$$

The following definition formalizes the class of generalized H-systems considered in the remaining of the paper. We will set $\mathrm{Sprt}(u)$ to denote the support of a given function u.

Definition 2. *A generalized H-system is called admissible if for each n there exists $A \in \mathcal{B}_n$, a generating atom, such that we have $Sprt(u_k) \subset A$ for all $k_n < k \leq k_{n+1}$.*

For the remaining of the paper we will restrict ourselves to admissible generalized H-systems and therefore will drop the word *admissible* and only refer to generalized H-systems. Clearly, this H-systems can be realized by tree data structures.

3. Overview of Results

Generalized H-systems is the guiding general framework for the class of orthonormal systems constructed in this paper, the actual construction given by the VGS algorithm will realize a special class of these systems. In particular, the VGS construction will be optimized to a given collection of input random variables $x[1], \ldots, x[d]$, the optimization is achieved by collecting the given random variables into a single vector valued random variable X. We emphasize the fact that the underlying (scalar) orthonormal system is common to all the given random variables, this is crucial for compression applications as the relative cost of storing the orthonormal system decreases as d increases.

Definition 3. *Given $A \in \mathcal{A}$, $P(A) > 0$, a function Ψ is called an admissible (vector valued) function on A if there exist $A_i \in \mathcal{A}$, $A_i \subseteq A$, $i = 0, 1$, $\Psi = B_0\, \mathbf{1}_{A_0} + B_1\, \mathbf{1}_{A_1}$, $B_0, B_1 \in \mathbb{R}^d$, and*

$$\int_\Omega \Psi(\omega)\, dP(\omega) = 0, \qquad \int_\Omega \|\Psi\|^2(\omega)\, dP(\omega) = 1. \tag{4}$$

Remark 1. *If , $A_0 \cap A_1 = \emptyset$ and $A = A_0 \cup A_1$ is also satisfied, we call Ψ a (vector valued) Haar function on A.*

The following two dictionaries will be key for our developments. For a fixed $A \in \mathcal{A}$ define

$$\mathcal{D}_A^0 \equiv \{\Psi : \text{ a Haar function on A}\} \subseteq$$

$$\mathcal{D}_A^1 \equiv \{\Psi : \text{an admissible function on A and } A_0 \cap A_1 = \emptyset\}.$$

Our results will apply to the two sets $\mathcal{D}_A^k$, $k = 0, 1$ (using specific arguments in each case) and the notation $\mathcal{D}_A$ will be used as a generic notation to denote any of the two sets $\mathcal{D}_A^k$. We will also use the notation $\mathcal{D}^k \equiv \cup_{A\in\mathcal{A}}\ \mathcal{D}_A^k$. In Appendix A we explain how one may extend our approach to functions of the form $\Psi = B_0\ \varphi_0 + B_1\ \varphi_1$, $B_0, B_1 \in \mathbb{R}^d$, $\varphi_i \in \mathcal{A}$, $\text{Sprt}(\varphi_i) \subseteq A$, $0 \leq \varphi_i \leq 1$, $i = 0, 1$.

Consider $\Psi = B_0\mathbf{1}_{A_0} + B_1\mathbf{1}_{A_1} \in \mathcal{D}_A$, let $B' \equiv \frac{B_1}{\|B_1\|}$, hence $B' \in S^d$, where S^d is the unit sphere in $\mathbb{R}^d$. Notice that using (4) we can write:

$$\Psi(B') = \Psi = \sqrt{\frac{u_0\ u_1}{(u_0 + u_1)}}\ B'\left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0}\right). \tag{5}$$

We will show in Theorem 1 that, for a fixed $X \in L^2(\Omega, \mathbb{R}^d)$, there exists $\Psi_A^0 \in \mathcal{D}_A$ such that

$$[X, \Psi_A^0] = \sup_{\Psi\in\mathcal{D}_A} [X, \Psi].$$

Together with the companion function Ψ_A^1, defined in (79) (see also (83)), these two functions are used to define the vector valued orthonormal system $\mathcal{H} = \{U_k\}$ defined in (40). The vectors $D_k \equiv [X, U_k]U_k$ are a martingale sequence with respect to $\mathcal{F}_n \equiv \sigma(U_0, \ldots, U_n)$ which gives the vector valued martingale sequence

$$\sum_{k=0}^{n} D_k = \sum_{k=0}^{n} [X, U_k]U_k.$$

In Appendix B we relate $\mathcal{H} = \{U_k\}$ to the generalized H-system $\mathcal{G} = \{u_k\}$ defined in (81), this allow us to prove that for all $n \geq 0$,

$$X_{\mathcal{A}_{k_n}} \equiv \mathbf{E}(X|u_0, u_1, \ldots, u_{k_n}) = \sum_{k=0}^{k_n} [X, U_k]\ U_k. \tag{6}$$

We also note that $\mathcal{A}_k \equiv \sigma(u_0, \ldots, u_k) = \sigma(U_0, \ldots, U_k)$. Actually, (6) follows from

$$X_{\mathcal{A}_{k_n}}[i] = x_{\mathcal{A}_{k_n}}[i] \equiv \mathbf{E}(x[i]|u_0, u_1, \ldots, u_{k_n}) = \sum_{k=0}^{k_n} [x[i], u_k]_s\ u_k = \sum_{k=0}^{k_n} [X, U_k]\ U_k[i],$$

which is basic in our paper and is proved in Appendix B. Our proof of pointwise convergence

$$X(w) = \lim_{n\to\infty} X_{\mathcal{A}_{k_n}}(w)$$

relies on the conditional expectation expression (6) for our approximation. In fact, $\{u_k\}$ is an unconditional martingale basis for the spaces $L^p((\Omega, \sigma(X), P), \mathbb{R})$, $1 < p < \infty$, using the filtration $\mathcal{B}_n \equiv \mathcal{A}_{k_n}$.

4. Optimization Results

This section establishes the results needed to setup the computational realization of the VGS algorithm for the function classes $\mathcal{D}_A$, possible extensions are also described in Appendixes A and B. The main result needed in the construction is the following theorem.

Theorem 1. *Let $A \in \mathcal{A}$ be an arbitrary measurable set and $X \in L^2(\Omega, \mathbb{R}^d)$ then there exists $\Psi_A^0 \in \mathcal{D}_A$ so that*

$$[X, \Psi_A^0] = \sup_{\Psi \in \mathcal{D}_A} [X, \Psi]. \tag{7}$$

Proof. In order to evaluate the supremum in (7), consider $\Psi = B_0\ \mathbf{1}_{A_0} + B_1\ \mathbf{1}_{A_1} \in \mathcal{D}_A$ and set $u_0 = P(A_0), u_1 = P(A_1) \in (0, P(A))$. With this notation we now write the inner product [,]

$$[X, \Psi] \equiv \int_\Omega \langle X(w), \Psi(w) \rangle dP(w) = \int_A \langle X(w), \Psi(w) \rangle dP(w) =$$

$$\frac{-u_1}{u_0} \int_{A_0} \langle X(w), B_1 \rangle\ dP(w) + \int_{A_1} \langle X(w), B_1 \rangle\ dP(w),$$

and introduce the following functional to be optimized,

$$\gamma(B', u_0, u_1, A_0, A_1) \equiv [X, \Psi] = \tag{8}$$

$$\|B_1\|\ u_1 \left(\frac{1}{u_1} \int_{A_1} \langle X(w), B' \rangle\ dP(w) - \frac{1}{u_0} \int_{A_0} \langle X(w), B' \rangle\ dP(w) \right) =$$

$$\left(\frac{1}{u_1} \int_{A_1} \langle X(w), B' \rangle dP(w) - \frac{1}{u_0} \int_{A_0} \langle X(w), B' \rangle\ dP(w) \right).$$

Where we have used

$$\|B_1\| = \sqrt{\frac{u_0}{u_0\ u_1 + u_1^2}} \tag{9}$$

with B' as introduced in (5).

The supremum in (7) can be written as iterated suprema, the two innermost suprema will depend on the specific class of functions $\mathcal{D}_A^k$, $k = 0, 1$, introduced earlier. To deal with the different cases we need some notation. For $A \in \mathcal{A}$ with $P(A) > 0$ given, define

$$I^0 \equiv \{(u_0, u_1) :\ u_0 + u_1 = P(A), 0 < u_i < P(A), i = 0, 1\},$$

$$I^1 \equiv \{(u_0, u_1) :\ 0 < u_0 + u_1 \leq P(A), 0 < u_i < P(A), i = 0, 1\}.$$

The notation $\bar{I^k}$ represents the closure (in $\mathbb{R}^2$) of the set I^k, $k = 0, 1$. We will use the notations I and $\bar{I}$ as a generic reference to any of the sets I^k, $\bar{I^k}$ $k = 0, 1$. Define for fixed $0 < u_0, u_1 < P(A)$,

$$\sup_{A_0, A_1}{}^0 \equiv \sup_{A_0, A_1 \in \mathcal{A}, A_0 \cap A_1 = \emptyset,\ A_0 \cup A_1 = A, P(A_i) = u_i, i = 0, 1}.$$

Define for *fixed* $0 < u_0, u_1 < P(A)$

$$\sup_{A_0, A_1}{}^1 \equiv \sup_{A_0, A_1 \in \mathcal{A}, A_0 \cap A_1 = \emptyset,\ A_0, A_1 \subseteq A, P(A_i) = u_i, i = 0, 1}.$$

Moreover,

$$\sup_{u_0,u_1}{}^k \equiv \sup_{(u_0,u_1)\in I^k}, \ k = 0, 1.$$

We will use the notations $\sup_{A_0,A_1}$ and $\sup_{u_0,u_1}$ as generic notation for any of $\sup_{A_0,A_1}{}^k$ and $\sup_{u_0,u_1}{}^k$, $k = 0, 1$. Finally let,

$$\sup_{B'} \equiv \sup_{B'\in S^d}.$$

Therefore (7) can be written as iterated suprema as follows

$$\sup_{\Psi\in\mathcal{D}_A}[X,\Psi] = \sup_{B'}\left[\sup_{u_0,u_1}\left(\sup_{A_0,A_1}\gamma(B',u_0,u_1,A_0,A_1)\right)\right]. \tag{10}$$

In the above expression, and for later use below, we set $\gamma(B', u_0, u_1, A_0, A_1) = 0$ if for given u_0 and u_1 there is no A_0 or A_1 so that $u_0 = P(A_0)$ or $u_1 = P(A_1)$. The maximization of this functional is done through a series of results described next. In this way, we will then obtain (7) from Corollary 1 and Proposition 4. □

We concentrate first on evaluating the innermost supremum above i.e.

$$\sup_{A_0,A_1}\gamma(B',u_0,u_1,A_0,A_1).$$

Therefore, for fixed $(u_0, u_1) \in I$ and $B' \in S^d$, we need to maximize the following functional for the variables A_i

$$\gamma_{B',u_0,u_1}(A_0,A_1) \equiv \gamma(B',u_0,u_1,A_0,A_1). \tag{11}$$

Using (62) and (63) from Appendix A, suitable interpreted when restricted to the given set A, we define φ_0 and φ_1 are given by

$$\varphi_0(\,.\,) = \mathbf{1}_{\{w\in A:\langle X(w),B'\rangle<y_{u_0}\}}(\,.\,) + c_{u_0}\,\mathbf{1}_{\{w\in A:\langle X(w),B'\rangle=y_{u_0}\}}(\,.\,), \tag{12}$$

$$\varphi_1(\,.\,) = \mathbf{1}_{\{w\in A:\langle X(w),B'\rangle>z_{u_1}\}}(\,.\,) + d_{u_1}\,\mathbf{1}_{\{w\in A:\langle X(w),B'\rangle=z_{u_1}\}}(\,.\,). \tag{13}$$

Notice that $\varphi_i = \varphi_i(B', u_i)$, $i = 0, 1$.

For a fixed $B' \in S^d$, define for *any* $(u_0, u_1) \in I$,

$$\rho_{B'}(u_0,u_1) \equiv \rho_{B'}(u_0,u_1,\varphi_0(B',u_0),\varphi_1(B',u_1)) \equiv \tag{14}$$

$$\sqrt{\frac{u_0\,u_1}{u_0+u_1}}\ \left[\left(\frac{1}{u_1}\int\langle X,B'\rangle\,\varphi_1 dP - \frac{1}{u_0}\int\langle X,B'\rangle\ \varphi_0 dP\right)\right].$$

Proposition 2. *Fix $(u_0, u_1) \in I$, $A \in \mathcal{A}$ with $P(A) > 0$ and $B' \in S^d$, then*

$$\sup_{A_0,A_1}\gamma_{B',u_0,u_1}(A_0,A_1) \le \max\{\rho_{B'}(u_0,u_1),0\}. \tag{15}$$

Proof. If there are $A_i \in \mathcal{A}$, $A_i \subseteq A$, $A_0 \cap A_1 = \emptyset$, so that $u_i = P(A_i)$, for $i = 0, 1$, then (15) follows from Theorem 5 and Corollary 2 , stated in Appendix A. If either such A_0 or A_1 do not exist the left hand side of (15) is equal to 0. □

Remark 2. *Theorem 5 and Corollary 2 are actually applied to the set A considered as a measure space (this structure is inherited from the probability space* $(\Omega, P, \mathcal{A})$*).*

Lemma 1. *Fix* $B' \in S^d$ *and* $A \in \mathcal{A}$ *with* $P(A) > 0$*, then* $\rho_{B'}(u_0, u_1)$ *is continuous on each of the sets* I^k*. Moreover, if* $K \equiv \bar{I} \setminus I$ *denotes the boundaries of the sets* $\bar{I}$*, then by setting*

$$\rho_{B'}(u_0, u_1) \equiv 0, \text{ for all } (u_0, u_1) \in K \tag{16}$$

$\rho_{B'}(u_0, u_1)$ *can be extended continuously to* $\bar{I}$*.*

Proof. Continuity of $\rho_{B'}(u_0, u_1, \varphi_0(B', u_0), \varphi_1(B', u_1))$, on each I^k, $k = 0, 1$, follows from the fact that y_{u_0} and c_{u_0} as well as z_{u_1} and d_{u_1} depend continuously on u_0 and u_1 respectively.

We study next continuity at the boundaries. Consider first the set I^0 and its boundary $K^0 = \{(0, P(A)), (P(A), 0)\}$. Given $(u_0, u_1) \in I^0$, we will show that (14) converges to 0 as $(u_0, P(A) - u_0) \to (0, P(A))$. The summation term in (14) converges to 0 given that $\sqrt{\frac{u_0(P(A)-u_0)}{P(A)}}$ approaches 0. The second term in (14) also converges to 0 by an application of Jensen's inequality, details for this argument are provided in [4]. The same arguments prove that (14) converges to 0 as $(u_0, P(A) - u_0) \to (P(A), 0)$.

Now consider the set I^1 and its boundary $K^1 \equiv \bar{I}^1 \setminus I^1 = \{(0, u_1), 0 \le u_1 \le P(A)\} \cup \{(u_0, 0), 0 \le u_0 \le P(A)\}$. Fix $(\hat{u}_0, 0)$ with $\hat{u}_0 \in (0, P(A)]$, we will show that for a given $\epsilon > 0$, there is a $\delta > 0$ so that for all $(u_0, u_1) \in I^1$ with $\|(u_0, u_1) - (\hat{u}_0, 0)\| < \delta$ then $|\rho_{B'}(u_0, u_1, \varphi_0(B', u_0), \varphi_1(B', u_1))| < \epsilon$. To check this it is enough to indicate that the term $\sqrt{\frac{u_0\ u_1}{(u_0+u_1)}}\ \frac{1}{u_1} \int \langle X, B' \rangle\ \varphi_1$ in (14) is controlled by $\frac{1}{\sqrt{u_1}} \int \langle X, B' \rangle\ \varphi_1$ (as long as u_0 remains close to $\hat{u}_0 > 0$) which goes to zero as $u_1 \to 0^+$ by applying Jensen's inequality as detailed in [4]. The term $\sqrt{\frac{u_0\ u_1}{(u_0+u_1)}}\ \frac{1}{u_0} \int \langle X, B' \rangle\ \varphi_0$ in (14) is controlled by $\sqrt{u_1} \int \langle X, B' \rangle\ \varphi_0$ (as long as u_0 remains close to $\hat{u}_0 > 0$) which goes to zero as $u_1 \to 0^+$. The same arguments apply to the case of a boundary point of the form $(0, \hat{u}_1)$ for a fixed $\hat{u}_1 \in (0, P(A)]$.

It remains to consider the boundary point $(0, 0)$, notice that $\sqrt{\frac{u_0\ u_1}{(u_0+u_1)}} \le \min(\sqrt{2\ u_0},\ \sqrt{2\ u_1})$, hence

$$|\rho_{B'}(u_0, u_1, \varphi_0(B', u_0), \varphi_1(B', u_1))| \le \left(\frac{\sqrt{2u_1}}{u_1}\ \left| \int X\ \varphi_1 \right| + \frac{\sqrt{2u_0}}{u_0}\ \left| \int X\ \varphi_0 \right| \right), \tag{17}$$

each of the two terms in the right hand side of (17) can be made arbitrarily small (as we argued previously) by taking (u_0, u_1) sufficiently close to $(0, 0)$. □

Using Lemma 1, let (u_0^*, u_1^*) denote the element in $\bar{I}$ such that $\rho_{B'}(u_0, u_1) \le \rho_{B'}(u_0^*, u_1^*)$ for all $(u_0, u_1) \in \bar{I}$. Given that the optimization is carried over the set I, it follows that

$$y_{u_0^*} \le z_{u_1^*}. \tag{18}$$

It is also important to keep in mind that:

$$u_i^* \equiv u_i^*(B'), \quad i = 0, 1. \tag{19}$$

Proposition 3. *Fix* $B' \in S^d$ *and* $A \in \mathcal{A}$ *with* $P(A) > 0$*. If* $\langle X, B' \rangle$ *is constant a.e. on A then*

$$\sup_{(u_0,u_1)\in I} \sup_{A_0,A_1} \gamma(B', u_0, u_1, A_0, A_1) = 0. \tag{20}$$

If $\langle X, B'\rangle$ is not constant on A, then, without loss of generality, we may assume $(u_0^*, u_1^*) \in I$. Moreover,

$$c_{u_0^*} = 0 \text{ or } c_{u_0^*} = 1, \tag{21}$$

and

$$d_{u_1^*} = 0 \text{ or } d_{u_1^*} = 1. \tag{22}$$

If $c_{u_0^*} = 0$ set $A_0^* \equiv \{w \in A : \langle X, B'\rangle < y_{u_0^*}\}$ and if $c_{u_0^*} = 1$ set $A_0^* \equiv \{w \in A : \langle X, B'\rangle \leq y_{u_0^*}\}$.
If $d_{u_1^*} = 0$ set $A_1^* \equiv \{\langle X, B'\rangle > z_{u_1^*}\}$, and $d_{u_1^*} = 1$ set $A_1^* \equiv \{w \in A : \langle X, B'\rangle \geq z_{u_1^*}\}$. In any case, equations (12) and (13) become:

$$\varphi_i = \varphi_i(B', u_i^*) = \varphi_i(u_i^*(B')) = \mathbf{1}_{A_i^*},$$

and

$$0 < \sup_{(u_0,u_1)\in I} \sup_{A_0,A_1} \gamma(B', u_0, u_1, A_0, A_1) = \rho_{B'}(u_0^*, u_1^*, \varphi_0(u_0^*(B')), \varphi_1(u_1^*(B'))) = \tag{23}$$

$$\gamma(B', u_0^*, u_1^*, A_0^*, A_1^*).$$

Proof. If $\langle X, B'\rangle$ is constant on A, then $[X, \Psi] = 0$ for all $\Psi \in \mathcal{D}_A$, hence (20) follows from (65). Whenever $\langle X, B'\rangle$ is not constant on A, it follows that $\rho_{B'}(u_0, u_1) > 0$ for some $(u_0, u_1) \in I$. Also $\rho_{B'}$ is equal to zero in the boundary of I by (16); hence, without loss of generality, we may assume $(u_0^*, u_1^*) \in I$. Therefore, we need only to establish (21) and (22) as (23) will then follow from Proposition 2.

