Aston University

Some pages of this thesis may have been removed for copyright restrictions.

If you have discovered material in Aston Research Explorer which is unlawful e.g. breaches copyright, (either yours or that of a third party) or any other law, including but not limited to those relating to patent, trademark, confidentiality, data protection, obscenity, defamation, libel, then please read our Takedown policy and contact the service immediately (openaccess@aston.ac.uk)

SPARSE IMAGE APPROXIMATION

SHABNAM BIBI

Master of Science by Research

ASTON UNIVERSITY

December, 2012

©Shabnam Bibi, 2012

Shabnam Bibi asserts her moral right to be
identified as the author of this thesis

This copy of the thesis has been supplied on condition that anyone who consults it is understood to recognise that its copyright belongs to its author and that no quotation from the thesis and no information derived from it may be published without appropriate permission or acknowledgement.

Abstract

This thesis addresses the problem associated with the approximation of signals as linear superposition of elementary components often called 'atoms'. After highlighting the limitations of using only orthogonal elements, the approximation technique is extended to consider the selection of atoms from a large redundant set, called a 'dictionary'. In particular, a highly correlated 'mixed dictionary' is considered, from which the atoms are selected through highly non-linear techniques known as Matching Pursuit Strategies. These techniques evolve by stepwise selection of dictionary atoms. In particular, a relatively new strategy named Block wise Orthogonal Matching Pursuit is considered. This technique operates on images divided into blocks and extends the stepwise selection of dictionary atoms to also select the blocks to be approximated at each iteration step. The implementation of block selection introduces extra storage requirements, which has motivated the *Macro Processing Scheme* proposed in this thesis. The project also focuses on the effectiveness of the approach with regards to processing time. In this respect, a C++ implementation of the Block wise Orthogonal Matching Pursuit technique has been developed to operate in MATLAB environment. Using the developed tools a number of comparative tests, with respect to sparse image representation, have been performed and analysed.

Keywords: *Matching pursuit, block, compression, non-linear, wavelet, macro*

Dedication

This thesis is dedicated to the memory of my late niece Ishaal Ali, and to my loving parents, Ishfaq Hussain and Qalsoom Begum.

Acknowledgements

I am greatly appreciative of all the staff at Aston University who provided the necessary resources to carry out this project. I would like to give a special thanks to James Bowley for his knowledge on MEX files.

In particular my gratitude goes out to my supervisor Dr Laura Rebollo-Neira who is responsible for the successful completion of the project. Laura's untiring effort, commitment, encouragement, guidance and support helped me greatly in the understanding and writing of the project.

To my ever supportive parents, I would like to thank you for the financial, moral, and spiritual support you have provided over the years. To my brothers and sisters, I would like to say thank you for your constant belief in me, without you, mum and dad, nothing would be achievable. To my nephews, you brought a lot of chaos into the world but I still love you all.

Contents

Items Included

Item 1: CD-ROM provides files for testing the BWOMP2D implementations. For use with a software named MATLAB.

List of Abbreviations

DCT Discrete Cosine Transform
DWT Discrete Wavelet Transform
BWMP2D Block Wise Matching Pursuit in 2D
BWOMP2D Block Wise Orthogonal Matching Pursuit in 2D
MP2D Matching Pursuit in 2D
MWT Macro Wavelet Transform
OMP2D Orthogonal Matching Pursuit in 2D

List of Figures

List of Tables

List of Contributions

The following list summarizes the contributions of the research project:

- Macro processing scheme for processing large images with Block Wise Orthogonal Matching Pursuit in 2D (BWOMP2D).
- Translation to C++ of an available MATLAB BWOMP2D function, to be executable on a MATLAB platform through what is called a C++ MEX file.
- Successful application of the BWOMP2D approach in the wavelet domain for avoiding blocking artefacts when processing images which are partitioned into small blocks.
- Successful application of the BWOMP2D approach in the wavelet domain for perceptually lossless approximations.
- Experiments leading to conclusion on the effectiveness of the BWOMP2D approach when used in the wavelet domain with a particular mixed dictionary.

1 Introduction

An approximation is an inexact representation of something that is still close enough to be useful.

Approximation is applied to numbers or functions, which is related to data representation in the form of signals. A signal is any quantity varying with one or more parameters. It is often represented by a function and approximated by a linear combination of other known approximating functions. Approximating functions are either orthogonal or non-orthogonal, and depending on what they are, effects the way a signal is approximated. The project focuses on highly non-linear approximations which need to be tackled by highly non-linear techniques. Considerations are restricted to Matching Pursuit (MP) Strategies. In particular an effective implementation of a relatively new strategy: block wise Orthogonal Matching Pursuit strategy in 2D (BWOMP2D) is developed. The BWOMP2D operates on a 2D image divided into small blocks. It selects the block to be approximated at each iteration step and uses the method Orthogonal Matching Pursuit in 2D (OMP2D) to approximate the blocks. Approximations of images by the BWOMP2D strategy are restricted to medium size images, due to the storage demands. As we discuss here, while the technique can be successfully applied up to some image size, there is the challenge of its application on much larger images. Motivated by the need of overcoming this limitation we propose a scheme to process large images, that we call the macro processing scheme. We further address the issue of an effective implementation of the BWOMP2D approach by developing a C++ implementation, to be applied in a MATLAB platform by what is called a C++ MEX file. The developed tools are used in numerical experiments which are conceived for well defined comparative purposes.

The thesis is organised as follows: Section 2 is dedicated to revise some preliminary mathematics needed for the project. Section 3 focuses on linear and non-linear approximations based on orthonormal functions. The next section, Section 4 addresses the matters arising by extending non-linear approximations to consider non-orthogonal functions, which gives rise to highly non-linear approximations. In this section the greedy techniques considered by this project, namely the block wise extension of the Orthogonal Matching Pursuit, BWOMP, are introduced. Section 5 is dedicated to comparing approaches for image approximation. Section 6 moves on to producing perceptually lossless approximations with the BWOMP2D approach. Section 7 further examines the effectiveness of the BWOMP2D strategy on different image resolutions. Section 8 demonstrates how to enhance performance of the BWOMP2D approach, and the macro processing scheme is introduced in Section 9. The scheme is finally tested in Section 10, and Section 11 draws final conclusions.

2 Preliminary Mathematics

We revise here the basic mathematical concepts involved in the thesis.

2.1 Vector Space

A vector space over a field $\mathcal{F}$ is a set V together with two operations vector addition, denoted $v + w \in V$ for $v, w \in V$ and scalar multiplication, denoted $av \in V$ for $a \in \mathcal{F}$ and $v \in V$, such that the following axioms are satisfied:

1. $v + w = w + v,\ v, w \in V$
2. $u + (v + w) = (u + v) + w,\ u, v, w \in V$
3. There exists an element $0 \in V$, called the zero vector, such that $v + 0 = v$, $v \in V$
4. There exists an element $\widetilde{v}$, called the additive inverse of v such that $v + \widetilde{v} = 0,\ v \in V$
5. $ab(v) = (ab)v,\ a, b \in \mathcal{F}$ and $v \in V$
6. $a(v + w) = av + aw,\ a \in \mathcal{F}$ and $v, w \in V$
7. $(a + b)v = av + bv,\ a, b \in \mathcal{F}$ and $v \in V$
8. $1v = v,\ v \in V$, where 1 denotes the multiplicative identity in $\mathcal{F}$

The elements of a vector space are called vectors. [11].

2.2 Inner product

An inner product on a vector space V is a map from V to $\mathcal{F}$ which satisfies the following axioms. Let u, v and $w \in V$ and $a \in \mathcal{F}$.

- $< u + v, w >=< u, w > + < v, w >$
- $< av, w >= a < v, w >$
- $< v, w >=< w, v >$ or for a complex vector space $< v, w >= \overline{< w, v >}$
- $< v, v > \geqslant 0$ and equal to zero if and only if $v = 0$

A vector space together with an inner product on it, is called a inner product space. [11].

2.3 Inner Product Spaces

2.3.1 Euclidean space

Euclidean n-space, sometimes called Cartesian space or simply n-space, is the space of all n-tuples of real numbers, $\mathbf{x} = (x_1, x_2, ..., x_n)$. It is commonly denoted $\mathbb{R}$. [11]. For the Euclidean space $\mathbb{R}^n$ the inner product is given by the dot product

$$< \mathbf{x}, \mathbf{y} >= x_1 y_1 + x_2 y_2 + ... + x_n y_n$$

2.3.2 $L^2(a,b)$ space

The $L^2(a,b)$ space is the space of functions $f(x)$ such that

$$\|f(x)\| = \sqrt{\int_a^b |f(x)|^2} < \infty \tag{1}$$

Furthermore the inner product for two complex functions f and g in $L^2(a,b)$ is defined as

$$< f(x), g(x) >= \int_b^a f(x)\overline{g(x)}dx. \tag{2}$$

Since $f(x) \in L^2(a,b)$ and $g(x) \in L^2(a,b)$ we have,

$$\|f(x)\|^2 = \int_a^b |f(x)|^2 \, dx < \infty \tag{3}$$

and

$$\|g(x)\|^2 = \int_a^b |g(x)|^2 \, dx < \infty, \tag{4}$$

and it can be proved that

$$-\infty << f(x), g(x) >= \int_b^a f(x)\overline{g(x)}dx < \infty. \tag{5}$$

Proof

Since

$$< f(x), g(x) >= \int_b^a f(x)\overline{g(x)}dx$$

we have

$$|< f(x), g(x) >| = \left| \int_b^a f(x)\overline{g(x)}dx \right|$$

From the Cauchy-Schwarz inequality (see Appendix A for the proof),

$$|< f(x), g(x) >|^2 \leqslant \|f(x)\|^2 \|g(x)\|^2$$

and by taking the square root,

$$|< f(x), g(x) >| \leqslant \|f(x)\| \|g(x)\|$$

Since $f(x)$ and $g(x)$ are in $L^2(a,b)$, $\|f(x)\| < \infty$ and $\|g(x)\| < \infty$, therefore

$$|< f(x), g(x) >| \leqslant \|f(x)\| \|g(x)\| < \infty$$

which implies

$$|< f(x), g(x) >| = \int_b^a f(x)\overline{g(x)}dx < \infty$$

QED

2.4 Basis and Orthogonal Basis in finite Dimension

A basis of a finite dimension vector space $\mathbb{R}^n$ is defined as a set $\{v_i\}_{i=1}^n$ of vectors in $\mathbb{R}^n$ that are linearly independent and span $\mathbb{R}^n$, i.e $\mathbb{R}^n$ = span $\{v_i\}_{i=1}^n$. Consequently, every $v \in \mathbb{R}^n$ can be uniquely written as,

$$v = c_1 v_1 + c_2 v_2 + ... + c_n v_n$$

$$v = \sum_{i=1}^{n} c_i v_i$$

where $c_1, c_2..., c_n$ are in general complex numbers.

Two vectors v_1, v_2 are orthogonal if $< v_1, v_2 >= 0$, and a basis is orthogonal if all vectors belonging to the basis are pairwise orthogonal.

A basis is orthonormal if it is orthogonal and all vectors belonging to the basis have unit length. So,

$$< v_i, v_j >= \delta_{ij} \text{ where } \delta_{ij} = \begin{cases} 0 \text{ if } i \neq j \\ 1 \text{ if } i = j \end{cases} \tag{6}$$

For a finite dimension subspace of $L^2(a, b)$ an orthonormal basis is a set $\{\psi_i(x)\}_{i=1}^N$ spanning the subspace such that $< \psi_i(x), \psi_j(x) >= \delta_{ij}$, with $< \psi_i(x), \psi_j(x) >= \int_b^a \psi_i(x)\overline{\psi_j(x)}dx$. In order to span the whole space the basis needs to be infinite.

2.5 Wavelet Transform

2.5.1 Continuous Case

Many signals and images exhibit piecewise smooth behaviour punctuated by transients. For example, speech signals are characterised by short bursts encoding consonants followed by steady-state oscillations indicative of vowels, and natural images have edges. Unlike the Fourier basis, wavelet bases are adept at sparsely representing piecewise regular signals and images, which include transient behaviour. The advantage of wavelets is that they are capable of revealing aspects of data that other signal analysis techniques miss, aspects like trends, breakdown points, discontinuities in higher derivatives, and self-similarity. They also have the capability to compress a signal without noticeable degradation.

A wavelet function $\phi \in L^2(\mathbb{R})$, (often referred to as the mother wavelet) can be described as a function effectively limited in time domain, i.e. ϕ has values in a certain range and zero elsewhere. One property of the mother wavelet is that it has zero mean and has a non-zero norm.

Given

$$\int_{-\infty}^{\infty} \phi(t)\ dt = 0$$

and

$$\|\phi(t)\|^2 = \int_{-\infty}^{\infty} \phi(t)\phi^*(t)\ dt = 1$$

the mother wavelet can form an orthonormal basis set denoted by

$$\left\{\phi_{s,u}(t) = \frac{1}{\sqrt{s}}\phi\left(\frac{t-u}{s}\right)\right\}\Big|_{s\in\mathbb{R},u\in\mathbb{R}^+}$$

where s is the scaling parameter greater than zero, and u is the translating parameter used to locate the region of interest. Parameter s is inversely proportional to the frequency, so if we want to focus on low frequencies larger s is used, while for higher frequencies small s is used.

