DOCTOR OF PHILOSOPHY

Sparse image representation with encryption

James Bowley

2013

Aston University

Sparse Image Representation with Encryption

James Roger Bowley

Doctor Of Philosophy

Aston University

April 2013

Aston University

Sparse Image Representation with Encryption

James Roger Bowley

Doctor Of Philosophy, 2013

Thesis Summary

In this thesis we present an overview of sparse approximations of grey level images. The sparse representations are realized by classic, Matching Pursuit (MP) based, greedy selection strategies. One such technique, termed Orthogonal Matching Pursuit (OMP), is shown to be suitable for producing sparse approximations of images, if they are processed in small blocks. When the blocks are enlarged, the proposed Self Projected Matching Pursuit (SPMP) algorithm, successfully renders equivalent results to OMP. A simple coding algorithm is then proposed to store these sparse approximations. This is shown, under certain conditions, to be competitive with JPEG2000 image compression standard. An application termed image *folding*, which partially secures the approximated images is then proposed. This is extended to produce a self contained *folded* image, containing all the information required to perform image recovery. Finally a modified OMP selection technique is applied to produce sparse approximations of Red Green Blue (RGB) images. These RGB approximations are then *folded* with the self contained approach.

Keywords: Orthogonal Matching Pursuit, Sparse Approximations, Image Folding, Greedy Algorithms

Acknowledgements

I acknowledge my supervisor Dr. Laura Rebollo-Neira.

Firstly I would like to extend my gratitude to all my friends from university, Dr Erik Casagrande, Donnelle Christian, Dr. Thomas Drew, Dr. Dan Harvey, Matthew Hird, Dr. Richard Jones, Dr. George Lychnos, Antonio Oliveira, Dr. Benjamin Tocher and of course Sofia Topakas. Special thanks go to Dr. Jan Duracz, Dr. Harry Goldingay and Dr. Ryszard Macoil for their consultations and advice regarding this work.

Secondly I would like to thank both Colin Charlwood and Sultan Hussain, for offering me employment, enabling me to financially support myself until the completion of my PhD.

It gives me great pleasure to thank my mother Judy for making this all possible, through supporting my education, and encouraging me to attend University.

Lastly I would like to dedicate this thesis to my Angie, who has lost more nights sleep than anyone as a result. She has been there for me every step of the way, I did this for her!

Contents

List of Tables

List of Figures

Nomenclature

$\{\cdot\}$	set of elements
$\{\emptyset\}$	the empty set
$\mathcal{A}\backslash\mathcal{B}$	relative complement of $\mathcal{B}$ in $\mathcal{A}$
$(\cdot)^\dagger$	Moore-Penrose pseudo inverse
$(\cdot)^T$	transpose of a vector or matrix
$\tilde{(\cdot)}$	indicates that distortion may have been introduced
$\lceil x \rceil$	the smallest integer not less than x
$\lfloor x \rfloor$	the largest integer not greater than x
$\langle \cdot \rangle$	inner product
$\langle \cdot \rangle_F$	Frobenius inner product defined in Appendix A.3.3
$\lvert \cdot \rvert$	absolute value
$\lVert \cdot \rVert_0$	l_0 norm equal to the count of the non zeros elements of a vector
$\lVert \cdot \rVert_1$	l_1 norm equal to the sum of the absolute value of the elements of a vector
$\lVert \cdot \rVert$	Euclidean or l_2 norm defined in Appendix A.7
$\lVert \cdot \rVert_F$	Frobenius norm defined in Appendix A.3.4
$\otimes$	Kronecker product defined in Appendix A.4
$\widehat{\mathbf{Vec}}(\cdot)$	relabels elements of a matrix to become a vector by applying the convention in Appendix A.2.2
$\widehat{\mathbf{Mat}}(\cdot, N, N)$	relabels elements of a vector to become an $N \times N$ matrix by applying the convention in Appendix A.2.1

$\widehat{\mathbf{Flat}}(\cdot)$	combines a set of vectors into a single vector with the convention described in Appendix A.2.3
$\widehat{\mathbf{Set}}(\cdot)$	splits a large vector into several smaller vectors with the convention described in Appendix A.2.4
$\text{sign}(\cdot)$	sign function defined in equation (2.25)
$\widehat{\mathbf{Orth}}(\cdot)$	orthonormalizes a set of vectors, performed experimentally with the MATLAB `qr()` routine
$\widehat{\mathbf{FQuant}}(\cdot, \triangle)$	forward quantization operation defined in (2.24)
$\widehat{\mathbf{RQuant}}(\cdot, \triangle)$	inverse quantization operation defined in (2.27)
$P(\cdot)$	calculate the number of permutations with equation (3.24)
$C(\cdot)$	calculate the number of combinations with equation (3.25)
$\widehat{\mathbf{UInt}}(\cdot)$	converts a vector of u bits to an unsigned u bit integer by applying equation (4.13)
$\widehat{\mathbf{Bin}}(\cdot, u)$	converts an unsigned integer to a u bit representation
$\text{Spark}(\cdot)$	Spark defined in Definition 2
$\hat{P}_{\mathbb{V}_K}$	orthogonal projection onto $\mathbb{V}_K$
$\hat{P}_{\mathbb{V}_{K_q}}$	orthogonal projection onto $\mathbb{V}_{K_q}$
$\hat{P}_{\ddot{\mathbf{D}}}$	orthogonal projection onto the span of the columns of $\ddot{\mathbf{D}}$
$\hat{P}_{\S}$	orthogonal projection onto the space §
$\mathcal{O}$	big $\mathcal{O}$ indicating the order of the complexity
$f(\cdot)$	discrete value function
$g(\cdot)$	continuous function in 2D representing an intensity image
$p(\mathbf{t}(l))$	probability of the occurrence of $\mathbf{t}(l)$ in $\mathbf{t}$
m_1	method for storing the separable indices with the "pixel" scheme
m_2	method for storing the row and column indices with the "pixel" scheme
π	pi

$\triangle$	quantization step size
$\triangle_1$	quantization step size applied to $\mathbf{I}^{f_1}$
$\triangle_2$	quantization step size applied to $\mathbf{I}^{f_2}$
Err	error in recovery of $\tilde{\mathbf{e}}$ defined in equation (3.17)
$\overline{\text{Err}}$	percentage mean Err over a set of N_I images
ϵ	perturbation applied to $\tilde{\mathbf{G}}$
ϵ^1	perturbation applied to $\tilde{\mathbf{G}}(r_r^1, r_c^1)$
ϵ_R^1	perturbation applied to $\tilde{\mathbf{G}}(r_r^1, r_c^1)$ at the recovery stage
ϵ^2	perturbation applied to $\tilde{\mathbf{G}}(r_r^2, r_c^2)$
ϵ_R^2	perturbation applied to $\tilde{\mathbf{G}}(r_r^2, r_c^2)$ at the recovery stage
ϵ_R	perturbation used at the recovery stage
ϵ_{min}	minimum perturbation which is guaranteed to perturb $\tilde{\mathbf{G}}$
H	entropy of an image array
$max_{p,1}$	maximum pixel intensity in $\mathbf{I}^{f_1}$
$min_{p,1}$	minimum pixel intensity in $\mathbf{I}^{f_1}$
$min_{p,2}$	minimum pixel intensity in $\mathbf{I}^{f_2}$
PSNR^a	PSNR resulting from performing an approximation
$\overline{\delta\text{PSNR}}$	average δPSNR over N_I images
δPSNR	percentage difference between the PSNR^a and the PSNR^f defined in equation (3.18)
PSNR^f	PSNR resulting from image *folding*
s_{SR}	standard deviation of the SR over a set of test images
τ	ILS-DLA convergence parameter
$\bar{t}$	mean processing time over a set of test images
ρ	error tolerance

x	continuous variable, where required
$\bar{x}_{CR}$	mean CR over a set of test images
$\bar{x}_{SR}$	mean SR over a set of test images
y	continuous variable, where required
a	redundancy factor of DC dictionary
h	discrete index, initialized when required
H_1	number of host blocks
H_2	number of "ad hoc" host blocks
i_c	complementary index
i_c^c	complementary column index
i_c^r	complementary row index
i	discrete index, initialized when required
J	number of iterations of the ILS-DLA
j	length of support
k	current iteration or index to K
K	number of atoms chosen
l	discrete index, initialized when required
L	number of levels which an intensity pixel can have
$\ddot{L}$	number of levels in a given intensity image
M	number of columns in a redundant matrix
m	index, initialized when required
M_c	number of column vectors in column dictionary
m_c	index of column $\mathbf{d}^c$ in $\mathbf{D}^c$
M_r	number of column vectors in row dictionary
m_r	index of column $\mathbf{d}^r$ in $\mathbf{D}^r$

N	number of elements in a vector or rows and columns in a square matrix
n	index, initialized when required
N_b	number of bits required to store a compressed representation
$N_b^{f_1}$	number of bits in $\mathbf{b}^{f_1}$
$N_b^{f_2,\triangle}$	number of bits in $\mathbf{b}^{f_2,\triangle}$
$N_b^{f_2}$	number of bits required to represent $\mathbf{I}^{f_2}$ produced by the "pixel" scheme
$N_b^{m_1}$	number of bits in $\mathbf{b}^{m_1}$
$N_b^{m_2}$	number of bits in $\mathbf{b}^{m_2}$
N_c	number of columns
n_c	index of column
N_c^1	number of columns in $\mathbf{I}^{f_1,\triangle}$
N_c^2	number of columns in $\mathbf{I}^{f_2,\triangle}$
N_c^3	number of columns in $\mathbf{I}^{f_2}$ when the "pixel" scheme is applied with m_1
N_e	dimension of the space $\mathbb{V}^{\perp}$
N_h	number of elements in $\mathbf{h}$
N_I	number of test images
N_K	total number of *folded* coefficients
N_{-1}	number of -1 entries in $\mathbf{T}(:,1)$
N_p	number of pixels in the original image $\mathbf{I}$
N_1	number of padding elements included in $\mathbf{I}^{f_1}$
N_2	number of padding elements included in $\mathbf{I}^{f_2}$
N_ϵ	number of perturbations ϵ applied to $\tilde{\mathbf{G}}$
$\#_{pert}$	number of perturbation sizes which can be applied to $\tilde{\mathbf{G}}$
N_r	number of rows
n_r	index of row

N_r^1	number of rows in $\mathbf{I}^{f_1,\triangle}$
N_r^2	number of rows in $\mathbf{I}^{f_2,\triangle}$
N_r^3	number of rows in $\mathbf{I}^{f_2}$ when the "pixel" scheme is applied with m_1
p	projection step for SPMP
p^k	number of vectors chosen at each MP stage of SPMP
Q	number of $N \times N$ image samples required to train a TS dictionary
q	index of image block
Q^N	number of $N \times N$ blocks of pixels $\mathbf{I}$ can be divided into
r	reserved symbol which is not a valid index
r_c^1	random column within $\tilde{\mathbf{G}}$
r_c^2	random column within $\tilde{\mathbf{G}}$
r_r^1	random row within $\tilde{\mathbf{G}}$
r_r^2	random row within $\tilde{\mathbf{G}}$
s_2	seed for initializing pseudorandom generator required to produce $\mathbf{U}$
s_3	seed for initializing pseudorandom generator required to produce $\mathbf{m}$
s_1	seed for initializing pseudorandom generator required to produce $\mathbf{Z}$
s_{max}	maximum number of seed which can be used to initialize a PRNG or CSPRNG
s_{max}^1	maximum number values s_1 can take on
s_{max}^2	maximum number values s_2 can take on
u	number of bpp or bit depth
u_1	number of bits which each pixel in $\mathbf{I}^{f_1}$ is quantized to
u_2	number of bits which each pixel in $\mathbf{I}^{f_2}$ is quantized to
z	index of the colour plane, $z = 1, 2, 3$ correspond respectively to Red, Green and Blue