We rewrite (14) as follows

$$\rho_{B'}(u_0, u_1) = \sqrt{\frac{u_0\, u_1}{u_0 + u_1}}\ [(\frac{1}{u_1}\int_{\{\langle X,B'\rangle > z_{u_1}\}\cap A} (\langle X, B'\rangle - z_{u_1})\, dP - \tag{24}$$

$$\frac{1}{u_0}\int_{\{\langle X,B'\rangle < y_{u_0}\}\cap A} (\langle X, B'\rangle - y_{u_0})\, dP) + (z_{u_1} - y_{u_0})],$$

and define

$$\phi(u_0, u_1) \equiv \sqrt{\frac{u_0\, u_1}{u_0 + u_1}}\ [\frac{1}{u_1}\int_{\{\langle X,B'\rangle > z_{u_1^*}\}\cap A} (\langle X, B'\rangle - z_{u_1^*})\, dP -$$

$$\frac{1}{u_0}\int_{\{\langle X,B'\rangle < y_{u_0^*}\}\cap A} (\langle X, B'\rangle - y_{u_0^*})\, dP + (z_{u_1^*} - y_{u_0^*})].$$

We will first consider $u_0^* + u_1^* = P(A)$, handling this case will also prove all the required statements for the case when $\mathcal{D}_A = \mathcal{D}_A^0$. Notice that under this condition $y_{u_0^*} = z_{u_1^*}$. If we take $u_1 = P(A) - u_0$, with some abuse of notation, we then have

$$\phi(u_0) \equiv \phi(u_0, P(A) - u_0) =$$

$$\frac{1}{\sqrt{P(A)u_0(P(A) - u_0)}}\left[u_0\int_A (\langle X, B'\rangle - y_{u_0^*}) - P(A)\int_{\{\langle X,B'\rangle < y_{u_0^*}\}\cap A} (\langle X, B'\rangle - y_{u_0^*})\right] =$$

$$\frac{\sqrt{P(A)}\ [P(A)\, A_2 - u_0\, A_1]}{\sqrt{u_0(P(A) - u_0)}}$$

where

$$A_1 \equiv \frac{-1}{P(A)} \int_A (\langle X, B' \rangle - y_{u_0^*}), \;\; A_2 \equiv \frac{-1}{P(A)} \int_{\{\langle X,B' \rangle \leq y_{u_0^*}\} \cap A} (\langle X, B' \rangle - y_{u_0^*}).$$

Notice that

$$a_0 \equiv P(\{w \in A : \langle X(w), B' \rangle < y_{u_0^*}\}) \leq u_0^* \leq a_1 \equiv P(\{w \in A : \langle X(w), B' \rangle \leq y_{u_0^*}\}),$$

and that

$$y_{u_0} = y_{u_0^*} \;\text{ for all } u_0 \in [a_0, a_1].$$

It follows that

$$\phi(u_0) = \rho_{B'}(u_0, P(A) - u_0) \quad \text{for all } u_0 \in [a_0, a_1]. \tag{25}$$

In particular $\phi(u_0^*) = \rho_{B'}(u_0^*, P(A) - u_0^*)$ therefore, it is enough to consider $u_0 \in [a_0, a_1]$ in the analysis that follows. A computation gives the derivative of ϕ with respect to u_0; by noticing that $A_2 \geq A_1$, a simple analysis allow us to conclude that $\phi(u_0)$ reaches its maximum value at a_0 or a_1. This result combined with (25) proves that $u_0^* = a_0$ or $u_0^* = a_1$ which is exactly (21). Now, under the present case, namely, $u_0^* = P(A) - u_1^*$, $c_{u_0^*} = 0$ gives $d_{u_1^*} = 1$. Similarly $c_{u_0^*} = 1$ implies $d_{u_1^*} = 0$, therefore (22) also holds.

We will consider next the case when $u_0^* + u_1^* < P(A)$. The idea is the same as above, namely equations (21) and (22) are a necessary condition of the fact that (u_0^*, u_1^*) gives the maximum value.

Introducing the notation

$$B_1 \equiv -\int_{\{\langle X,B' \rangle > z_{u_1^*}\} \cap A} (\langle X, B' \rangle - z_{u_1^*}), \;\; B_0 \equiv -\int_{\{\langle X,B' \rangle < y_{u_0^*}\} \cap A} (\langle X, B' \rangle - y_{u_0^*}).$$

we can then write,

$$\phi(u_0, u_1) = \frac{u_1 B_0 - u_0 B_1 + (z_{u_1^*} - y_{u_0^*})\, u_0\, u_1}{\sqrt{u_0\, u_1\, (u_0 + u_1)}}.$$

In order to prove (21) we will consider $u_1 = u_1^*$ fixed and $u_0 \in [a_0, a_1]$, a_0, a_1 as defined above. As we indicated, $y_{u_0} = y_{u_0^*}$ for all $u_0 \in [a_0, a_1]$, this gives

$$\phi(u_0, u_1^*) = \rho_{B'}(u_0, u_1^*) \text{ for all } \;\; u_0 \in [a_0, a_1]. \tag{26}$$

Therefore, it is enough to consider $u_0 \in [a_0, a_1]$ in the analysis that follows. Calculating the derivative ϕ' of $\phi(u_0, u_1^*)$ with respect to u_0, a simple analysis using the fact that $B_1 u_1^* \geq 0$ shows that $\phi(u_0, u_1^*)$ reaches a maximum at a_0 or a_1. This implies (21). In order to prove (22), notice that

$$b_0 \equiv P(\{w \in A : \langle X(w), B' \rangle > z_{u_1^*}\}) \leq u_1^* \leq b_1 \equiv P(\{w \in A : \langle X(w), B' \rangle \geq z_{u_1^*}\}).$$

We will then consider $\phi(u_0^*, u_1)$ with $u_1 \in [b_0, b_1]$. Computing the derivative of ϕ with respect to u_1 and using the fact that $B_1 u_0^* \leq 0$ reveals that $u_1^* = b_0$ or $u_1^* = b_1$ which concludes the proof. □

It is important to remark that, as noted in (19),

$$A_i^* = A_i^*(u_i^*(B')), \;\; i = 0, 1. \tag{27}$$

Corollary 1. *Fix $A \in \mathcal{A}$ with $P(A) > 0$, then: X is constant on A a.e. if and only if* $\sup_{\Psi \in \mathcal{D}_A}[X, \Psi] = 0$. *Moreover, if X is not constant on A:*

$$0 < \sup_{\Psi \in \mathcal{D}_A}[X, \Psi] = \sup_{B'} \left[\sup_{u_0, u_1} \left(\sup_{A_0, A_1} \gamma(B', u_0, u_1, A_0, A_1) \right) \right] = \sup_{B'}[X, \Psi_{u_0^*, u_1^*}(B')]. \tag{28}$$

Where

$$\Psi_{u_0^*, u_1^*}(B') \equiv \sqrt{\frac{u_0^*\, u_1^*}{u_0^* + u_1^*}}\, B' \left(\frac{\mathbf{1}_{A_1^*}}{u_1^*} - \frac{\mathbf{1}_{A_0^*}}{u_1^*} \right) \in \mathcal{D}_A, \tag{29}$$

and we have used the notation introduced in Proposition 3 and (5).

Proof. If X is constant on A a.e. then $\sup_{\Psi \in \mathcal{D}_A}[X, \Psi] = 0$ is obvious, conversely, assume X is not constant on A, then there is $B' \in S^d$ and a constant c such that $A_0 \equiv \{w \in A : \langle X(w), B' \rangle \leq c\}$ and $A_1 \equiv \{w \in A : \langle X(w), B' \rangle > c\}$ with $P(A_i) > 0, i = 0, 1$, it is then clear that there exists $\Psi \in \mathcal{D}_A$ so that $[X, \Psi] > 0$.

Equation (28) follows from (23), it only remains to check that $\Psi_{u_0^*, u_1^*}(B') \in \mathcal{D}_A$, in turn, this only requires that we check if $A_0^* \cap A_1^* = \emptyset$ (up to sets of measure zero). From (18), and the definitions of A_0^* and A_1^* introduced in Proposition 3, it follows that it is enough to consider the case in which $y_{u_0^*} = z_{u_1^*}$ and $P(\{w : \langle X(w), B' \rangle = y_{u_0^*}\} \cap A) > 0$ and $A_0^* = \{w \in A : \langle X(w), B' \rangle \leq y_{u_0^*}\}$ and $A_1^* = \{w \in A : \langle X(w), B' \rangle \geq z_{u_1^*}\}$. Notice that this situation will be impossible as it contradicts the optimization constraint $u_0^* + u_1^* \leq P(A)$. □

Proposition 4. *Given $A \in \mathcal{A}$ with $P(A) > 0$, there exists $\Psi_A^0 \in \mathcal{D}_A$ so that*

$$[X, \Psi_A^0] = \sup_{B'}[X, \Psi_{u_0^*, u_1^*}(B')]. \tag{30}$$

Proof. We may assume that X is not constant on A, then, using the notation described in (14) and (27) we define the following function of $B' \in S^d$:

$$\delta(B') \equiv \rho_{B'}(u_0^*(B'), u_1^*(B'), \mathbf{1}_{A_0^*(u_0^*(B'))}, \mathbf{1}_{A_1^*(u_1^*(B'))}), \text{ whenever } \langle X, B' \rangle \text{ is not constant a.e. on } A$$

$$\text{and } \delta_X(B') \equiv 0, \text{ otherwise.}$$

Then, (30) will follow if we can prove that $\delta(B')$ is continuous on S^d for $d \geq 2$. The proof of (30) in the simpler case $d = 1$ (which degenerates into $B' \in \{-1, 1\}$) was treated in [4]. Consider first $\hat{B}' \in S^d$ with $\delta(\hat{B}') = 0$. Given $\epsilon > 0$ define the neighborhood $\|B' - \hat{B}'\| \leq \beta$ with $2^{3/2}\, P(A)\, \|X\|\, \beta \leq \epsilon$. Let B' be in such a neighborhood, we may assume $\langle X, B' \rangle$ is not constant on A, it follows then that

$$|\delta(B') - \delta(\hat{B}')| = |\delta(B')| \leq 2^{3/2}\, P(A)\, \|X\|\, \beta \leq \epsilon. \tag{31}$$

Consider next $\hat{B}' \in S^d$ with $\delta(\hat{B}') \neq 0$. Then, it can be seen that there exists a neighborhood of $\hat{B}'$ such that for any B' in such a set, there is a subset of A of full measure where $\langle X, B' \rangle$ is not constant on that subset. We remark that this statement is easy to prove when X is discrete but not easy in the general case. If we concentrate on such a small neighborhood, continuity at $\hat{B}'$ will follow from the continuity of $u_i^* = u_i^*(B')$ as a function of B'. This in turn follows from the following argument. Consider,

$$\int_{A_i} \langle X, B' \rangle\, dP \tag{32}$$

where A_0 is of the form $\{w \in A : \langle X, B' \rangle < y_{u_0}\}$ or $\{w \in A : \langle X, B' \rangle \leq y_{u_0}\}$. With similar expressions for A_1. To complete the argument, notice that the expression in (32) is continuous as a function of both arguments, u_i and B', this fact plus the definition (14) for $\rho_{B'}$ proves that $u_i^*(B')$ depends continuously on B'. □

From Proposition 4 it follows that there exists $B'^* \in S^d$ so that

$$\Psi_A^0 = \Psi_{u_0^*, u_1^*}(B'^*) \tag{33}$$

where we have used the notation in (29).

Allowing for some abuse in the notation used in (19), from now on (and in particular in the next Proposition) we will use the following notation

$$u_i^* \equiv u_i^*(B'^*), i = 0, 1,$$

and so, of course, $A_i^* = A_i^*(u_i^*(B'^*))$.

The next proposition offers a useful necessary condition satisfied by A_0^*, A_1^*, B'^* and u_i^* (which, of course, obey the constraint $u_i^* = P(A_i^*)$). Equation (34) plays a key role in the results described in Appendix B, in turn, those results are crucial in our proof of convergence presented in Section 5.

Proposition 5. *The following identity is satisfied by A_0^*, A_1^*, B'^*, u_0^* and u_1^**

$$B'^* = \frac{\frac{1}{u_1^*}\int_{A_1^*} X \; dP - \frac{1}{u_0^*}\int_{A_0^*} X \, dP}{\sqrt{\sum_{k=1}^{d}\left(\frac{1}{u_1^*}\int_{A_1^*} X[k] \, dP - \frac{1}{u_0^*}\int_{A_0^*} X[k] \, dP\right)^2}}. \tag{34}$$

Proof. Consider the following function of $B' \in S^d$

$$\beta(B') \equiv \gamma(B', u_0^*, u_1^*, A_0^*, A_1^*) = \sqrt{\frac{u_0^* \, u_1^*}{u_0^* + u_1^*}} \times \tag{35}$$

$$\left(\frac{1}{u_1^*}\int_{A_1^*} \langle X(w), B' \rangle dP(w) - \frac{1}{u_0^*}\int_{A_0^*} \langle X(w), B' \rangle \, dP(w)\right),$$

where we have used (65). Moreover, $A_0^* \equiv \{w \in A : \langle X, B'^* \rangle < y_{u_0^*}\}$ or $A_0^* \equiv \{w \in A : \langle X, B'^* \rangle \leq y_{u_0^*}\}$ and $A_1^* \equiv \{\langle X, B'^* \rangle > z_{u_1^*}\}$ or $A_1^* \equiv \{w \in A : \langle X, B'^* \rangle \geq z_{u_1^*}\}$ accordingly to Proposition 3. The function β is a linear function under the constraint $B' \in S^d$, therefore, it can be optimized by Lagrange multipliers. Performing this optimization shows that it has a single maximum $\hat{B}'$ which satisfies the following identity

$$\hat{B}' = \frac{\frac{1}{u_1^*}\int_{A_1^*} X \; dP - \frac{1}{u_0^*}\int_{A_0^*} X \, dP}{\sqrt{\sum_{k=1}^{d}\left(\frac{1}{u_1^*}\int_{A_1^*} X[k] \, dP - \frac{1}{u_0^*}\int_{A_0^*} X[k] \, \varphi_0 dP\right)^2}}. \tag{36}$$

Therefore $\beta(\hat{B}') \geq \gamma(B'^*, u_0^*, u_1^*, A_0^*, A_1^*) = [X, \Psi_A^0]$. It follows then that, without loss of generality, we may take $\hat{B}' = B'^*$, hence B'^* satisfies (36) as we wanted to prove. □

5. Formal Description of the Vector Greedy Splitting (VGS) Algorithm

Given a set $A \in \mathcal{A}$ let Ψ_A^0 and Ψ_A^1 be the functions introduced in (30) and (79) respectively. The partition of A into A_0^*, A_1^* (as introduced in Proposition 3) and $A_2^* \equiv A \setminus \left(A_0^* \cup A_1^*\right)$ will be called the *best split* of A and the sets A_i^*, $i = 0, 1, 2$ are called the best children of A. Notice that $A_2^* = \emptyset$ is possible, in particular this will be the case when we restrict the optimizations to $\mathcal{D}_A^0$ (i.e. the Haar case).

The VGS algorithm will be described for both classes of functions, $\mathcal{D}_A^k$, simultaneously by introducing the indexes $J_A^0 \equiv 1$ for the case $\mathcal{D}_A^0$ and, for the case $\mathcal{D}_A^1$, we set $J_A^1 \equiv 1$ if $P(A_2^*) = 0$ and $J_A^1 \equiv 2$ otherwise. Analogously to how we have done elsewhere in this paper, we will use J_A to denote any of the numbers J_A^k.

Given $X \in L^2(\Omega, \mathbb{R}^d)$ (this hypothesis will be required in the remaining of the paper but not made explicit again), the VGS algorithm builds a sequence of partitions $\mathcal{Q}_n$ of Ω indexed by $n = 1, 2, ...$; this index will be referred as the n-th iteration of the VGS algorithm. The partitions are defined recursively:

- Let $\mathcal{Q}_0 = \{\Omega\}$
- Assuming that $\mathcal{Q}_n$ has been created, then $\mathcal{Q}_{n+1}$ is generated as follows:
  Consider $\hat{A} \in \mathcal{Q}_n$ with $P(\hat{A}) > 0$, such that it satisfies

$$\|[X, \Psi_{\hat{A}}^0]\| \geq \|[X, \Psi_A^0]\| \text{ for all } A \in \mathcal{Q}_n. \tag{37}$$

  Now, if

$$[X, \Psi_{\hat{A}}^0] = 0 \text{ or } P(A) = 0 \text{ for all } A \in \mathcal{Q}_n,$$

  the algorithm terminates and $\mathcal{Q}_p \equiv \mathcal{Q}_n$ for all $p \geq n$. Otherwise, i.e. $P(\hat{A}) > 0$ and $[X, \Psi_{\hat{A}}] \neq 0$, we set

$$\mathcal{Q}_{n+1} = \mathcal{Q}_n \setminus \{\hat{A}\} \bigcup_{i=0}^{J_{\hat{A}}} \{\hat{A}_i^*\} \tag{38}$$

  where, as indicated previously, the sets $\hat{A}_i^*$ are the best children of $\hat{A}$.