The wavelet transform (WT) of a one-dimensional signal $f(t) \in L^2(\mathbb{R})$ is given by

$$\begin{aligned} Wf(s,u) &= < f(t), \phi_{s,u}(t) > \\ &= \int_{-\infty}^{\infty} f(t)\phi^*_{s,u}(t)\ dt \\ &= \int_{-\infty}^{\infty} f(t)\frac{1}{\sqrt{s}}\phi^*\left(\frac{t-u}{s}\right)\ dt \end{aligned}$$

and the inverse wavelet transform (IWT) is given by

$$f(t) = \frac{1}{C_\phi}\int_0^{\infty}\int_{-\infty}^{\infty} Wf(s,u)\frac{1}{\sqrt{s}}\phi^*\left(\frac{t-u}{s}\right)\ du\frac{ds}{s^2}$$

where C_ϕ is defined as

$$C_\phi = \int_0^{\infty}\frac{|\varphi(\omega)|^2}{\omega}\ d\omega < \infty$$

and $\varphi(\omega)$ is the Fourier transform of the mother wavelet ϕ.

Wavelet transformation can be applied to image approximation in the following way. Let X be the input image and Y the approximated image. One possibility of an image process is OMP2D.

$$X \xrightarrow{\text{WT}} \widetilde{X} \xrightarrow{\text{Process}} \widetilde{Y} \xrightarrow{\text{IWT}} Y$$

2.5.2 Discrete Case

Definition

The Discrete Wavelet Transform (DWT) can be described as a series of filtering and down sampling (decimating in time). One type of filter is a low pass filter. A low pass filter is an electronic filter that passes low-frequency signals but rejects (reduces the amplitude of) signals with frequencies higher than the cutoff frequency. The actual amount of reject for each frequency varies from filter to filter. A high pass filter is the opposite of a low pass filter and a band pass filter is a combination of a low pass and a high pass.

Calculating Coefficients

To obtain the DWT the coefficients are calculated by applying a high pass wavelet filter to the signal and down sampling the result by a factor of 2. At the same level, a low pass scale filter is also performed followed by down sampling to produce the signal for the next level. The resolution of the signal, which is a measure of the amount of detail information in the signal, is determined by the filtering operations, and the scale is determined by sub sampling operations.

At each decomposition level, the half band filters produce signals spanning only half the frequency band. This doubles the frequency resolution as the uncertainty in frequency is reduced by half. Thus, while the half band low pass filtering remove half of the frequencies and thus halves the resolution, the decimation by 2 doubles the scale. With this approach, the time resolution becomes arbitrarily good at high frequencies, while the frequency resolution becomes arbitrarily good at low frequencies. The filtering and decimation process is continued until the desired level is reached. The maximum number of decomposition levels depends on the length of the signal.

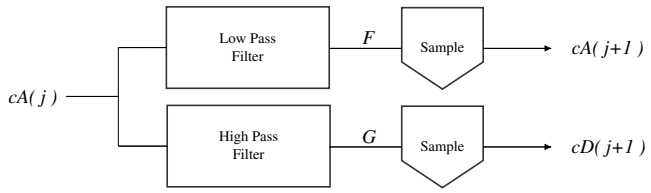
1D DWT and Inverse DWT

Figure 1(a) depicts the multi-level decomposition process for the 1D case of DWT. Starting form $A(j)$, this level provides two sets of coefficients: approximation coefficients $cA(j+1)$, and detail coefficients $cD(j+1)$. These vectors are obtained by convolving $A(j)$ with the low pass filter for approximation, and with the high pass filter for detail, followed by decimation. We down sample by keeping the even indexed elements. The first level of a signal would start with $A(0)$, where $A(0)$ is the signal. In Figure 1(a), the length of each filter is equal to $2N$, and if $n = length(cA(j))$, then the signals F and G are of length $n+2N-1$, and the coefficients $cA(1)$ and $cD(1)$ are of length $floor\left(\frac{n-1}{2}\right)+N$. The next level decompose approximation coefficients $cA(1)$ in two components using the same scheme, (replacing $cA(0)$ by $cA(1)$ in Figure 1(a), and producing $cA(2)$ and $cD(2)$, and so on. For the 1D case of DWT the original signal is then obtained by concatenating all the coefficients starting from the last level of decomposition. The wavelet decomposition of the signal s analysed at level j has the structure $[cA(j), cD(j), \ldots, cD(1)]$.

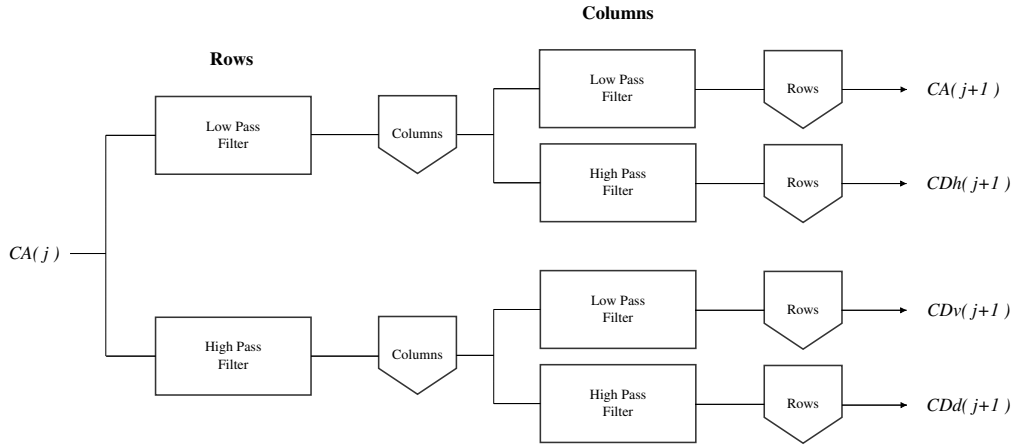
The Inverse Discrete Wavelet Transform (IDWT) is the reverse process of 1D decomposition. The approximation and detail coefficients at every level are up sampled by a scale factor of 2, and passed through the low pass and high pass filters and then added. This process is continued through the same number of levels as in the decomposition process to obtain the original signal. For 1D Inverse DWT we up sample by adding zero at the odd indexed elements.

2D DWT and Inverse DWT

For multi level 2D Discrete Wavelet Transform the decomposition process in depicted in Figure 1(b). When X is represented by a m by n matrix, the output arrays cA, cDh, cDv, cDd are also m by n matrices. Matrices cDh, cDv and



(a) 1D DWT. Decomposition provides two components: the approximation at level $j + 1$, and the detail.

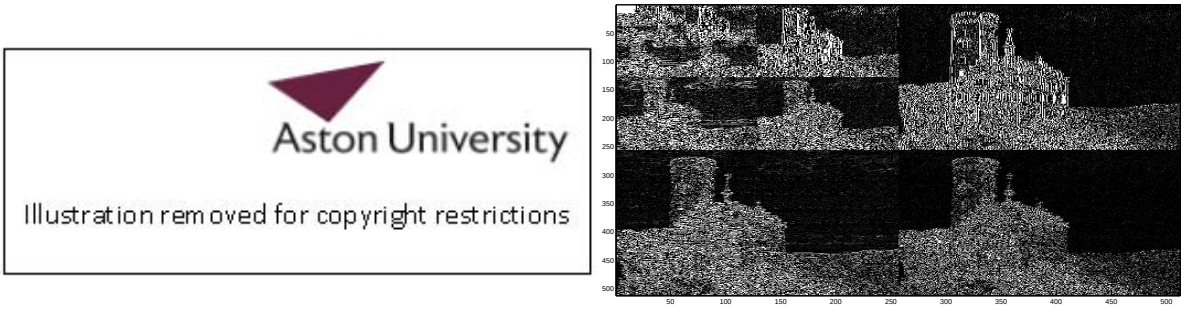


(b) 2D DWT. Decomposition provides four components: the approximation at level $j + 1$, and the details in three orientations.

Figure 1: Multi-level Discrete Wavelet Transform. Decomposition of approximation coefficients at level j.

cDd are the detail coefficients in three orientations, horizontal, vertical, and diagonal respectively. For 2D DWT we down sample the columns by keeping the even indexed columns, and then down sample the rows by keeping the even indexed rows.

For 2D IDWT is the reverse process of 2D decomposition. We would up sample the rows by adding zeros at the odd indexed elements, and up sample the columns by adding zeros at the odd indexed elements. Figure 2(b) illustrates the results of DWT applied to the Lichtenstein Castle image shown in Figure 2(a), with 5 levels of decomposition.



(a) Lichtenstein Castle (b) Wavelet Transform

Figure 2: Discrete Wavelet Transform of level 5 applied to the image shown in Figure 2(a).

3 Approximation based on Orthonormal Functions

There are two types of approximations with orthonormal functions, one is the linear approximation and the other is the non-linear approximation.

3.1 Linear Approximation

The linear approximation with orthonormal functions considers only the first M elements for approximation. Therefore a signal $f(x)$ is approximated by $f^A(x)$, where

$$f^A(x) = \sum_{i=1}^{M} c_i \psi_i(x),$$

and the coefficients can be calculated as

$$c_i =< f(x), \psi_i(x) >= \int_a^b f(x)\overline{\psi_i(x)}dx.$$

3.2 Non-linear Approximation

For the non-linear approximation with orthonormal functions only the M largest absolute value coefficients are considered.

The coefficients are calculated as $c_i =< f(x), \psi_i(x) >$, $i = 1, \ldots, N$ and the M largest absolute values are found by ordering the quantities $|c_i|$, $i = 1, \ldots, N$. The largest absolute value coefficients are taken because this minimises the norm error

between $f(x)$ and the approximated function $f^A(x)$, i.e minimises $\|f(x) - f^A(x)\|$. In this case the non-linear approximation with orthonormal functions is

$$f^A(x) = \sum_{i \in I} c_i \psi_i(x),$$

where I is a set containing the indexes corresponding to the M largest absolute value coefficients.

3.3 Difference between Linear and Non-linear Approximation

The difference between the linear and non-linear approximation is that the linear approximation satisfies the linear properties, i.e it satisfies

$$L(c_1 f(x) + c_2 g(x)) = c_1 L(f(x)) + c_2 L(g(x))$$

where L is the linear approximation operation.

Proof

Consider the approximation of f and g where f and $g \in L^2(a, b)$

$$f^A(x) = \sum_{i=1}^{M} a_i \psi_i(x) \text{ with } a_i = \int_a^b f(x)\overline{\psi_i(x)}dx$$

and

$$g^A(x) = \sum_{i=1}^{M} b_i \psi_i(x) \text{ with } b_i = \int_a^b f(x)\overline{\psi_i(x)}dx$$

Let $h(x) = c_1 f(x) + c_2 g(x)$ and

$$h^A(x) = \sum_{i=1}^{M} d_i \psi_i(x) \text{ with } d_i = \int_a^b h(x)\overline{\psi_i(x)}dx$$

Hence

$$\begin{aligned} d_i = & \int_a^b (c_1 f(x) + c_2 g(x))\, \overline{\psi_i(x)}dx \\ = & \; c_1 \int_a^b f(x)\overline{\psi_i(x)}dx + c_2 \int_a^b g(x)\overline{\psi_i(x)}dx \\ = & \; c_1 a_i + c_2 b_i \end{aligned}$$

and

$$\begin{aligned} h^A(x) = & \sum_{i=1}^{M} d_i \psi_i(x) \\ = & \sum_{i=1}^{M} (c_1 a_i + c_2 b_i)\, \psi_i(x) \\ = & \sum_{i=1}^{M} c_1 a_i \psi_i(x) + \sum_{i=1}^{M} c_2 b_i \psi_i(x) \\ = & \; c_1 f^A(x) + c_2 g^A(x). \end{aligned}$$

This means function $h(x)$ can be approximated by an approximating function $h^A(x)$ - which is the sum of other linear approximating functions $f^A(x)$ and $g^A(x)$.

This allows approximations to be calculated separately. For the non-linear approximation this relation does not hold. In general the norm error of the non-linear approximation is less than or equal to the norm error of the linear approximation.

3.4 Calculating the Error of Linear Approximation

Definitions

Point-wise Error

$$E(x) = \left|f(x) - f^A(x)\right|^2 \tag{7}$$

Error in norm over the range $[a, b]$

$$\|E(x)\|^2 = \int_a^b \left|f(x) - f^A(x)\right|^2 dx \tag{8}$$

Proposition

The error in (8) can be written $\left\|f(x) - f^A(x)\right\|^2 = \sum_{i=M+1}^{\infty} |c_i|^2$

Proof

We have $f(x) = \sum_{i=1}^{\infty} c_i\psi_i(x)$ where $c_i =< f(x), \psi_i(x) >$ and $f^A(x) = \sum_{i=1}^{M} c_i\psi_i(x)$ where $c_i =< f^A(x), \psi_i(x) >$ and so

$$f(x) - f^A(x) = \sum_{i=M+1}^{\infty} c_i\psi_i(x)$$

Let $\Delta = f(x) - f^A(x) = \sum_{i=M+1}^{\infty} c_i\psi_i(x)$ then

$$\begin{aligned}\left\|f(x) - f^A(x)\right\|^2 &= < f(x) - f^A(x), f(x) - f^A(x) > \\ &= < f(x) - f^A(x), \Delta > \\ &= < f(x), \Delta > - < f^A(x), \Delta >\end{aligned}$$

where $< f^A(x), \Delta >= 0$ since

$$\begin{aligned}< f^A(x), \Delta >= & < \textstyle\sum_{i=1}^{M} c_i\psi_i(x), f(x) - f^A(x) > \\ = & < \textstyle\sum_{i=1}^{M} c_i\psi_i(x), \sum_{j=M+1}^{\infty} c_j\psi_j(x) > \\ = & \textstyle\sum_{i=1}^{M} c_i < \psi_i(x), \sum_{j=M+1}^{\infty} c_j\psi_j(x) > \\ = & \textstyle\sum_{i=1}^{M} c_i \sum_{j=M+1}^{\infty} \overline{c_j} < \psi_i(x), \psi_j(x) > \\ = & \textstyle\sum_{i=1}^{M} \sum_{j=M+1}^{\infty} c_i\overline{c_j}\delta_{ij}\end{aligned}$$

and because i never equals j, $\delta_{ij} = 0$, hence $< f^A(x), \Delta >= 0$

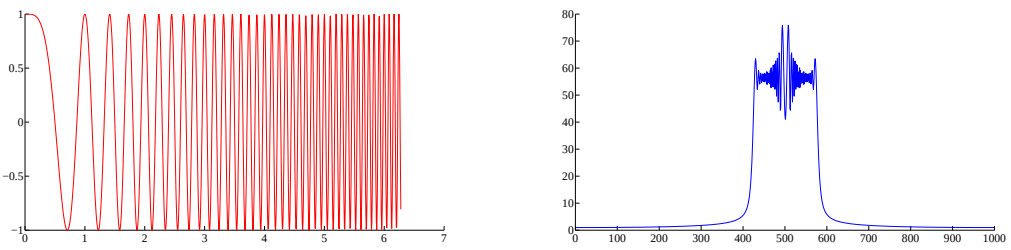
So,

$$\begin{aligned}
\left\|f(x) - f^A(x)\right\|^2 &= < f(x), \Delta > - < f^A(x), \Delta > \\
&= < f(x), \Delta > -0 \\
&= < f(x), \textstyle\sum_{i=M+1}^{\infty} c_i \psi_i(x) > \\
&= \textstyle\sum_{i=M+1}^{\infty} \overline{c_i} < f(x), \psi_i(x) > \\
&= \textstyle\sum_{i=M+1}^{\infty} \overline{c_i} c_i \\
&= \textstyle\sum_{i=M+1}^{\infty} |c_i|^2
\end{aligned}$$

This concludes by taking the M largest absolute value coefficients it minimises the norm error between $f(x)$ and the approximated function, as the norm error is calculated by taking the sum of the remaining absolute value coefficients squared.