$\mathbf{A}$	matrix of inner products
$\mathbf{B}$	biorthogonal matrices
$\mathbf{C}$	temporary matrix required in calculation of the biorthogonal matrices $\mathbf{B}$
$\mathbf{D}^{c,(i)}$	column dictionary at iteration i of the ILS-DLA
$\mathbf{D}^{(K)}$	matrix $\mathbf{D}$ at iteration K of Optimized Orthogonal Matching Pursuit (OOMP)
$\mathbf{D}^{r,(i)}$	row dictionary at iteration i of the ILS-DLA
$\ddot{\mathbf{D}}$	matrix constructed with columns chosen from $\mathbf{D}$
$\mathbf{D}$	full rank, redundant matrix who's columns are the vectors $\mathbf{d}$
$\mathbf{D}_1^c$	RDC dictionary
$\mathbf{D}_2^c$	matrix who's N columns form a basis for the space $\mathbb{R}^N$
$\mathbf{D}^c$	column dictionary
$\mathbf{D}_3^c$	RDBS dictionary
$\mathbf{D}_4^c$	RDU column dictionary
$\mathbf{D}_5^c$	RDW column dictionary
$\mathbf{D}_6^c$	RR column dictionary
$\mathbf{D}_9^c$	B-Spline dictionary constructed with equation (2.22)
$\mathbf{D}^r$	row dictionary
$\mathbf{D}_1^{c,1}$	RDC column dictionary, $\mathbf{D}_1^{c,1} = \mathbf{D}_1^c$ (RDC)
$\mathbf{D}_1^{r,1}$	RDC row dictionary, $\mathbf{D}_1^{r,1} = \mathbf{D}_1^c$ (RDC)
$\mathbf{D}_2^{c,1}$	combined RDC and RDBS column dictionary, $\mathbf{D}_2^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_3^c]$ (RDC-RDBS$_1$).
$\mathbf{D}_2^{r,1}$	combined RDC and RDBS row dictionary, $\mathbf{D}_2^{r,1} = [\mathbf{D}_1^c, \mathbf{D}_3^c]$
$\mathbf{D}_3^{c,1}$	combined RDC, Euclidean Basis and RDW, column dictionary, $\mathbf{D}_3^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_5^c]$ (RDC-RDW)

$\mathbf{D}_4^{c,1}$ combined RDC, Euclidean Basis and RR, column dictionary, $\mathbf{D}_4^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_6^c]$ (RDC-RR)

$\mathbf{D}_5^{c,1}$ combined RDC, Euclidean Basis, RDU and the smaller B-spline $\mathbf{D}_9^c$, column dictionary, $\mathbf{D}_5^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_3^c, \mathbf{D}_9^c]$ (RDC-RDBS$_2$)

$\mathbf{D}_5^{r,1}$ combined RDC, Euclidean Basis, RDU and the smaller B-spline $\mathbf{D}_9^c$, row dictionary, $\mathbf{D}_5^{r,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_3^c, \mathbf{D}_9^c]$ (RDC-RDBS$_2$)

$\mathbf{D}_6^{c,1}$ TS column dictionary trained on 10 astronomical images

$\mathbf{D}_6^{r,1}$ TS row dictionary trained on 10 astronomical images

$\mathbf{D}_7^{c,1}$ TS column dictionary trained on 10 natural images

$\mathbf{D}_7^{r,1}$ TS row dictionary trained on 10 natural images

$\mathbf{D}_8^{c,1}$ combined RDC and Euclidean Basis, column dictionary, $\mathbf{D}_8^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c]$

$\mathbf{D}_8^{r,1}$ combined RDC and Euclidean Basis, row dictionary, $\mathbf{D}_8^{r,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c]$

$\mathbf{D}_1^{c,2}$ RDC column dictionary, $\mathbf{D}_1^{c,2} = \mathbf{D}_1^c$ (RDC) from Chapter 4

$\mathbf{D}_2^{c,2}$ column dictionary $\mathbf{D}_2^{c,2} = \mathbf{D}_8^{c,1} = [\mathbf{D}_1^{c,2}, \mathbf{D}_2^c]$

$\mathbf{D}_3^{c,2}$ column dictionary $\mathbf{D}_3^{c,2} = [\mathbf{D}_2^{c,2}, \{U_2(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_2}]$

$\mathbf{D}_4^{c,2}$ column dictionary $\mathbf{D}_4^{c,2} = [\mathbf{D}_3^{c,2}, \{b_i Y_2^2(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_2}]$

$\mathbf{D}_5^{c,2}$ column dictionary $\mathbf{D}_5^{c,2} = [\mathbf{D}_4^{c,2}, \{U_4(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_4}]$

$\mathbf{D}_6^{c,2}$ column dictionary $\mathbf{D}_6^{c,2} = [\mathbf{D}_5^{c,2}, \{b_i Y_2^3; (n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_3}]$

$\mathbf{D}_7^{c,2}$ column dictionary $\mathbf{D}_7^{c,2} = [\mathbf{D}_6^{c,2}, \{U_6(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_6}]$

$\mathbf{D}_8^{c,2}$ column dictionary $\mathbf{D}_8^{c,2} = \mathbf{D}_5^{c,1} = [\mathbf{D}_7^{c,2}, \{b_i Y_2^4(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_4}]$, (RDC-RDBS$_2$

$\mathbf{E}$ embedded matrix in $\mathbb{V}^{\perp}$

$\mathbf{F}$ matrix who's columns are $\widehat{\mathbf{Vec}}(\mathbf{S})$

$\mathbf{G}$ matrix who's null space is $\mathbb{V}^{\perp}$

$\tilde{\mathbf{G}}$ perturbed version of $\mathbf{G}$

$\mathbf{H}$ embedded matrix in $\mathbb{W}^{\perp}$

$\tilde{\mathbf{H}}$	recovered matrix $\mathbf{H}$
$\mathbf{I}$	matrix representing a digital grey level intensity image
$\mathbf{I}^{f_1}$	*folded* image containing the hidden coefficients
$\mathbf{I}^{f_2}$	*folded* image containing hidden indices
$\mathbf{I}_z^{f_2}$	*folded* image containing the indices $\mathbf{l}_z$
$\mathbf{I}^{f_1,\triangle}$	coded version of $\mathbf{I}^{f_1}$ where each pixel is represented with u bits
$\mathbf{I}^{f_2,\triangle}$	coded version of $\mathbf{I}^{f_2}$ where each pixel is represented with u bits
$\mathbf{I}_z^{f_1,\triangle}$	quantized version of $\mathbf{I}_z^{f_1}$
$\tilde{\mathbf{I}}^{f_1}$	quantized representation of $\mathbf{I}^{f_1}$
$\tilde{\mathbf{I}}^{f_2}$	quantized representation of $\mathbf{I}^{f_2}$
$\mathbf{I}^{f}$	self contained *folded* image
$\mathbf{I}_z^{f}$	self contained *folded* image containing all the information to recover $\mathbf{I}_z^K$
$\mathbf{I}_z^{f_1}$	*folded* image containing the coefficients $\mathbf{c}_z$
$\mathbf{I}_z^K$	K sparse approximation of $\mathbf{I}_z$
$\mathbf{I}_z$	one of the three matrices representing an RGB intensity image
$\mathbf{I}^K$	K sparse representation of $\mathbf{I}$
$\ddot{\mathbf{J}}$	"ad hoc" intensity array
$\mathbf{J}$	normalized to unity matrix
$\mathbf{K}$	sparse matrix required by the ILS-DLA
$\mathbf{M}$	matrix, initialized when required
$\mathbf{N}$	matrix, initialized when required
$\mathbf{R}$	matrix representing the residual
$\mathbf{R}^K$	residual of K sparse representation $\mathbf{I}^K$
$\mathbf{R}_z^K$	Residual of K sparse representation $\mathbf{I}_z^K$
$\mathbf{S}$	large separable matrix

$\mathbf{S}^{\perp}$	matrix forming part of the basis for $\mathbb{V}^{\perp}$
$\mathbf{T}$	table containing all the information required to recover an image approximation
$\mathbf{U}$	pseudorandom matrix required in the random security scheme
$\mathbf{W}^{\perp}$	matrix forming part of the orthonormal basis for $\mathbb{W}^{\perp}$
$\mathbf{X}$	image samples for training TS dictionaries
$\mathbf{Y}^{\perp}$	temporary matrix required to construct $\mathbf{S}^{\perp}$ in the random security scheme
$\mathbf{Z}$	pseudorandom matrix required in the random security scheme
$\mathbf{Z}^{\perp}$	matrices in $\mathbb{V}^{\perp}$
$\mathcal{D}$	set of elements forming a dictionary
$\mathcal{E}$	set of perturbations $\{10^{\imath}\}_{\imath=-20}^{1}$
$\mathcal{E}_1$	set of perturbations $\{10^{\imath}\}_{\imath=-16}^{1}$
$\mathcal{E}_2$	set of perturbations $\{10^{\imath}\}_{\imath=-16}^{-3}$
$\mathcal{G}$	set of indices
$\mathcal{L}$	set of indices
$\mathbb{N}$	set of natural numbers
$\mathbb{R}$	set of real numbers
$\mathbb{S}$	space spanned by the vectors in $\mathcal{S}$
$\mathcal{S}$	set of vectors chosen by SPMP
$\mathcal{V}$	set of column vectors
$\mathbf{E}_q$	embedded matrix in $\mathbb{V}_q^{\perp}$
$\mathbb{V}_K$	space spanned by K elements of a dictionary
$\mathbb{V}^{\perp}$	orthogonal complement of $\mathbb{V}_K$ in $\mathbb{R}^{N_r \times N_c}$
$\mathbb{W}$	space spanned by the single matrix $\mathbf{J} \in \mathbb{R}^{N \times N}$
$\mathbb{W}^{\perp}$	orthogonal compliment of $\mathbb{W}$ in $\mathbb{R}^{N \times N}$

$\mathbf{a}$	vector of inner products
$\mathbf{b}$	bit stream (vector containing only 0's and 1's)
$\mathbf{b}^{f_1}$	vector of bits representing the pixels of $\mathbf{I}^{f_1,\triangle}$
$\mathbf{b}^{f_2,\triangle}$	vector of bits representing the pixels of $\mathbf{I}^{f_2,\triangle}$
$\mathbf{b}^{m_1}$	vector of bits representing the indices $\mathbf{l}_q, q = 1, \ldots, Q^N$
$\bar{\mathbf{b}}^{m_1}$	elements of $\mathbf{b}^{m_1}$ who's order is determined by the elements in $\mathbf{m}$
$\mathbf{b}^{m_2}$	vector of bits representing the index pairs $(\mathbf{l}_q^c, \mathbf{l}_q^r), q = 1, \ldots, Q^N$
$\mathbf{c}$	vector of coefficients
$\mathbf{c}^a$	coefficients sorted to correspond to the correct index in $\mathbf{l}^a$
$\mathbf{l}^{(i)}$	vector of coefficients for approximation of $\mathbf{X}$ at iteration i of the ILS-DLA
$\mathbf{c}^{\triangle f}$	vector containing the vectors of coefficients $\mathbf{c}_q, q = 1, \ldots, Q^N$
$\mathbf{c}_1^{\triangle f}$	vector containing the absolute value of each entry in $\mathbf{c}^{\triangle f}$
$\mathbf{c}_2^{\triangle f}$	vector containing the result of the sign() function on each entry in $\mathbf{c}^{\triangle f}$
$\mathbf{c}_z$	vector of coefficients for representing $\mathbf{I}_z^K$
$\mathbf{d}$	column vector producing sparse decomposition
$\mathbf{d}^c$	column of the column dictionary $\mathbf{D}^c$
$\mathbf{d}^r$	column of the row dictionary $\mathbf{D}^r$
$\mathbf{e}$	vector of numbers hidden in $\mathbf{E}$
$\mathbf{f}$	column vector to be approximated
$\mathbf{f}^K$	K sparse representation of $\mathbf{f}$
$\mathbf{f}_2^{\triangle}$	vector of quantization indices representing the pixels of $\mathbf{I}^{f_2}$
$\mathbf{f}_1^{\triangle}$	vector of quantization indices representing the pixels of $\mathbf{I}^{f_1,\triangle}$
$\mathbf{f}^k$	representation of $\mathbf{f}$ at iteration k
$\mathbf{g}$	temporary vector, initialized when required
$\mathbf{h}$	vector of numbers hidden in $\mathbb{W}^{\perp}$

$\mathbf{l}$	vector of indices
$\mathbf{l}^a$	indices in $\mathbf{l}$ sorted in ascending numerical order
$\mathbf{l}^c$	vector containing chosen columns of $\mathbf{D}^c$
$\mathbf{l}^{c,a}$	indices in $\mathbf{l}^c$ sorted in ascending numerical order
$\mathbf{l}^{c,d}$	vector containing the difference between each element in $\mathbf{l}^{c,a}$
$\mathbf{l}^{c,(i)}$	vector of column indices for approximation of $\mathbf{X}$ at iteration i ILS-DLA
$\mathbf{l}^d$	vector containing the difference between each element in $\mathbf{l}^a$
$\mathbf{l}^f$	vector containing the vectors of indices $\mathbf{l}_q, q = 1, \ldots, Q^N$
$\mathbf{l}^i$	original position of each element of $\mathbf{l}^a$ in $\mathbf{l}$
$\mathbf{l}^r$	vector containing chosen columns of $\mathbf{D}^r$
$\mathbf{l}^{r,a}$	indices in $\mathbf{l}^r$ sorted to correspond to the correct index in $\mathbf{l}^{c,a}$
$\mathbf{l}^{r,(i)}$	vector of row indices for approximation of $\mathbf{X}$ at iteration i ILS-DLA
$\mathbf{l}_z$	vector of indices for representing $\mathbf{I}_z^K$
$\mathbf{m}$	pseudorandom vector required in the "pixel" method
$\mathbf{n}$	column vector, initialized when required
$\mathbf{n}_e$	vector containing $N_{e,q}, q = 1, \ldots, H_1$
$\mathbf{n}_h$	vector where each element is N_h
$\mathbf{n}_K$	vector containing the number of coefficients *folded* in each block
$\mathbf{n}_{\tilde{K}}$	vector containing the number of coefficients *folded* in each block, recovered from $\mathbf{I}^{f_2}$
$\mathbf{p}_1$	vector of padding included in $\mathbf{I}^{f_1}$
$\mathbf{p}_2$	vector of padding included in $\mathbf{I}^{f_2}$
$\mathbf{q}$	vector containing the block index corresponding to each element in $\mathbf{c}^{\triangle f}$
$\mathbf{r}$	residual
$\mathbf{r}^K$	residual of K sparse representation of $\mathbf{f}^K$

$\mathbf{r}^k$	residual at iteration k
$\mathbf{r}^\perp$	residue of the MP approximation having no component in $\mathcal{S}$
$\mathbf{s}_1$	vector of initialization seeds s_1
$\mathbf{s}^\perp$	element of the basis for the null space of $\mathbf{F}$
$\mathbf{s}_2$	vector of initialization seeds s_2
$\mathbf{t}$	temporary vector, initialized when required
$\mathbf{u}$	column vector, initialized when required
$\mathbf{v}$	column vector, initialized when required
$\mathbf{v}^\triangle$	vector of quantization indices
$\mathbf{w}$	vector of normalization factors

Acronyms

1D one dimension.