The VGS algorithms builds a tree $\mathcal{T}$ where its nodes are atoms from the partitions $\mathcal{Q}_n$. Define first the n-iteration tree by

$$\mathcal{T}_n = \bigcup_{i=0}^{n} \mathcal{Q}_i \text{ and } \mathcal{T} = \bigcup_{n=0}^{\infty} \mathcal{Q}_n.$$

The parent-children relationship is given by the *best split* relationship mentioned previously.

We will define an increasing sequence of orthonormal systems $\mathcal{H}_n$, for $n \geq 0$, corresponding to the n-th. iteration of the VGS algorithm as follows: $\mathcal{H}_0 \equiv \{U_0 \equiv \Psi_\emptyset^0\}$ where

$$\Psi_\emptyset^0 \equiv C\, \mathbf{1}_\Omega \text{ and } C[i] = \frac{\int_\Omega X[i]\, dP}{\|X\|}. \tag{39}$$

Assume, recursively, that $\mathcal{H}_n = \{U_0, \ldots, U_{k_n}\}$ has been constructed, we then let,

$$\mathcal{H}_{n+1} \equiv \mathcal{H}_n \bigcup_{i=0}^{J_{\hat{A}}-1} \{\Psi_{\hat{A}}^i\}$$

where $\hat{A}$ is the set in (37), also set $U_{k_n+i+1} \equiv \Psi^i_{\hat{A}}$ for $i = 0, \ldots, J_{\hat{A}} - 1$, so $\Psi^1_A \in \mathcal{H}_{n+1}$ only when $J_{\hat{A}} = 2$. We also define

$$\mathcal{H} \equiv \bigcup_{n \geq 0} \mathcal{H}_n. \tag{40}$$

Clearly, $\mathcal{H}$ is a vector valued orthonormal system.
We will associate the obvious approximation to the tree $\mathcal{T}_n$, this approximation will only involve internal nodes i.e. it will exclude leafs (final nodes). We will use the following notation,

$$\mathcal{L}(\mathcal{T}_n) \equiv \{ \text{ set of leafs of } \mathcal{T}_n\},\ \mathcal{T}_n^\circ \equiv \mathcal{T}_n \setminus \mathcal{L}(\mathcal{T}_n). \text{ Also set } \mathcal{T}_0^\circ \equiv \{\emptyset\}.$$

Given $\mathcal{T}_n$, the associated *VGS approximation* is defined by the following equation

$$X_{\mathcal{T}_n} \equiv \sum_{A \in \mathcal{T}_n^\circ} \sum_{i=0}^{J_A - 1} [X, \Psi^i_A]\, \Psi^i_A. \tag{41}$$

Clearly, the outer summation in (41) can be rewritten recursively as follows, starting with the first iteration (taking $J_\emptyset \equiv 1$),

$$X_{\mathcal{T}_0} = [X, \Psi^0_\emptyset]\, \Psi^0_\emptyset. \tag{42}$$

In general

$$X_{\mathcal{T}_{n+1}} = X_{\mathcal{T}_n} + \sum_{i=0}^{J_{\hat{A}^*} - 1} [X, \Psi^i_{\hat{A}}]\, \Psi^i_{\hat{A}}.$$

It is a simple observation that the terms $[X, \Psi^i_{\hat{A}}]\, \Psi^i_{\hat{A}}$, $i = 0, 1$, are martingale differences with respect to the sigma algebra generated by VGS, so (41) is a martingale sequence. The following theorem establishes more than this as it shows that we actually have a conditional expectation martingale. The result, as well as Theorem 3, relies on connecting the vector valued approximation to a generalized H-system. The technical details are provide in Appendix B, the following Theorem makes use of $\mathcal{G} = \{u_k\}$ introduced in (81).

Theorem 2. *The sequence $\{X_{\mathcal{T}_n}\}$, $n \geq 0$, of $\mathbb{R}^d$-valued random variables is a martingale sequence with respect to the filtration $\mathcal{F}_n \equiv \sigma(\mathcal{Q}_n)$. Moreover,*

$$X_{\mathcal{T}_n}[i] = X_{\mathcal{A}_{k_n}}[i] = x[i]_{\mathcal{A}_{k_n}} \equiv \mathbf{E}(x[i] | u_0, u_1, \ldots, u_{k_n}) =$$

$$\sum_{k=0}^{k_n} [x[i], u_k]_s\, u_k = \sum_{k=0}^{k_n} [X, U_k]\, U_k[i],$$

and $\mathcal{A}_k \equiv \sigma(u_0, \ldots, u_k) = \sigma(U_0, \ldots, U_k)$.

Proof. The result follows immediately from Theorem 6 from Appendix B after noticing that $\sigma(\mathcal{Q}_n) = \mathcal{B}_n \equiv \mathcal{A}_{k_n} \equiv \sigma(u_0, \ldots, u_{k_n})$ where $\mathcal{G}_n \equiv \{u_0, \ldots, u_{k_n}\}$ is the generalized H-system used in Theorem 6. □

We now embark in the proof of convergence of the VGS algorithm (Theorem 3), for completeness we state the following simple lemma.

Lemma 2. *Given $A \in \mathcal{Q}_{n_0}$ and if X restricted to A is not constant, then there exists $n_1 > n_0$ such that VGS splits A before or at iteration n_1.*

Proof. From

$$\sum_{k\geq 0} |[X, U_k]|^2 \leq \|X\|^2,$$

it follows

$$\lim_{k\to\infty} [X, U_k] = 0. \tag{43}$$

By hypothesis $|[X, \psi_A^0]| > 0$, then using (43) find n_1 such that $[X, U_k]| < |[X, \psi_A]|$ for all $k > n_1$; by the definition of VGS we then know that VGS will best split A before or at iteration n_1. □

For simplicity we will assume X is a *discrete* $\mathbb{R}^d$-valued random variable. We will make this assumption explicitly whenever we will make use of it; the hypothesis is general enough for the scope of the paper but most likely the results hold with more generality. The hypothesis of discreteness means X takes values $\{C_k\}$, $C_k \in \mathbb{R}^d$, with some abuse of notation we will use the same notation for the pre-images, namely, $C_k = \{w \in \Omega : X(w) = C_k\}$, these sets will be called *cells*. We also allow $P(C_j) = 0$ for some values of j. the next Lemma shows that VGS does not split cells.

Lemma 3. *Assume X is a discrete $\mathbb{R}^d$-valued random variable, given $C_k \subseteq A \in \mathcal{A}$ with $P(A) > 0$ and A_i^*, $i = 0, 1, 2$, the best children of A (with A_2^* possibly empty), then $C_k \subseteq A_j^*$ for some $j \in \{0, 1, 2\}$.*

Proof. The proof follows from (21) and (22) and the definition of the VGS algorithm. □

The above lemma allows to define in an obvious way, for a given cell C_k, a sequence of atoms $\{A^n(C_k)\}$ generated by VGS (i.e. $A^n(C_k) \in \mathcal{Q}_m$ for some $m \geq n$) such that $A^{n+1}(C_k) \subseteq A^n(C_k)$ and $C_k \subseteq A^n(C_k)$ for all $n \geq 0$. Define then

$$Q(C_k) \equiv \cap_n A^n(C_k).$$

The next Lemma is crucial in order to establish Theorem 3.

Lemma 4. *Assume X is discrete as indicated above. For any C_k with $P(C_k) > 0$*

$$X(w) = C_k \ \textit{for almost every} \ \ w \in Q(C_k),$$

and so

$$Q(C_k) = C_k \ \textit{up to sets of P-measure} \ \ 0.$$

Proof. Notice that $C_k \subseteq Q(C_k)$ so $P(Q(C_k)) > 0$. If X is constant a.e. on $Q(C_k)$ the results follow. Assume then, that X is not constant on $Q(C_k)$. The proof then follows as in Lemma 7 from [4], we reproduce it here for completeness.

We set

$$K = \sup\{|[X, \Psi]| : \Psi \in \mathcal{D}_{Q(C_k)}\}.$$

We observe that $K > 0$. We claim there exists $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$,

$$\sup\{|[X, \Psi]| : \Psi \in \mathcal{D}_{A^n(C_k)}\} > \frac{K}{2}.$$

Notice that because $\mathcal{D}_A^0 \subseteq \mathcal{D}_A^1$, we can concentrate on $\mathcal{D}^0$ without loss of generality. In fact, we have that there exist $\Psi \equiv B_0 1_{A_0} + B_1 1_{A_1} \in \mathcal{D}^0_{Q(C_k)}$ such that $|[X,\Psi]| > \frac{K}{2}$. Let $C^n \equiv A^n(C_k) \setminus Q(C_k)$ and define $\Psi^n \equiv B_0^n 1_{A_0 \cup C^n} + B_1^n 1_{A_1} \in \mathcal{D}^0_{A^n(C_k)}$ where

$$A_0^n = \sqrt{\frac{P(A_0) + P(C^n)}{P(A_1)(P(A_0) + P(A_1) + P(C^n))}}$$

and

$$A_1^n = -\sqrt{\frac{P(A_1)}{(P(A_0) + P(C^n)(P(A_0) + P(A_1) + P(C^n))}}.$$

The monotone convergence theorem implies that $\lim_{n\to\infty} P(C^n) = 0$, hence $\lim_{n\to\infty} \Psi^n = \Psi$ a.e.; it follows that $\lim_{n\to\infty} |[X,\Psi^n]| = |[X,\Psi]|$. This shows that there exist n_0 such that for $n > n_0$, $\sup\{|[X,\Psi]| : \Psi \in \mathcal{D}^0_{A^n(C_k)}\} \geq |[X,\Psi^n]| > \frac{K}{2}$. But $|[X,U_n]| = \sup\{|[X,\Psi]| : \Psi \in \mathcal{D}_{A^n(C_k)}\}$, then we have that $|[X,U_n]| > \frac{K}{2}$ for all $n > n_0$, which is impossible. □

Theorem 3. *Assuming X is a discrete $\mathbb{R}^d$-valued random variable (as described above), then:*

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

Proof. Using Theorem 6 from Appendix B (or, equivalently, Theorem 2) we can see that in order to prove the theorem it is enough to prove

$$\lim_{n\to\infty} \frac{1}{P(A^n(C_k))} \int_{A^n(C_k))} X\, dP = C_k \quad \text{for all} \quad k \quad \text{such that} \quad P(C_k) > 0.$$

This last equation follows easily using $\lim_{n\to\infty} P(A^n(C_k)) = P(Q(C_k))$ and Lemma 4. □

Proposition 6. *If X is a simple function that takes q distinct values, then VGS terminates in a finite number of steps N which satisfies $N \leq q \leq |\Omega|$.*

Proof. Let $\mathcal{R}_\Omega(X) = \{C_1, \ldots, C_q\}$ be the finite set of the range values of X on Ω. As we did before, we also use the same notation for the cells defined by $C_i \equiv \{w \in \Omega : X(w) = C_i\}$, we will assume without loss of generality that $P(C_i) > 0$ for all $i = 1, \ldots, q$. It is easy to see that any $A \in \mathcal{Q}_n$, where $n \geq 0$ is arbitrary, can be written as a finite union of some the cells. It then follows that each best split of a given $A \in \mathcal{Q}_n$ will produce best children which can be written as union of a strictly smaller number of generating sets. From Lemma 2 this process will continue until the remaining atoms are made up of a single generating set or X is constant on all these atoms. In any of these cases X will be constant for all the given atoms and hence VGS terminates. If we let N denote the smallest integer such that $\mathcal{Q}_n = \mathcal{Q}_N$ for all $n \geq N$, a simple counting argument proves that $N \leq q$. □

The following Theorem summarizes some formal properties of the scalar orthonormal systems constructed in this paper. One could use this Theorem to derive Theorem 3 but we preferred the direct proof of that theorem instead.

Theorem 4. *If X is a discrete random variable we have*

$$\mathcal{A}_\infty = \bigvee Q_n = \sigma(X). \tag{44}$$

The generalized H-system $\mathcal{G} = \{u_k\}$ introduced in (81) is an unconditional basis for the spaces $L^p(\Omega, \sigma(X), P), \mathbb{R})$, $1 < p < \infty$.

Proof. $\mathcal{A}_\infty = \bigvee Q_n$ is trivial by construction, $\bigvee Q_n = \sigma(X)$ is the key equality and it follows from Lemma 4. The rest of the statements follow from general results in martingale theory (as described in [12]). □

Despite the fact that the VGS algorithm deals with arbitrary geometrical regions it is a computational efficient algorithm. The reason for this is that the atoms processed by the algorithm are level sets of the input data and they can be efficiently manipulated with a computational cost proportional to the size of the range of values of the data. In fact, for a given $B' \in S^d$, the two innermost suprema in (10) are reduced (by means of the Bathtub theorem) to an optimization over the range of values of a scalar random variable over the given atom A. The exhaustive optimization is on the outermost supremum for $B' \in S^d$. This optimization can be implemented efficiently, by combining the local characterization provided by Proposition 5, which can be used as an iterative procedure to find local maxima very quickly, with a global search. Full details on computational costs and implementation are provided in [2]. The implementation we used, based on C++, runs in real time for images of size 128×128 pixels. Computations can be organized in a convenient multi-resolution analysis algorithm as described in [4].

In order to illustrate the algorithm we consider four (so $d = 4$) simple geometrical images taking a reduced number of gray levels. Figure 1 shows the original 256 gray-levels images of 128×128 pixels. In this illustration we will not perform compression (which is the main content of later sections) and the approximations are steps of the algorithm constructing the full tree.

Figure 1: Simple Geometrical set, $d = 4$

Figure 2 shows the VGS approximation using the scalar (see Section 6.4) Haar case (i.e. when $\mathcal{D} = \mathcal{D}^0$) approximation at different stages of convergence. The first approximation (top left) shows the approximation with only one component and clearly the algorithm splits the domain in two atoms in each image. The second approximation (top right) shows the approximation with two components, it is possible to see that the top images are approximated using a partition associated with the bottom images. The third approximation (bottom left) shows that the bottom left image is completely approximated and the top images have almost converged. The last approximation (bottom right) is the approximation using seven components and it is almost a perfect reconstruction but the algorithm needs one more iteration to complete the bottom right image.

Figure 3 shows the VGS scalar approximation when $\mathcal{D} = \mathcal{D}^1$ at different stages of convergence. The first approximation (top left) shows the approximation with only one component,

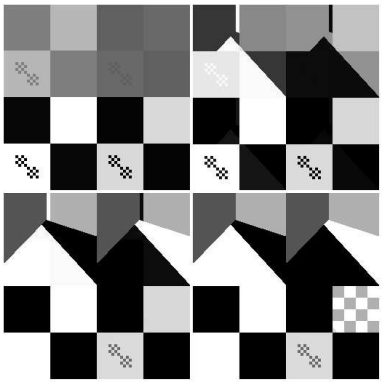

Figure 2: $\mathcal{D} = \mathcal{D}^0$ (Haar) Scalar Approximations. Top left: one component, Top right: two components, Bottom left: three components, Bottom right: seven components

the difference between this method and the previous one, is that the approximating functions ψ in this case can take three values in each node. It is clear that in this case the bottom images were selected to approximate the whole image, and in the second and third approximations it is possible to appreciate how the algorithm tries to approximate the top images based on the bottom images. The last approximation shows almost a complete reconstruction with only seven components, each taking three different values at each node.

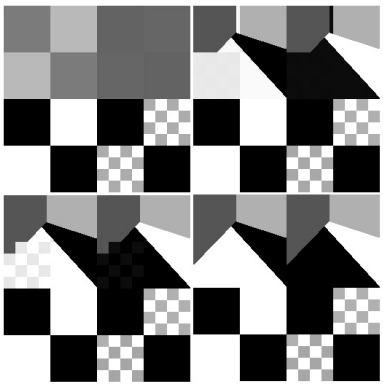

Figure 3: $\mathcal{D} = \mathcal{D}^1$ Scalar Approximations. Top left: one component, Top right: two components, Bottom left: three components, Bottom right: seven components

6. Encoding Of Approximations

Our constructions provide the vector valued orthonormal system $\mathcal{H} = \{U_k\}$, defined in (40), and the H-system $\mathcal{G} = \{u_k\}$ defined in Appendix B (81).

Equation (70) and (80) show that for any finite index set $I \subseteq \mathbb{N}$ we have the fundamental identity

$$\sum_{k\in I} [X, U_k]\ U_k[i] = \sum_{k\in I} [x[i], u_k]_s\ u_k \quad \text{for all} \quad i = 1, \ldots, d. \tag{45}$$

This is a basic result and shows, along with the convergence result from Theorem 3, that one could use the vector valued orthonormal system $\mathcal{H}$ to approximate X or one could use the scalar valued orthonormal system $\mathcal{G}$ to approximate each $x[i]$, $i = 1, \ldots, d$. For practical compression matters, the two systems $\mathcal{H}$ and $\mathcal{G}$ are not equivalent when one considers the optimized expansions as we explain next.

Let $h : \mathbb{N} \to \mathbb{N}$ be a re-ordering function for $\mathcal{H}$ in such a way that

$$|[X, U_{h(0)}]| \geq |[X, U_{h(1)}]| \geq \ldots. \tag{46}$$

We then have the n-term VGS optimized *vector approximation*

$$X_n = \sum_{k=0}^{n-1} [X, U_{h(k)}]\ U_{h(k)}. \tag{47}$$

In practice, the integer n is chosen to satisfy some error criteria, say an vector error level ϵ_v is given so we can find $n = n(\epsilon_v)$ so that $\|X - X_n\| \leq \epsilon_v$. The corresponding n-nodes A_k, which satisfy $\Psi_{A_k} = U_{h(k)}$, will be called *active nodes* (for the given ϵ_v.)

One can define the same notions for the orthonormal system $\mathcal{G}$, let $g_i : \mathbb{N} \to \mathbb{N}$ be one such re-ordering function for each $i = 1, \ldots, d$, so that

$$|[x[i], u_{g_i(0)}]_s| \geq\ |[x[i], u_{g_i(1)}]_s| \geq \ldots. \tag{48}$$

We then define the n-term VGS optimized *scalar approximation(s)* by

$$x[i]_n = \sum_{k=0}^{n-1} [x[i], u_{g(k)}]_s\ u_{g(k)}. \tag{49}$$

(A caution to the reader: the i-th. component of the n-term vector approximation is denoted by $X_n[i]$ while the n-term approximation of the scalar i-th. component is denoted by $x[i]_n$).