3.5 Numerical Example

We will now look at two examples of signal approximation using linear and non-linear approximations, and assess how well the signals are approximated by looking at the norm errors. Firstly we will approximate a chirp signal $f(x)$ in the range [0,7] and calculate the norm error for both types of approximation. Figure 3(a) shows the chirp signal to be approximated using Fourier functions, $\psi_n(x) = e^{\frac{2\pi n x i}{7}}$. For this signal when M is small, e.g. $M = 90$, the first M coefficients (shown in the middle of Figure 3(b)), are not the M largest absolute value coefficients selected for non-linear approximation, and so the relative error norms are not equal. Table 1 shows that as you get to a certain number of coefficients, e.g. M=130, the error norm becomes equal for both types of approximations. This is because the absolute values of the coefficients of the Fourier Transform shown in Figure 3(b) monotonically decay as M increases, which means that the first M coefficients for the linear approximation are in actual fact the M largest absolute value coefficients used for the non-linear approximation, hence the corresponding error norms are equal. If we approximate another signal this is not likely to happen. Table 1 indicates using 200 coefficients the linear and non-linear approximation may produce an acceptable approximation of the signal. Indeed, the norm of the error is about 6% of the signal norm, and the signal is given by 1000 points. This implies that the representation by coefficients reduces the dimension of the data 5 times for a error norm of 6%.



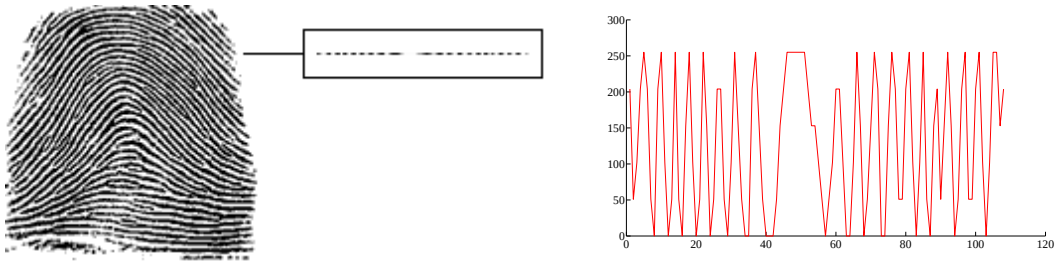
(a) Chirp signal in [0,7]

(b) Absolute value coefficients

Figure 3: A chirp signal in [0,7], and the absolute value coefficients of the Fourier Transform.

Number of Coefficients	Linear	Non-linear
90	0.5227	0.4640
100	0.4408	0.3845
110	0.3426	0.2947
120	0.1931	0.1921
130	0.1170	0.1170
140	0.0911	0.0911
180	0.0646	0.0646
200	0.0591	0.0591

Table 1: The relative error norm to the chirp signal shown in Figure 3(a), for linear and non-linear approximation verse the number of coefficients.



(a) Finger print

(b) The signal in [0,120] corresponding to the single line.

Figure 4: A finger print and a signal corresponding to the single line of the finger print.

Number of Coefficients	Linear	Non-linear
48	0.3735	0.1580
58	0.1438	0.1154
68	0.1028	0.0890
78	0.0809	0.0685
88	0.0568	0.0476
98	0.0468	0.0236

Table 2: This table shows the relative error norm corresponding to the finger print signal shown in Figure 4(b) for linear and non-linear approximation.

The other signal we approximate using Fourier functions is a single line of a human finger print. The finger print is shown in Figure 4(a), and the single line signal to be approximated is shown in Figure 4(b). It is evident from Table 2 that as you increase the number of coefficients the error norm of the linear and non-linear approximations decrease. However, even though the non-linear approximation is better than the linear, since it produces smaller error norm, it still does not provide a useful approximation for the finger print line. Certainly, considering that the signal is given by only 108 sample points a quality approximation would require too many coefficients, thereby, making the approximation not useful. From this example one can conclude that *there are signals that can not be usefully approximated by linear or non-linear approximation* using Fourier functions.

4 Highly Non-linear Approximations

We have discussed the approximation of a function $f(x)$ using orthogonal functions by the expansion $f^A(x) = \sum_{i=1}^{n} c_i \psi_i(x)$ where $< \psi_i(x), \psi_j(x) >= \delta_{ij}$. In this case, $f^A(x)$ is the orthogonal projection of $f(x)$ in the subspace $V_n = \text{span}\,\{\psi_i(x)\}_{i=1}^{n}$ and therefore is the unique function in V_n minimising the distance to $f(x)$, see Appendix B. Moreover, if the approximation were to be improved by considering more terms in the expansion, the new approximation is obtained by adding more terms, but the coefficients in $f^A(x)$ are NOT modified.

When one wants to approximate $f(x)$ with non-orthogonal functions $\{\Phi_i(x)\}_i^n$, the problem is more complicated: The approximation still takes the form

$$f^A(x) = \sum_{i=1}^{n} c_i \Phi_i(x)$$

however, since $< \Phi_i(x), \Phi_j(x) >\neq \delta_{ij}$, the coefficient c_i cannot be calculated as in the orthogonal case. Furthermore, each time an approximation is to be improved by adding more terms in the expansion, all the coefficients have to be recalculated to guarantee that the new approximation minimises the distance to the given signal. The other problem that this sort of approximation has to consider is the one of choosing the functions involved in the approximation. For these reasons an approximation by non-orthogonal functions is said to be a highly non-linear approximation.

4.1 Matching Pursuit Strategies

As already discussed, by using non-linear approximations with orthogonal functions one may not obtain an economical approximation for some signals, i.e., to produce a good approximation we may have to use almost as many coefficients as sampling points giving the signal. With the limitation of orthogonal functions it is useful to release the orthogonal property and approximate signals as a linear superposition of elementary functions often called atoms, where the atoms are selected from a large redundant set, called a dictionary. This thesis emphasizes the fact that, when appropriate dictionaries are used, highly non-linear techniques, such as Matching Pursuit Strategies, may produce far more economical approximations. One can expect this outcome because those techniques give more freedom to choose the functions for the signal expansion. Consequently less coefficients should be required to produce a good level of approximation. We restrict consideration to two greedy strategies: the first is called Orthogonal Matching Pursuit (OMP) [6] and the second is Bock wise Orthogonal Matching Pursuit (BWOMP) [10].

4.2 Selection Strategies

The OMP algorithm is a greedy selection technique for approximating signals by recursive selection of elementary components taken from a dictionary. Especially when applied using separable dictionaries which are a mixture of a redundant cosine component and a component consisting of atoms of small support, OMP in

2D (called OMP2D) has been shown to be very effective for approximating natural images [1, 2], in particular astronomical images [3]. Since the approximations considered in those papers aim to yield high approximations of high PSNR, the possibility of blocking artefacts is not considered. However, for some classes of images containing large sections with uniform background, or some classes of X-ray images, blocking artefacts may become an issue. It is the aim of this project to illustrate the fact that by processing images in the wavelet domain blocking is not visually detected as such. Moreover, using appropriate dictionaries, the levels of achieved sparsity is very high.

Sparse image approximation in the wavelet domain has been successfully applied for image coding by schemes that apply the greedy Matching Pursuit (MP) strategy using localised atoms, which are chosen by stepwise selection involving the whole transformed image of regions corresponding to different wavelet bands [4, 5]. Since we restrict considerations to high quality approximations, (yielding high PSNR), the greedy technique OMP2D is expected to perform considerably better than MP. Hence we discuss here an improvement of OMP2D which also selects atoms in an stepwise manner involving large regions of an image (if not the whole image itself) but maintaining the blocking processing structure to keep the technique effective in terms of processing time.

4.2.1 OMP2D selection

Given an image $\mathbf{I} \in \mathbb{R}^{L_x \times L_y}$ and two 1D dictionaries $\mathcal{D}^x = \{\mathbf{d}_n^x \in \mathbb{R}^{L_x}\}_{n=1}^{M_x}$ and $\mathcal{D}^y = \{\mathbf{d}_m^y \in \mathbb{R}^{L_y}\}_{m=1}^{M_y}$ the greedy procedure OMP2D for approximating $\mathbf{I}$ by the superposition

$$I^K(i,j) = \sum_{n=1}^{K} c_n^K d_{\ell_n^x}^x(i) d_{\ell_n^y}^y(j) \ , \quad i = 1, \ldots, L_x, \quad j = 1, \ldots, L_y \tag{9}$$

with atoms $d_{\ell_n^x}^x$ and $d_{\ell_n^y}^y$, $n = 1, \ldots, K$, selected from $\mathcal{D}^x$ and $\mathcal{D}^y$ respectively, evolves as follows.

On setting $\mathbf{R}^0 = \mathbf{I}$ at iteration $k+1$ the algorithm selects the atoms $\mathbf{d}_{\ell_{k+1}^x}^x \in \mathcal{D}^x$ and $\mathbf{d}_{\ell_{k+1}^y}^y \in \mathcal{D}^y$ that maximise the absolute value of the Frobenius inner products $\langle \mathbf{d}_n^x, \mathbf{R}^k \mathbf{d}_m^y \rangle_{\mathrm{F}}$, $n = 1, \ldots, M_x$, $m = 1, \ldots, M_y$, i.e.,

$$\ell_{k+1}^x, \ell_{k+1}^y = \underset{\substack{n=1,\ldots,M_x \\ m=1,\ldots,M_y}}{\arg\max} \Big| \sum_{\substack{i=1 \\ j=1}}^{L_x, L_y} d_n^x(i) R^k(i,j) d_m^y(j) \Big|$$

with

$$R^k(i,j) = I(i,j) - \sum_{n=1}^{k} c^k(n) d_{\ell_n^x}^x(i) d_{\ell_n^y}^y(j). \tag{10}$$

The coefficients $c^k(n)$, $n = 1, \ldots, k$ in the above expansion are such that $\|\mathbf{R}^k\|_{\mathrm{F}}$ is minimum, where $\|\cdot\|_{\mathrm{F}}$ is the Frobenius norm. This is ensured by requesting that $\mathbf{R}^k = \mathbf{I} - \hat{P}_{\mathbb{V}_k}\mathbf{I}$, where $\hat{P}_{\mathbb{V}_k}$ is the orthogonal projection operator onto $\mathbb{V}_k = \mathrm{span}\{\mathbf{d}_{\ell_n^x}^x \otimes \mathbf{d}_{\ell_n^y}^y\}_{n=1}^k$. A straightforward generalisation of the implementation

discussed in [6, 7] for the 1D case provides us with the representation of $\hat{P}_{\mathbb{V}_k}\mathbf{I}$ as given by,

$$\hat{P}_{\mathbb{V}_k}\mathbf{I} = \sum_{n=1}^{k} \mathbf{A}_n \langle \mathbf{B}_n^k, \mathbf{I} \rangle_F = \sum_{n=1}^{k} c^k(n)\mathbf{A}_n, \tag{11}$$

where each $\mathbf{A}_n \in \mathbb{R}^{L_x \times L_y}$ is an array with the selected atoms $\mathbf{A}_n = \mathbf{d}^x_{\ell^x_n} \otimes \mathbf{d}^y_{\ell^y_n}$ and $\mathbf{B}_n^k$, $n = 1, \ldots, k$ are the concomitant reciprocal matrices, which are the unique elements of $\mathbb{R}^{L_x \times L_y}$ satisfying the conditions:

i) $\langle \mathbf{A}_n, \mathbf{B}_m^k \rangle_{\mathrm{F}} = \delta_{n,m} = \begin{cases} 1 & \text{if } n = m \\ 0 & \text{if } n \neq m. \end{cases}$

ii) $\mathbb{V}_k = \mathrm{span}\{\mathbf{B}_n^k\}_{n=1}^k$.