2D two dimensions.

AES Advanced Encryption Standard.

BOOMP Backward Optimized Orthogonal Matching Pursuit.

BP Basis Pursuit.

BPDN Basis Pursuit Denoising.

bpp bits per pixel.

CDF9/7 Cohen Daubechies Feauveau 9/7.

CR Compression Ratio.

CSPRNG Cryptographically Secure Pseudorandom Number Generator.

DC Discrete Cosine.

DCT Discrete Cosine Transform.

DES Data Encryption Standard.

FOCUSS FOcal Underdetermined System Solver.

HSI Hue Saturation Intensity.

ILS-DLA Iterative Least Squares Dictionary Learning Algorithm.

IRLS Iterative Reweighted Least Squares.

LASSO Least Absolute Shrinkage and Selection Operator.

MOF Method Of Frames.

MP Matching Pursuit.

MP2D Matching Pursuit in 2D.

MSE Mean Squared Error.

NTSC National Television System Committee.

OLS Orthogonal Least Squares.

OMP Orthogonal Matching Pursuit.

OMP2D Orthogonal Matching Pursuit in 2D.

OOMP Optimized Orthogonal Matching Pursuit.

ORMP Order Recursive Matching Pursuit.

PRNG Pseudorandom Number Generator.

PSNR Peak Signal to Noise Ratio.

RDBS Redundant Discrete B-Spline.

RDC Redundant Discrete Cosine.

RDU Redundant Discrete Uniform.

RDW Redundant Discrete Wavelet.

RGB Red Green Blue.

RLS-DLA Recursive Least Squares Dictionary Learning Algorithm.

RR Redundant Random.

RSA Ron Rivest, Adi Shamir and Leonard Adleman.

SPIHT Set Partitioning in Hierarchical Trees.

SPMP Self Projected Matching Pursuit.

SPMP2D$_{10}$ Self Projected Matching Pursuit in 2D with step size $p = 10$.

SPMP2D$_1$ Self Projected Matching Pursuit in 2D.

SR Sparsity Ratio.

SVD Singular Value Decomposition.

TS Trained Separable.

TV television.

1 Introduction

1.1 Motivation

In todays digital age, the storage paradigm has shifted. Digital files, are no longer kept at home or in the office, now everything is virtual, instead being kept on the cloud. All types of digital media including documents, books, photo's and videos, must now be stored online, allowing users access, anywhere, and any time. Consequently, there is a corresponding need for implementations and technologies, which, utilizing the current infrastructure, offer solutions satisfying this ever increasing demand.

Almost simultaneously the requirement for higher and higher quality digital media is everywhere. Ranging from the proliferation of mobile phones, both taking and displaying photos, with equivalent resolution to dedicated digital cameras. All the was to the provision of high definition television services. With users constantly requesting higher quality, on demand, the need for new approaches to compression, is now more important than ever.

The shift has been partly fueled by businesses adapting to this digital revolution. By embarking into additional markets, a fresh demand for digital content which did not

previously exist, has been created. The provision of online television services is an example of one such emerging market, bringing with it, its own set of requirements. Services such as these often operate by offering a lower quality or partially secured version for free, aiming to entice the user into subscribing to a paid service. Due to the size of digital media, the priority is satisfying the demand, with security being less important. A direct result of this business model is more widespread interest into partial or selective security schemes.

1.1.1 Overview

The focus of this thesis is on grey level digital images, and the application of partial image security to their sparse representations. An overview of the main content of each Chapter is given below:

- This Chapter begins with an introduction to digital images which are the focus of the applications in this thesis. A discussion is then presented regarding sparse representations and approximations. The emphasis is placed on greedy selection algorithms which are used throughout this work. The next Section provides a brief overview of the current situation regarding image security. Finally the main contributions of this thesis are put forward.

- Chapter 2 explores sparse representations of grey level images produced by greedy algorithms. The first Section introduces an alternative to the original MP algorithm, termed SPMP. The following Sections then adapt the SPMP approach to work in 2D, before evaluating its performance with respect other greedy selection approaches. The sparsity of image approximations with alternative dictionary constructions is then examined. Finally a sparse image coding scheme is designed. Its performance over two sets of images is then compared, with implementations of both the JPEG and JPEG2000 image compression standards.

- Chapter 3 describes and analyses a method for hiding information in the null space created from a sparse approximation of an image. Two different approaches for securing this hidden information are then proposed, termed the SVD and random methods. The final part of the Chapter is devoted to determining the size of the *keyspace* for each method, by running a number of different simulations.

- Chapter 4 expands the image *folding* procedure described in Chapter 3 to make it self contained. That is a prescription is given for creating a *folded* image which

contains all the information required to recover the original sparse approximation. The additional information required to make the images self contained is then secured in two ways, referred to as the "ad hoc" and "pixel" approaches. The suitability of each of these methods is then assessed with respect to both the SR and CR of their representations.

The remainder of the Chapter is devoted extending the procedure to RGB colour images . A single and multi channel approach to finding sparse approximations of colour images is advanced. Both approaches are then implemented before the *folding* stage. The CR produced by these different implementations is then compared, before finally giving an example of the application of both schemes to a colour image.

- Chapter 5 offers conclusions and avenues of further investigation.

1.2 Contributions

The following work investigates the application of sparse approximations to both, digital image compression, and security. The author considers the main contributions of this to be:

- Examination of two theoretically equivalent algorithms (OMP and SPMP), applied to produce sparse representations of images (Chapter 2).
- Proposal of a strategy for storing the atomic decomposition of images, which, produces a bit stream of comparable size to that of JPEG2000 (Chapter 2).
- Proposal of a novel scheme for partially securing grey, and RGB colour, digital images, termed image *folding* (Chapters 3 and 4) .

Other contributions include:

- Implementation in C++ of all greedy algorithms considered, enabling the execution of experiments over large sets of test images.
- Evaluation of alternative pursuit strategies, MP, OMP, SPMP and OOMP with reference to the sparsity of the representations they produce (Chapter 2).
- Investigation into the effect of dictionary choice and block partition size on sparse representations and its application to image *folding* (Chapters 2-4) .

Chapters 2, 3 and 4 extend and complete work realized initially in collaboration and are supported by published papers [2–4].

1.3 Digital Images

This Section provides a short introduction to digital images and there representations. This discussion will be useful for understanding the material in remaining Chapters.

1.3.1 Grey Level Intensity Images

A monochrome grey level image can be considered to be a 2D continuous function $g(x, y)$ where x and y are the spatial coordinates, and the value of the function at any pair of coordinates (x, y) is the intensity of the image at that point [5]. A digital version of this image, or a digital image, is produced by restricting both x and y, and $g(x, y)$, to be a discrete set of values [6]. A convenient way of representing this digital version is as a raster image array $\mathbf{I}$ [7], containing N_r rows and N_c columns, where each picture element or pixel [8] $\mathbf{I}(n_r, n_c), n_r = 1, \ldots, N_r, n_c = 1, \ldots, N_c$ contains the discrete value representing the intensity of the image at that point.

To convert an image to a digital version the values of N_r, N_c and the number of intensity levels L have to be chosen. The image can then be sampled at each of the $N_r \times N_c$ points, with the intensity at each of these points being quantized to one of the L grey levels. Both N_r and N_c can take on any values as long as they are positive integers. Because of storage and quantization hardware, the value of L is usually an integer power or 2 [9], that is

$$L = 2^u,$$

where u is the bit depth or number of bits required to represent L different levels.

The range of grey levels is usually $[0, 1, \ldots, L-1]$, with 0 represents black and $L-1$ representing white. A common choice for is $u = 8$ bits, resulting in $L = 256$ grey levels $[0, 1, \ldots, 255]$ [5]. Larger values of u are also commonly used for specific tasks in fields such as astronomy and medical imaging [7], where $u = 8$ is not sufficient. An image requiring u bits to represent each pixel is commonly referred to as a u bit image [9]. Therefore the image described above, requiring 8 bits per pixel (bpp), can be referred to as an 8 bit grey level image.

1.3.2 Colour Images

Colour representation is important because whilst we can only distinguish between a few dozen grey levels [9, 10], we can distinguish between thousands of different colours.

Representation of colour in digital images is performed with a colour model which defines the colours in a standard way [9]. There are several colour models each designed

for a specific purpose. Two examples of commonly used colour models are described below, the first termed the RGB model is an additive model suitable for implementation in hardware, the second termed the Hue Saturation Intensity (HSI) model, is designed to correspond to the way humans perceive and interpret colour [11].

- The RGB model represents each colour as the addition of three primary components, representing the contribution of red green and blue light. This is the most common model used [11] and is a convenient representation for use with display hardware where colours are reproduced by emitting different levels of red green and blue light to generate a single colour [9].

- The RGB colour model is particularly well suited for use in hardware and image processing however it is not intuitive to humans, that is a person would not describe a colour as the sum of its red green and blue components. To enable humans to describe and manipulate colours in a more natural way the HSI colour model was developed. In this model the hue, saturation and intensity components can be manipulated to achieve the desired colour, where hue can be thought of as the dominant perceived colour, saturation is the colourfulness or relative purity of the colour and intensity is the brightness [9, 12].

Colour images will be considered in Chapter 4 where they will be represented with the RGB model described above. This is achieved by three image arrays $\mathbf{I}_z, z = 1, \ldots, 3$, which respectively will store a numeric representation of the intensity of the Red Green and Blue components of each image.

The images considered in Chapter 4 will all use $u = 8$ bits to represent the intensity of each colour component. This will results in $L = 256$ intensity values for each colour and $(2^8)^3 = 16,777,216$ different colours. This is a typical choice for the number of colours [6] and referred to as a true colour [9] representation. The bit depth of each pixel is now 24, hence a true colour image will be referred to as a 24 bit image.

It is sometimes desirable to convert RGB images to grey level intensity images. This can be achieved in several different ways, the convention adopted here is to use the standard conversion from RGB to luminance used in the standard television (TV) broadcasting system, the National Television System Committee (NTSC) standard [11]. This is used in broadcasting to separate the grey scale information (luminance) from the hue and saturation and is the standard way of converting RGB images to grey level used in MATLAB [13].

The conversion from an RGB image array $\mathbf{I}_z, z = 1, \ldots, 3$ to a grey level image array

$\mathbf{I}$ will be performed by applying the weighted sum of the colour components shown below

$$\mathbf{I} = 0.2989\mathbf{I}_1 + 0.587\mathbf{I}_2 + 0.114\mathbf{I}_3. \tag{1.1}$$

1.4 Digital Image Compression

The digital image arrays mentioned above are generally very large, and redundant containing three main types of redundancy [6, 9]:

1) Coding redundancy, which occurs when the representation used for each pixel is not the most efficient [6]. The definition of the entropy of an image $\mathbf{I}$ [9, 11, 14], containing $\ddot{L}$ different intensity levels $\mathbf{t}(l), l = 1, \ldots, \ddot{L}$ is

$$H = -\sum_{l=0}^{\ddot{L}} p(\mathbf{t}(l)) \log_2(\mathbf{t}(l))), \tag{1.2}$$

where $p(\mathbf{t}(l))$ is the probability of the occurrence of the symbol $\mathbf{t}(l)$ in $\mathbf{t}$.