Given an scalar error level ϵ_s we will find integers n_i such that

$$\|x[i] - x[i]_{n_i}\| \equiv \sqrt{[x[i] - x[i]_{n_i}, x[i] - x[i]_{n_i}]_1} \leq \epsilon_s \ \text{ for all } \ i = 1, \ldots, d. \tag{50}$$

The corresponding n-nodes A_k, which satisfy $\psi_{A_k,s} = u_{g(k)}$, are called *active nodes* (for the given ϵ_s). For each type of approximation, we will *prune* the tree by keeping only the corresponding active nodes for further processing. Retaining only the active nodes from the full VGS tree results into a data structure which is not a tree.

In the vector case one needs to store the following information at the active nodes: numbers of the form $[X, \Psi_A^0]$ and/or $[X, \Psi_A^1]$ and a corresponding vector B'_A. In the scalar case one needs to store some (or all) of the following numbers: $[X[i], \psi_{A,s}^0]_1$ and/or $[X[i], \psi_{A,s}^1]$, $i = 1, \ldots, d$. Besides the scalar and vector approximations, there exists another approach described in Section 6.7, this approach uses the information on the leaves after the tree has been pruned, it will be called *Leaves average approximation*.

At this point it may be useful to outline some general structures related to the information needed by the decoder to reconstruct the approximation to a given set of images.

The *Partition Map* (PM) encodes the partition which results after the tree has been pruned. Of course, the partition constructed by VGS is input-dependent and needs to be encoded entirely, details are described in Section 6.1. The *Significance Map* (SM) encodes what remains of the tree

structure after pruning has taken place, it contains the information on the number of children and pointers from active nodes to the partition sets (from PM). Additional indexing information is also required to account for particulars inherent to the vector case or scalar case. We will analyze this map in Section 6.2. The *Quantization Map* (QM) is used to quantify the quantized information required for reconstruction at active nodes, it is described in Section 6.3. This information could be inner products, B' or actual averages. In most of our numerical illustrations the bit costs of SM and QM are added together and reported as a single cost, the quantized inner products are naturally stored in the SM data structure and, in most cases, they represent the largest part of the bit costs (beyond the PM costs).

The Greek letter λ will be used throughout this section and the next one to denote inner products. Depending on the context, these inner products will be of the form $[X, \Psi_A^0]$or $[X, \Psi_A^1]$ or $[x[i], \psi_{A,s}^0]_s$ or $[x[i], \psi_{A,s}^1]_s$. Variations on this type of notation, mainly used in the diagrams, should be self explanatory. Some of the images used for illustration refer to input data used for experimentation, this data is formally presented in Section 7.

Sections 6.1, 6.2 and 6.3 describe practical issues related to the encoding of the data structures used in our implementation.

6.1. *Partition Map ($\mathcal{M}_\Pi$)*

Definition: Consider $n \equiv |\Pi(\Omega)|$, where $\Pi(\Omega)$ is a finite partition of Ω , a function $\mathcal{M}_\Pi : \Omega \to N$ is called a *Partition Map* if for each $A_k \in \Pi(\Omega)$, $k = 1, \ldots, n$ it satisfies:

$$\mathcal{M}_\Pi(w) = v_k \quad \forall\, w \in A_k, \text{ if } k \neq j \;\Rightarrow\; v_k \neq v_j.$$

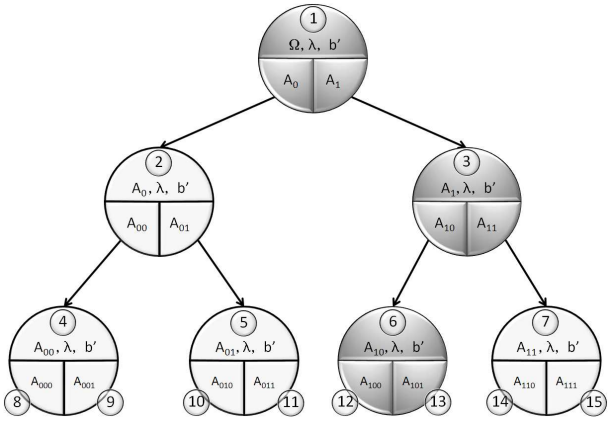

Figure 4: Full tree with active nodes

We describe how the Partition Map is created by means of an example. Figure 4 displays a full tree obtained after three iterations, the partition associated to the full tree is shown in Figure 5 a). Assuming that nodes $\{1, 3, 6\}$ are the only active nodes, the resulting partition is shown in Figure 5 b). Notice that node 2 is not active and hence the atom associated to node 1 is not further split in this case (unless a descendant of node 2 were actually active.)

Remark 3. *The partition map shares the same domain as the input images, and the maximum number of atoms is equal to the number of pixels. In general, an upper bound number of bits to store the partition map is* $\log_2 |\Omega|$.

Re-ordering Partition Values: The partition map could be interpreted as an image even though the values assigned to the atoms are not related. Technically, whenever we restrict to gray scale images, the partition map is not an image because the values in the range of the partition map could be greater than 256. Although it seems to be difficult to reduce the number of values without losing crucial information, there exists the possibility to re-order the values of the atoms in the partition map in such a way that if the distance between two different atoms is small then its corresponding values should be near too; then we need to find a way to measure a distance between atoms. There are different methods to carry out this task. The following is one such method:

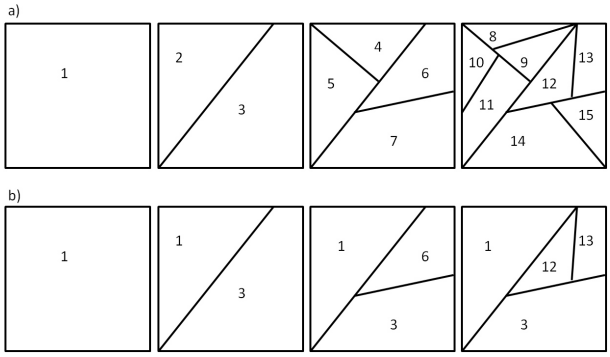

Figure 5: a) Partition using the full tree, b) Partition using the compressed tree

For a given atom $A_k \in \Pi(\Omega)$ we compute the average of the input set in this atom

$$V_k = \frac{1}{d\,|A_k|} \sum_{i=0}^{d} \sum_{w \in A_k} x[i](w),$$

now using V_k is possible to re-order the values associated to each A_k. Defining the sorted set

$$\{V_{h_1}, V_{h_2}, \ldots, V_{h_n}\} \quad \text{such that} \quad V_{h_1} \leq V_{h_2} \leq \cdots \leq V_{h_n}$$

where n the number of elements in $\Pi(\Omega)$, then

$$\mathcal{M}_\Pi(w) = k \quad \forall\, w \in A_{h_k},\ k = 1, \ldots, n.$$

Entropy Encoding:
The word *symbol* will be used to refer to the value assigned to each atom in $\Pi(\Omega)$. The average number of bits needed to encode each symbol is given by

$$H_{\mathcal{M}_\Pi} = -\sum_{i=0}^{n} p_i \log_2 p_i \quad \text{where} \quad p_i = \frac{|A_i|}{|\Omega|},$$

and $N_s = |\Omega|$ is the number of pixels in Ω and so p_i becomes the relative frequency of each symbol. The (theoretical) cost associated to the partition is given by

$$C_{\mathcal{M}_\Pi} = H_{\mathcal{M}_\Pi} \times N_s.$$

Consider an example of two images each of 128×128 pixels (actually, the standard Barbara and Lena images). If we run VGS to an approximation of PSNR = 38.38 db and then we apply the re-order of the partition, the relative frequency of the symbols gives $H_{\mathcal{M}_\Pi} = 7.83$ bpp (bits per pixel) and $C_{\mathcal{M}_\Pi} = 16384 \times 7.83 = 128439.67$ bits, approximately 97.99% of the maximum when $H_{\mathcal{M}_\Pi} = \log_2 256 = 8$. So, there is almost no compression at all, and it becomes obvious when we check that the relative frequency of the symbols is quasi uniform.

Spatial Correlation: There are several approaches to take advantage of any spatial correlation; we have found, in our setup, that one of the best techniques is the following. We suppose that there is a spatial correlation between pixels by assuming that one column is similar to the next, if $w = (x, y)$ then

$$\mathcal{M}_\Pi(x + 1, y) - \mathcal{M}_\Pi(x, y) \sim 0, \tag{51}$$

the relation between lines is also true. Then, it is advantageous to store the difference of the columns instead of the original values. Consider the previous example of the two images but now using the differences in (51). The relative frequency of the symbols is shown in Figure 6, where it is possible to appreciate that most of the symbols are equal to zero.

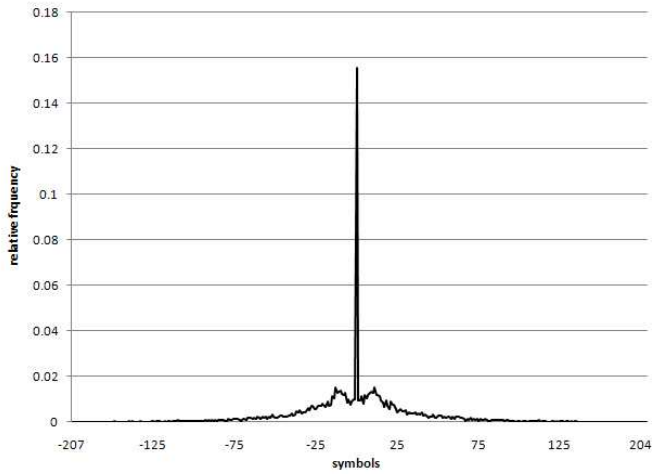

Figure 6: Relative frequency using spatial correlation

The results now are better than before, $H_{\mathcal{M}_\Pi} = 6.8$ bpp and $C_{\mathcal{M}_\Pi} = 16384 \times 6.8 = 111,423.84$ bits, approximately 85.03% of the maximum when $H_{\mathcal{M}_\Pi} = \log_2 256 = 8$, it means a 15% of compression. Although it seems to be not a good compression rate, we have to take in consideration that it is lossless compression, and also if JPEG2K is used to compress the partition image, with a PSNR = 45 db, the size will be approximately equal to the 60% of the image. Therefore the lossless compression using differences seems to work well. There exists also the possibility to apply a lossy compression algorithm to the partition, but the results showed that the algorithm is very sensitive to a change in the partition values. Therefore because the distortion has to be very small, the compression rate for a lossy method is practically the same as the lossless method.

6.2. Significance Map ($\mathcal{M}_S$)

For a given tree, as in Figure 7, we assume in general that we have a function called `compressTree` that selects a number of nodes based on some criteria, a typical output is shown in Figure 8.

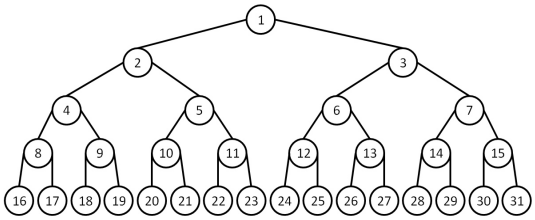

Figure 7: Full tree

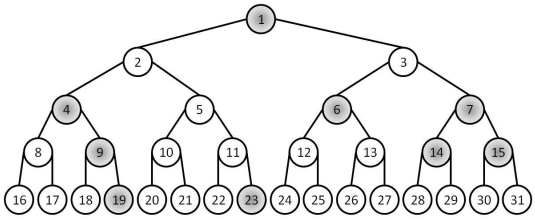

Figure 8: Compressed tree

The significance map stores what remains of the tree information after pruning has taken place. Each node contains the information associated to the approximation on each atom, e.g. inner products and B'. Notice that the significance map also needs to include links from nodes to the partition sets encoded by the partition map. This is the main difference with other methods that use trees to encode inner products. Our case is more difficult due to the fact that the encoder should encode the inner products and the links to the atoms at the same time.

As we can see in Figure 8, if a node is selected we *do not require the ancestors to be included*. Most approaches at this point [1], [13], assume that, under suitable conditions, a significant node does not have any significant children nodes. The zero-trees proposed in [15] make use of this property. In our vector case the extension of that proposition is not clear and also the problem to include a node and not its ancestors can be solved without including much more extra information or introducing any extra computational cost. Due to the complexity of the algorithm, we will describe it by means of an example.

Three different type of symbols are employed to encode the tree, the symbols are used to create a string of symbols called the *significant string* and denoted with $\mathcal{S}$. The symbols are: • Q (Active node), • V (Link to the partition) and • D (Dummy node). Tree nodes are visited using a pre-order traversal method. The algorithm continues until the following string is constructed.

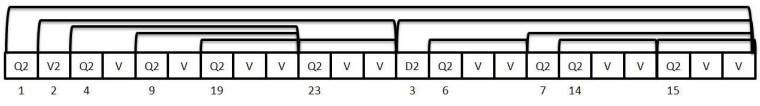

Figure 9: Encoded string

Figure 9 shows the encoded string and Figure 10 shows the decoded tree that is equivalent for reconstruction to the original tree. As indicated, the number of symbols proposed is three, but if we adjoin the number of children to the symbol we can check that the sequence of symbols

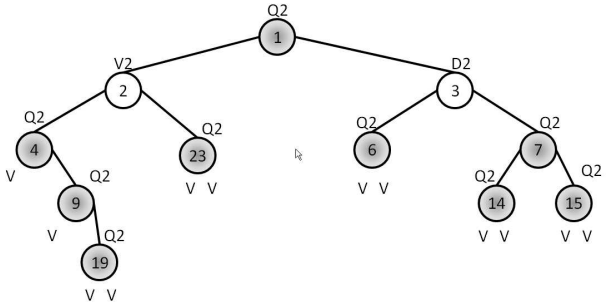

Figure 10: Equivalent decoded tree

"{Q2, V, V}" has a high probability. We can then introduce a new symbol called Q2VV, this is analogous to the zero tree symbol introduced in [15].

Entropy Encoding:

Definition 4. *The algorithm described in the example above can be used to define a function* $\mathcal{M}_S : \mathcal{S} \to \mathbb{Z}$ *called the Significance Map.*

The above definition considers $\mathcal{S}$ as a set constructed as the disjoint union of the symbols used by the algorithm to construct the significance string. Therefore, symbols are included in the set $\mathcal{S}$ as many times as they appear in the significance string.
Set $\mathcal{S}_k = \{s \in \mathcal{S} : \mathcal{M}_S(s) = k \text{ and } k \in \mathbb{Z}\}$ and define the symbol set $\mathcal{J}_S = \{\mathcal{S}_k \subset \mathcal{S} : \mathcal{S}_k \neq \emptyset\}$.
Using entropy encoding as before, the theoretical cost associated to the significance map is given by

$$C_{\mathcal{M}_S} = H_{\mathcal{M}_S} \times |\mathcal{S}|. \tag{52}$$

For the example above using the encoded string in Figure 9,

$$\mathcal{S} = \{Q2, V2, Q2, V, Q2, V, Q2VV, Q2VV, D2, Q2VV, Q2, Q2VV, Q2VV\}.$$

The average number of bits $H_{\mathcal{M}_S} = 2.038$ bits per symbol and the theoretical total cost is equal to $C_{\mathcal{M}_S} = 26.49$ bits, using the standard coding without taking in consideration the entropy, is equal to 30.18 bits it means that we have saved 12.3%. In a real application the number of nodes in a compressed tree are near to 1000 then the number of bits needed to store such a tree is close to 2000.
We remark that the cost associated to the significance map is, relatively speaking, the lowest cost when compared with cost to encode the partition map or the quantization map. In some instances the cost of the significance map may increase [15].

6.3. Quantization Map ($\mathcal{M}_Q$)

The quantization map stores the information (quantized) needed for the reconstruction in each node; entropy encoding is used afterwards. We have found, through experimentation, that the uniform quantization works the best. Given that the actual details of the quantization schemes depend on the different cases, details on quantization will be provided in the appropriate sections. We will also use those sections to summarize the *total* bit costs for each case.

6.4. $\mathcal{D}^0$ (Haar) Case: Scalar Approximation.

This case is referred to as SHVGS (Scalar Haar VGS) in later sections. The information needed in this specific case at each node consists of the scalar inner products $[x[i], \psi_s]_s$ and the partition that is encoded by the partition map. Therefore, the information to be quantized consists of d real numbers, where d is the number of input images. We will quantize these numbers and then use entropy to encode them. The two techniques can be combined and performed simultaneously as in the case of the arithmetic coding, see [15], [14].

Let us call $\lambda_k = [X[i], \psi^0_{A,s}]_s$ to the largest inner products kept after pruning a full tree by means of the scalar approximation. In a real application (from the hand video set, see Figure 18), using the full tree, $\lambda_k \in (0.0001, 200)$, Figure 11 shows the values of λ_k sorted by $|\lambda_k|$. Figure 12 shows the information on each node, our `compressTree` algorithm selects a node if at least one scalar product λ_i is needed at the node.

We will define next the quantization map $\mathcal{M}_Q : Q \to \mathbb{Z}$, where $Q = \{\lambda_i\}$.

Quantization: We have verified, through experimentation, that the best quantization technique for our algorithm is the uniform quantization defined by the following *quantization map*

$$\mathcal{M}_Q(\lambda_k) = \left\lfloor \left\lfloor \frac{\lambda_k}{c} \right\rfloor \times c \right\rfloor \quad \text{and} \quad c > 0. \tag{53}$$

The value of c is found empirically through experimentation.