Such matrices can be adaptively constructed through the recursion formula:

$$\mathbf{B}_n^{k+1} = \mathbf{B}_n^k - \mathbf{B}_{k+1}^{k+1} \langle \mathbf{A}_{k+1}, \mathbf{B}_n^k \rangle_{\mathrm{F}}, \quad n = 1, \ldots, k$$

where

$$\mathbf{B}_{k+1}^{k+1} = \mathbf{C}_{k+1} / \|\mathbf{C}_{k+1}\|_{\mathrm{F}}^2, \text{ with } \mathbf{C}_1 = \mathbf{A}_1 \text{ and } \mathbf{C}_{k+1} = \mathbf{A}_{k+1} - \sum_{n=1}^{k} \frac{\mathbf{C}_n}{\|\mathbf{C}_n\|_{\mathrm{F}}^2} \langle \mathbf{C}_n, \mathbf{A}_{k+1} \rangle_{\mathrm{F}}. \tag{12}$$

For numerical accuracy in the construction of the set $\mathbf{C}_n$, $n = 1, \ldots, k+1$ at least one re-orthogonalisation step is usually needed. It implies that one needs to recalculate these matrices as

$$\mathbf{C}_{k+1} = \mathbf{C}_{k+1} - \sum_{n=1}^{k} \frac{\mathbf{C}_n}{\|\mathbf{C}_n\|_{\mathrm{F}}^2} \langle \mathbf{C}_n, \mathbf{C}_{k+1} \rangle_{\mathrm{F}}. \tag{13}$$

The coefficients in (11) are obtained from the inner products $c^k(n) = \langle \mathbf{B}_n^k, \mathbf{I} \rangle_{\mathrm{F}}$, $n = 1, \ldots, k$. The algorithm iterates up to step, say K, for which, for a given ρ, the stopping criterion $\|\mathbf{I} - \mathbf{I}^K\|_{\mathrm{F}} < \rho$ is met. The MATLAB function OMP2D, and the corresponding MEX file in C++ for faster implementation of the identical function, are available at [8].

4.2.2 Block wise OMP2D selection

For effective processing of images with the OMP2D approach it has to be applied on small blocks of size, say, $L_x \times L_y$ pixels, partitioning the image. From here on a block of size 8×8 pixels will be defined as block of size 8, i.e., block of size 8 will consist of $8 \times 8 = 64$ pixels, and a block of size 16 will consist of $16 \times 16 = 256$ pixels. One possibility for this is to process the blocks totally independently and then assemble them together to produce the approximated image. Alternatively, we consider here the possibility for stepwise selection not only of the atoms but also the blocks for which the atoms are chosen. In other words, if an image $\mathbf{I}$ is assumed to the composition of Q identical blocks, i.e.,

$$\mathbf{I} = \cup_{q=1}^{Q} \mathbf{I}_q,$$

where each block $\mathbf{I}_q$ is approximated as

$$I^{k_q}(i,j) = \sum_{n=1}^{k_q} c^{k_q}(n) d^x_{\ell^{x,q}_n}(i) d^y_{\ell^{y,q}_n}(j), \quad i,j = 1,\dots L_q. \tag{14}$$

The atoms $d^x_{\ell^{x,q}_n}$ and $d^y_{\ell^{y,q}_n}$ are recursively chosen to improve, at step $k+1$, the approximation of the block, q, such that

$$\begin{aligned} \ell^{x,q}_{k+1}, \ell^{y,q}_{k+1} = & \underset{\substack{n=1,\dots,M_x \\ m=1,\dots,M_y \\ h=1,\dots,Q}}{\arg\max} \; | \sum_{\substack{i=1 \\ j=1}}^{L_x,L_y} d^x_n(i) R^{k_h}(i,j) d^y_m(j)| \\ & \text{with} \\ R^{k_h}(i,j) = I(i,j) - & \sum_{n=1}^{k_h} c^{k_h}(n) d^x_{\ell^{x,h}_n}(i) d^y_{\ell^{y,h}_n}(j). \end{aligned} \tag{15}$$

This selection forces the storage of matrices (12), for each block partitioning the image. While the complexity order of this variation of the technique does not change with respect to the block independent version of OMP2D, the storage demands significantly increase. The corresponding increment is in direct relation with the partition cardinality. Hence, for large images the process needs to be dedicated to deal with images of that nature. As already mentioned, a central aim of this project is to investigate the possibility of applying, in an effective manner, the block wise selection strategy, that we term BWOMP2D, in conjunction with dictionaries that were specially designed to work within the wavelet domain.

4.3 Mixed dictionary for sparse image representation

The success in the economy of a representation depends on the dictionary being suitable for the signal in hand. We introduce at this point the dictionaries that will be used in the forthcoming sections.

Let dictionary D_a be defined as

$$D_a = \{\mathbf{v}_i;\, v_{j,i} = p_i \cos(\frac{\pi(2j-1)(i-1)}{2M}),\, j = 1,\dots,N\}_{i=1}^{M},$$

with p_i, $i = 1,\dots,M$ normalisation factors and the notation $v_{j,i}$ indicating the component j of vector $\mathbf{v}_i \in \mathbb{R}^N$. If $M = N$ this set is a Discrete Cosine (DC) orthonormal basis for $\mathbb{R}^N$. If $M = 2lN$, with l a positive integer, the set is a DC dictionary with redundancy $2l$ which we will call Redundant Discrete Cosine (RDC) dictionary.

For constructing a mixed dictionary for image approximation, in addition to the dictionary D_a we further consider the set D_b, which is a discrete Euclidean Basis (EB) i.e.,

$$D_b = \{\mathbf{e}_i \in \mathbb{R}^N;\, e_{j,i} = \delta_{i,j},\, j = 1,\dots,N\}_{i=1}^{N},$$

where $\delta_{i,j} = 1$ if $i = j$ and zero otherwise.

The third set D_c is obtained by the translation of prototype atoms (PrA). Altogether there are 7 different types. Atom type 1 and type 3 are triangles ('Hats') of base 3 and base 5, and type 2 and type 4 are the complements to unity of type 1 and type 2 respectively. Types 5, 6 and 7 are Haar wavelets of 2 points, 4 points and 6 points.

It is the joint dictionary $D_{abc} = D_a \cup D_b \cup D_c$ (RDC $\cup$ EB $\cup$ PrA) that will be used with the selection strategies in the wavelet domain.

To achieve highly sparse image approximations it is important to use the 'best' dictionary. For general processing we only consider a mixed dictionary formed by redundant Discrete Cosine components, discrete Euclidean Basis and components obtained by the translation of prototype atoms, so using 'better' dictionaries one would only enhance on the results obtained in this thesis. Figure 5 illustrates one of the many atoms available from 8 different types of atoms, this includes the Euclidean Basis.

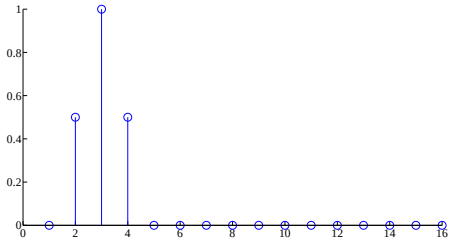
4.4 Quality of a technique for signal approximation

The suitability of a technique for signal approximation can be quantified by a few factors one of which is the speed. A fast technique is, of course, always preferable. However, as already discussed a fast orthogonal technique may produce poorer results than a non-orthogonal one, in relation to the number of coefficients needed for representing the signal as a linear combination. When the number of coefficients is much less than the number of sample points needed to represent the signal, the decomposition is said to be *sparse*. The sparsity quality is the central property that we aim to achieve in this project. Highly non-linear techniques are specially suitable for this purpose. The price to be paid is in terms of computation time. However, as we will demonstrate in subsequent sections, by segmenting the signal the required time becomes very affordable and one can process large dimensional signals such as 2D images using a laptop. We shall enhance the importance of non-linear techniques for achieving a sparse representation of an image by comparing the results with those yielded by two popular transforms: the Discrete Cosine Transform (DCT) [12] and the Discrete Wavelet Transform [9, 12].

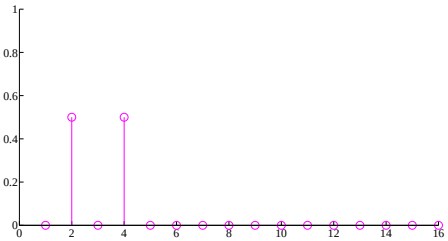
4.5 Quality of an approximate image

The quality of an approximate signal can be measured by calculating the Peak Signal to Noise Ratio (PSNR). The PSNR is generally used to analyse the quality of an image, sound and video files in decibels (dB). In particular the PSNR calculation of two images of equal size, one original and an altered (approximate) image describes how far two images are equal. The formula for the calculation is given by

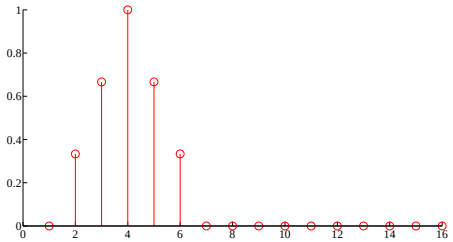
$$\text{PSNR(dB)} = 10 \log \left(\frac{I_{\max}}{2} \mu^2 \right)$$



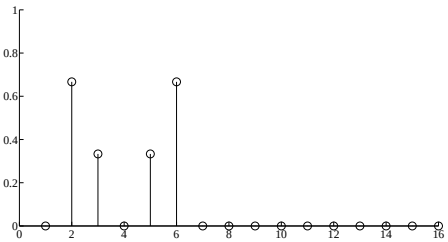
(a) Type 1: Hat base 3: Atom 4



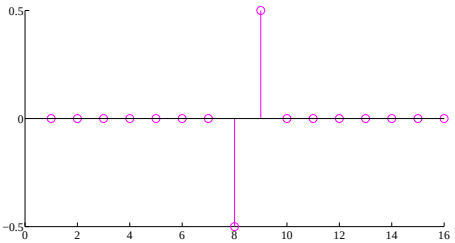
(b) Type 2: Complement to unity of type 1: Atom 4



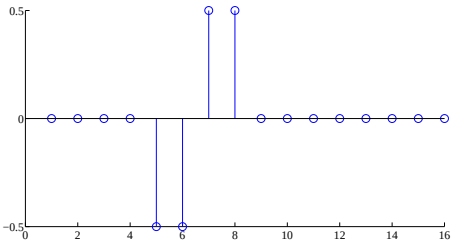
(c) Type 3: Hat base 5: Atom 6



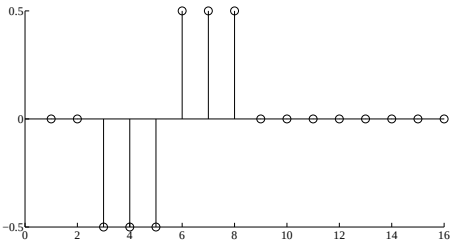
(d) Type 4: Complement to unity of type 3: Atom 6



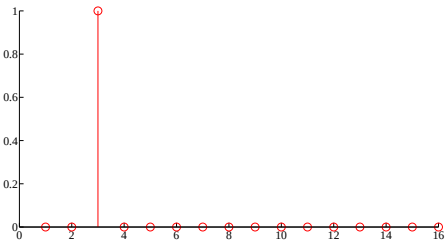
(e) Type 5: Haar 2 point: Atom 8



(f) Type 6: Haar 4 point: Atom 8



(g) Type 7: Haar 6 point: Atom 8



(h) Type 8: Euclidean Basis: Atom 3

Figure 5: Illustrates an atom from 8 different types of atoms.

with

$$\mu^2 = \sum_{i=1}^{L_y} \sum_{j=1}^{L_x} \frac{|B_{ij} - A_{ij}|^2}{L_x\, L_y}$$

- A is the matrix representation of the original image.
- B is the matrix representation of the approximate image.
- μ^2 is the mean square error.
- L_x is the number of pixels along the vertical axis.
- L_y is the number of pixels along the horizontal axis.
- $L_x\, L_y$ is the total number of pixels in the image.
- $I_{\max} = 2^{L_B} - 1$, where L_B is the number of bits used to represent the original image.

Numerically the higher the PSNR the better the quality of the approximate image. Nevertheless it is possible to produce a pleasant looking approximation with a not too high PSNR and vice versa. For other images the same PSNR may not produce a visually pleasing approximation.

4.6 Measure of sparsity of an approximated image

The sparsity of an image gives detail on the level of gain achieved with respect to the image. The whole aim is to be able represent an image with fewer data points, and there by reducing the data storage requirements. The measure of sparsity, or the Sparsity Ratio (SR) of an image is defined as follows,

$$\text{SR} = \frac{\text{Number of pixels in the image}}{\text{Number of coefficients used for approximating the image}}$$

When considering the best strategy for sparse image representation it is important to keep the quality (PSNR value) of the comparable images the same, and compare the respective SR values. The higher the SR value the sparser the approximation.

5 Comparing Selection Strategies

In the following section the aim is to examine the effectiveness of the BWOMP2D strategy in the wavelet domain. To do this BWOMP2D is first compared with the Discrete Cosine Transform (DCT) in the pixel domain to see artefacts formed as a result of block processing. The technique is also compared with existing pursuit strategies OMP2D, MP in 2D (MP2D) and its extension BWMP2D in the wavelet domain.

MP2D differs from OMP2D in (10) in that the coefficients c^k are calculated simply as

$$\sum_{\substack{i=1 \\ j=1}}^{L_x, L_y} d_n^x(i) R^{k-1}(i,j) d_m^y(j)$$

and the selected atoms may be selected more than once.

5.1 Experiment 1: DCT, OMP2D and BWOMP2D: Artefacts

Aim

The aim of this experiment is to compare the visual result (blocking effect) of approximate images using an existing orthonormal transform DCT to see if by working in the pixel domain blocking artefacts are observed. For this experiment the quality (PSNR value) of the approximate images are fixed, and the sparsity (SR value) is compared. This experiment will determine which strategy produces visually appealing approximated images under the current experimental conditions.