The logarithm base 2 in equation (1.2) means that H represents the average information in bits of the pixels in the image $\mathbf{I}$. This is a lower bound on the average number of bits which can be used to represent the pixels in the image $\mathbf{I}$ [9]. Coding redundancy is present when the average number of bits used to represent each pixel in $\mathbf{I}$ is greater than H.

2) Spatial redundancy. When neighbouring pixels in an image have the same or similar values [9].

3) Psycho visual redundancy. Where an image contains information which cannot be perceived by an observer [6].

There are two approaches to removing this redundancy, which are described in the next two Sections. Both approaches can be used on their own or in combination with each other.

Lossless Compression

This type of compression is used when image quality is the main priority and as the name suggests will result in a compressed version of an image which when decompressed will be identical to the original [11]. One approach to this is to employ a statistical coding method, for example the Huffman or arithmetic coding algorithms [14], to reduce the coding redundancy mentioned above. There are many such algorithms all of which when applied to the pixels of an image $\mathbf{I}$, aim to produce a representation with an average of close to H bits per pixel.

Lossy Compression

This compression methodology produces an image which is an approximation of the original [11] and relies on the human visual system being less sensitive to loss of particular types of information [6]. Quantization, which is a mapping of many input values to a limited number of output values [9], is one application of this. A simple example of applying quantization to image compression would be to reduce the number of intensity levels L in a grey level image. There are many approaches to this from basic scalar to vector quantization [14]. What all these approaches have in common is that information is lost, and the process is therefore irreversible or lossy.

The information which is lost by applying a lossy compression method to an image needs to be measured to indicate the quality of the resulting compressed image. A standard measure used to indicate this is the PSNR [15], measured in decibels (dB). The calculation of the PSNR between two images matrices $\mathbf{I}$ and $\mathbf{I}^K$, each containing N_r rows and N_c columns is defined as:

Definition 1.

$$PSNR = 10\log_{10}\left(\frac{MAX_{\mathbf{I}}^2}{MSE}\right), \tag{1.3}$$

where $MAX_{\mathbf{I}}$ is the maximum possible pixel value in the image $\mathbf{I}$ and the Mean Squared Error (MSE) is defined as:

$$MSE = \frac{1}{N_r N_c}\sum_{n_r=1}^{N_r}\sum_{n_c=1}^{N_c}(\mathbf{I}(n_r,n_c) - \mathbf{I}^K(n_r,n_c))^2. \tag{1.4}$$

1.4.1 Transform Coding

Many image compression approaches including both the JPEG [16] and JPEG2000 [17] compression standards, use a combination of lossy and lossless compression approaches in what is known as transform coding [15]. The idea behind this approach is to apply a transform to the image to produce a representation where most of the image information is concentrated in only a few coefficients [11]. The lossy stage then quantizes the coefficients, keeping the ones containing most of the image information. These quantized coefficients are then stored.

While both JPEG and JPEG2000 have lossless modes [17] the main baseline [9] coding system utilizes a lossy compression mode. A brief description highlighting the differences between the JPEG and JPEG2000 standards when applied to a grey level image $\mathbf{I}$ is given below:

- The JPEG standard processes an image in independent square 8×8 non overlapping blocks of pixels, therefore before processing the image $\mathbf{I}$ is first partitioned into these blocks. The next step is to apply the orthogonal DCT [18], also known as the DCT-II [19], in 2D [12] to each image block to calculate an 8×8 block of coefficients. The coefficients are then quantized according to there location within the block. Huffman coding is then used to entropy code the blocks of quantized coefficients [16].

- In contrast to JPEG, JPEG2000 processes the whole image $\mathbf{I}$ at once. The first stage is then to decompose $\mathbf{I}$ into sub bands, applying the Cohen Daubechies Feauveau 9/7 (CDF9/7) filter pair [20]. Each sub band is then quantized separately with sub band dependant parameters before being arithmetically coded [17].

1.5 Approximation

1.5.1 Orthogonal Basis

Given a vector $\mathbf{f} \in \mathbb{R}^N$ representing a discrete signal, and the set $\{\mathbf{d}_n \in \mathbb{R}^N\}_{n=1}^N$, forming a basis for the space $\mathbb{R}^N$, $\mathbf{f}$ can be decomposed as,

$$\mathbf{f} = \sum_{n=1}^{N} \langle \mathbf{f}, \mathbf{d}_n \rangle \mathbf{d}_n. \tag{1.5}$$

The following is then an approximation of $\mathbf{f}$ with $K < N$ basis vectors,

$$\mathbf{f}^K = \sum_{n \in \mathcal{L}} \langle \mathbf{f}, \mathbf{d}_n \rangle \mathbf{d}_n,$$

where the set $\mathcal{L}$ contains the indices of the K basis vectors from $\{\mathbf{d}_n\}_{n=1}^N$. The approximation error is then

$$\|\mathbf{f} - \mathbf{f}^K\|^2 = \sum_{n \in \mathcal{G}} |\langle \mathbf{f}, \mathbf{d}_n \rangle|^2, \tag{1.6}$$

where $\mathcal{G} = \{1, \ldots, N\} \backslash \mathcal{L}$, with $\backslash$, denoting the relative complement of $\mathcal{L}$ in $\{1, \ldots, N\}$, $\|\cdot\|$ denoting the euclidean or l_2 norm, defined in Appendix A.3.2, and $|\cdot|$ denoting the absolute value. The error in equation (1.6) can then be minimized, for any choice of K, by choosing $\mathcal{L}$ to be the indices of the K vectors, having the largest absolute inner product $|\langle \mathbf{f}, \mathbf{d}_n \rangle|$.

This is the traditional approach for approximating a signal, which was adopted by the JPEG standard, with the Discrete Cosine basis. An alternative, which makes it possible to improve the approximation error in equation (1.6) [21], is to approximate $\mathbf{f}$ in a redundant set of linearly dependent vectors $\{\mathbf{d}_m\}_{m=1}^M$, with $M > N$. The problem of choosing the K vectors minimizing $\|\mathbf{f} - \mathbf{f}^K\|^2$, is now a combinatorial one.

In the redundant case it is convenient to write equation (1.5) as

$$\mathbf{f} = \mathbf{D}\mathbf{c} \tag{1.7}$$

where the columns of $\mathbf{D} \in \mathbb{R}^{N \times M}$ are constructed as the vectors $\{\mathbf{d}_m \in \mathbb{R}^N\}_{m=1}^M$, and $\mathbf{c} \in \mathbb{R}^M$ is a vector of coefficients. With $M > N$, a closed form solution for $\mathbf{c}$ can be found as the linear least squares solution

$$\mathbf{c} = \mathbf{D}^\dagger \mathbf{f},$$

where the superscript † denotes the Moore-Penrose pseudo inverse. This is an application of the Method Of Frames (MOF) [22] which out of all the solutions to (1.7) chooses the one where $\mathbf{c}$ has the minimum euclidean norm:

$$\min \|\mathbf{c}\| \quad \text{subject to} \quad \mathbf{f} = \mathbf{D}\mathbf{c}.$$

This solution where it is possible for $\mathbf{c}$ to have M non zero values is not sparsity preserving [23], therefore other solutions to this problem are currently being pursued as detailed in the next two Sections.

1.5.2 Sparse Representations

A K sparse representation of $\mathbf{f} \in \mathbb{R}^N$, is $\mathbf{c} \in \mathbb{R}^M$ having only K non zero values. If a K sparse representation exists then as mentioned above finding it is a combinatorial problem which is the solution to the following minimization problem

$$\min \|\mathbf{c}\|_0 \quad \text{subject to} \quad \mathbf{f} = \mathbf{D}\mathbf{c}, \tag{1.8}$$

where $\min \|\mathbf{c}\|_0$ is the l_0 norm equal to the count of non zero entries in $\mathbf{c}$.

The MOF above does not look for this solution, and because it is not sparsity preserving is unlikely to produce it. Therefore other alternative solutions to this problem have been sought. One approach is to replace the l_0 norm in equation (1.8) with l_1, where $\|\mathbf{c}\|_1 = \sum_{m=1}^M |\mathbf{c}(m)|$, and (1.8) becomes

$$\min \|\mathbf{c}\|_1 \quad \text{subject to} \quad \mathbf{f} = \mathbf{D}\mathbf{c}. \tag{1.9}$$

In [24] it was show that it is possible to determine from the matrix $\mathbf{D}$, when the solution to (1.9) is unique, that is when it is the l_0 solution. The requirement on $\mathbf{D}$ for the solution to be unique is

$$\|\mathbf{c}\|_0 < \frac{\text{Spark}(\mathbf{D})}{2}, \tag{1.10}$$

with the definition of the Spark below, taken from [24].

Definition 2. *Given a matrix* $\mathbf{D}$ *we define Spark*($\mathbf{D}$) *as the smallest possible number such that there exists a sub-group of Spark*($\mathbf{D}$) *columns from* $\mathbf{D}$ *that are linearly dependent.*

From the above, it is clear that there are cases when the solution to (1.9), will find the K sparse representation of $\mathbf{f} \in \mathbb{R}^N$, however it is not guaranteed to do so. More recently it has also been demonstrated in [25], their are practical cases in image approximation when the converse is true, that is where l_0 is superior to l_1.

Finding a solution to (1.9) is referred to as Basis Pursuit (BP) [23], which can be reformulated and tackled by classic linear programming techniques [24]. Other ways of attacking (1.8) to make the problem more tractable is to relax the l_0 constraint. The FOcal Underdetermined System Solver (FOCUSS) algorithm [26] applies this approach with an l_p norm for some $p = (0, 1]$.

All of these approaches above aim to recover the K sparse representation of $\mathbf{f}$ if it exists, but what happens if it does not and we are actually interested in is an approximation of $\mathbf{f}$. In this case an alternative approach described in the next section is undertaken.

1.5.3 Sparse Approximation

The minimization problem in equation (1.9) is reformulated as

$$\min \|\mathbf{c}\|_1 \quad \text{subject to} \quad \|\mathbf{f} - \mathbf{D}\mathbf{c}\| \leq \rho, \tag{1.11}$$

where an error of ρ can be tolerated. Problem (1.11) is referred to as Basis Pursuit Denoising (BPDN) [23], which can be re-written as a standard optimization problem. This problem can then be tackled with algorithms such as Iterative Reweighted Least Squares (IRLS) [27] or Least Absolute Shrinkage and Selection Operator (LASSO) [28].

Greedy Algorithms

An alternative viewpoint to minimizing (1.8), is that adopted by the greedy algorithm [29] approach. Instead of searching for the sparse representation $\mathbf{c}$, a greedy algorithm iteratively searches through the columns of $\mathbf{D}$ for those corresponding to non zero entries in $\mathbf{c}$. In other words at each iteration a greedy algorithm will choose columns from $\mathbf{D}$ satisfying some selection criteria. It will build up a matrix say $\ddot{\mathbf{D}}$ containing all the chosen columns, then the sparse representation $\mathbf{c}$ at any iteration can be calculated as the least squares solution to $\mathbf{c} = \ddot{\mathbf{D}}^{\dagger}\mathbf{f}$.

There are many variations of this, including but not limited to MP [30], OMP [31], OOMP [32], Backward Optimized Orthogonal Matching Pursuit (BOOMP) [33] and re-

finements [34], all offering alternatives to the original MP algorithm. Below is an overview of three of the main algorithms, MP, OMP and OOMP.

Each algorithm begins at iteration 0 by initializing the residual error $\mathbf{r}^0 = \mathbf{f}$. Then at iteration k a column of $\mathbf{D}(:, m) \in \mathbb{R}^{N \times M}, m = 1, \ldots, M$ is chosen with the current residual. The matrix $\ddot{\mathbf{D}}$ is updated to include this column and the residual error is updated ready for use in the next iteration. Below is an overview of the three algorithms highlighting the main differences in the selection and updating procedure:

- The first and most straight forward to explain of the three is MP, described in more detail in Section 2.1. At each iteration a new column of $\mathbf{D}$, $\mathbf{D}(:, \mathbf{l}(k))$ is chosen as one corresponding to the largest inner product $\langle \mathbf{r}^{k-1}, \mathbf{D}(:, m) \rangle, m = 1, \ldots, M$. The current residue error $\mathbf{r}^k$ is then updated to become

$$\mathbf{r}^k = \mathbf{r}^{k-1} - \langle \mathbf{r}^{k-1}, \mathbf{D}(:, \mathbf{l}(k)) \rangle \mathbf{D}(:, \mathbf{l}(k)).$$

  By construction at each iteration the updated residue error is orthogonal to the most recently chosen vector $\mathbf{D}(:, \mathbf{l}(k))$, however it is not guaranteed to be orthogonal to all the previously chosen vectors. An alternative termed OMP which guarantees that at each iteration the current residual is orthogonal to all previously chosen vectors in $\ddot{\mathbf{D}}$ is described next.