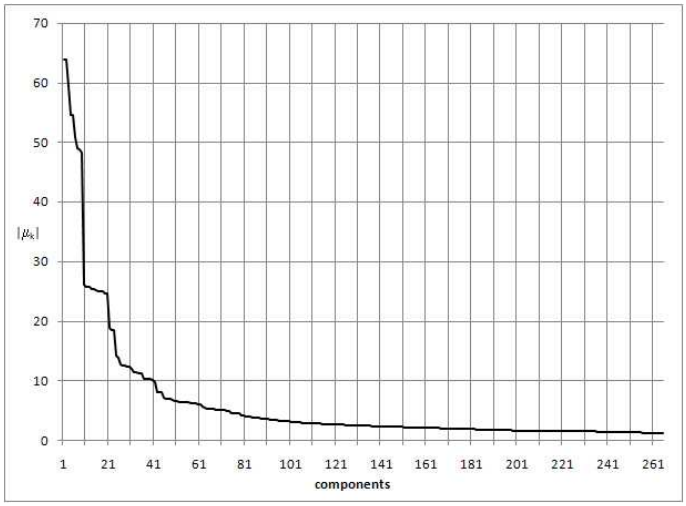

Figure 11: Ordered scalar inner products

Entropy Encoding: Let $Q_k = \{q \in Q : \mathcal{M}_Q(q) = k \quad \text{and} \quad k \in \mathbb{Z}\}$ and so the symbol set is $\mathcal{J}_Q = \{Q_k \subset Q : Q_k \neq \emptyset\}$. Using entropy encoding, as we have done previously, the theoretical total cost associated with the quantization map is denoted by

$$C_{\mathcal{M}_Q} = H_{\mathcal{M}_Q} \times |Q|. \tag{54}$$

Indices Information: The indexing information can be encoded using different approaches. We describe a single such a method here (two other methods are described in [2]). The approach

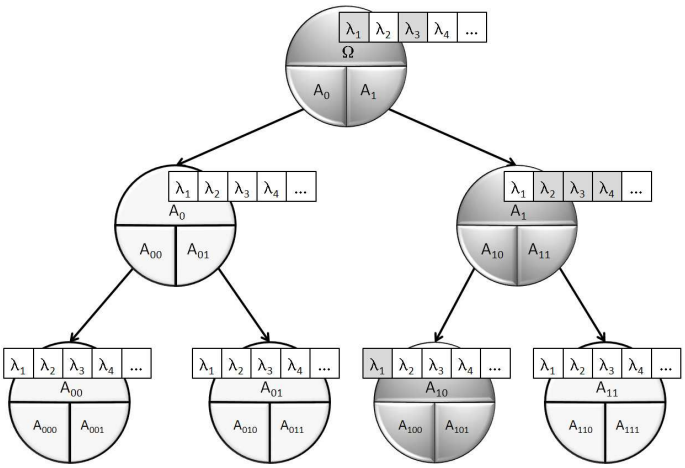

Figure 12: Haar case tree for the scalar approximation

uses d bits to encode whether an inner product is included or not. Then, for a given node n the associated indexing cost for this approach is given by

$$C_{I_n} = d + k\, H_{\mathcal{M}_Q}, \tag{55}$$

where d is, as usual, the number of inputs, $H_{\mathcal{M}_Q}$ is the average bits per scalar inner product, and k is the number of inner products being used at node n. Then the total indexing cost is

$$C_I = \sum_n C_{I_n}.$$

The indexing cost is generally low but one could construct examples where it can become significant. The total cost C_T for this case (namely the SHVGS case) is given by

$$C_T = C_{\mathcal{M}_\Pi} + C_{\mathcal{M}_S} + C_I.$$

Where $C_{\mathcal{M}_\Pi}$ is the cost associated with the partition, $C_{\mathcal{M}_S}$ is the cost associated with the tree, and C_I is the indexing cost given above. The cost associated with the quantized coefficients $C_{\mathcal{M}_Q}$ (as given by (54)), is clearly included in C_I.

6.5. $\mathcal{D}^0$ (Haar) Case: Vector Approximation.

For the purpose of reconstructing the vector approximation, we need to store, at a given active node A, B'^*, as given in (34) and $[X, \Psi_A^0]$. The partition information is also required. From (34), we see that $B'^*[i]$ is given by the normalized difference of two expected values $\mathbf{E}_{A_1}(X[i]) - \mathbf{E}_{A_0}(X[i])$. Therefore, we only need to store the result of such differences because the normalization can be done a posteriori. Now let us define

$$\Delta_i = \mathbf{E}_{A_1}(X_i) - \mathbf{E}_{A_0}(X_i),$$

then

$$B'^*[i] = \frac{\Delta_i}{\sqrt{\sum_{k=1}^{d} [\Delta_k]^2}}.$$

Quantization Map for the Vector Haar Approximation: The quantization map used for this special case is just the integer part of the difference of the expected values defined above,

$$\mathcal{M}_Q(\Delta_i) = \lfloor \Delta_i + 0.5 \rfloor . \tag{56}$$

Figure 13 shows an example of the relative frequency of the quantized differences Δ_i. The Faces set (introduced in [4]) was used with a PSNR = 40. As we have done previously, defining the

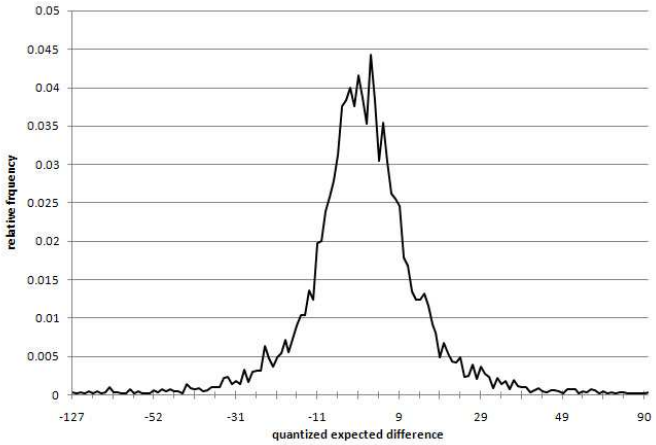

Figure 13: Relative frequency of the quantized difference of the expected values

corresponding symbol set $\mathcal{J}_Q$, we obtain the entropy cost associated with the above quantization map

$$C_{\mathcal{M}_Q} = H_{\mathcal{M}_Q} \times |Q|. \tag{57}$$

The total cost C_T for this case is calculated as

$$C_T = C_{\mathcal{M}_\Pi} + C_{\mathcal{M}_S} + C_{\mathcal{M}_Q}.$$

$C_{\mathcal{M}_\Pi}$ and $C_{\mathcal{M}_S}$ are the partition and significance bit costs respectively. Both have been completely described in previous sections. Without providing any further details we just indicate that $C_{\mathcal{M}_Q}$ is given by (57) plus the cost associated with the (quantized) coefficients $[X, \Psi^0]$ (only one per node).

6.6. $\mathcal{D}^1$ Case: Scalar Approximation.

This more general case is referred to as SMDVGS (Scalar Martingale Differences VGS) in later sections. It is the case when optimization is carried over $\mathcal{D}^1$. It is similar to the Haar scalar approximation, but in this case there could exist two types of inner products at each node, corresponding to ψ^0 and ψ^1. Figure 14 shows the tree structure for this case, the two classes of inner products are denoted by $\{\lambda_1^0, \lambda_2^0, \lambda_3^0, \lambda_4^0, \dots\}$ and $\{\lambda_1^1, \lambda_2^1, \lambda_3^1, \lambda_4^1, \dots\}$, the encoding scheme is similar to the one we have described for the Haar scalar case but in the present case we need extra information to decide whether an inner product belongs to the first set or to the second set.

For or a given node n the indexing cost is

$$C_{I_n} = 2\, d + (k_0 + k_1)\, H_{\mathcal{M}_Q},$$

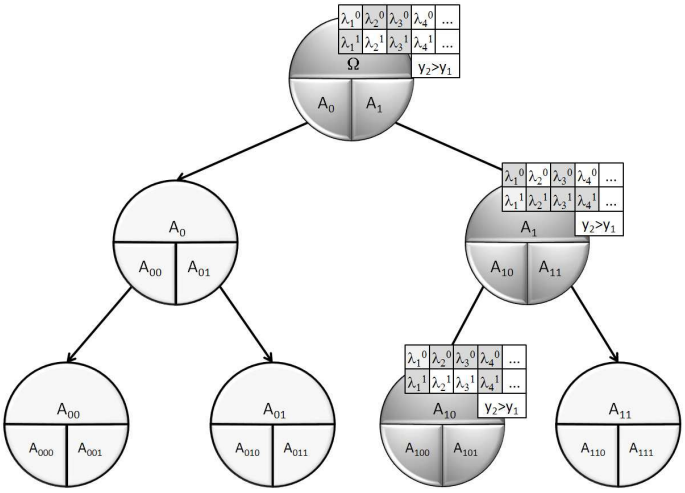

Figure 14: Tree for the scalar approximation for the case $\mathcal{D}^1$

where d is the number of inputs, k_0 and k_1 the number of inner products of each set included in the representation, $H_{\mathcal{M}_Q}$ is the average numbers of bits needed to store the inner products. The total cost C_T for this case is calculated as

$$C_T = C_{\mathcal{M}_\Pi} + C_{\mathcal{M}_S} + C_I,$$

with $C_I = \sum_n C_{I_n}$. Analogous remarks as in Section 6.4 apply.

6.7. Leaves Average Approximation.

This case is referred to as AVGS (Averages VGS) or Haar-AVGS in later sections. For this specific approximation, we assume that the VGS algorithm was applied to an input set and, after pruning, a partition is obtained. Given such a partition $\Pi(\Omega)$ we compute the integer part of the average of each input image over each atom,

$$\lambda_{ij} = \left\lfloor \frac{1}{|A_j|} \sum_{w \in A_j} X[i](w) \right\rfloor \quad \text{and} \quad A_j \in \Pi(\Omega),$$

where $\lambda_{ij} \in \mathbb{Z}$. Now, defining

$$\Lambda = \{\lambda_{ij} \ \text{ for all } \ i = 1, \ldots, d \ \text{ and } \ j = 1, \ldots, n\}$$

where $n = |\Pi(\Omega)|$ and d is the number of input images, then $|\Lambda| = n \times d$.
The approximation X_Π is given by

$$X_\Pi(w) = (\lambda_{1j}, \lambda_{2j}, \ldots, \lambda_{dj}) \quad \forall\, w \in A_j.$$

Entropy Encoding: Let us define Λ_k as the set of all values λ_{ij} equal to k

$$\Lambda_k = \{\lambda_{ij} \in \Lambda : \lambda_{ij} = k\},$$

and also define the symbol set $\mathcal{J}_\Lambda = \{\Lambda_k \subset \Lambda : \Lambda_k \neq \emptyset\}$. The theoretical cost associated to the set of averages is given by

$$C_\Lambda = H_\Lambda \times n \times d. \tag{58}$$

The total cost associated to this approximation is the cost associated to the partition, plus the cost C_Λ associated to encode the average values Λ,

$$C_{Total} = C_{\mathcal{M}_\Pi} + C_\Lambda.$$

In practice C_Λ will actually represent the cost of encoding the quantized difference of the average values as explained at the end of the present section.

Using a set of four standard images formed by {Barbara, Lena, Boats, Peppers}, of size 128×128 pixels, the VGS algorithm for the Haar case was run until the PSNR = 40, and the partition obtained contained 6983 atoms. It is possible to appreciate that there exists a slight common structure, that is shown in Figure 15. Notice that the range of the images is given by $[0, 255]$ therefore the worst case can be encoded with $\hat{H}_\Lambda = \log_2 256 = 8$ bpp and the maximum cost $\hat{C}_\Lambda = 8 \times 6983 \times 4 = 223456$ bits.

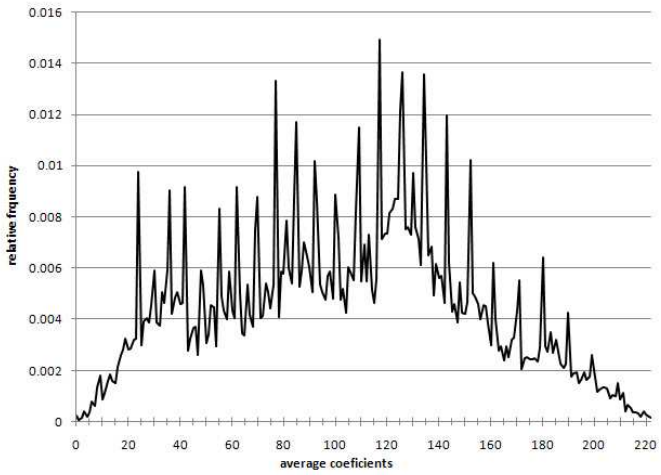

Figure 15: Relative frequency, for all input images, of averages for the Leaves Average Approximation

The values obtained for this particular case are $H_\Lambda = 7.499$ bpp and $C_\Lambda = 209479.22$ bits, approximately a compression of about 6.25%. Again from Figure 15 it is possible to observe that the values in the mid-range should be encoded with less bits than the values in the extrema.

Quantization: A quantization over the values taken by this approximation entails a quantization over the range of values taken by the image, i.e. a reduction of the gray levels of the image. Although the algorithm seem to be not so sensible to this quantization, in terms of the image degradation, the effects, specially on the smooth parts are not desirable. The goal is to find a quantization function $\mathcal{M}_Q : \mathbb{Z} \to \mathbb{Z}$ that once applied to the λ_{ij}'s the distortion remains small enough. A quantization can be done using many different approaches, but the best results were obtained using a uniform quantization algorithm described below:

Uniform Quantization: The uniform quantization function affects all values alike.

$$\mathcal{M}_Q(\lambda_{ij}) = \left\lfloor \left\lfloor \frac{\lambda_{ij}}{c} \right\rfloor \times c \right\rfloor \quad \text{and} \quad A_j \in \Pi(\Omega)$$

and $c > 0$.

There exists other types of quantization but the results of our experiments were not as good as the uniform quantization model.

Frame Correlation: It is very common in video applications to use the fact that consecutive frames are similar. Some algorithms take the difference between those frames, the computations reach a cumulative error and at that time the algorithm re-starts the encoding again. We use this property too, but with a different meaning. If two frames are similar then, the average value in a given atom should be similar too, so $\lambda_{ij} \sim \lambda_{i+1\ j}$ then we define the difference average values as

$$\alpha_{i+1\ j} = \lambda_{ij} - \lambda_{i+1\ j} \text{ and } \alpha_{0j} = \lambda_{0j}.$$

Then the α_{ij}'s are now the new input symbols for the entropy encoding block.

The results over the image set, without using quantization, are not good; the reason of this is because those images have nothing in common, considering the set as a video sequence. But using the first 9 images of the hand video set, see Figure 18, the results are quite impressive. Again we are using a good approximation of the video sequence with a PSNR= 40, the partition has 673 atoms. Using equation (58), we find that the maximum cost associated to store the average information is equal to in this case $H_\Lambda = 8$, $n = 673$ and $d = 9$ then $\hat{C}_\Lambda = 8 \times 673 \times 9 = 48456$.

In order to show the contrast we present first the results without using frame correlation. The theoretical average number of bits to store each average value is 7.76. Figure 16 shows the relative frequencies histogram. The cost associated to this is equal to $C_\Lambda = 7.76{\times}673{\times}9 = 47002$ approximately a 3% compression rate.

Now using the image correlation information the average number of bits is 4.93. Figure (17) shows the histogram and it becomes obviously that most of the values are zeros. The cost for this case is equal to $C_\Lambda = 4.93 \times 673 \times 9 = 29861$, what is close to a 39% compression rate.

6.8. Bits Tradeoffs

In this Section, when reporting bit values of the significance map we will be actually reporting the bit cost of encoding quantized values for $[X, \psi_A^0]$ and $[X, \psi_A^1]$. Bit values of the partition map will represent the cost of encoding a lossless compressed version of the corresponding data structure (we have found that the partition map is very sensitive to quantization). Therefore, in effect, in the present section the bit cost of $\mathcal{M}_S$ includes the bit cost of $\mathcal{M}_Q$.

The notation $C_{\mathcal{M}_S}[i](d)$ indicates 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. This is reasonable as it was indicated in the previous Section 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.

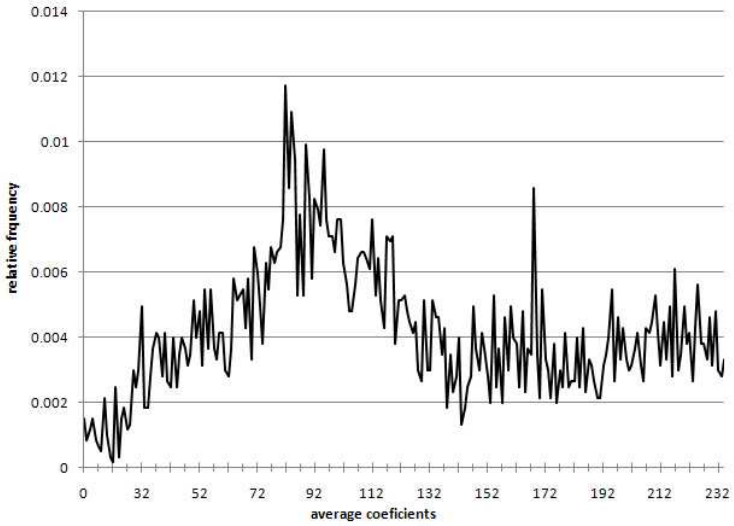

Figure 16: Histogram the average coefficients for the video sequence

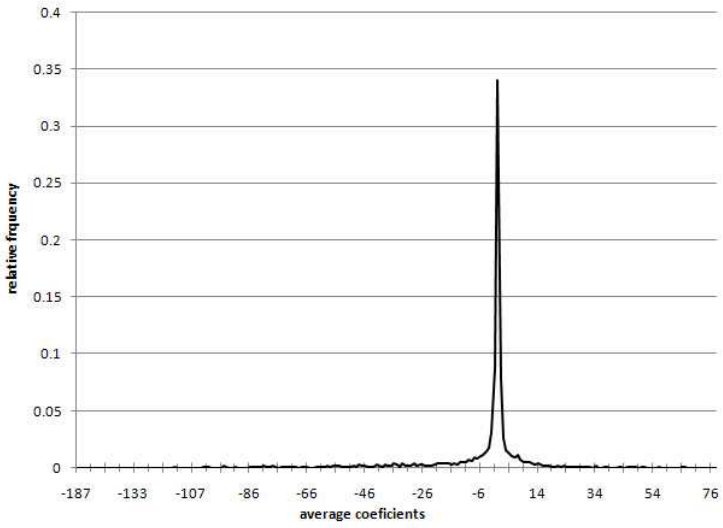

Figure 17: Histogram of the difference of the average coefficients for the video sequence

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.