Methodology

When working with blocks an approximated image can be prone to blocking artefacts. This experiment involves approximating two images, the Lichtenstein Castle Figure 6(a) and the Planet Figure 6(b), using DCT in the pixel domain, and OMP2D and BWOMP2D in the wavelet domain to see if such observations are found. The Lichtenstein Castle image will be approximated using blocks of size 8 and to a PSNR value of 43. The Planet approximation image will be approximated to a higher PSNR value of 45. DCT approximations use the Discrete Cosine (DC) dictionary, and OMP2D and BWOMP2D use the Redundant Discrete Cosine (RDC) dictionary mentioned in Section 4.3. OMP2D and BWOMP2D will use Discrete Wavelet Transform (DWT) of level 5 prior to processing, and after processing revert back to the pixel domain using Inverse Discrete Wavelet Transform (IDWT) of level 5.

Discussion of results

For the both the Lichtenstein Castle and Planet approximation it is evident, by processing an image in the pixel domain blocking artefacts are still visual for PSNR values of 43 and 45 respectively, but other effects are not detected for images processed in the wavelet domain. The blocking artefacts are shown in Figure 7(b), and Figure 8(b). For both images, Table 3 in terms of sparsity show BWOMP2D to be the sparser strategy. The Planet approximation results recorded in Table 2(a) shows BWOMP2D produces a SR value of 37.42, compared to OMP2D value of 29.53 and DCT value of 26.51.

Conclusion

One of the main conclusions observed is that by working in the wavelet domain blocking artefacts produced by block approximation are avoided. The reason for this is because it is difficult to represent edges in a block, and so when a approximate image is visualised in the same domain as being processed, the irregularity of joining blocks is more evident. For this reason when an image is converted to a different domain, i.e., the wavelet domain, the artefacts still exist but when translated back to the pixel domain are less visual to the eye. This experiment

also gives promising results for the BWOMP2D strategy, and it produces sparser results compared to OMP2D and DCT.

(a) Lichtenstein Castle (512 × 512) (b) Planet (512 × 512)

Figure 6: Test images: Sample 1

(a) Lichtenstein (512 × 512)

PSNR = 43			
Strategy	DCT	OMP2D	BWOMP2D
SR	4.1523	5.3118	5.5429

(b) Planet (512 × 512)

PSNR = 45			
Strategy	DCT	OMP2D	BWOMP2D
SR	26.5113	29.5307	37.4224

Table 3: DCT, OMP2D and BWOMP2D approximation results, using blocks of size 8, for the Lichtenstein Castle and Planet image shown in Figure 6. PSNR is fixed for all images. Approximations use DC and RDC dictionary.

(a) Lichtenstein Castle Original

(b) DCT

(c) OMP2D

(d) BWOMP2D

Figure 7: Lichtenstein Castle (512×512) approximations at PSNR 43 using blocks of size 8.

(a) Planet Original

(b) DCT

(c) OMP2D

(d) BWOMP2D

Figure 8: Planet (512×512) approximations at PSNR 45 using blocks of size 8.

5.2 Experiment 2: MP2D vs OMP2D

Aim

The aim of this experiment is to illustrate if OMP2D produces sparser results than MP2D in the wavelet domain for sample 1 test images shown in Figure 6. The aim is also to realise which strategy has faster processing times.

Methodology

For this experiment the Lichtenstein Castle and Planet image shown in Figure 6, are approximated using strategies MP2D and OMP2D. For this experiment each image is processed using blocks of size 8, 16 and 32, and the respective SR values are compared for both strategies. For fair comparison the PSNR values for all block approximations of an image are fixed to 2 decimal places and DWT of level 5 is applied prior to processing. For both strategies we give processing times for MATLAB implementation.

(a) Lichtenstein Castle (512 × 512)

PSNR = 43				
Strategy	MP2D		OMP2D	
Block Size	SR	MAT	SR	MAT
8	8.8691	0.3360	9.7198	0.3986
16	9.1955	0.9716	10.3758	1.2116
32	9.0075	5.6019	10.3614	12.5316

(b) Planet (512 × 512)

PSNR = 45				
Strategy	MP2D		OMP2D	
Block Size	SR	MAT	SR	MAT
8	51.3907	0.0623	54.8648	0.0714
16	66.1980	0.1331	74.8769	0.1506
32	64.1881	0.8110	75.4806	1.2332

Table 4: MP2D and OMP2D approximation results for images shown in Figure 6 with fixed PSNR.

Discussion of results

Sparsity results shown in Table 4 for the Lichtenstein Castle image and the Planet image, show OMP2D produces higher sparsity values than MP2D for all block sizes, and as the images are relatively small the processing times are minimal for both strategies. For both images, OMP2D results using blocks of size 32 show processing times are around twice as much as MP2D. However for block size 8, the processing times are similar but yet OMP2D has significantly larger sparsity values.

Conclusion

From Table 4 it is possible to conclude for sample 1 test images shown in Figure 6, OMP2D produces higher sparsity results than MP2D, and for approximations

that use blocks of size 8 processing times for both strategies are almost identical.

As the BWOMP2D approach is an extension of the OMP2D approach, then based on Experiment 1 conclusions, it implies for the same set of images, BWOMP2D produces sparser results than MP2D.

5.3 Experiment 3: BWMP2D vs BWOMP2D

Aim

We compare the BWOMP2D approach with the block wise extension of MP2D (BWMP2D) to see which strategy produces sparser approximations.

Methodology

This experiment is exactly the same as Experiment 2, but using pursuit strategies BWOMP2D and BWMP2D, where BWMP2D is the block wise equivalent extension of the BWOMP2D strategy.

(a) Lichtenstein Castle (512 × 512)

PSNR = 43				
Strategy	BWMP2D		BWOMP2D	
Block Size	SR	MAT	SR	MAT
8	9.4395	0.4513	10.4096	1.3390
16	9.4259	0.9249	10.6265	1.6679
32	9.2624	4.8952	10.6135	11.0712

(b) Planet (512 × 512)

PSNR = 45				
Strategy	BWMP2D		BWOMP2D	
Block Size	SR	MAT	SR	MAT
8	78.9828	0.1106	88.1156	0.2324
16	75.5023	0.1624	86.3452	0.2490
32	71.7613	0.6676	83.3261	1.1173

Table 5: BWMP2D and BWOMP2D approximation results for images shown in Figure 6 with fixed PSNR.

Discussion of results

BWMP2D vs BWOMP2D results in Table 5, show BWOMP2D to have slower processing times than BWMP2D. Processing times for BWOMP2D are around twice as much for the Lichtenstein image, but as the Planet image is sparse, processing times do not differ as much. For both images, approximations that use blocks of size 8 for processing have faster processing times.

For both images and both strategies, most approximations that use blocks of size 8 produce the sparser approximation, In other cases, approximations of block size 8 are still comparable with approximations that use blocks of size 16, and blocks of size 32.

For both strategies BWOMP2D produces sparser results than BWMP2D. One example for the Planet image using blocks of size 32 shows BWOMP2D produces a sparsity value of 83, compared to the lesser value of 71.7 for BWMP2D.

Conclusion

For the set of images shown in Figure 6 results from this experiment allow to conclude BWOMP2D produces sparser results than BWMP2D, and from conclusions obtained from Experiment 1 and 2, sparser results than OMP2D and MP2D. Thus, the results obtained for this set of images motivates further analysis of the BWOMP2D approach.

6 Perceptually Lossless Approximations

In this section the BWOMP2D approach is used with a mixed dictionary to approximate images in the wavelet domain. The aim is to produce 'perceptually lossless' approximations. These are approximations that do not have any visual degradation with respect to the raw image at the original resolution. One of the applications where perceptually lossless approximations are required concerns representation of X-ray medical images. Such images are called radiography and are useful for:

- the detection of pathologies of the skeletal system,
- the detection of some disease processes in soft tissue.

Radiography can be used for instance to identify lung diseases such as pneumonia, lung cancer or pulmonary edema. The abdominal X-ray can also detect intestinal obstruction or pathologies such as gallstones or kidney stones. These applications highlights the importance of visually retaining all details on approximated X-ray images, otherwise the approximation could be misleading and, as a result, could potentially be life threatening. Thus, perceptually lossless approximations are tested here, using the set of images shown in Figure 9

6.1 Experiment 1: OMP2D vs BWOMP2D Further Analysis

Aim

The aim of this experiment is to conclude if the BWOMP2D approach produces sparser results than the OMP2D approach for perceptually lossless approximations. The experiment will involve approximating the 6 different X-ray images depicted in Figure 9.

Methodology

For this experiment OMP2D is compared against BWOMP2D. For each image and each strategy, the varying factor of the experiment is the size of the block used for approximation. Thus, in order to conduct a fair experiment it is essential the X-ray images are *prepared* before approximation.

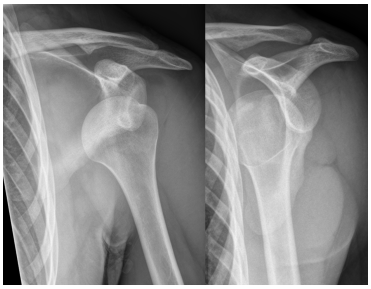
Preparation of an image involves manually cropping the image in such a way that all the varying blocks sizes divide perfectly into the dimensions of the cropped image, i.e., leaving no unprocessed parts of the cropped image. When cropping

(a) Osteopetrosis pelvis (629 × 829)

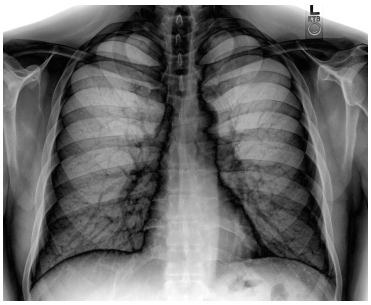
(b) Chest (755 × 927)

(c) Sinus frontalis (1006 × 1074)

(d) Cervical spine (1110 × 992)



(e) Dislocated shoulder (1005 × 1305)



(f) Chest Inverted (1956 × 2412)

Figure 9: Test images: Sample 2

an image, areas of most detail are considered: the aim behind this is to retain detail. The cropped image should now *act* as the original image, and be processed as usual with varying block sizes.

For comparison the 'original' image should first be approximated using OMP2D. For OMP2D processing the PSNR value is fixed to 2 decimal places, and the number of atoms (NAT) used for the varying block approximations are recorded. The image is then processed with the BWOMP2D approach using the same NAT. In this way it is possible to compare the PSNR values for OMP2D and BWOMP2D processing at fixed sparsity (SR). For this experiment DWT of level 5 is applied before an image is processed. The experiment requires the use of an industrial computer to deal will the high storage demand of the block wise strategy.

Discussion of results

By observing Table 6, it is possible to conclude for each image and each block size, the BWOMP2D strategy produces a higher quality approximate image for the given sparsity, compared to OMP2D. The amount of increase in quality varies depending on the size of the blocks used for approximation. For block size 8 approximations, there is a large increase in PSNR value, and for block size 32 not as large of an increase, but still an increase that can be appreciated since the images where originally approximated to a very high level (visually indistinguishable). Results in Table 6 for the Osteopetrosis pelvis X-ray image shows block size 8 approximation has an increase of 2.2 dB, block size 16 has an increase of 0.66 dB and finally block size 32 has an increase of 0.36 dB. It is important to note for the Chest Inverted X-ray shown in Figure 9(f), BWOMP2D with block size 32 fails to process (FtoP) the very large 1952×2400 pixel image.

Conclusion

The sparser results achieved by the BWOMP2D strategy indicate the benefits of the BWOMP2D strategy. It was also noticed that block size 8 approximations produced a large increase in dB. This conclusion is beneficial in terms of processing times as approximations that use blocks of size 8 deal with smaller matrices, as a result produce faster processing times.

Results for this experiment were achieved using an industrial computer. Thus, the aim now is to focus on ways to apply the strategy on large images.

(a) Osteopetrosis pelvis X-ray image (608 × 800)

Block Size	8	16	32
Strategy	SR = 29.0614	SR = 37.7523	SR = 39.4421
OMP2D	46.6760	46.6798	46.6809
BWOMP2D	48.8707	47.3307	47.0429

(b) Chest X-ray image (736 × 896)

Block Size	8	16	32
Strategy	SR = 22.9960	SR = 25.8894	SR = 26.1990
OMP2D	47.7359	47.7387	47.7395
BWOMP2D	48.1122	47.8195	47.7966

(c) Sinus frontalis X-ray image (992 × 1056)

Block Size	8	16	32
Strategy	SR = 7.4119	SR = 7.7642	SR = 7.8746
OMP2D	46.7195	46.7244	46.7160
BWOMP2D	46.8823	46.7895	46.7989

(d) Cervical spine X-ray image (1088 × 992)

Block Size	8	16	32
Strategy	SR = 28.4033	SR = 35.4682	SR = 37.0396
OMP2D	46.8579	46.8604	46.8599
BWOMP2D	48.0985	47.2185	47.0493

(e) Dislocated shoulder X-ray image (992 × 1280)

Block Size	8	16	32
Strategy	SR = 17.2686	SR = 18.9104	SR = 19.2472
OMP2D	47.0155	47.0159	47.0192
BWOMP2D	47.4321	47.1545	47.1365

(f) Chest X-ray Inverted image (1952 × 2400)

Block Size	8	16	32
Strategy	SR = 5.3625	SR = 5.6227	SR = 5.7661
OMP2D	46.4196	46.4217	46.4181
BWOMP2D	46.4630	46.4437	FtoP

Table 6: Details of results of the X-ray images shown in Figure 9 for OMP2D and BWOMP2D approximation. All approximations are to a visually indistinguishable quality.