- OMP guarantees that at each iteration k a new vector from $\mathbf{D}$ is chosen, ensuring that the iteration is equal to the number of columns in $\ddot{\mathbf{D}}$. This is indicated by the replacement of k with K in this description.

  At each iteration the selection criteria adopted by the OMP algorithm is the same as for MP above, that is the chosen column $\mathbf{D}(:, \mathbf{l}(K))$ is the one resulting in the largest inner product $\langle \mathbf{r}^{K-1}, \mathbf{D}(:, m) \rangle, m = 1, \ldots, M$. The difference with MP is the update step

$$\mathbf{r}^K = \mathbf{r}^{K-1} - \hat{P}_{\ddot{\mathbf{D}}} \mathbf{r}^{K-1},$$

  where the operator $\hat{P}_{\ddot{\mathbf{D}}}$ is the orthogonal projection onto the span of the columns of $\ddot{\mathbf{D}}$. Thus guaranteeing that the approximation at each iteration, say $\mathbf{f}^K$ is equal to the best, least squares approximation with the columns of the matrix $\ddot{\mathbf{D}}$, that is $\mathbf{f}^K = \hat{P}_{\ddot{\mathbf{D}}} \mathbf{f}$.

- The last variation is referred to here as OOMP although it is also known by other names including Orthogonal Least Squares (OLS) [35] and Order Recursive Matching Pursuit (ORMP) [36]. This algorithm first initializes a temporary matrix $\mathbf{D}^{(0)} = \mathbf{D}$.

Then at each iteration the selection and update step is equivalent to MP with the exception that the chosen vector is $\mathbf{D}^{(K-1)}(:, \mathbf{l}(K))$. That is the update step is

$$\mathbf{r}^K = \mathbf{r}^{K-1} - \langle \mathbf{r}^{K-1}, \mathbf{D}^{(K-1)}(:, \mathbf{l}(K)) \rangle \mathbf{D}^{(K-1)}(:, \mathbf{l}(K)),$$

with $\mathbf{D}^{(K-1)}(:, \mathbf{l}(K))$ the column of $\mathbf{D}^{(K-1)}$ with the largest inner product $\langle \mathbf{r}^{K-1}, \mathbf{D}^{(K-1)}(:, m) \rangle, m = 1, \ldots, M$. What stands OOMP apart from MP is that at iteration K the unchosen columns in $\mathbf{D}^{(K-1)}$ are orthogonalized with respect to the chosen column $\mathbf{D}^{(K-1)}(:, \mathbf{l}(K))$. Denoting the orthogonalized matrix as iteration K by $\mathbf{D}^{(K)}$, this guarantees two things;

1) the unchosen columns in $\mathbf{D}^{(K)}$ will span the same space as the residue error. Therefore the one chosen at the next iteration will minimize the residue error [32], and
2) the residue error $\mathbf{r}^K$ will be orthogonal to all previously chosen columns, stored in $\ddot{\mathbf{D}}$. Therefore the approximation $\mathbf{f}^K$ will be the best least squares approximation $\mathbf{f}^K = \hat{P}_{\ddot{\mathbf{D}}}\mathbf{f}$.

If the above algorithms iterate up until the exact solution $\mathbf{f} = \mathbf{D}\mathbf{c}$ is found then these selection techniques can be applied to finding a solution to the minimization problem in (1.8). However they can easily be adapted, and offer an approach to finding a solution to,

$$\min \|\mathbf{c}\|_0 \quad \text{subject to} \quad \|\mathbf{f} - \mathbf{D}\mathbf{c}\| \leq \rho, \tag{1.12}$$

by stopping the algorithm when the condition $\|\mathbf{f} - \mathbf{D}\mathbf{c}\| \leq \rho$ is satisfied. This makes them appealing for use in sparse approximation [37], which is investigated in Chapter 2.

1.5.4 Sparse Compression

Sparse approximations, discussed above, have many digital image processing applications, including, image deblurring [38], image inpainting [39], image denoising [40] and image compression. The focus of Chapter 2 is on applying sparse approximations of images, to image compression. To motivate this idea, a short overview of current results in this field, is detailed below.

Sparse image compression has two main stages, the first described in Section 1.5.3, is to find a sparse approximation of the original image. The second stage is then, to code this information so it can be stored in an efficient manner.

Most of the recent work on sparse image approximation for compression has involved, as its first step, applying greedy algorithms with trained dictionaries. A trained dictionary

($\mathbf{D}$ in equation (1.11)) is calculated from a set of training images taken from a given image corpus. The main difference between the currently proposed approaches is the method of calculating, or training, the dictionary. However the idea behind the approaches is always the same, to calculate a matrix $\mathbf{D}$ (1.11) which is suitable for producing sparse approximations, of a chosen image corpus.

There has been a lot of alternative approaches to the problem of sparse image compression, such as [41], involving overcomplete curvelets. However in this and other work, the comparison of compression performance has not been against current image compression standards. Therefore the overview below only includes reference to methods where a comparison is made with current image compression standards.

A common approach to facial image compression, has been to first split an image up into a fixed number of square sections. Each section then represents a specific part of the face, for example, in every single image, the left eye should appear in the same square section. The training is then performed separately on each of the image sections, producing a dictionary suitable for representing each section. This has been successfully applied in [42] to outperform JPEG2000 at low bit rates. In this method the dictionary was trained by the K-SVD algorithm [43] and the sparse approximation made by OMP. The final stage involved applying uniform quantization and Huffman coding, to produce the compressed image.

The main problem with processing the above facial images in blocks, is that at low bit rates, blocking artefacts appear at the boundaries between the chosen sections. The authors of [42] apply a linear deblocking approach in [44], to reduce these artefacts. This increased the PSNR for a given bit rate and subsequently increased the compression performance in relation to JPEG2000.

In [45] the K-LMS algorithm was proposed for compressing facial images. This is a variation of the K-SVD algorithm, reducing the time required to train the dictionaries. The authors again chose to produce the sparse approximation with OMP and the compressed image with Huffman coding. The compression results at low bit rates were again better than JPEG2000.

The above results are promising however the approaches are very specific to facial images, relying on each feature always being in the same section of the image. In [46] this approach is generalized by classifying regions of images and training dictionaries suitable for each of these regions. The image is then coded with a Set Partitioning in Hierarchical Trees (SPIHT) [47] like entropy coder. The performance of this approach is shown to be comparable to JPEG2000 over a more general set of standard test images, at 0.5bpp.

More recently in [48] dictionaries trained by the Recursive Least Squares Dictionary Learning Algorithm (RLS-DLA) were used to compress natural images. The compression results were just below that of JPEG2000, however the evaluation was performed over much higher bit rates than in [46]. In this paper sparse approximations were made of images following a transformation into the CDF9/7 wavelet domain. These were found to produced smaller compressed images than those performed in the pixel domain. The bit stream was again produced by Huffman coding the results from the sparse approximation, however the authors chose to use the more computationally expensive OOMP algorithm to produce the initial approximation.

From the above it is clear that a lot of the work on sparse image compression has historically been focussed on compressing facial images. More recently promising results have been achieved on natural images as well by training dictionaries with the RLS-DLA algorithm. In Chapter 2 sparse approximations of natural and astronomical image sets is examined by training dictionaries with a variation of the RLS-DLA algorithm.

1.6 Image Security

The need for digital image security arises from the ease with which digital images can be both distribute and duplicated [49]. These two concerns are addressed with the following two techniques:

1) Encryption, providing end to end security for image distribution [50].

2) Watermarking, providing copyright protection through owner identification [51].

Unlike conventional encryption strategies where the objective is to prevent access to the *plaintext* to all except those in possession of the correct *private key*. Image security can be applied in many different ways depending on the specific requirements which need to be satisfied.

If the requirement is for full end to end security, then the traditional approach would be to encrypt the entire image, with a standard algorithm like the Data Encryption Standard (DES) [52]. As discussed in Section 1.6.2 digital images have unique requirements which make the direct application of standard encryption algorithms unsuitable for every use case.

For example, the provider of online television services, clearly only wants to provide there content to fee paying subscribers. However they may wish to encourage future subscribers by supplying a low quality version for free. This can be achieved by applying

a transparent encryption strategy, partially degrading the broadcast video stream. The decryption process can then be realized by fee paying customers, who would be provided with the decryption *key*.

Encryption

Encryption can be described as a scheme which enables two parties to exchange messages with each other in the presence of an adversary, who can intercept these messages [49]. Broadly speaking encryption is the process of securing these *plaintext* messages [53]. In this process the encryption algorithm converts the *plaintext* to what is known as *ciphertext* [53]. Decryption is then the process of recovering the message from the *ciphertext*.

The study of techniques for securing information is called cryptography [54]. In its modern form there are two main approaches to securing *plaintext* messages against an adversary, these are:

1) Private or symmetric key cryptosystems, such as the DES and Advanced Encryption Standard (AES) [55]. In this paradigm the *plaintext* is encrypted with a value referred to as the *private key*, this can then be decrypted with the same *private key* or a *key* which can be explicitly determined from it [50]. This cryptosystem is termed symmetric because knowledge of the same *private key* is enough to successfully encrypt and decrypt the *plaintext*.

2) Public or asymmetric key cryptosystems, such as the Ron Rivest, Adi Shamir and Leonard Adleman (RSA) algorithm [55]. The idea here is that the *plaintext* can be encrypted with a *public key* and then decrypted by a separate *private key*. This is clearly asymmetric, with a different *key* being required at the encryption and decryption stages [54].

Before discussing applications of encryption to digital images, a brief overview of several types of attack, which, a digital image encryption scheme should be resistant to, is given in the next Section.

1.6.1 Cryptanalysis

Cryptanalysis can be described as, the art of deciphering encrypted messages without prior knowledge of the decryption key [50]. Depending on the amount of available information an adversary has there are several methods of attack which a cryptanalyst may use:

- *Ciphertext* only attacks, where an adversary only has access to one or more encrypted

messages. This level of attack is the minimum that an encryption scheme should be secure against.

- Brute force attacks. This type of attack involves an exhaustive search through all possible *private keys* until one is found which successfully decrypts the *ciphertext*. The set of all possible *private keys* is referred to as the *keyspace* and a necessary condition for an encryption scheme is that the *keyspace* is large enough to prevent a brute force attack [53, 56].

- Known *plaintext* attack. In this type of attack an adversary in possession of the *ciphertext* has some knowledge regarding the *plaintext* from which it was generated. This knowledge is used to help determine either part or all of the *private key*.

- Chosen *plaintext* attack. In this scenario the attacker can choose the *plaintext* to be encrypted. They can then use the knowledge of the *plaintext*, *ciphertext* pairs to obtain either part or all of the *private key*.

- Chosen *ciphertext* attack. Contrary to the chosen *plaintext* attack, here the attacker can choose the *ciphertext* to be decrypted which can then be used to help determine either part or all of the *private key*.

A secure encryption scheme should be resistant to all of these attacks. The security is then measured by the amount of computational effort required when the best known attack on the system is used [53, 54]

1.6.2 Image Encryption

From the above the natural question is, why not just apply existing public or private encryption approaches, to either to the raw or compressed version of a digital image.

Firstly this naive approach poses several problems, one of which, as mentioned above, is that images require a lot of storage space even when compressed. Therefore applying classic encryption methods to this type of data would be computationally expensive.

Another problem which arises from the properties of encrypted data [57], is the trade off between encryption and compression. If encryption is performed first, the compressibility of the resulting data is significantly reduced. Therefore to reduce the size of encrypted images, compression always has to be performed first.

A further consideration is the quality of the encrypted data. In [52] it was highlighted that the redundancy present in images can reduce the effectiveness of encryption when it is applied to the raw pixel values.

These concerns have been addressed by two alternative approaches which either partially or fully encrypt the image data. The first termed partial or selective encryption only encrypts part of the image data, reducing the amount processing involved, but also reducing the quality of the encryption.

Partial encryption has been proposed to secure JPEG compressed images with various approaches, including encryption of only selected DCT transform coefficients [58, 59], and encryption of only the sign of each transform coefficient [60, 61]. Alternatively partial encryption has been applied to both raw image data, encrypting only selected bit planes [62] and wavelet decompositions of digital images [63].

Many of these selective techniques are susceptible to *ciphertext* only attack due to visible information remaining in the encrypted images [57]. Therefore an alternative to is to use a full encryption method specifically designed for processing digital images, such as a recently proposed scheme [64] which is applied in the spatial domain applying bit manipulation to the pixels to fully encrypt the raw image data.

There are many other proposed full image encryption schemes, a large proportion of which are based on chaotic maps. In these methods chaotic maps are used either to generate pseudorandom bits [65] for use in a classic encryption approaches [66] or to perform 2D permutations in the spatial domain [67, 68].

Image Degradation

Unlike the partial encryption discussed above where the aim is to reduce the overall processing time of the encryption step while maintaining a high level of security. Transparent encryption can be applied to content distribution systems to degrade or encrypt only a proportion of the multimedia data [61, 69–71], thus enticing new customers to purchase the full service.