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 consider that $C_{FixedBasis}[i]$ denotes 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

$$\gamma(d) \equiv \frac{C_{\mathcal{M}_\Pi}(d)}{d} + \frac{\sum_{i=1}^{d} C_{\mathcal{M}_S}[i](d)}{d}. \tag{59}$$

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

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

Using the average cost per image, instead of using the total cost, not only allow us to determine the optimal number of images for which VGS will deliver best compression performance but also provides a reasonable bit- scale to compare with other methods (Section 8).
Considering that a (slow changing) video sequence could be an optimal situation for our algorithm, we propose the following test set shown in Figure 18. This test set was down-sampled from 640×480 pixels to 128×128 using bi-cubic interpolation. We have run VGS on increas-

Figure 18: Hand Video Set: length 1 second, frame size: 128×128, color depth: 8bpp, 25 fps (frames per second)

ingly larger subsets of the video sequence by adding one image at a time to the previous subset. The results are plotted in Figure 19, the term $\frac{C_{\mathcal{M}_\Pi}(d)}{d}$ in (59) (average cost of Partition Map) is denoted PM (Partition Map) in the Figure, and the term in (59) (average total cost of the Quantization Map and Significance Map) $\frac{\sum_{i=1}^{d} C_{\mathcal{M}_S}[i](d)}{d}$ is denoted by $QM + SM$. The total cost is equal to $C_{\mathcal{M}_Q} + C_{\mathcal{M}_S} + C_{\mathcal{M}_\Pi}$. The display corresponds to the case of 45db and shows that the minimum occurs at $d = 9$ for this case.

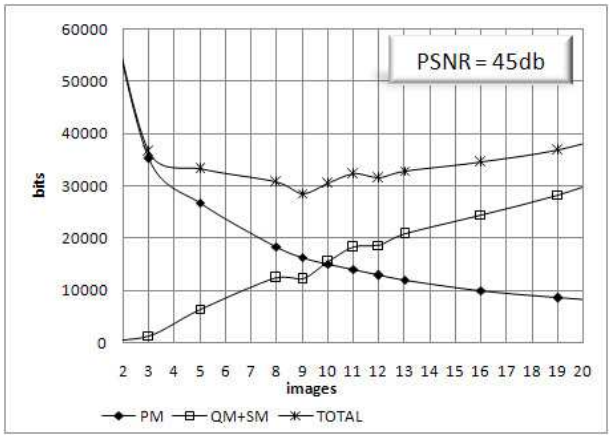

Figure 19: Average $C_{\mathcal{M}_\Pi}$ and average $C_{\mathcal{M}_Q} + C_{\mathcal{M}_S}$ plotted against d

7. Numerical Illustrations of the VGS Algorithm

We present a collection of test cases to illustrate, test and compare the VGS algorithm with other techniques. The results also indicate advantages and some shortcomings of the VGS algorithm (at least in its present form). The reader will note that only scalar approximations are discussed, i.e. no results on vector approximations are presented. This is done by reasons of space, moreover, in a limited number of experiments we have found the results to be comparable. When reporting distortion values for a sequence of images or a video sequence (we will refer to any of these two instances generically by *vector input set*), we make use of the PSNR as defined by (60) and (61).

Definition 5. *For the purposes of this section, an image is defined as a function* $I : \Omega \to \mathbb{N}$ *such that* $I(x, y) = v$ *and* $v \in \mathbb{N}$ *where*

$$\Omega = U \times V, \ \ where \ \ U = \{0, \ldots, N-1\} \ \ and \ \ V = \{0, \ldots, M-1\} \quad N, M \in \mathbb{N}.$$

Practically the VGS algorithm has no restriction on the size of the images, but for simplicity we will consider $N = M$ most of the times. Also, we will use $V = 256$-gray scale images with 8 bpp (bits per pixel). We will only consider the case when P is the uniform measure on Ω and we set $\mathcal{A} = \mathcal{P}(\Omega)$ (where $\mathcal{P}$ is the power set operation). Therefore, the L^2 error measure and the PSNR assume the following expressions in the present discrete setting:

$$\mathbf{MSE}[i] = \frac{1}{N_s} \sum_{w \in \Omega} (x[i](w) - \hat{x}[i](w))^2, \ \ \mathbf{PSNR}[i] = 10 \ \log_{10}\left(\frac{V^2}{\mathbf{MSE}[i]}\right), \tag{60}$$

where $N_s = |\Omega|$ and $\hat{X}$ denotes one of our approximations. Then, the total **MSE** is given by $\mathbf{MSE}_T \equiv \sum_{i=1}^{d} \mathbf{MSE}[i]$. Given that we deal with a vector of input images and the VGS algorithm operates on all images simultaneously, it is practical, in order to report the performance of the algorithm, to use the PSNR per image (or *average distortion* per image or *average* PSNR) defined by

$$\mathbf{PSNR}_A = \frac{1}{d} \sum_{i=1}^{d} \mathbf{MSE}[i]. \tag{61}$$

As mentioned before, in our software implementation the case when the optimization is carried over $\mathcal{D} = \mathcal{D}^0$ is referred as the Haar case. The case when $\mathcal{D} = \mathcal{D}^1$ is referred as the *martingale difference* (MD) case. In the following sub-sections we introduce a variety of inputs used as examples, they illustrate important characteristics of the algorithm.

7.1. Geometrical Set

Figure 20 displays twelve simple geometrical images used to further analyze some of the characteristics of the algorithm. Figure 21 shows the inner product curve for the *geometrical*

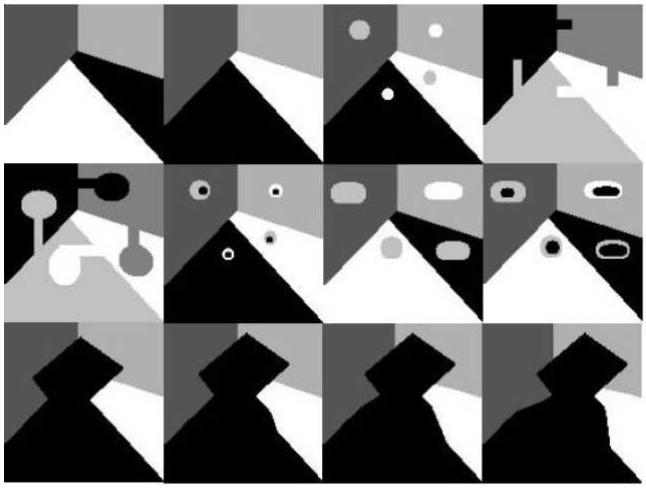

Figure 20: Geometrical Set, each of size 128×128, color depth: 8bpp

set using the Scalar Haar (i.e. $\mathcal{D} = \mathcal{D}^0$) VGS algorithm (denoted SHVGS, see Section 6.4), the inner products are sorted and plotted vs. the number of component (i.e., as we add more inner products to our list, and, hence *components* to our approximation), the comparison is done using different values of $d = 4$, 9 and 12 and the full tree was calculated. The dependency of the speed of decay of the inner products as a function of d can be observed.

Figure 22 shows a comparison between SHVGS and the Scalar Martingale Differences (so $\mathcal{D} = \mathcal{D}^1$) Vector Greedy Splitting algorithm (denoted SMDVGS, see Section 6.6), it is possible to observe that for larger number of components both algorithm have the same behavior for this case.

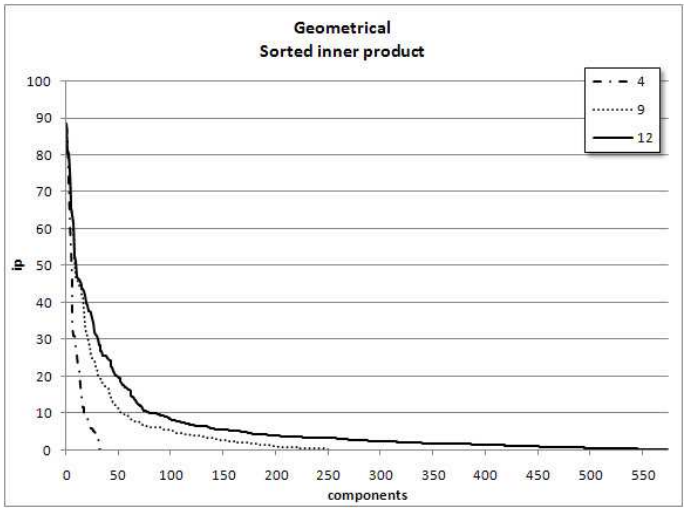

Figure 21: Inner products curve for Geometrical Set, SHVGS, $d = 4, 9$ and 12

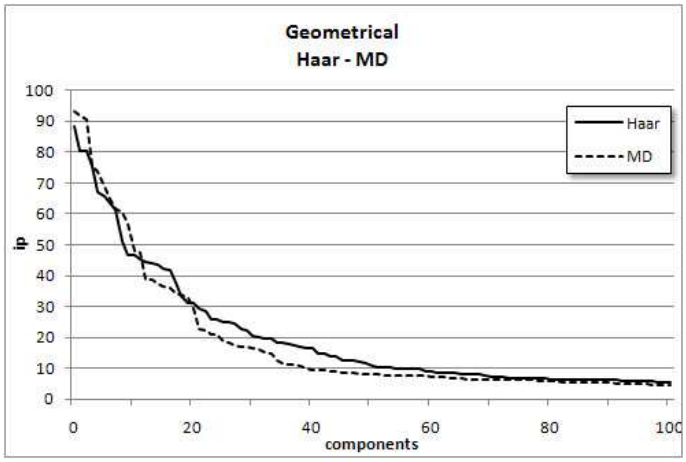

Figure 22: Inner products curve, Geometrical Set, SHVGS and SMDVGS comparison, d=9

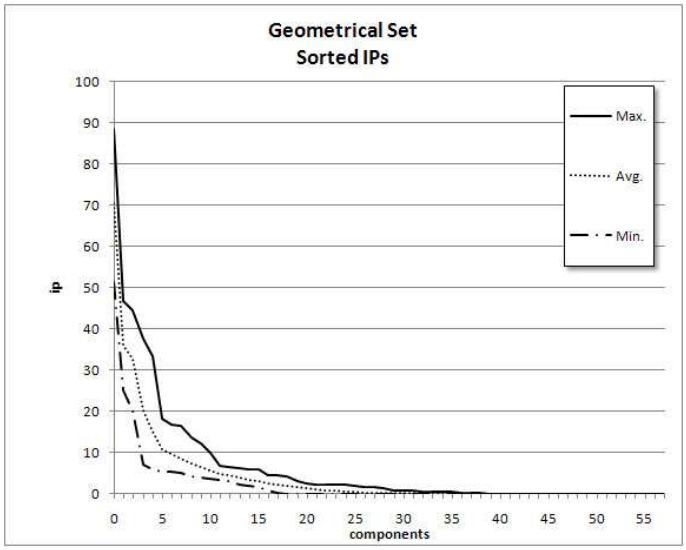

Figure 23: Inner products curve, Maximum, minimum and average per component, Geometrical Set, SHVGS, d=9

7.2. Translations Set

Figure 24 displays three input vectors, we refer to each of these input vectors as Translations Sets $1, 2, 3$. Each vector is represented by the corresponding square of $d = 9$ images (the three blocks, of nine images each, are pasted together just for easy of display). Images in each of the input vectors are constructed by translating a (smooth) single object. In each of the translation sets, the object used is larger and so the common sigma algebra to represent the nine images becomes more complex.

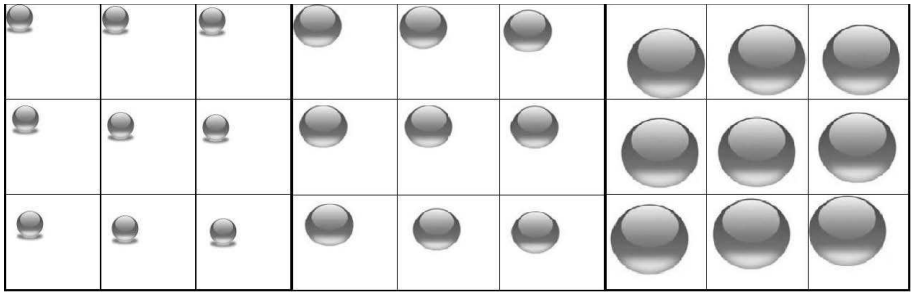

Figure 24: All Translation sets, d=9

Figure 25 shows the average distortion vs the total rate graph for the three translations sets, the degradation in the compression power of the VGS algorithm becomes evident. Figure 26 compares how the inner products decrease for each of the three translations sets.

7.3. Akiyo Video Set

The following images display results for the set Akiyo, Image 27 only shows nine representative frames from the video sequence used. Figures 28 to 30 show the average cost $C_{\mathcal{M}_\Pi}$ and average

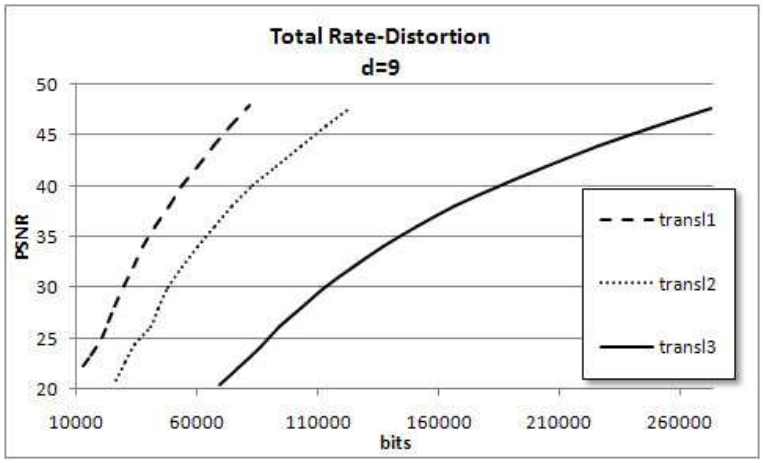

Figure 25: Average distortion vs total rate; translation sets 1,2 and 3, d=9, SHVGS

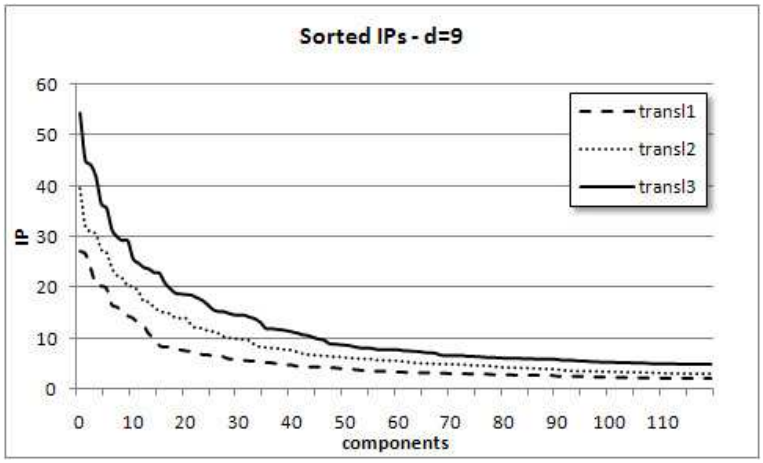

Figure 26: Sorted Inner products for translation sets 1,2 and 3, d=9, SHVGS

Figure 27: Test Set Akiyo - QCIF : 176x144 - 8bpp

$C_{\mathcal{M}_Q} + C_{\mathcal{M}_S}$ plotted vs d for different average distortion values. Figure 31 shows the average

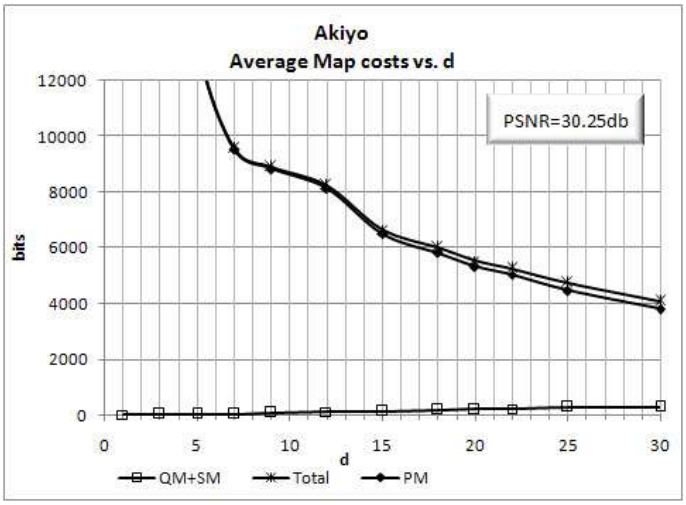

Figure 28: Average $C_{\mathcal{M}_\Pi}$ and average $C_{\mathcal{M}_Q} + C_{\mathcal{M}_S}$ vs. d, SHVGS

distortion vs the total cost for $d = 9$. More results on the Akiyo test set appear in the Section 8.