7 Resolution

The detail in an image is determined by resolution. Resolution can be expressed as the total number of dots, or pixels, used to display an image. A higher resolution image uses more pixels than a lower resolution image, resulting in a crisper, cleaner appearance. The resolution of an image may effect the results obtained by sparse image approximation. Thus, approximations on difference resolutions of an image are tested here.

7.1 Experiment 1: BWOMP2D, Wavelet and DCT

Aim

The aim of this experiment is to fix sparsity for BWOMP2D approximations for all block sizes and resolutions, and to identify any trends that may appear for increasing resolution. Additionally the aim is to compare BWOMP2D, Wavelet and DCT approximations using blocks of size 8 to see if similar trends are identified. The experiment will involve approximating variations of the 6 different X-ray images depicted in Figure 9.

Methodology

This experiment approximates various resolutions of the X-ray images shown in Figure 9. Each resolution is approximated using blocks of size 8, 16 and 32 at a fixed SR. The common problem with most resolutions is that when processed with a particular block size, there are some pixels of the original image left unprocessed. For large resolution images if there are some unprocessed pixels it would not make a major effect on the approximation. However if there are the same number of unprocessed pixels in a smaller resolution a lot more detail is lost compared to the higher resolution. So for a fair experiment the images need to be *prepared*.

To prepare the resolution images for a particular block size, one calculates the scale factors between the smallest resolution and the higher resolutions. A scale factor for a higher resolution is calculated by dividing the dimensions by the dimensions of the smallest resolution. One then calculates the amount of unprocessed pixels that remain for the smallest resolution (for that particular block size). Then on the higher resolutions manually crop out the same pixels or detail by removing 'scale factor of that resolution $\times$ the number of unprocessed pixels from the smallest resolution'. This way the same amount of detail is removed from higher resolution images as was for the smallest resolution. The same rescaling technique is applied to prepare images for other block sizes.

This experiment will apply DWT of level 5 to an image before it is processed with BWOMP2D. Approximations will be made on sample 2 test images shown in Figure 9, where sparsity for all block sizes and resolutions are fixed. Since this experiment involves approximation of large images, they will be performed using an industrial computer. For further analysis, Wavelet and DCT approximations using blocks of size 8 are compared with BWOMP2D approximations.

Discussion of results

Results show to a certain stage where there was an increase in resolution there was an increase in PSNR. It is also clear for top end resolutions the PSNR values lower. This effect is expected as larger images have courser detail. Wavelet and DCT approximations show similar trends as BWOMP2D block size 8 results when resolution is increased. Looking at the the results obtained for different block sizes, Table 7 shows that the change in block size does not produce much better approximatations. This suggests that it would be more convenient to work with blocks of size 8, as, in Section 5 it was shown to have faster processing times. It is important to note for the Chest Inverted image BWOMP2D approximation fails to process (FtoP) using blocks of size 32.

Conclusion

This experiment concludes BWOMP2D approximations produce sparser approximations than the other popular techniques. It also shows for all techniques the relation is maintained at all resolutions.

(a) Osteopetrosis pelvis (629 × 829)

SR = 29.9					
Resolution	8	16	32	Wavelet	DCT
240 × 316	41.3679	41.2347	41.0015	36.8254	32.9012
480 × 633	46.5515	46.5259	46.3902	41.4305	38.6408
600 × 791	48.7979	48.8694	48.7370	42.9821	40.2701
629 × 892	48.8081	48.7350	48.5763	43.3478	40.5982

(b) Chest (755 × 927)

SR = 24.4					
Resolution	8	16	32	Wavelet	DCT
240 × 295	45.8464	45.9761	45.8730	41.2904	38.2943
480 × 589	49.2196	49.3881	49.3041	45.0619	42.2588
600 × 737	49.7066	49.8375	49.6642	46.0143	43.3778
755 × 927	47.9732	48.0437	47.9815	45.4207	43.4493

(c) Sinus frontalis (1006 × 1074)

SR = 7.6					
Resolution	8	16	32	Wavelet	DCT
240 × 256	46.7453	46.8104	46.8336	40.4111	39.4066
600 × 640	47.9568	48.3227	48.4640	41.9306	41.3698
768 × 820	47.9356	48.3370	48.1842	41.9110	41.3552
1006 × 1074	46.8288	47.2204	47.1734	40.8867	40.3295

(d) Cervical spine (1110 × 992)

SR = 28					
Resolution	8	16	32	Wavelet	DCT
239 × 214	38.0355	37.8863	37.3840	33.8926	31.2811
480 × 429	43.0248	42.9619	42.3995	39.0081	35.8824
600 × 536	44.3316	44.2543	43.6217	40.4955	37.4007
1110 × 992	47.9123	47.9921	47.3245	44.3929	41.6030

(e) Dislocated shoulder (1005 × 1305)

SR = 16.9					
Resolution	8	16	32	Wavelet	DCT
240 × 312	42.0965	41.8928	41.5640	37.1574	35.1293
480 × 623	44.3526	44.3978	44.1212	40.8218	39.2167
768 × 997	46.1144	46.1540	45.9895	42.5026	41.2302
1005 × 1305	47.6465	47.7014	47.6043	43.8443	42.5136

(f) Chest Inverted (1956 × 2412)

SR = 5.8					
Resolution	8	16	32	Wavelet	DCT
240 × 296	45.5202	45.7460	45.3843	38.1533	37.1478
480 × 592	47.3282	47.6725	47.7164	40.8712	40.2420
600 × 740	47.4016	47.7649	47.9021	41.0122	40.4954
768 × 947	47.0364	47.4534	47.7373	40.6268	40.1835
1024 × 1263	46.4781	46.9348	47.3631	39.9730	39.5775
1956 × 2412	45.5663	46.0655	FtoP	38.6962	38.5407

Table 7: BWOMP2D resolution approximation results for X-ray images shown in Figure 9 with fixed SR. Wavelet and DCT approximations use blocks of size 8.

8 Enhance Performance using C++ MEX files

What is a MEX file?

C, C++, or Fortran subroutines can be called from the MATLAB command line as if they were built-in functions. These programs, called binary MEX files, are dynamically-linked subroutines that the MATLAB interpreter loads and executes. MEX stands for 'MATLAB executable'.

MEX Application

MATLAB scripts are very slow when it comes to loops, as in MATLAB, loops are done implicitly. Sometimes, however, the loops can be avoided by vectorizing the code. This means that a function is applied to all input/data at once. In the cases where a computation is not vectorizable, and speed is important, it is possible to re-write the performance-critical MATLAB routines in C or C++. A MATLAB function written in a compiled language is called a MEX file.

Here is a demonstration on how to implement a C++ MEX file of certain codes performed in MATLAB. A case where a C++ MEX file can be considered is when a loop is used to iterate through a structure with MATLAB implementation.

For example, if a script named Test1.m calls upon the arrayProduct.m routine, and it is found the arrayProduct.m routine takes some time to iterate through a large array, then the use of a C++ MEX file could enhance performance. The way to enhance performance of the Test1.m script would be to implement the arrayProduct routine in C++. To be able to execute the C++ file on a MATLAB platform, there is a need for a MEX file (arrayProduct.cpp) that acts as the 'gateway' to the platform for the C++ routine arrayProduct.h.

In other words, MATLAB routine calls,

1) Test1.m (MATLAB) → arrayProduct.m (MATLAB)

are converted to C++ MEX calls,

2) Test1.m (MATLAB) → arrayProduct.cpp (MEX) → arrayProduct.h (C++).

Implementations for 1) and 2) are shown in Appendix C.

The above described construction of C++ MEX files was used to produce the C++ MEX version of the MATLAB function BWOMP2D. This implementation, that is called BWOMP2D_Mex is a direct translation of the available MATLAB function. It reproduces the same results but with faster processing time. The BWOMP2D_Mex routine is given in the attached CD.

9 Macro Processing Scheme

As discussed in Section 4.2.2, block-independent OMP2D processing involves partitioning an image into small blocks, henceforth to be called *micro blocks* of size, say $L_x \times L_y$ pixels and optimise, in an stepwise manner, the selection of the dictionary atoms for approximating each micro block. Thus, the storage demands of this implementation depend only on the micro block size and not on the actual image size. Another feature of this type of processing is that the computational complexity of the whole image increases linearly with the number of micro blocks.

Since, in addition of the best atoms, BWOMP2D processing also optimises in a stepwise manner which is the block to be approximated at each iteration step, and so the processing of micro blocks becomes slightly dependent. Consequently, within this framework the image size does matter and large images cannot be processed as a whole. In order to implement BWOMP2D for large images, and also at a competitive running time, we introduce the *macro processing scheme* described below.

As done for the independent processing, the images is first partitioned into micro blocks of size $L_x \times L_y$. Additionally, the algorithm breaks the image into *macro blocks* (henceforth to be also called macros) of size say, $N_x \times N_y$. The size of the macros can be conveniently set, with the only restriction that it satisfies being a multiple of the block size, i.e. $N_x = a\,L_x$ and $N_y = b\,L_y$, where $a, b \in \mathbb{Z}^+$. Next the large image is split into macro blocks. Each macro is to be processed by BWOMP2D. The macro partition is illustrated by the light pink macros in Figure 10.

Our macro processing-based algorithm for implementation of BWOMP2D has an additional functionality to process the remainder of an image: The algorithm singles out the part of the image which is not covered by the the macro partition and, if applicable, calculates the size of the largest blocks that can fit that section of the image. Again, the size of the fitted macro blocks are a multiple of the block size. The result of these steps are illustrated by the red, green and dark blue regions in Figure 10.

Finally, depending on the size of the edges, the algorithm either keeps all the edge macros or processes them using BWOMP2D. The edges are illustrated by the orange and sky blue macros in Figure 10. In Figure 10 all the pixels highlighted by the purple macros are simply kept. To obtained the approximated image, all macro results are concatenated.

Code snippets of the macro processing scheme are shown in Appendix D.

A numeric example of macro partitioning

Figure 10 image size is 372×491.

User defined:
Block size ($L_x \times L_y$) is 16×16,
Light pink macro size ($N_x \times N_y$) is 128×128.

Calculation of the next largest macros:
Red macro size 112×128,
Green macro size 128×96,
Dark blue macro size 112×96.

Remainder:
Orange macro size $(RL_x \times N_y)$ is 4×128, where block size $(RL_x \times L_y)$ is 4×16,
Sky blue macro size $(N_x \times RL_y)$ is 128×11, where block size $(L_x\ RL_y)$ is 16×16,
Purple macro are of sizes 4×96, 112×11 and 4×11.

For the orange macros, if $RL_x < 3$ simply keep all the pixels, otherwise process each macro using BWOMP2D.

For the sky blue macros, if $RL_y < 3$ simply keep all the pixels, otherwise process each macro using BWOMP2D.

For the purple macros, keep all the pixels.

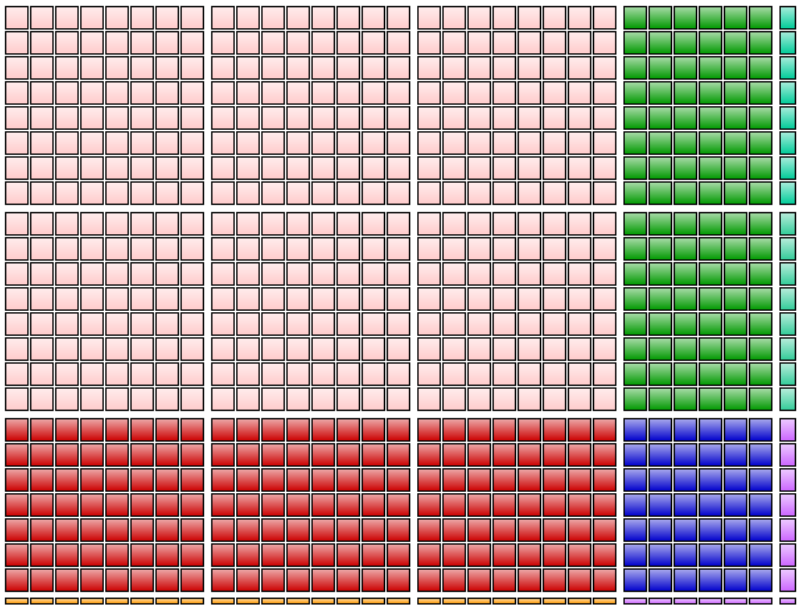


Figure 10: Illustration of macro processing an image of size 372×491, using blocks of size 16×16 pixels, and macro blocks of size 128×128 pixels.

10 Macro Approximation

10.1 Experiment 1: Wavelet Application

Aim

To examine the effectiveness of the macro processing scheme it is important to find an effective technique, of applying DWT to an image formed of macros. The aim

of this experiment is to get an insight of how to best deal with an image before it is passed to the macro processing scheme.

Methodology

The experiment involves comparing two different ways of processing an image by altering the regions where DWT is applied. Method 1 would be to perform DWT on the whole image, and then process the whole image using the BWOMP2D strategy. Method 2 would be to perform DWT on regions within an image and then process the whole image using the BWOMP2D strategy. An example would be to take an image say of dimension 512×512 such that the dimensions are divisible by blocks of size 8, 16 and 32. Then, for method 2, for each block size apply DWT on regions within the image. Examples of possible sectioning, henceforth to be called 'region size' for an image of size 512×512 would be to have regions of size 128×128, or 256×256. A region of size 512×512 on a image of size 512×512 would be the same result as applying method 1 to the image.

As the image size would divide perfectly into region size and block size, it would allow one to examine the effect of block size and region size on each method. To compare method 1 and method 2 the number of atoms used (NAT) will remain fixed, and the achieved level of PSNR will be compared. The difference between the two methods is: method 1, the image would contain the combined DWT information of the whole image before it is processed, and method 2, the image would have the DWT information of regions within the image before it is processed. Comparing these two methods would allow one to see if processing an image with its full DWT information produces higher quality approximations, than processing the image with DWT information of regions. Since this experiment fixes all parameters it would mean any difference in result would be as a result of applying the two different methods. For analysis, a new set of images shown in Figure 11, along with sample 1 test images shown in Figure 6 are to be used.