In [70] the author proposes a scheme which applies a linear transformation to the raw pixel values before applying the compression stage. The amount of degradation to the image can then be controlled by the broadcaster depending on the business requirements. This scheme has a number of drawbacks one being that the costly linear transformation has to be applied to every single pixel in the original image. To improve on this in [71] a scrambling operation is applied instead to the coefficients resulting from the DCT transformation.

In line with the above, Chapters 3-4 propose a novel application of sparse image representations to partial image security, termed image *folding*. The application of the *folding* approach results in digital images where only the top section is left unencrypted. However

the method can easily be applied to only encrypt other sections of the image such as alternate rows or columns, which would be achieved by applying a simple reordering of the image pixels, before applying the *folding* procedure.

1.7 Experimental Set up

All of the Experiments in the following Chapters were performed within the MATLAB [72] R2012a programming environment. Unless otherwise stated, all calculations were performed by inbuilt functions. Specifically three routines were required to secure the images in Chapters 3-4 , these are given below with details of there contribution.

- `qr()` calculates the orthogonal-triangular decomposition of a matrix. Required to perform the $\widehat{\mathbf{Orth}}(\cdot)$ operation.
- `svd()` calculates the singular value decomposition when the SVD security method is applied, shown in Section 3.3.1.
- `rand()` generates pseudorandom numbers. These were required by the
  - random security scheme, described in Section 3.3.2, to construct the matrices $\mathbf{Z}$ and $\mathbf{U}$, and by the
  - "pixel" method, to construct the vectors $\mathbf{m}$, described in Section 4.7.2.

  The Pseudorandom Number Generator (PRNG) which produced all the pseudorandom numbers required above was the Mersenne Twister MT19937 [73] algorithm.

Additionally entropy coding and decoding was performed inside MATLAB using a mex implementation of the C++ adaptive arithmetic coding algorithm provided by Dr. Amir Said [74], available from [75].

The matrix and vector notation, described in Appendix A, is designed to be equivalent to that used in MATLAB programming environment, described in Appendix A.

2 Sparse Image Representation with Greedy Algorithms

This Chapter explores sparse representations of grey level images, generated by greedy algorithms.

The Chapter begins with a description of a modification to the original Matching Pursuit (MP) algorithm, termed Self Projected Matching Pursuit (SPMP) [2]. The next Section introduces alternative implementations of some greedy selection approaches to operate in two dimensions (2D). Experiments are performed to evaluate these approaches. The comparison was based on the sparsity of the representations achieved, and the execution time when performed on a standard personal computer. The most suitable approach for the application in hand is then selected.

The next Section introduces several dictionary constructions. A comparison of the sparsity of approximations, made on grey level images, by the adopted selection technique with these dictionaries, is then explored. Finally the Chapter introduces and discusses the implementation of a simple coding scheme to store the sparse image information. This coding scheme is then evaluated with respect to both the JPEG and JPEG2000 image compression algorithms.

2.1 Matching Pursuit

The MP algorithm originally proposed in [30], is an iterative approach to approximating a signal by choosing elements from a dictionary. At each iteration of the algorithm, the residual error of the current approximation, is calculated to be orthogonal to the most recently chosen dictionary element. An overview of the MP approach, applied to approximate a vector $\mathbf{f}$, by choosing columns from a redundant matrix or dictionary $\mathbf{D} \in \mathbb{R}^{N \times M}$, with $M > N$, is given next.

The residual at iteration 0 is initialized to $\mathbf{r}^0 = \mathbf{f}$. Denoting by $\mathbf{l}(k)$ the index of the column $\mathbf{D}(:, \mathbf{l}(k))$ chosen at the k'th iteration, the MP algorithm evolves as follows. At iteration k the current residual $\mathbf{r}^{k-1}$ is projected onto the vector $\mathbf{D}(:, \mathbf{l}(k))$. The resulting residual $\mathbf{r}^k$ is then

$$\mathbf{r}^k = \mathbf{r}^{k-1} - \langle \mathbf{r}^{k-1}, \mathbf{D}(:, \mathbf{l}(k)) \rangle \mathbf{D}(:, \mathbf{l}(k)). \tag{2.1}$$

Since $\mathbf{r}^k$ in (2.1) is orthogonal to $\mathbf{D}(:, \mathbf{l}(k))$,

$$\|\mathbf{r}^k\|^2 = \|\mathbf{r}^{k-1}\|^2 - |\langle \mathbf{r}^{k-1}, \mathbf{D}(:, \mathbf{l}(k)) \rangle|^2. \tag{2.2}$$

Therefore to minimize the current residual $\|\mathbf{r}^k\|^2$ the MP approach chooses $\mathbf{l}(k)$ such that $|\langle \mathbf{r}^{k-1}, \mathbf{D}(:, \mathbf{l}(k)) \rangle|$ is a maximum.

At iteration k, the residual $\mathbf{r}^k$, is calculated to be orthogonal to the most recently selected column $\mathbf{D}(:, \mathbf{l}(k))$, however is is not guaranteed to be orthogonal to all the previously selected columns. As a result a column $\mathbf{D}(:, \mathbf{l}(i)), i = 1, \ldots, k-1$ which has been chosen at a previous iteration, may be chosen again. An improvement to this mentioned in the original paper [30], discusses modifying the original MP approach with an orthogonal projection. This improvement is realized by orthogonally projecting the current approximation onto the set of already selected atoms, at each iteration of the algorithm. The projection step can be implemented in several ways, the original MP paper [30] suggested the conjugate gradient method, for calculating this "back-projection" step. One approach to this is that undertaken by OMP, described in Section 2.3, which explicitly calculates the orthogonal projection at each iteration. An alternative which is discussed in the next Section, is to apply the MP approach itself, to calculate the orthogonal projection onto the set of already selected atoms.

2.2 Self Projected Matching Pursuit

The approach alluded to above is termed SPMP [2]. A description of its application to render a sparse approximation of a vector $\mathbf{f}$ with a dictionary $\mathbf{D} \in \mathbb{R}^{N \times M}$, with $M > N$,

follows.

Begin at iteration 0 choosing a projection step p and an approximation tolerance ρ. Initialize, the set $\mathcal{S} = \{\emptyset\}$ to store the chosen vectors, and the residual $\mathbf{r}^0 = \mathbf{f}$. The algorithm then evolves as follows:

i) Apply a maximum of p iterations of MP to the current residual, selecting columns $\{\mathbf{D}(:, \mathbf{l}(i))\}_{i=1}^{p^k}$ with $p^k \leq p$ from $\mathbf{D}$. The chosen vectors are then assigned to $\mathcal{S}$ as $\mathcal{S} \leftarrow \mathcal{S} \cup \{\mathbf{D}(:, \mathbf{l}(i))\}_{i=1}^{p^k}$. With K equal to the cardinality of the updated set $\mathcal{S}$, the approximation so far is denoted as $\mathbf{f}^K$, and the residual of this approximation is $\mathbf{r}^K = \mathbf{f} - \mathbf{f}^K$. The number of chosen vectors p^k is determined by the termination of the MP selection procedure. This is either following p iterations, or when $\|\mathbf{r}^K\| < \rho$.

ii) Apply the MP approach to approximate $\mathbf{r}^K$ with only the selected set $\mathcal{S}$ as the dictionary. This guarantees the asymptotic convergence to the approximation $\hat{P}_{\mathbb{S}}\mathbf{r}^K$ of $\mathbf{r}^K$, where $\hat{P}_{\mathbb{S}}$ is the orthogonal projection onto the space $\mathbb{S} = \text{span}\, \mathcal{S}$. The residual of this MP approximation is $\mathbf{r}^\perp = \mathbf{r}^K - \hat{P}_{\mathbb{S}}\mathbf{r}^K$, having no component in $\mathbb{S}$.

iii) Update the current residual $\mathbf{r}^K \leftarrow \mathbf{r}^\perp$, and approximation $\mathbf{f}^K \leftarrow \mathbf{f}^K + \hat{P}_{\mathbb{S}}\mathbf{r}^K$, and repeat steps i) and ii), until, for a required ρ, the condition $\|\mathbf{r}^K\| < \rho$ is reached.

For $p = 1$ the above refinement gives, asymptotically [76], the orthogonal projection approximation at each iteration, thereby reproducing the results of OMP. As illustrated by the example below, significant improvement upon the original MP approach may be achieved for values of $p > 1$.

Example. This numerical example, similar to the one given in the "Sparse Representations of Astronomical Images" [2] paper, is a hard test for MP. Consider a matrix $\mathbf{D}_1^c$ representing a Redundant Discrete Cosine (RDC) dictionary given by:

$$\mathbf{D}_1^c(n, m) = \mathbf{w}(m) \cos(\frac{\pi(2n-1)(m-1)}{2M}),\ n = 1, \ldots, N, m = 1, \ldots, M, \qquad (2.3)$$

with $\mathbf{w}$ containing the normalization factors. For $M = N$, this set is a Discrete Cosine (DC) orthonormal basis for the Euclidean space $\mathbb{R}^N$. For $M = 2aN$, with $a \in \mathbb{N}$, the set is a RDC dictionary with redundancy $2a$. The redundancy here is fixed to 2, with $a = 1$.

To represent the modified chirp signal $e^{-0.1x}\cos(2\pi x^2)$ depicted in Fig. 2.1, an equidistant partition of the interval $[0, 8]$ consisting of $N = 2000$ points is taken, and the chirp is sampled at those points $\mathbf{f}(n)$, $n = 1, \ldots, N$. The aim is to find an approximation of these points up to precision $\rho = 0.001\|\mathbf{f}\|$. Considering $M = N = 2000$, the above definition of $\mathbf{D}_1^c$ provides orthonormal basis, therefore both the MP and OMP methods give the sparsest decomposition of the signal. For an approximation to the given precision

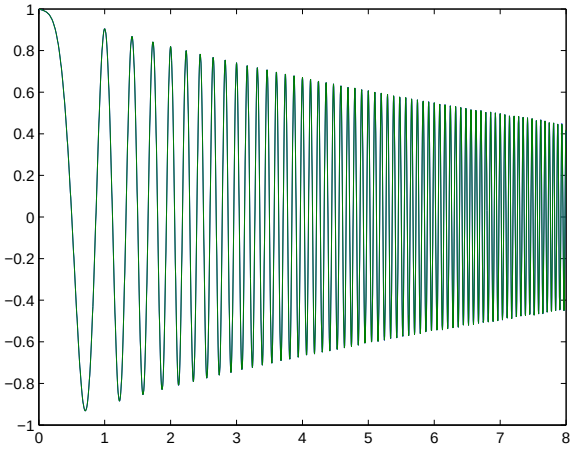

Figure 2.1: Chirp signal approximated up to error $\rho = 0.001\|\mathbf{f}\|$ by i) $K = 542$ orthogonal DC components taken from (2.3) with $N = M = 2000$. ii) $K = 281$ vectors selected by OMP from (2.3) when $M = 2N = 4000$, or $K = 729$ selected with MP. iii) $K = 300$ vectors selected by SPMP with $p = 10$ from (2.3) with $M = 2N = 4000$, or $K = 281$ with $p = 2$.

(coinciding visually with the theoretical chirp in Fig.2.1) it is necessary to use $K = 542$ orthogonal elements from (2.3). Setting $M = 2N = 4000$ the dictionary $\mathbf{D}_1^c$ is no longer an orthogonal basis but a redundant *tight frame* [77], now the algorithms MP and OMP produce very different decompositions. While OMP improves the sparsity of the representation requiring only $K = 281$ components, MP needs $K = 729$ different atoms, i.e. significantly more than with the orthonormal basis. The reason for the poor performance of MP is, the redundant dictionary contains highly correlated atoms and MP is picking linearly dependent ones, something which cannot occur with OMP. However, when applying the proposed refinement SPMP with projection step $p = 10$ the number of required components drops to $K = 300$. For $p = 2$ the number of required components falls to that of OMP, i.e. $K = 281$. In this example there is no need for the SPMP approach, because the already established algorithm OMP performs the decomposition faster. The result serves to illustrate the fact that SPMP can provide an effective alternative to OMP, when, as is the case with 2D images, OMP becomes slow or its storage demands cannot be met. Further details for the 2D implementation of SPMP will be discussed in Section. 2.3.

2.3 2D Implementation of the Selection Strategies with Separable Dictionaries

The set $\mathcal{D} = \{\mathbf{S}_m \in \mathbb{R}^{N_r \times N_c}\}_{m=1}^{M_r M_c}$ is a dictionary if $M_r M_c > N_r N_c$ and it spans the space $\mathbb{R}^{N_r \times N_c}$. With $\|\mathbf{S}_m\|_F = 1, m = 1, \ldots, M_r M_c$, where $\|\cdot\|_F$ denotes the Frobenius norm defined in Appendix A.3.4, a suitable approach to finding a sparse approximation of an

image $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$ in $\mathcal{D}$ is to apply OMP.