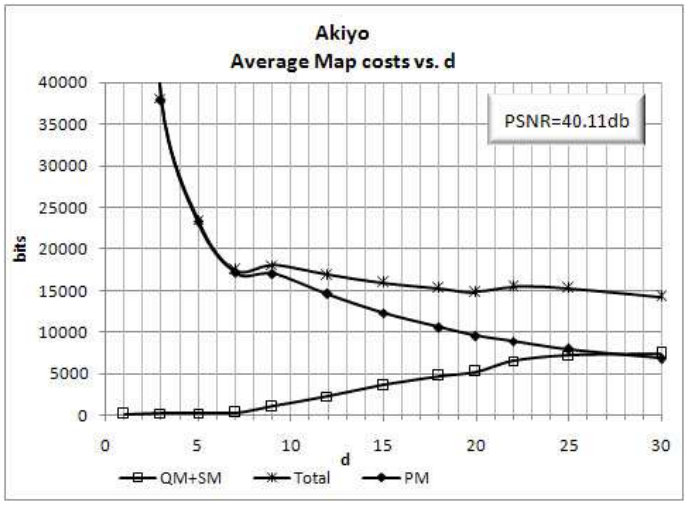

Figure 29: Average $C_{\mathcal{M}_\Pi}$ and average $C_{\mathcal{M}_Q} + C_{\mathcal{M}_S}$ vs. d, SHVGS

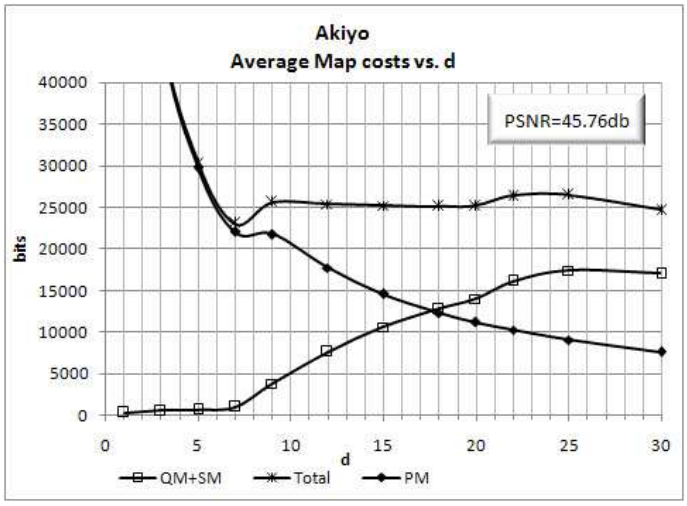

Figure 30: Average $C_{\mathcal{M}_\Pi}$ and average $C_{\mathcal{M}_Q} + C_{\mathcal{M}_S}$ vs. d, SHVGS

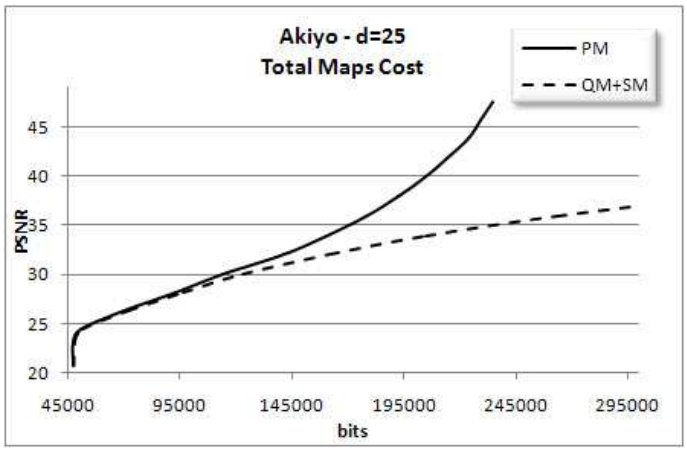

Figure 31: Average distortion vs total $C_{\mathcal{M}_\Pi}$ and total $C_{\mathcal{M}_Q} + C_{\mathcal{M}_S}$, d=9, SHVGS

8. Comparisons

8.1. JPEG2000 Static Comparison

In this section we will compare the JPEG2000 (JPEG2K) and the VGS algorithm in a static environment. When we say *static* we mean that JPEG2K does not make use of any temporal correlation among frames (this temporal correlation will be accounted for when we compare with MPEG). When running JPEG2K over a set of images, we collect them into a single larger image as JPEG2K's performance is enhanced in this way. Figure 19 reveals that VGS performs best by taking $d = 9$; also we will use the VGS-AVG algorithm (*Leaves Average Approximation Algorithm*, see Section 6.7) that performs well for video images. There are two versions of our algorithm and both outperform the JPEG2K results. The first version is the standard VGS-AVG approximation and the second one is the VGS-AVG approximation using the Lempel-Ziv algorithm to encode the partition and the quantization map (lossless compression). Figure 32 shows the bit cost vs distortion (average per image), comparing the JPEG2K with our algorithms. It is clear from the figure that the VGS-AVG using the Lempel-Ziv encoding algorithm outperforms the others for this special video sequence. Table 8.1 shows numerically the same information as in Figure 32. The bit cost for a given distortion and the last column contains the difference between the JPEG2K and the VGS-Haar-AVG LZ, it is possible to see that the VGS-Haar-AVG LZ is within $20\% - 35\%$ better than JPEG2K for this case.

Figure 33, for the Akiyo set, shows the average rate-distortion (i.e. bit rate and PSNR per image) for different values of d but all in the same graph. Figure 34 shows the average rate-distortion for $d = 16$, for different methods, SHVGS, SMDVGS and AVGS-LZIV (average leaves using Lempel-Ziv algorithm to compress the partition and quantization maps). Figure 35 shows the average rate-distortion, for the Faces set (introduced in [4]) for $d = 9$, comparing different methods.

8.2. MPEG4-3 Comparisons

The video sequences have been sampled at a slow rate of 15fps (frames per second) and this is not the standard sampling rate which is at least 30fps. This means that our video sequences have

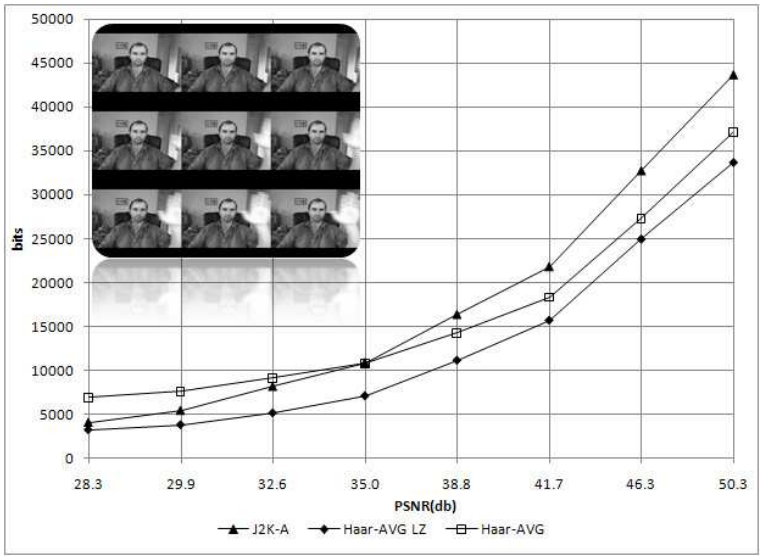

Figure 32: Bit cost vs. distortion per image for JPEG2K, VGS-HAAR and VGS-HAAR-LZ

PSNR(db)	JPEG2K	Haar-AVG	Haar-AVG LZ	DIFF
28.3	4096	6888	3209	22%
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%
50.3	43686	37131	33740	23%

Table 1: Numerical bit cost vs. distortion comparison

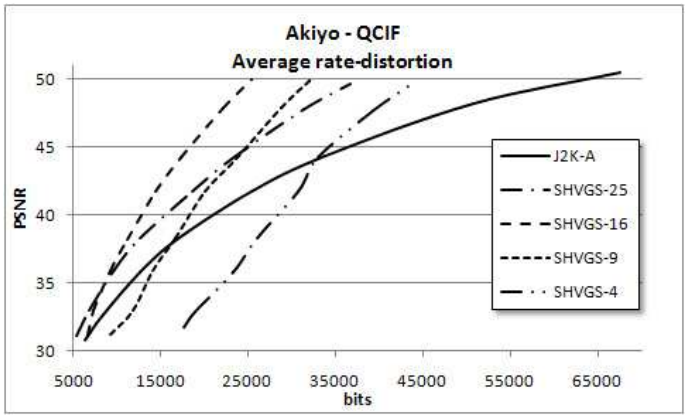

Figure 33: Average rate-distortion, d=4 to 25, SHVGS

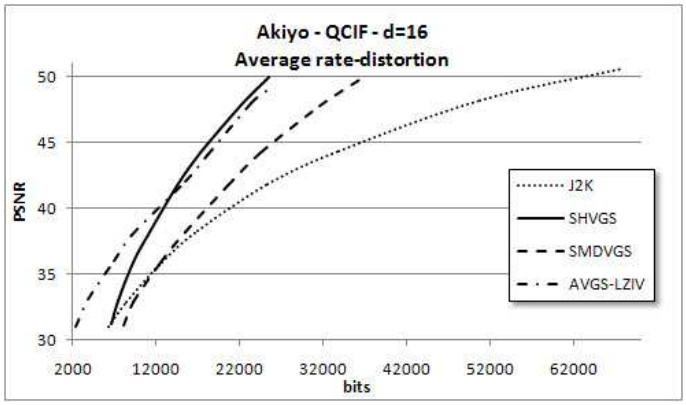

Figure 34: Comparison average rate-distortion, d=16 for different methods

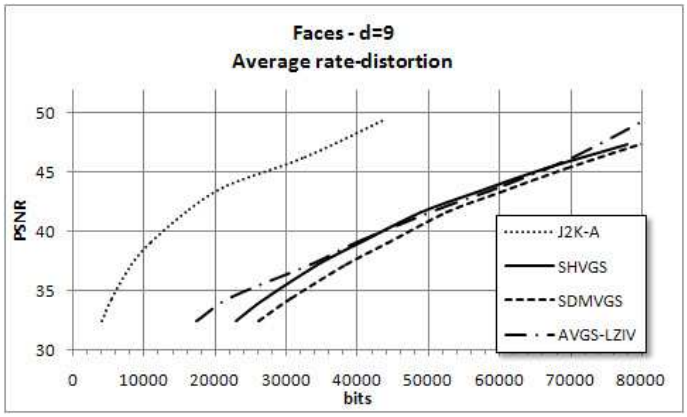

Figure 35: Average rate-distortion, d=9

more variation among the frames. Clearly this represents a disadvantageous comparison situation for the VGS algorithm. Given the availability of color only implementations of MPEG4-3, we also implemented VGS for color components for this comparison.

8.2.1. Hand Video Sequence

This sequence, which we do not display for lack of space, is the color version of the sequence displayed in Figure 18, the uncompressed video size is 450, 796 bytes. Different values of PSNR and compressed sizes are shown in Table 8.2.2 for the VGS approximation and for the MPEG4 algorithm. Notice that in the present case (hand video sequence) the VGS compares favorably in terms of the PSNR. We remark, that the image quality (once sufficiently amplified) seems to be better in the MPEG4-3 approximation. Figure 36 shows a detail comparison between the MPEG4 and the VGS approximation.

Figure 36: Video Hand approximation detail, a) Input, b) MPEG4-3 approximation PSNR=36.417, c) HAAR-AVG LZ approximation PSNR=36.79

8.2.2. More Examples of Video Sequences

Next we provide numerical results for several more examples of different video sequences. The information displayed is the same than the one displayed for the previous hand video sequence and, hence, should be self-explanatory. Displays of the video sequences can be seen in [2].
Notice that the video sequences: *Doll* and *Doll 2* are slow varying videos and VGS outperforms MPEG. The remaining example, *Princess*, is a faster paced video and, therefore, there is more variation among the frames. The performance of the VGS algorithm degrades accordingly for this example. Many more examples of the performance of the VGS algorithm on video sequences are described in [3].

9. Conclusions

We have described a simple greedy algorithm that does a fast optimization over a very large dictionary. The general nature of the dictionary makes it expensive to encode the approximating functions. The extra imposed structure, namely, the vector setting and the tree structure are used to offset the high cost of storing the approximating functions as this cost gets ameliorated (in relative terms) as d increases.

The VGS algorithm is designed to optimize the inner products appearing in the approximating expansions, given the type of approximating functions used, this results in efficient representation of basic geometrical images containing some common structure and slow changing video frames. From our experimentation, images with smooth variation make the VGS to under-perform. This

Video	HAAR-AVG LZ				MPEG4-3		Diff %	YCbCr PSNR		
	PSNR	$C_{\mathcal{M}_\Pi}$	$C_{\mathcal{M}_Q}$	Total	PSNR	Total		Y	Cr	Cb
Hand	36.79	13721	4003	17724	36.41	19422	8.74%	39	45	45
	37.36	14390	4654	19044	36.41	19422	1.95%	40	45	45
	36.48	14581	4746	19327	36.41	19422	0.49%	41	42	42
	37.9	15474	5476	20950	36.41	19422	-7.87%	41	45	45
Doll	37.46	15492	3363	18855	35.29	19245	2.03%	42	45	45
Doll2	38.97	12714	3538	16252	34.93	20302	19.95%	42	42	42
	40.79	14510	5058	19568	34.93	20302	3.62%	45	45	45
Princess	33.88	19189	23145	42334	36.67	34460	-22.85%	37	45	45

Table 2: Video compression comparison, Haar-AVG LZ vs. MPEG4-3, the costs are measured in bytes and the distortion (PSNR) in decibels (db)

is reasonable, we work on a general probability space and our approximating functions do not have any sense of smoothness (they, essentially, only obey the zero mean value constraint). Interestingly, a form of smoothness, which in our general setup appears as constraints in the higher moments of the approximating functions, emerges as a solution of a more thorough optimization (as sketched at the end of Appendix A).

We believe our work offers an explicit setup where important tradeoffs, namely the one between the expense to code the selected elements of a large dictionary and the efficiency of the transform expansion, can easily be seen and studied. Interesting questions, for further work, resulting from our study are: Can one characterize a given finite collection of functions so that the VGS approximation is optimal in terms of total bit cost? What is the role of the measure P? Notice that we have taken the uniform measure for P, presumably it could also be adapted to the input vector.

A. Bathtub Optimization and Alternative Dictionaries

In the main text we make use of the following Bathtub Principle which we borrow from [9].

Theorem 5. *For a given number $u_0 > 0$ and $x(w)$, a real valued measurable function, defined on Ω, set:*

$$\mathcal{E}_{u_0} = \{\varphi : 0 \le \varphi(w) \le 1 \text{ for all } w \text{ and } \int_\Omega \varphi(w)\, dP(w) = u_0\}.$$

Then the minimization problem $I = \inf_{\varphi \in \mathcal{E}_{u_0}} \int_\Omega x(w)\, \varphi(w)\, dP(w)$, is solved by

$$\varphi_0(w) = \mathbf{1}_{\{x < y_{u_0}\}}(w) + c_{u_0}\, \mathbf{1}_{\{x = y_{u_0}\}}(w), \text{ and} \tag{62}$$

$$I = \int_{x < y_{u_0}} x(w)\, dP(w) + c_{u_0}\, y_{u_0}\, P(x = s).$$

Where

$$y_{u_0} = \sup\{t : P(x < t) \le u_0\}, \text{ and } c_{u_0}\, P(x = y_{u_0}) = u_0 - P(x < y_{u_0}).$$

We also make use of the dual version of Theorem 5, we present this result in the following corollary.

Corollary 2. *Assume the same hypothesis as Theorem 5, then for a fixed number* $u_1 > 0$

$$I = \sup_{\varphi \in \mathcal{E}_{u_1}} \int_\Omega x(w)\, \varphi(w)\, dP(w),$$

is solved by

$$\varphi_1(w) = \mathbf{1}_{\{x > z_{u_1}\}}(w) + d_{u_1}\, \mathbf{1}_{\{x = z_{u_1}\}}(w), \tag{63}$$

and

$$I = \int_{x > z_{u_1}} x(w)\, dP(w) + d_{u_1}\, P(x = z_{u_1}).$$

Where

$$z_{u_1} = \inf\{t : P(x > t) \leq u_1\},$$

and

$$d_{u_1}\, P(x = z_{u_1}) = u_1 - P(x > z_{u_1}).$$

The general setup of the Bathtub result suggests an extension of our approach. Namely instead of optimizing over admissible functions of the form $\Psi = B_0 \mathbf{1}_{A_0} + B_1 \mathbf{1}_{A_1}$ as in (3), we could work with the following class of functions,

$$\Psi = B_0\, \varphi_0 + B_1\, \varphi_1, \text{ with } 0 \leq \varphi_i \leq 1,\ i = 0, 1,\ \varphi_i(w) = 0 \text{ for } w \notin A, \tag{64}$$

we will use the following notation

$$u_i = \mathbf{E}(\varphi_i),\ i = 0, 1,\ \ u_2 = \mathbf{E}(\varphi_0^2),\quad u_3 = \mathbf{E}(\varphi_2^2),\ \ u_4 = \mathbf{E}(\varphi_1 \varphi_2),$$

and assume $u_4 = 0$.
As usual we require

$$\int_\Omega \Psi(w)\, dP(w) = 0, \text{ and } \int_\Omega \|\Psi\|^2(w)\, dP(w) = 1.$$

Therefore,

$$[X, \Psi] = \|B_1\|\, u_1 \left(\frac{1}{u_1} \int_A \langle X(w), B_0' \rangle\, \varphi_1(w)\, dP(w) - \frac{1}{u_0} \int_A \langle X(w), B_0' \rangle\, \varphi_0(w)\, dP(w) \right), \tag{65}$$

where

$$\|B_1\| = \frac{u_0}{\sqrt{u_1^2 u_2 + u_3 u_0^2}},\ B' = \frac{B_1}{\|B_1\|} \in S^d. \tag{66}$$

A close study of the construction in our paper shows that in order to extend our work to the class given in (64) we need to solve the following optimization problems.

$$I = \mathtt{opt}_{\varphi \in \mathcal{E}_{u,v}} \int_\Omega x(w)\, \varphi(w)\, dP(w), \quad \text{where} \tag{67}$$

$$\mathcal{E}_{u,v} = \{\varphi : 0 \leq \varphi \leq 1,\ \ E(\varphi) = u,\ E(\varphi^2) = v\}\ ,$$

where opt covers the two cases of sup and inf.

Notice that, assuming $P(\Omega) = 1$, we have the following constraints

$$0 < u^2 \leq v \leq u \leq 1,$$

where we used Jensen's inequality. Transforming the inequalities into equalities, namely $u^2 = v = u$ gives, a posteriori, the setup of the present paper. Once (67) is tackled (it is a linear optimization with linear and quadratic constraints), one needs to perform optimizations over u_i, $i = 0, \ldots, 3$ under the above constraints. In general, the resulting functions φ_i will not be characteristic functions anymore but will take more than one single value and the constraints on $\mathbf{E}(\varphi_i^2)$ will result on smoother functions (as measured by their variation).

B. Scalar and Vector Valued Basis Functions

In order to establish the convergence of our algorithm we need to relate each Ψ_A^0 function (as defined in (33)) to an associated scalar function. We will assume $A \in \mathcal{A}$ is given and, in order to simplify the notation, we will dispense with the subset subscript. In particular, we will write Ψ^0 in place of Ψ_A^0 whenever possible. Moreover, at some points in this appendix, for the sake of generality, the sets A_i, that partition the given set A, will be left quite arbitrary and hence we will work with a general function Ψ^0 which becomes the optimized function constructed in (33)) when specializing the sets A_i to the sets A_i^*. Each of these instances will be indicated explicitly.