Discussion of results

For all tested images, Table 8 results indicate method 2 produces lower quality approximations when compared with method 1. The difference between the two methods is less than 1 dB. Nevertheless, results indicate method 1 produces higher quality approximations. In general no block size is more superior than the other.

Conclusion

Results show with method 1 processing, higher quality approximations are obtained. This implies before an image is sent to the BWOMP2D strategy it is better to send all the DWT information of that image. This would mean when preparing macros under the macro processing scheme all the DWT information of that particular macro should be sent to the strategy. This method of preparing macros for the macro processing scheme is to be termed Macro WT (or MWT), and from now on will be used for all macro processing approximations.

(a) Lichtenstein Castle (512 × 512)

SR = 10.31 (NAT=25418)			
Region Size	8	16	32
128 × 128	42.9975	43.1825	43.1871
256 × 256	43.0855	43.2843	43.2839
512 × 512	43.1070	43.3467	43.3333

(b) Planet (512 × 512)

SR = 87.38 (NAT=3000)			
Region Size	8	16	32
128 × 128	44.6085	44.4078	44.2455
256 × 256	44.8780	44.6745	44.5499
512 × 512	45.0523	44.9212	44.7096

(c) Landscape (1024 × 1024)

SR = 9.9998 (NAT=104860)			
Region Size	8	16	32
128 × 128	42.0720	42.6047	42.7857
256 × 256	42.1878	42.7069	42.8902
512 × 512	42.2417	42.7803	42.9567
1024 × 1024	42.2697	42.8082	43.0040

(d) Rib (1024 × 1024)

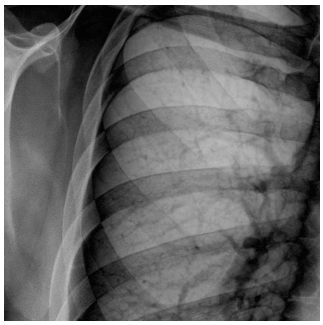
SR = 5.2 (NAT 201650)			
Region Size	8	16	32
128 × 128	47.6761	48.2334	48.5673
256 × 256	47.7444	48.2934	48.6467
512 × 512	47.7719	48.3373	48.6844
1024 × 1024	47.7853	48.3503	48.7046

(e) Gentleman (1024 × 1024)

SR = 4.6517 (NAT 225418)			
Region Size	8	16	32
128 × 128	46.3111	46.8257	47.1104
256 × 256	46.3694	46.8756	47.1778
512 × 512	46.3973	46.9159	47.2141
1024 × 1024	46.4117	46.9325	47.2348

Table 8: BWOMP2D approximation results for sample 1 and sample 3 test images shown in Figure 6 and 11. SR value is fixed for each image. Method 1: WT is applied on the whole image, and processed as a whole image. Method 2: WT is applied to each region, then all regions are processed as a whole image.

(a) Landscape (1024 × 1024)



(b) Rib (1024 × 1024)

(c) Gentleman (1024 × 1024)

Figure 11: Test images: Sample 3

10.2 Experiment 2: Uniform macros

Aim

The aim of this experiment is to successfully apply the macro processing scheme on images using uniform macros. The objective is to compare the quality (PSNR values) of the approximate images for varying macro size and micro size, at fixed sparsity.

Another aim is to test the speed of our BWOMP2D C++ MEX implementation by comparing processing times of MATLAB and C++ implementations. The sole purpose for the C++ implementation was to be able to apply the BWOMP2D strategy on large images, and on a larger scale, but at a reduced time when compared to MATLAB implementation. It will be shown later C++ implementation is faster.

Successful application of the macro processing scheme, using uniform macros will be tested on sample 1 and sample 3 test images shown in Figure 6, and 11.

Methodology

The experiment will involve approximating five images with varied macros sizes using the BWOMP2D approach. Each macro will be processed with micro blocks of size 8, 16 and 32. The chosen images for testing will be suitable for uniform macros since the chosen macro size will be exactly divisible into the image size. For this reason, there would be no need to prepare images for fair micro size/ macro size comparison. This also implies there will be no remaining unprocessed parts

of the image. Previously, it was decided that DWT of level 5 would be applied to each macro (MWT) before the macro was passed to the BWOMP2D selection strategy. For all approximations, SR values will be fixed to 2 decimal places and PSNR values compared.

Discussion of results

Results in Table 9 show the time reduction advantage of BWOMP2D C++ implementation. Examples of time reduction for the Rib (1024×1024) approximations using macros of size 128×128, shown in Table 8(d) include, micro size 8: time reduced by a factor of 12.5, micro size 16: time reduced by a factor of 6.6, and micro size 32: time reduced by a factor of 4.7. For all images, the larger the macro size the higher the quality (PSNR value) of the approximate images. Since the sparsity of the approximate images is higher for larger macro, the processing time also reduces. This is because the strategy has fewer matrices to manage. One of the main noticeable features of the BWOMP2D strategy indicated by Table 9 results, show for most images micro size of 8 produces higher quality results compared micro size 16, and 32, and in cases where this is not the case, micro size 8 is comparable.

Conclusion

This experiment shows successful application of the macro processing scheme using uniform macros for image approximation. As expected the time reduction achieved by the BWOMP2D C++ MEX implementation is noticeable. The gain makes the BWOMP2D C++ implementation a much needed tool, especially for approximations that involve large images, due to the high matrix complexity. Another important conclusion raised by this experiment is that micro blocks of size 8 produce higher quality approximations or comparable results, when compared with approximations that use micro blocks of size 16 and 32. This is a advantageous, since processes that use micro blocks of size 8 are faster when compared to processes that use micro blocks of size 16 and 32. This experiment leads to the conclusion that *the BWOMP2D strategy is efficient and effective using micro blocks of size 8, and other micro blocks need not to be considered*.

(a) Lichtenstein Castle (512 × 512)

SR = 10									
Micro Size	8			16			32		
Macro Size	PSNR	MAT	C++	PSNR	MAT	C++	PSNR	MAT	C++
128 × 128	37.5820	1.2088	0.1106	37.6848	1.7374	0.3976	37.6880	9.8750	2.4582
256 × 256	38.5534	1.2323	0.1016	38.6716	1.6681	0.3353	38.6775	10.2783	2.4966
512 × 512	43.4586	0.9065	0.1114	43.7009	1.7035	0.3563	43.6987	10.9445	FtoP

(b) Planet (512 × 512)

SR = 35									
Micro Size	8			16			32		
Macro Size	PSNR	MAT	C++	PSNR	MAT	C++	PSNR	MAT	C++
128 × 128	47.7661	0.3482	0.0584	47.7931	0.5029	0.1233	47.4964	2.4843	0.6758
256 × 256	50.0745	0.4005	0.0420	49.9511	0.4452	0.1076	49.8058	2.6290	0.6833
512 × 512	51.0300	0.4525	0.0420	50.9031	0.5023	0.1065	50.7084	2.8091	FtoP

(c) Landscape (1024 × 1024)

SR = 10									
Micro Size	8			16			32		
Macro Size	PSNR	MAT	C++	PSNR	MAT	C++	PSNR	MAT	C++
128 × 128	43.9583	9.0809	0.7402	44.6687	15.0453	2.3912	44.9954	101.1875	24.6379
256 × 256	44.8851	7.1574	0.5944	45.9706	14.8089	2.4266	45.9208	102.7200	26.3414
512 × 512	45.2707	6.6772	0.6613	45.9706	12.6212	2.4349	FtoP	FtoP	FtoP

(d) Rib (1024 × 1024)

SR = 5									
Micro Size	8			16			32		
Macro Size	PSNR	MAT	C++	PSNR	MAT	C++	PSNR	MAT	C++
128 × 128	47.2266	9.5996	0.7671	47.8134	17.6978	2.6959	48.1649	119.0933	25.2797
256 × 256	47.4237	9.5689	0.7242	48.0035	17.2726	2.6240	48.3649	119.6354	21.9996
512 × 512	47.8651	8.6680	0.7855	48.4638	18.1708	2.6288	48.8196	145.0425	FtoP

(e) Gentlemen (1024 × 1024)

SR = 4.2									
Micro Size	8			16			32		
Macro Size	PSNR	MAT	C++	PSNR	MAT	C++	PSNR	MAT	C++
128 × 128	47.1442	8.6315	0.8619	47.7488	15.8265	3.3221	48.0819	162.2858	29.5583
256 × 256	47.4484	9.0473	0.8265	48.0370	15.6827	3.3352	48.3965	162.3457	30.3997
512 × 512	47.8635	9.2573	0.9053	48.4777	16.6082	3.3303	FtoP	FtoP	FtoP

Table 9: BWOMP2D macro approximation results for sample 1 and sample 3 test images shown in Figure 6 and 11. MWT of level 5 is applied to the image. SR is fixed to 2 decimal places.

10.3 Experiment 3: Non-Uniform

Aim

The aim of this experiment is to successfully apply the macro processing scheme on images using non-uniform macros. The objective is to compare SR results for macro size and micro size of approximations of the same quality. Successful application of the macro processing scheme, using non-uniform macros will be tested on sample 1 and sample 3 test images shown in Figure 6, and 11.

Methodology

The experiment will involve approximating six images with varied macros sizes using the BWOMP2D strategy. Each macro will be processed with micro blocks of size 8, 16, 32. The chosen images for testing will be suitable for non-uniform macros since the chosen macro size will not be exactly divisible into the image size. The macro processing scheme will be responsible for the remaining unprocessed parts of the image, and deal with using non-uniform macros. Earlier, it was decided that DWT of level 5 would be applied to each macro before the macro was passed to the BWOMP2D selection strategy. For all approximations, PSNR values will be fixed to 2 decimal places and SR values compared.

Discussion of results

Observations of the results obtained for the non-uniform macro processing shown in Table 10 show an increase in macro size gives rise to an increase in sparsity. Results for the Dislocated Shoulder image shows there is an increase in sparsity from 17.1443 for macro size 128×128, to 19.1148 for macro size 800×800. For non-uniform macro processing, micro size 32 produces a slight increase in sparsity compared to micro size 16, and 8, but the gain would be canceled since it was found earlier processes that use micro blocks of size 32 have slower processing times.

Conclusion

This experiment shows successful application of the macro processing scheme using non-uniform macros for image approximation. Results also show larger macros produce sparser results. This experiment reconfirms the findings of Section 10 Experiment 2; where it was found that results for micro blocks of size 8 were very competitive with results for micros of size 16 and 32.

(a) Osteopetrosis pelvis (629 × 829)

PSNR = 47			
Macro Size	8	16	32
128 × 128	30.4740	30.0473	28.2073
256 × 256	35.6419	34.9538	32.9588
512 × 512	38.6195	39.8259	39.9357
608 × 608	39.1208	39.7895	40.2688

(b) Chest (755 × 927)

PSNR = 47			
Macro Size	8	16	32
128 × 128	23.7547	24.6014	25.1423
256 × 256	30.4390	31.0536	31.6432
512 × 512	32.5498	33.6015	34.3367
736 × 736	33.4889	34.3687	34.5179

(c) Sinus frontalis (1006 × 1074)

PSNR = 47			
Macro Size	8	16	32
128 × 128	7.0664	7.3383	7.4293
256 × 256	7.2238	7.4808	7.5819
512 × 512	7.3255	7.5955	7.7089
800 × 800	7.4907	7.7681	7.8486

(d) Cervical spine (1110 × 992)

PSNR = 47			
Macro Size	8	16	32
128 × 128	30.5884	31.4696	31.5128
256 × 256	33.3198	33.9401	34.4186
512 × 512	34.8975	35.8309	36.1035
800 × 800	33.6858	34.5460	34.8676

(e) Dislocated shoulder (1005 × 1305)

PSNR = 47			
Macro Size	8	16	32
128 × 128	17.1443	17.7516	18.0980
256 × 256	18.1385	18.7637	19.2467
512 × 512	19.0203	19.7912	20.1389
800 × 800	19.1148	19.8220	20.0040

(f) Chest Inverted (1956 × 2412)

PSNR = 47			
Macro Size	8	16	32
128 × 128	4.8202	5.0489	5.1800
256 × 256	4.9610	5.1879	5.3274
512 × 512	5.0317	5.2715	5.4043
800 × 800	5.0758	5.3164	5.4465

Table 10: BWOMP2D macro approximation results for X-ray images shown in Figure 9. PSNR is fixed to 47 to 2 decimal places. MWT of level 5 is applied to each image.

11 Conclusion

It is clear from the results presented in this thesis that, while the application of orthogonal transforms for approximating images is a simple task with available tools, it does not produce optimal approximations with regards to sparsity. On the contrary, releasing the condition of orthogonality and creating a large redundant dictionary for choosing approximating atoms, the sparsity property is improved. This type of approximation is known as highly non-linear.

Approximating images with a highly non-linear approximation involves the problem of i) using the appropriate dictionary for the signal/image in hand, ii) choosing the right functions from the dictionary. In the project the dictionary was fixed. The selection of atoms from such a dictionary was addressed by stepwise greedy techniques which are known as Pursuit Strategies.

The block selection strategy BWOMP2D was compared with other widely used strategies such as the block independent strategy OMP2D, block independent strategy MP2D and the block selection strategy BWMP2D. The examples given in this thesis illustrate the improvements in sparsity achieved by the BWOMP2D technique.