The OMP selection criteria chooses at iteration K the index $\mathbf{l}(K)$ corresponding to the atom that maximize the absolute value of the inner product given by,

$$\mathbf{l}(K) = \underset{m=1,\ldots,M_r M_c}{\operatorname{argmax}} |\langle \mathbf{S}_m, \mathbf{R}^{K-1} \rangle_F| \quad \text{with,} \quad \mathbf{R}^{K-1} = \mathbf{I} - \sum_{k=1}^{K-1} \mathbf{c}(k)\mathbf{S}_{\mathbf{l}(k)}, \tag{2.4}$$

where $\langle \cdot \rangle_F$ denotes the Frobenius inner product defined in Appendix A.3.3,

This is performed in the OMP algorithm with a large matrix $\mathbf{F} \in \mathbb{R}^{N_r N_c \times M_r M_c}$ who's m'th column is the vector $\widehat{\mathbf{Vec}}(\mathbf{S}_m) \in \mathbb{R}^{N_r N_c}, m = 1, \ldots, M_r M_c$. The $\widehat{\mathbf{Vec}}(\cdot)$ operation relabels the elements of an $N_r \times N_c$ matrix, in row major order, to become the elements of a $N_r N_c \times 1$ vector, with the convention described in Appendix A.2.2. The index of the largest element of the vector

$$\mathbf{a} = |\mathbf{F}^T(\widehat{\mathbf{Vec}}(\mathbf{R}^{K-1}))|, \tag{2.5}$$

where the superscript T indicates the transpose operation defined in Appendix A.1, is then equal to $\mathbf{l}(K)$ in equation (2.4). The selection step shown in equation (2.5) has complexity $\mathcal{O}(N_c N_r M_c M_r)$, where $\mathcal{O}(\cdot)$ denotes the order of the complexity. This complexity can be reduced if each $\mathbf{S}_m \in \mathbb{R}^{N_c \times N_r}, m = 1, \ldots, M_r M_c$ is a separable matrix constructed from vectors $\mathbf{d}^r_{m_r} \in \mathbb{R}^{N_r}, m_r = 1, \ldots, M_r$ and $\mathbf{d}^c_{m_c} \in \mathbb{R}^{N_c}, m_c = 1, \ldots, M_c$. Such a matrix is constructed as,

$$\mathbf{S}_m = \mathbf{d}^r_{m_r} \otimes \mathbf{d}^c_{m_c}, \quad m = 1, \ldots, M_r M_c, \quad \text{and} \quad m_r = 1, \ldots, M_r, m_c = 1, \ldots, M_c, \tag{2.6}$$

with $\otimes$ indicating the Kronecker product between two vectors, defined in Appendix A.4.

2.3.1 Reducing the Complexity of the Selection Procedure

Instead of the decomposition shown in equation 2.5 involving a single matrix, the problem is reformulated here with two matrices $\mathbf{D}^r \in \mathbb{R}^{N_r \times M_r}$ and $\mathbf{D}^c \in \mathbb{R}^{N_c \times M_c}$ who's columns are respectively the vectors $\mathbf{d}^r_{m_r} \in \mathbb{R}^{N_r}, m_r = 1, \ldots, M_r$ and $\mathbf{d}^c_{m_c} \in \mathbb{R}^{N_c}, m_c = 1, \ldots, M_c$ from equation (2.6).

Given the identity [78]

$$(\mathbf{D}^r \otimes \mathbf{D}^c)\widehat{\mathbf{Vec}}(\mathbf{R}^{K-1}) = \widehat{\mathbf{Vec}}((\mathbf{D}^c)^T \mathbf{R}^{K-1} \mathbf{D}^r), \tag{2.7}$$

finding the the maximum element of $\mathbf{a}$ in equation (2.5), is equivalent to finding the the maximum element of,

$$\mathbf{A} = |\mathbf{D}^c \mathbf{R}^{K-1} (\mathbf{D}^r)^T|. \tag{2.8}$$

That is $\mathbf{A}(\mathbf{l}^c(K), \mathbf{l}^r(K)) = \mathbf{a}(\mathbf{l}(K))$, where $\mathbf{l}^c(K)$ and $\mathbf{l}^r(K)$ are the row and column indices where the maximum element of $\mathbf{A}$ in equation (2.8) occurred. Additionally the column of $\mathbf{F}$, $\widehat{\mathbf{Vec}}(\mathbf{S}_{\mathbf{l}(K)})$, which produces the largest value in $\mathbf{a}$ from equation (2.5), can be constructed as $\mathbf{D}^r(:, \mathbf{l}^r(K)) \otimes \mathbf{D}^c(:, \mathbf{l}^c(K))$ if required.

If $N_r, N_c = N$ and $M_r, M_c = M$ the complexity of (2.8) for redundant dictionaries with $M > N$ is $\mathcal{O}(M^2N)$, which is less than $\mathcal{O}(M^2N^2)$, the corresponding complexity of (2.5).

2.3.2 Separable Orthogonal Matching Pursuit

The separable implementation of OMP will be termed OMP2D. Within the adopted notation the algorithm evolves as follows: On setting $\mathbf{R}^0 = \mathbf{I}$ at iteration K the algorithm selects the vectors $\mathbf{D}^c(:, \mathbf{l}^c(K))$ and $\mathbf{D}^r(:, \mathbf{l}^r(K))$ corresponding to the column and row of the largest element of the matrix $\mathbf{A}$ in equation (2.8), with

$$\mathbf{R}^{K-1} = \mathbf{I} - \sum_{k=1}^{K-1} \mathbf{D}^c(:, \mathbf{l}^c(k))\mathbf{c}(k)(\mathbf{D}^r(:, \mathbf{l}^r(k)))^T. \tag{2.9}$$

The coefficients $\mathbf{c}(k)$, $k = 1, \ldots, K-1$ in the above expansion are such that $\|\mathbf{R}^{K-1}\|_F$ is minimum. This is ensured by requesting that $\mathbf{R}^{K-1} = \mathbf{I} - \hat{P}_{\mathbb{V}_{K-1}}\mathbf{I}$, where $\hat{P}_{\mathbb{V}_{K-1}}$ is the orthogonal projection operator onto $\mathbb{V}_{K-1} = \text{span}\{\mathbf{D}^r(:, \mathbf{l}^r(k)) \otimes \mathbf{D}^c(:, \mathbf{l}^c(k))\}_{k=1}^{K-1}$.

A straightforward generalization of the implementation in one dimension (1D) discussed in [32, 34] provides us with a suitable representation of $\hat{P}_{\mathbb{V}_{K-1}}\mathbf{I}$. This is given by

$$\hat{P}_{\mathbb{V}_{K-1}}\mathbf{I} = \sum_{k=1}^{K-1} \langle \mathbf{B}_k^{(K-1)}, \mathbf{I} \rangle_F \mathbf{S}_k = \sum_{k=1}^{K-1} \mathbf{c}(k)\mathbf{S}_k, \tag{2.10}$$

where

$$\mathbf{S}_k = \mathbf{D}^c(:, \mathbf{l}^c(k)) \otimes \mathbf{D}^r(:, \mathbf{l}^r(k)),$$

and $\mathbf{B}_k^{(K-1)}, k = 1, \ldots, K-1$ are the concomitant reciprocal matrices, which are the unique elements of $\mathbb{R}^{N_r \times N_c}$ satisfying the conditions:

i) $\langle \mathbf{S}_n, \mathbf{B}_k^{(K-1)} \rangle_F = \delta_{n,k} = \begin{cases} 1 & \text{if } n = k \\ 0 & \text{if } n \neq k. \end{cases}$

ii) $\mathbb{V}_{K-1} = \text{span}\{\mathbf{B}_k^{(K-1)}\}_{k=1}^{K-1}$.

Such matrices can be adaptively constructed through the recursion formula:

$$\begin{aligned}&\mathbf{B}_k^{(K)} = \mathbf{B}_k^{(K-1)} - \mathbf{B}_K^{(K)}\langle \mathbf{S}_K, \mathbf{B}_k^{(K-1)}\rangle_F, \quad k = 1, \ldots, K-1 \\ &\text{where} \\ &\mathbf{B}_K^{(K)} = \mathbf{C}_K / \|\mathbf{C}_K\|_F^2 \text{ with } \mathbf{C}_1 = \mathbf{S}_1 \text{ and } \mathbf{C}_K = \mathbf{S}_K - \sum_{k=1}^{K-1} \frac{\mathbf{C}_k}{\|\mathbf{C}_k\|_F^2}\langle \mathbf{C}_k, \mathbf{S}_K\rangle_F.\end{aligned} \tag{2.11}$$

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

$$\mathbf{C}_K = \mathbf{C}_K - \sum_{k=1}^{K} \frac{\mathbf{C}_k}{\|\mathbf{C}_k\|_F^2}\langle \mathbf{C}_k, \mathbf{C}_K\rangle_F.$$

The algorithm will iterate up to step K for which, for a given ρ, the stopping criteria

$$\|\mathbf{I} - \hat{P}_{\mathbb{V}_K}\mathbf{I}\|_F < \rho \tag{2.12}$$

is met.

If the coefficients in (2.9) are instead the largest elements of the matrix $\mathbf{A}$ in (2.8) then this reduces to Matching Pursuit in 2D (MP2D), a separable version of MP.

If a separable dictionary is considered, adopting the algorithm OMP2D instead of OMP can be very effective at reducing the execution time of the approximation. However the large matrices $\mathbf{B}_k^{(K)}, k = 1, \ldots, K$ required in equation (2.10) still need to be calculated. In the Experiment in Section 2.5 this is shown to become prohibitive in terms of execution time, as the size of the blocks $N_r \times N_c$ gets larger than 24×24.

As demonstrated in Section 2.2, SPMP can reproduce the results of OMP, removing the need for the calculation of the large matrices $\mathbf{B}_k^{(K)}, k = 1, \ldots, K$. Therefore applying SPMP with a separable dictionary, in a procedure called Self Projected Matching Pursuit in 2D (SPMP2D$_1$), will both reduce the complexity of the selection step and reduce the storage requirements of the algorithm. This is shown experimentally in Section 2.5.1, to significantly reduce the processing time over OMP2D when images are processed in blocks with $N_r = N_c = 32$.

The pseudo code for the SPMP2D$_1$ algorithm is shown in Appendix D.

2.4 Dictionaries for Sparse Representation of Images

The following discusses the construction of matrices for creating sparse representations of images $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$.

In this and the remaining Chapters all sparse image approximations are calculated from separable matrices with unit norm. A separable matrix $\mathbf{S}_m$ is constructed from

vectors $\mathbf{d}^c_{m_r} \in \mathbb{R}^{N_r}, m_r = 1, \ldots, M_r$ and $\mathbf{d}^r_{m_c} \in \mathbb{R}^{N_c}, m_c = 1, \ldots, M_c$, with equation (2.6). The condition $\|\mathbf{S}_m\|_F = 1, m = 1, \ldots, M_r M_c$ is guaranteed by requiring both $\|\mathbf{d}^c_{m_r}\| = 1, m_r = 1, \ldots, M_r$ and $\|\mathbf{d}^r_{m_c}\| = 1, m_c = 1, \ldots, M_c$. The dictionary $\mathcal{D}$ is therefore the set of matrices $\mathbf{S}_m \in \mathbb{R}^{N_c \times N_r}, m = 1, \ldots M_c M_r$ spanning the space $\mathbb{R}^{N_c \times N_r}$ with $M_c M_r > N_c N_r$.

The vectors $\mathbf{d}^c_{m_c} \in \mathbb{R}^{N_c}$ and $\mathbf{d}^r_{m_r} \in \mathbb{R}^{N_r}$ are stored, respectively as columns of two matrices $\mathbf{D}^c \in \mathbb{R}^{N_c \times M_c}$ and $\mathbf{D}^r \in \mathbb{R}^{N_r \times M_r}$, referred to as the column and row dictionaries. This representation is convenient because these are the two matrices from equation (2.8), required to perform the selection procedure in the separable algorithms, MP2D, OMP2D and SPMP2D$_1$. It is clear that, as the names imply, the vectors $\mathbf{d}^c_{m_c} \in \mathbb{R}^{N_c}$ and $\mathbf{d}^r_{m_r} \in \mathbb{R}^{N_r}$, respectively perform inner products with the columns and rows of the residual $\mathbf{R}^{K-1}$ in equation (2.8).

If the selection procedure cannot employ the separable product, which is the case with OOMP, the large separable matrices $\mathbf{S}_m \in \mathbb{R}^{N_c \times N_r}, m = 1, \ldots M_c M_r$ are calculated from the column and row dictionaries, $\mathbf{D}^c$ and $\mathbf{D}^r$ before applying the algorithm.