As we have done previously, we use the notation $\Psi^0 = B_0\, \mathbf{1}_{A_0} + B_1\, \mathbf{1}_{A_1} + B_2 \mathbf{1}_{A_2}$ with $A_0, A_1 \subseteq A$, $A_0 \cap A_1 = \emptyset$, $A_2 \equiv A \setminus (A_0 \cup A_1)$ and $B_2 = 0$. We call to the attention of the reader the notational difference between a given vector valued random variable Ψ and the associated scalar basis vector ψ_s. We will use the following notation for the associated scalar function (set $b_2 = 0$)

$$\psi_s = b_0 \mathbf{1}_{A_0} + b_1 \mathbf{1}_{A_1} + b_2 \mathbf{1}_{A_2} = b_0 \mathbf{1}_{A_0} + b_1 \mathbf{1}_{A_1}$$

we also require,

$$\int_\Omega \psi_s(w)\, dP(w) = 0, \text{ and } \int_\Omega \psi_s^2(w)\, dP(w) = 1. \tag{68}$$

Recall the notation $B_1 = \|B_1\|\, B'$ where $B' \in S^d$, also, an expression for $\|B_1\|$ is provided in (66). Some of the results in this Appendix require that the quantities A_i, $i = 0, 1$ and B' satisfy the following equation:

$$B'[i] = \frac{\dfrac{1}{u_1}\displaystyle\int_{A_1} X[i]\ dP - \frac{1}{u_0}\int_{A_0} X[i]\, dP}{\sqrt{\displaystyle\sum_{k=1}^{d}\left(\frac{1}{u_1}\int_{A_1} X[k]\, dP - \frac{1}{u_0}\int_{A_0} X[k]\, \varphi_0 dP\right)^2}}, \tag{69}$$

where, as usual, $u_i = P(A_i)$.

Proposition 7. *Fix an atom A and assume that (69) holds. Then*

$$[X, \Psi^0]\, \Psi^0[i](w) = [X[i], \psi_s^0]_s\ \psi_s^0(w)\ \ a.e.\ in\ \Omega. \tag{70}$$

It also follows from (70) that

$$[X, \Psi^0]^2 = \sum_{i=1}^{d} [X[i], \psi_s^0]_s^2.$$

Proof. The proof follows, essentially, by plugging the expression (69) into the left hand side of (70) and comparing with the right hand side of (70). Computing we obtain,

$$[X, \Psi^0] = \|B_1\| \, u_1 \left(\frac{1}{u_1} \int_{A_1} \langle X, B' \rangle \, dP - \frac{1}{u_0} \int_{A_0} \langle X, B' \rangle \, dP \right) =$$

$$\|B_1\| \, u_1 \left[\sum_{i=1}^{d} B'[i] \left(\frac{1}{u_1} \int_{A_1} X[i] \; dP - \frac{1}{u_0} \int_{A_0} X[i] \; dP \right) \right],$$

then using (69) in this last expression we obtain

$$[X, \Psi^0] = \|B_1\| \, u_1 \sqrt{\sum_{k=1}^{d} \left(\frac{1}{u_1} \int_{A_1} X[k] \, dP - \frac{1}{u_0} \int_{A_0} X[k] \, dP \right)^2}. \tag{71}$$

In order to re-write $[X, \Psi^0] \, \Psi^0[i]$ we perform the following manipulations

$$\Psi^{20} = B_0 \mathbf{1}_{A_0} + B_1 \mathbf{1}_{A_1} = \left(\frac{-B_1 \, u_1 \, \mathbf{1}_{A_0}}{u_0} + B_1 \mathbf{1}_{A_1} \right) = \|B_1\| \, u_1 \, B' \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right). \tag{72}$$

Therefore

$$\Psi^0[i] = \|B_1\| \, u_1 \, B'[i] \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right). \tag{73}$$

Using (71) and (73) we obtain

$$[X, \Psi^0] \, \Psi^0[i] = \|B_1\|^2 \, u_1^2 \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right) \left(\frac{1}{u_1} \int_{A_1} X[i] \, dP - \frac{1}{u_0} \int_{A_0} X[i] \, dP \right). \tag{74}$$

We concentrate now on the right hand side of (70). Let $b' = \frac{b_1}{|b_1|}$, so $b' \in \{-1, 1\}$ and write the scalar basis function as follows

$$\psi_s^0 = b_0 \, \mathbf{1}_{A_0} + b_1 \, \mathbf{1}_{A_1} = b_1 \, u_1 \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right) =$$

$$|b_1| \, u_1 \, b' \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right).$$

Notice that

$$|b_1| = \|B_1\|. \tag{75}$$

Therefore

$$\psi_s^0 = \|B_1\| \, u_1 \, b' \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right).$$

Moreover

$$[X[i], \psi_s^0]_s = b_1 \, u_1 \left(\frac{1}{u_1} \int_{A_1} X[i] \, dP - \frac{1}{u_0} \int_{A_0} X[i] \; dP \right) =$$

$$[X[i], \psi_s^0]_s = \|B_1\| \, u_1 \, b' \left(\frac{1}{u_1} \int_{A_1} X[i] \; dP - \frac{1}{u_0} \int_{A_0} X[i] \; dP \right).$$

Therefore (notice that $b'^2 = 1$),

$$[X[i], \psi_s^0]_s \, \Psi_s^0 = (\|B_1\| \, u_1 \, b')^2 \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_0}}{u_0} \right) \left(\frac{1}{u_1} \int_{A_1} X[i] \; dP - \frac{1}{u_0} \int_{A_0} X[i] \, dP \right) = \tag{76}$$

$$\|B_1\|^2 \, u_1^2 \left(\frac{\mathbf{1}_{A_1}}{u_1} - \frac{\mathbf{1}_{A_1}}{u_0} \right) \left(\frac{1}{u_1} \int_{A_1} X[i] \; dP - \frac{1}{u_0} \int_{A_0} X_i \; dP \right).$$

We just checked that (76) is equal to (74). □

Notice that

$$[X, \Psi_\emptyset^0] \, \psi_\emptyset^0[i] = \int_\Omega X[i] \, dP, \tag{77}$$

with $\Psi_\emptyset^0$ as defined in (39). We remark that for the Haar case, namely by restricting computations to the class $\mathcal{D}^0$, we can use (77), Proposition 7 and Proposition 5 to prove the following proposition.

Proposition 8. *Given $A \in \mathcal{A}$ with $P(A) > 0$, and taking Ψ_A^0 as given by (33) we have*

$$\sum_{A \in \mathcal{T}_n^\circ} [X, \Psi_A^0] \, \Psi_A^0[i](w) = \sum_{A \in \mathcal{T}_n^\circ} [X[i], \psi_{s,A}^0]_s \, \psi_{s,A}^0(w).$$

Proof. The proof follows by a simple induction along the iterations of the VGS algorithm. A key remark is that Proposition 7 is applicable, this is so as Proposition 5 shows that (69) holds for $\Psi_\emptyset^0$ (as defined in (39)). □

We will prove a similar result for the case when we add more than one function to the node, namely, Ψ_A^1. It is enough to consider only those nodes for which $J_A^1 = 2$; as the nodes with $J_A^1 = 1$ are covered by (70).

Consider now

$$\psi_s^1 = \sum_{k=0}^{2} e_k \mathbf{1}_{A_k} \tag{78}$$

to be the unique function satisfying the following three conditions

$$\|(\psi_s^1)^2\| = 1, \qquad \int \psi_s^1 dP(w) = 0, \quad [\psi_s^1, \psi_s^0]_s = 0.$$

In the present scalar case it is easy to find the solution to the above system of equations. Here is the explicit solution:

$$e_2 = \frac{-\, e_0 \, P(A_0) \, (b_1 - b_0)}{P(A_2) \, (b_1 - b_2)},$$

$$e_1 = \frac{-\, e_0 \, P(A_0) \, (b_2 - b_0)}{P(A_1) \, (b_2 - b_1)},$$

and

$$e_0^2 = \frac{P(A_1) \, P(A_2) \, (a_2 - a_1)^2}{P(A_0) \, \left[P(A_1)P(A_2)(a_2 - a_1)^2 + P(A_0)P(A_2)(a_2 - a_0)^2 + P(A_0)P(A_1)(a_1 - a_0)^2 \right]}.$$

In the above expressions $b_2 = 0$. Therefore taking $e_0 = \pm\sqrt{e_0^2}$ and using the above equations provides the desired solution (the $\pm$ sign does not affect the scalar components, i.e. the inner product times the basis element).

Define now Ψ^1 as follows

$$\Psi^1[i](w) \equiv \frac{[X[i], \psi_s^1]_s \, \psi_s^1(w)}{\sqrt{\sum_{j=1}^d [X[j], \psi_s^1]_s^2}}. \tag{79}$$

Notice that once Ψ^0 is given, Ψ^1 is uniquely defined.

Proposition 9. *For a given set $A \in \mathcal{A}$ with $P(A) > 0$, the above constructed vector valued function Ψ_A^1 is 0 outside A and it takes three distinct values on A, the pre-images of these constant values are exactly the sets A_k $k = 0, 1, 2$. Moreover,*

$$[\Psi^0, \Psi^1] = 0, \quad \|\Psi^1\| = 1, \quad \int_\Omega \Psi^1 \, dP = 0,$$

and

$$[X, \Psi^{(1)}] \, \Psi^1[i](w) = [X[i], \psi_s^1]_s \, \psi_s^1(w). \tag{80}$$

Proof. The fact that the best children of A are given as pre-images of the constant values of Ψ^1 follows directly from (78) and (79). From the definition (79) it follows that in order to prove (80) it is enough to prove the following equality

$$[X, \Psi^1]^2 = \sum_{i=1}^d [X[i], \psi_s^1]_1^2.$$

This last equation as well as the other properties follow from simple computations. □

Using (79), Proposition 7 can be extended to the general dictionary $\mathcal{D}$ by means of the following result.

Proposition 10. *Given $A \in \mathcal{A}$ with $PA) > 0$, taking Ψ_A^0 as given by (33) and Ψ_A^1 given by (79). Then the approximation defined in (41) satisfies*

$$X_{\mathcal{T}_n}[k](w) = \sum_{A \in \mathcal{T}_n^\circ} \sum_{i=0}^{J_A - 1} [X, \Psi_A^i] \quad \Psi_A^i[k](w) = \sum_{A \in \mathcal{T}_n^\circ} \sum_{i=0}^{J_A - 1} [X[k], \psi_{s,A}^i]_s \, \psi_{s,A}^i(w).$$

Next, we will define an increasing sequence of orthonormal systems $\mathcal{G}_n$, for $n \geq 0$ corresponding to the n-th. iteration of the VGS algorithm, as follows: $\mathcal{H}_0 \equiv \{u_0 \equiv \mathbf{1}_\Omega\}$ also, assume, recursively that $\mathcal{G}_n = \{u_0, \ldots, u_{k_n}\}$ has been constructed. We then let,

$$\mathcal{G}_{n+1} \equiv \mathcal{G}_n \cup_{i=0,\ldots,J_{\hat{A}}-1} \{\psi_{s,\hat{A}}^i\}, \tag{81}$$

where $\hat{A}$ is the set in (37), also set $u_{k_n+i+1} \equiv \psi_{s,\hat{A}}^i$ for $i = 0, \ldots, J_{\hat{A}} - 1$. Set also $\mathcal{G} \equiv \cup_n \mathcal{G}_n$.

The next theorem simply puts together what has been achieved by construction, namely, that $\mathcal{G}_n$ is a generalized H-system and draws the conclusions from that fact.

Theorem 6. *$\mathcal{G}_n$ is a generalized H-system (as in Definition 1) with $k_{n+1} = k_n + J_{\hat{A}}$, then Proposition 10 and (3) give:*

$$X_{\mathcal{T}_n}[k](w) = \sum_{i=0}^{k_n} [X[k], u_i]_s \, u_i = \mathbf{E}(X[k]|u_0, \ldots, u_{k_n}),$$

and so

$$X_{\mathcal{T}_n}(w) = \frac{1}{P(A)} \int_A X \, dP, \quad \textit{where } w \in A \ \in \mathcal{L}(\mathcal{T}_n).$$

Proof. Define

$$V_n \equiv \text{span } \{u \in \sigma(u_0, \ldots, u_{k_n})\}.$$

We will first check that

$$\dim(\text{span } \mathcal{G}_n) = \dim(V_n), \tag{82}$$

where $\dim(V)$ denotes the vector space dimension of a given vector space V. Notice that for $n = 0$ we are taking $k_0 = 0$ and $u_0 = \mathbf{1}_\Omega$ therefore $\dim(\text{span } \mathcal{G}_0) = 1$, clearly we also have $\dim(V_0) = 1$. We will proceed by induction, so assume (82) holds for n. Given the definition of the VGS algorithm, we may assume without loss of generality that for $\hat{A} \in \mathcal{Q}_n$ with $P(A) > 0$ ($\hat{A}$ as in (37)) and that X is not constant a.e. on $\hat{A}$, it is then easy to prove that $\dim(V_{n+1}) = \dim(V_n) + J_{\hat{A}}$. Now, from the orthogonality relations $[\Psi_A^k, \Psi_{A'}^{k'}] = 0$ whenever $A, A' \in \mathcal{T}_n$, with $P(A) > 0$ and $P(A') > 0$, and $A \neq A'$, or $k \neq k'$ for $k, k' \in \{0, 1\}$, it follows that $\mathcal{G}$ is an orthonormal system and hence $\dim(\text{span}\mathcal{G}_{n+1}) = \dim(\text{span}\mathcal{G}_n) + J_{\hat{A}}$.

For a given random variable z we have

$$P_{\mathcal{G}_n} z = \sum_{i=0}^{k_n} [z, u_i]_s \, u_i,$$

and from the definition of conditional expectations we have

$$E(z|u_0, \ldots, u_{k_n}) = P_{V_n} z,$$

where $P_{V_n} x$ denotes the projection of x onto the closed subspace V_n. Notice that $\mathcal{G}_n \subseteq V_n$, therefore (82) implies that $\mathcal{G}_n$ is an orthonormal basis for V_n, therefore

$$E(z|u_0, \ldots, u_{k_n}) = \sum_{i=0}^{k_n} [z, u_i]_s \, u_i.$$

□

The construction of the above Ψ^1 was guided by the goal to implement a conditional expectation martingale as in (6). If one intends to explore a vector valued martingale sequence that is not given as a conditional expectation one can perform a more aggressive optimization to obtain a different Ψ^1. The cost of storing such a Ψ^1 will increase correspondingly though. Next we outline here how this optimization can be done.

We look for a vector valued function of the form,

$$\Psi^1 = E_0 \mathbf{1}_{A_0} + E_1 \mathbf{1}_{A_1} + E_2 \mathbf{1}_{A_2}, \tag{83}$$

where the $E_i \in \mathbb{R}^d$. We will impose the following constraints :

$$1 = \|\Psi^1\|^2 = [\Psi^1, \Psi^1] = \langle E_0, E_0\rangle P(A_0) + \langle E_1, E_1\rangle P(A_1) + \langle E_2, E_2\rangle P(A_2), \tag{84}$$

$$0 = \int_A \Psi^1 \, dP(w) = E_0 \, P(A_0) + E_1 \, P(A_1) + E_2 \, P(A_2), \tag{85}$$

and (using the fact that $B_2 = 0$)

$$0 = [\Psi^0, \Psi^1] = \langle \, E_0, B_0\rangle \, P(A_0) + \langle \, E_1, B_1\rangle \, P(A_1). \tag{86}$$

Notice that we now have $3 \times d$ unknowns, namely the values $E_k[i]$, $k = 0, 1, 2$ and $i = 1, \ldots, d$ and only $d + 2$ equations. Also notice that the scalar case described in (79) is a special solution of the equations satisfying (84), (85) and (86). Clearly, the optimal Ψ^1 should maximize $[X, \Psi^1]$. Actually, this maximization can be done explicitly by using Lagrange multipliers. We will not present here this lengthy derivation.

Acknowledgements: We would like to acknowledge the use we have made of the software and output images created by the student Ariel Bernal during his MSc thesis ([2]) and as a research assistant, both works were performed under our supervision.

References

[1] D. Alani, A. Averbuch and S. Dekel, *"Image coding with geometric wavelets"*, IEEE Transactions on Image Processing, Vol.16, No.1, (2007) 69-77.

[2] A. J. Bernal, *Simultaneous Approximation of Images. Applications to Image and Video Compression"*. MSc Thesis, Ryerson University, January 2008, 153 pages.

[3] A.J. Bernal and S.E. Ferrando, *Adaptive orthonormal bases for video compression*. The 2008 International Workshop on Local and Non-Local Approximation in Image Processing, LNLA2008. Lausanne, Switzerland, August 2008. Available at: `http://ticsp.cs.tut.fi/reports/report-44.pdf`.

[4] P. J. Catuogno, S.E. Ferrando and A.L. Gonzalez, *"Adaptive Martingale Approximations"*. *Journal of Fourier Analyisis and Applications.* Volume 14, Issue 5, (2008) 712-745.

[5] A. Cohen, W. Dahmen, I. Daoubechies and R. De Vore, *"Tree approximation and optimal encoding"*, Applied and Computational Harmonical Analysis, Vol. 11, (2001) 192-226.

[6] A. Cohen, R.A. DeVore and R. Hochmuth *"Restricted nonlinear approximation"*, Constructicve Approx., 16, (2000) 85-13.

[7] S.E.Ferrando, E.J. Doolittle, A.J.Bernal, L.J.Bernal, *"Probabilistic matching pursuit with gabor dictionaries"*, Signal Processing, Vol. 80, Issue 10, (2000) 2099-2120.

[8] Richard F. Gundy , *"Martingale theory and pointwise convergence of certain orthogonal systems"*. *Trans. Amer. Math. Soc.*, **124**, (1966) 228–248.

[9] Elliott H. Lieb and Michael Loss. *Analysis. Second Edition*. Graduate Studies in Mathematics, Volumen 14, American mathematical Society, 2001.

[10] S. Mallat and Z. Zhang, *"Matching pursuits with time-frequency dictionaries"*. IEEE Transactions of Signal Processing, Vol. 41, (1993) 3397-3415.

[11] S. Mallat. *"A Wavelet Tour of Signal Processing"*. Academic Press, second edition 1999.

[12] J. Neveu. *"Discrete-Parameter Martingales"*. North Holland 1975.

[13] H. Radha, M. Vetterli and R. Leonardi *"Image compression using binary space partitioning trees"*, IEEE Transactions on Image Processing, Vol. 5, No. 12, (1996) 1610-1624.

[14] A. Said and W. Pearlman, *"An image multiresolution representation for lossless and lossy compression"*, IEEE Transaction on Image Processing, Vol. 5, No.9, (1996) 1303-1310.

[15] J.M. Shapiro,*"Embedded image coding using zerotrees of wavelet coefficients"*, IEEE Transactions on Signal Processing, Vol.41 - No.12, (1993) 3445-3462.