For a fixed mixed dictionary comprising of redundant DCT component and a component of localised atoms, one of the main findings using the BWOMP2D strategy was that *micro blocks of size 8 was the most convenient size, as it gave comparable results as micros of size 16 and 32, but the processing time was considerably faster.*

The inability to apply the block selection approach to large images lead to the proposal of a scheme that would allow to do that. The proposal enabled the application of the BWOMP2D strategy on larger images by decomposition into macro blocks. This scheme is called the macro processing scheme. Results obtained by this scheme showed that the use of large macros produced higher sparsity for an image. The implementation of the BWOMP2D C++ MEX file helped to reduce processing times.

References

[1] L. Rebollo-Neira, J. Bowley, A. Constantinides and A. Plastino, "Self contained encrypted image folding", *Physica A*, vol 391, pp. 5858–5870, 2012.

[2] J. Bowley, L. Rebollo-Neira, "Sparse image representation by discrete cosine/spline based dictionaries", arXiv:0909.1310v1.

[3] L. Rebollo-Neira, J. Bowley, "Sparse representation of Astronomical Images", preprint,

[4] Ryszard Maciol, Yuan Yuan and Ian T. Nabney, "Colour Image Coding with Matching Pursuit in the Spatio-frequency Domain", Image Analysis and Processing ICIAP 2011 Lecture Notes in Computer Science, vol 6978/2011, pp 306–317, 2011.

[5] Y. Yuan and D. M. Monro, "Improved matching pursuits image coding", In Proc.of International Conference on Acoustics, Speech,and Signal Processing, vol 2, pp 201–204, 2005.

[6] L. Rebollo-Neira and D. Lowe, Optimized orthogonal matching pursuit approach, IEEE Signal Processing Letters, vol, 9 137–140, 2002.

[7] M. Andrle and L. Rebollo-Neira, A swapping-based refinement of orthogonal matching pursuit strategies, Signal Processing, vol 86, 480–495, 2006.

[8] Highly nonlinear approximations for sparse signal representation. `http://www.nonlinear-approx.info/`

[9] I. Daubechies, "Ten lectures on wavelets", CBMS-NSF Series in Appl. Math., SIAM, 1991.

[10] L. Rebollo-Neira, Personal communication, 2012.

[11] S. H. Friedberg, A. J. Insel and L. E. Spence, "Linear Algebra", Pearson Education, California, 2003.

[12] S. Mallat. "A Wavelet Tour of Signal Processing", Academic Press, 1999.

Appendix A

Cauchy-Schwart Inequality Proof

The Cauchy-Schwart inequality is $|< f(x), g(x) >|^2 \leqslant \|f(x)\|^2 \|g(x)\|^2$ and by taking the square root gives $|< f(x), g(x) >| \leqslant \|f(x)\| \|g(x)\|$.

If $f(x) = \lambda g(x)$ for $f(x), g(x) \in L^2(a, b)$ where $\|f(x)\|$,$\|g(x)\| \neq 0$ and $\lambda \in C$, then $|< f(x), g(x) >| = \|f(x)\| \|g(x)\|$.

Define $P(\lambda) = \|\lambda g(x) - f(x)\|^2 \geqslant 0$, so

$$\begin{aligned} P(\lambda) &= \|\lambda g(x) - f(x)\|^2 \\ &= < \lambda g(x) - f(x), \lambda g(x) - f(x) > \\ &\quad \text{and from the inner product properties} \\ &= \lambda\overline{\lambda} < g(x), g(x) > -\lambda < g(x), f(x) > -\overline{\lambda} < f(x), g(x) > + < f(x), f(x) > \\ &= |\lambda|^2 \|g(x)\|^2 - 2Real(\lambda) < f(x), g(x) > + \|f(x)\|^2 \end{aligned}$$

and by substituting $\lambda = \frac{\|f(x)\|^2}{<f(x),g(x)>}$

$$\begin{aligned} P\left(\frac{\|f(x)\|^2}{< f(x), g(x) >}\right) &= \left|\frac{\|f(x)\|^2}{< f(x), g(x) >}\right|^2 \|g(x)\|^2 - 2Real\left(\frac{\|f(x)\|^2}{< f(x), g(x) >}\right) < f(x), g(x) > + \|f(x)\|^2 \\ &= \frac{\|f(x)\|^4}{|< f(x), g(x) >|^2} \|g(x)\|^2 - 2Real\left(\|f(x)\|^2\right) + \|f(x)\|^2 \\ &= \frac{\|f(x)\|^4}{|< f(x), g(x) >|^2} \|g(x)\|^2 - \|f(x)\|^2 \end{aligned}$$

and so,

$$\begin{aligned} \|f(x)\|^2 &= \frac{\|f(x)\|^4}{|< f(x), g(x) >|^2} \|g(x)\|^2 \\ \|f(x)\|^2 |< f(x), g(x) >|^2 &= \|f(x)\|^4 \|g(x)\|^2 \\ \|f(x)\|^2 |< f(x), g(x) >|^2 - \|f(x)\|^4 \|g(x)\|^2 &\geqslant 0 \end{aligned}$$

and dividing through by $\|f(x)\|^2$ gives

$$|< f(x), g(x) >|^2 - \|f(x)\|^2 \|g(x)\|^2 \leqslant 0$$

hence

$$|< f(x), g(x) >|^2 \leqslant \|f(x)\|^2 \|g(x)\|^2$$

QED

Appendix B

Minimising Distance Proof

Let f be an element of a vector space $\mathcal{H}$ and let V_k be a subspace of $\mathcal{H}$. The closest approximation of f that can be obtained by a function $g \in V_k$ can be written as $g = \hat{P}_{V_k} f$, where $\hat{P}_{V_k}$ is the orthogonal projector operator onto V_k

Proof

Since $\hat{P}_{V_k}$ is the orthogonal projector operator onto V_k it should satisfy that

i)$\hat{P}_{V_k} g = g$ for all $g \in V_k$
and
ii) $\hat{P}_{V_k} g^{\perp} = 0$ for all $g^{\perp} \in {V_k}^{\perp}$, where ${V_k}^{\perp}$ indicates the orthogonal complement of V_k in $\mathcal{H}$, i.e. V_k is the subspace of all the vectors in $\mathcal{H}$ that are orthogonal to all the vectors in V_k. In other words: $g \in V_k$ and $g^{\perp} \in {V_k}^{\perp}$ implies $< g, g^{\perp} >= 0$

Let us assume that g is an arbitrary function in V_k and write it as

$$g = g - \hat{P}_{V_k} f + \hat{P}_{V_k} . f$$

If we calculate the distance $\|f - g\|^2$, since $(f - \hat{P}_{V_k} f) \in V_k^{\perp}$ we have:

$$\|f - g\|^2 = \|f - g - \hat{P}_{V_k} f + \hat{P}_{V_k} f\|^2 = \|f - \hat{P}_{V_k} f\|^2 + \|\hat{P}_{V_k} f - g\|^2.$$

Hence, the distance is minimised if $g \equiv \hat{P}_{V_k} f$.

Appendix C

Examples of MEX/C++ Implementation

Test1.m (MATLAB)

```
% This is an example of a script implemented in MATLAB
xValue = 2;
n = 20;
yArray = rand(1,n); % creates a row array holding n number of random values
[result] = arrayProduct(xValue,yArray);
result % an array of size n containing the MATLAB result of y.*x
```

arrayProduct.m (MATLAB)

```
function [z] = arrayProduct(x,y)
% An example of a function arrayProduct implemented in MATLAB
sizeY = numel(y); % obtains the number of values in the input matrix y
z = zeros(1,sizeY); % creates a 1 x sizeY matrix to hold the multiplication results
    for i = 1:sizeY
        z(i) = x * y(i);
    end
end
```

arrayProduct.h (C++)

```
// An example of a function arrayProduct implemented in C++
void arrayProduct(double x, double *y, double *z, int n) {
    int i;
    for (i=0; i<n; i++) {
        z[i] = x *y[i];
    }
}
```

arrayProduct.cpp (MEX)

```
#include "mex.h"
#include "OMP2D_INTER.h"
using namespace std;

/** The MEX file arrayProduct.cpp in the most basic form could look like this.  **/

/** The gateway function **/
void mexFunction( int nlhs, mxArray *plhs[], int nrhs, const mxArray *prhs[]) {

    /** Variable declarations **/
    double multiplier; // input scalar
    double *inMatrix; // 1xN input matrix
    int ncols; // size of matrix
    double *outMatrix; // output matrix

    /* get the value of the scalar input */
```

```
    multiplier = mxGetScalar(prhs[0]);

    /* create a pointer to the real data in the input matrix */
    inMatrix = mxGetPr(prhs[1]);

    /* get dimensions of the input matrix */
    ncols = mxGetN(prhs[1]);

    /* create the output matrix */
    plhs[0] = mxCreateDoubleMatrix(1,ncols,mxREAL);

    /* get a pointer to the real data in the output matrix */
    outMatrix = mxGetPr(plhs[0]);

    /* call the arrayProduct routine in file arrayProduct.h */
    arrayProduct(multiplier,inMatrix,outMatrix,ncols);
}
```

To stabilise the MEX file it is important to validate input parameters passed from the MATLAB Test1.m file to the MEX file. Validation is most often done after declaring the gateway function and before declaring variables. An example of input validation is,

```
/* check for proper number of arguments */
if(nrhs!=2) {
    mexErrMsgTxt("Two inputs required.");
}
if(nlhs!=1) {
    mexErrMsgTxt("One output required.");
}

/* make sure the first input argument is scalar */
if( !mxIsDouble(prhs[0]) || mxIsComplex(prhs[0]) || mxGetNumberOfElements(prhs[0])!=1 ) {
    mexErrMsgTxt("Input multiplier must be a scalar.");
}

/* check that number of rows in second input argument is 1 */
if(mxGetM(prhs[1])!=1) {
    mexErrMsgTxt("Input must be a row vector.");
}
```

Appendix D

Macro Processing Scheme MATLAB Code Snippets

```
%====================
% Validate Macro Size
%====================

sO = size(mIOrig); % size of the image to be approximated
Approx = zeros(sO(1),sO(2)); % store to hold the approximated image

maxSizeX=800;
maxSizeY=800;

if sO(1) < maxSizeX
    maxSizeX = sO(1);
end
if sO(2) < maxSizeY
    maxSizeY = sO(2);
end

if input_PC_Nx > maxSizeX
    fprintf('\n \n !***  The chosen value for macro length Nx | is too large ***!  \n \n')
    possible_choices_Nx = [Lx:Lx:maxSizeX]' % choices for macro length Nx
    return
end
if input_PC_Ny > maxSizeY
    fprintf('\n \n !***  The chosen value for macro length Ny _ is too large ***!  \n \n')
    possible_choices_Nx = [Lx:Lx:maxSizeY]' % choices for macro length Ny
    return
end

if rem(input_PC_Nx,Lx) ~= 0
    fprintf('\n \n !***  Invalid choice for macro length Nx |   ***!  \n \n')
    possible_ choices_ Nx = [Lx:Lx:maxSizeX]'
    return
end
if rem(input_PC_Ny,Ly) ~= 0
    fprintf('\n \n !***  Invalid choice for macro length Ny _ ***!  \n \n')
    possible_choices_Ny = [Ly:Ly:maxSizeY]'
    return
end

%=================================
% Partition Image into Macros
% (Light pink macros in Figure 10)
%=================================

iMLx = input_PC_Nx; % indexing to help iterate through the image via macros
iMLy = input_PC_Ny;

NXBiMLx = floor(sO(1)/iMLx); % NXBiMLx * NYBiMLy = Total number of marco blocks
NYBiMLy = floor(sO(2)/iMLy);
iMLxFinal= NXBiMLx*iMLx;
iMLyFinal= NYBiMLy*iMLy;

for i= iMLx:iMLx:iMLxFinal
    for j= iMLy:iMLy:iMLyFinal
        mI = mIOrig(i-iMLx+1:i,j-iMLy+1:j); % obtain the macro from the image
        ...Process macro using BWOMP2D
        ...Store (light pink) macro result
        ...Store necessary information
    end
end

Management:  Store all (light pink) macro results into 'Approx' and destroy unwanted arrays

%=================================================
% Remainder:  Partition the bottom part of the Image
% (Red and Orange macros in Figure 10)
%=================================================

mIbottomImage = mIOrig( iMLxFinal+1:end, 1:iMLyFinal ); % Obtain the image covered by the red
and orange macros
sizeB=size(mIbottomImage)
s=floor(sizeB/Lx); % next largest macro size = [s(1)*Lx input_PC_Ny]
```

```
% ### (If applicable) Process red macros in Figure 10 ###

if s(1) ~=0;
    for j=iMLy:iMLy:iMLyFinal
        mI = mIbottomImage( 1:s(1)*Lx,j-iMLy+1:j);
        ...Process macro using BWOMP2D
        ...Store (red) macro result
        ...Store necessary information
    end
end

% ### (If applicable) Process orange macros in Figure 10 ###

remx=rem(sizeB(1),Lx);
if remx ~=0
    mIbottomRem = mIbottomImage( (s(1)*Lx)+1:end,:); % Obtain orange macros
    LxB=remx % Block size for orange macros = [LxB Ly], orange macro size= [LxB input_PC_Ny]
    LyB=Ly
    if (LxB<3)
        fprintf('\n \n No need for processing all Orange macros.  Keep all atoms \n \n ')
        ...Store necessary information
    else
        for j=iMLy:iMLy:iMLyFinal % process orange macros
            mI= mIbottomImage( (s(1)*Lx)+1:end,j-iMLy+1:j);
            ...Process macro using BWOMP2D (with block size = [LxB LyB])
            ...Store (orange) macro result
            ...Store necessary information
        end
    end
end

Management:  Store all (red and orange) macro results into 'Approx' and destroy unwanted arrays

Continue to process Green, Sky blue, Dark blue and Purple macros shown in Figure 10
```