A K-sparse representation of a given image $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$ constructed from the column and row dictionaries $\mathbf{D}^c$ and $\mathbf{D}^r$ described above is,

$$\mathbf{I}^K = \sum_{k=1}^{K} \mathbf{D}^c(:, \mathbf{l}^c(k)) \mathbf{c}(k) (\mathbf{D}^r(:, \mathbf{l}^r(k)))^T. \tag{2.13}$$

For the finite dimension Euclidean spaces $\mathbb{R}^{N_r}$ and $\mathbb{R}^{N_c}$ one can respectively construct dictionaries $\mathbf{D}^c$ and $\mathbf{D}^r$ with arbitrary vectors $\mathbf{d}^c_{m_c}$ and $\mathbf{d}^r_{m_r}$. The sets of vectors described in the following Sections were chosen because of the sparsity they produced in approximations of the form shown in equation (2.13).

The construction described for each of the first 5 dictionaries below is for the column dictionary $\mathbf{D}^c$, although it can also be applied to produce the row dictionary $\mathbf{D}^r$. On the other hand the last two, trained dictionaries, come as a pair, therefore the construction below is for both $\mathbf{D}^c$ and $\mathbf{D}^r$.

Redundant Discrete Cosine Dictionary

The matrix of Redundant Discrete Cosine vectors $\mathbf{D}^c_1$ which is referred to as the RDC dictionary contains all the vectors produced by equation (2.3) for $M_c = 2aN_c$, with $a = 1$. These vectors were chosen because of the significant increase in sparsity they produced over the DC basis demonstrated in the previous Example.

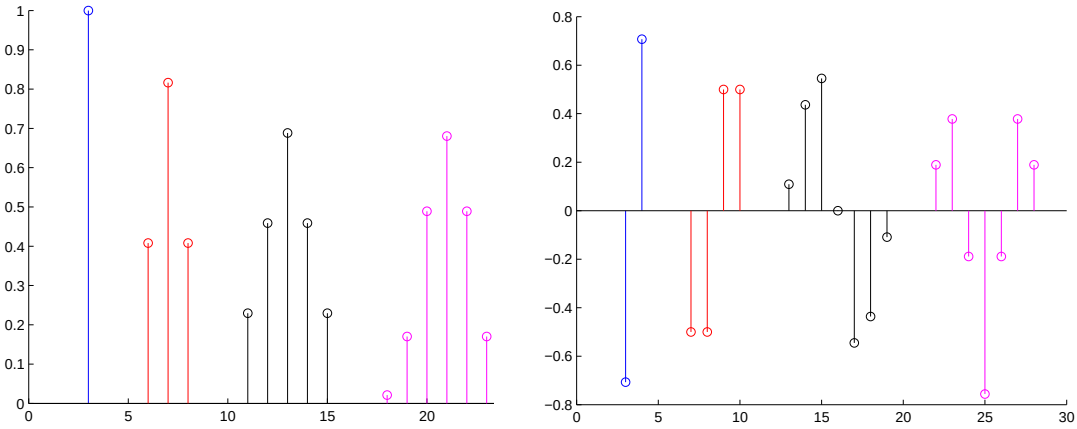

Figure 2.2: Prototype vectors as defined in (B.5) and (B.6). The columns of the RDBS dictionary are constructed by translations of these prototypes, applying the cut off approach at the boundaries.

Redundant Discrete B-Spline Dictionary

This matrix $\mathbf{D}_3^c$, referred to as the Redundant Discrete B-Spline (RDBS) dictionary is inspired by a general result holding for continuous spline spaces. Namely, that *spline spaces on a compact interval can be spanned by dictionaries of B-splines of broader support than the corresponding B-spline basis functions [79, 80].* The prototype vectors for this dictionary are shown in Figure 2.2 with their construction discussed in Appendix B.

Redundant Discrete Uniform Dictionary

This simple dictionary is formed by combinations of discrete value functions, $f_j(n)$ which have a constant value on a given interval given by:

$$f_j(n) = \begin{cases} \frac{1}{\sqrt{j}} & 0 < n \leq j \\ 0 & \text{otherwise} \end{cases} \tag{2.14}$$

with $n \in \mathbb{N}$.

The 3 sets of discrete uniform atoms of supports $j = 2, 4, 6$ required in the Experiments are given by,

$$\{f_j(n - m + 1); n = j, \ldots, N_c + j - 1\}_{m=1}^{M_j}, \quad j = 2, 4, 6.$$

The dictionary $\mathbf{D}_4^c$ referred to as the Redundant Discrete Uniform (RDU) dictionary is constructed with these vectors as its columns.

Redundant Discrete Wavelet Dictionary

The vectors for this set are discretized versions of the continuous wavelets given in [1], the form of which is very similar to the Mexican Hat wavelet. Three fractional scaling parameters were required to produce discrete wavelets of support $3, 5$, and 7, represented

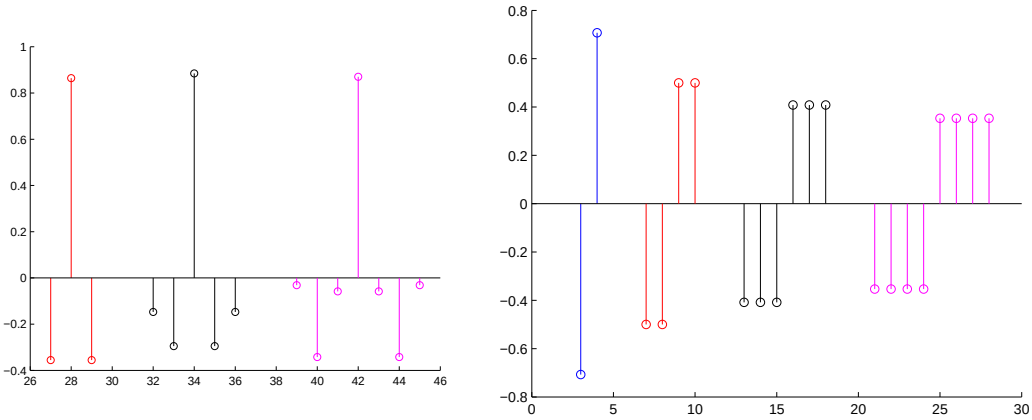

Figure 2.3: The left hand graph shows discretized versions of the continuous wavelets given in [1] and the right had graph shows discretized versions of Haar wavelets.

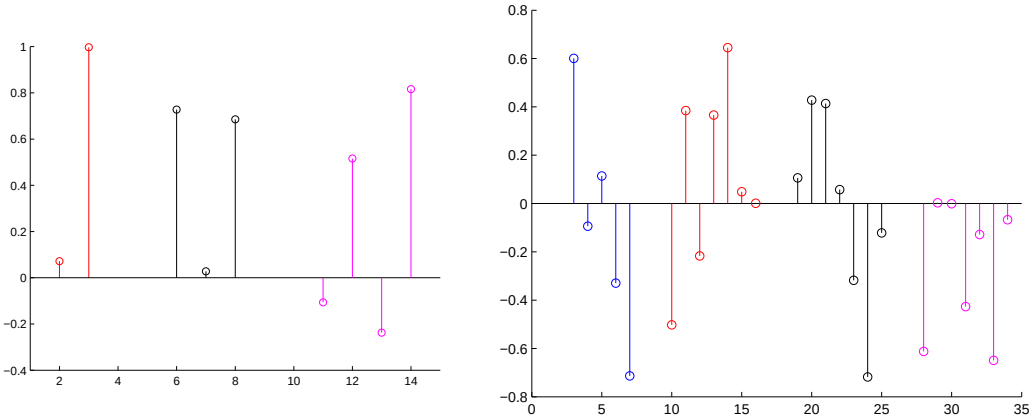

Figure 2.4: The two bottom graphs show prototype atoms of random shape defining a realization of the RR dictionary.

in the top left graph of Fig. 2.3. The other prototypes of support 2,4,6, and 8, represented in the top right graph of Fig. 2.3, are discretized versions of Haar wavelets.

The 7 sets of discrete vectors are formed by translating each of the 7 discrete wavelet prototypes, shown in Fig. 2.3, one sample point at each translation step. The 'cut off' approach described in Appendix B, is applied at the boundaries, keeping the elements of the vectors which have nonzero intersection with the discrete interval N_c being considered. The dictionary $\mathbf{D}_5^c$ referred to as the Redundant Discrete Wavelet (RDW) dictionary is constructed with these 7 sets of vectors as its columns.

Redundant Random Dictionary

The 3 dictionaries described above are each formed by translations of prototype supported vectors. To verify there sparsity 5 additional dictionaries formed by translating normally distributed pseudorandom supported vectors, were also constructed.

A single realization of the pseudorandom prototype vectors of support $j = 1, 2, 3, 4, 5, 7, 7$ and 7 is shown at the bottom of Figure 2.4

The 7 sets of discrete, atoms of identical support, are formed by translating each prototype pseudorandom atom one sample point at each translation step with the 'cut off' approach being applied at the boundaries. The dictionary $\mathbf{D}_6^c$ referred to as the RR dictionary is constructed with these 7 sets of vectors as its columns.

Trained Separable Dictionary

This type of dictionary is calculated or trained using test images which are representative of a given image corpus. Trained dictionaries may not generalize well to producing sparse representations of all images, however, the idea behind them is that they can produce sparser representations of there given corpus.

The training was performed with the Iterative Least Squares Dictionary Learning Algorithm (ILS-DLA), for unrestricted block based dictionaries [81]. This was implemented with a slightly modified version of the software provided by Karl Skretting [82], the implementation details of which are given in his PhD thesis [36]. The modification was to replace the OOMP algorithm with OMP2D to allow larger separable dictionaries to be trained. A description of this separable ILS-DLA with OMP2D is given in Appendix C.

Figure 2.5 shows the first 3 vectors from 4 Trained Separable (TS) column dictionaries, each one trained with a different N_c, with $N_c = 8, 16, 24$ and 32. The TS dictionaries were trained with Q, $N_c \times N_r$ blocks $\mathbf{X}_q$, randomly sampled from 10 of the top 100 images captured by the Hubble Telescope [83]. Figure 2.5 shows the first 3 vectors in each of the TS dictionaries, trained with $N_c = 8, 16, 24$ and 32. It is clear that for these vectors, the ones with the same index in each dictionary have similar shapes. This feature is present for the lower indexed atoms in all dictionaries, however the larger indexed atoms in all the dictionaries appear to have a random construction. This can be seen in Figure 2.6 which shows atoms $50, 51$ and 52 taken from the same 4 dictionaries with $N_c = 8, 16, 24$ and 32.

2.5 Sparsity of Greedy Algorithms

In the Experiment of this Section sparse image approximations are calculated by several greedy algorithms and a fixed separable dictionary. The investigation here is to access the performance of each algorithm in terms of sparsity and execution time.

Before a large image $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$ can be approximated by the greedy algorithms described in the previous Sections, it is first divided into smaller non overlapping square images or blocks $\mathbf{I}_q \in \mathbb{R}^{N \times N}, q = 1, \ldots, Q^N$. The approximation of these blocks requires less memory, resulting in a reduction in the overall processing time for an image. The K

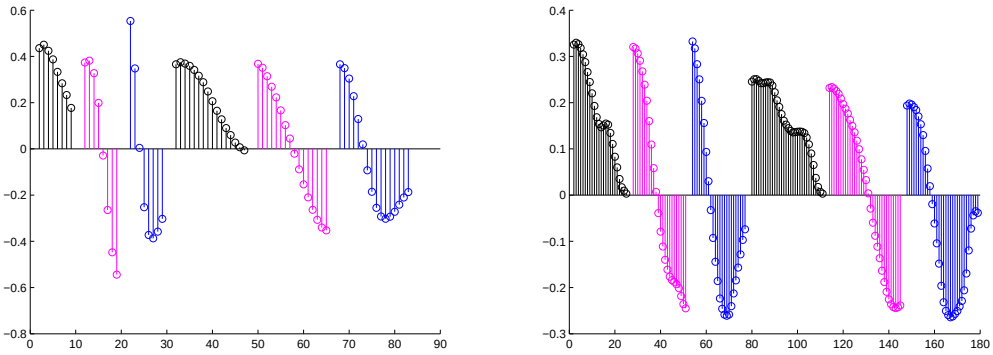

Figure 2.5: Vectors taken from the TS dictionaries calculated with the procedure described in Appendix C. Each dictionary was trained on Q blocks $\mathbf{X}_q$, randomly sampled from 10 of the top 100 images captured by the Hubble Telescope. The Figure on the left shows, from left to right, the first 3 atoms, taken from a dictionary trained with $N_c = 8$, and a dictionary trained with $N_c = 16$. The Figure on the right shows from left to right, the first 3 atoms, taken from a dictionary trained with $N_c = 24$, and a dictionary trained with $N_c = 32$.

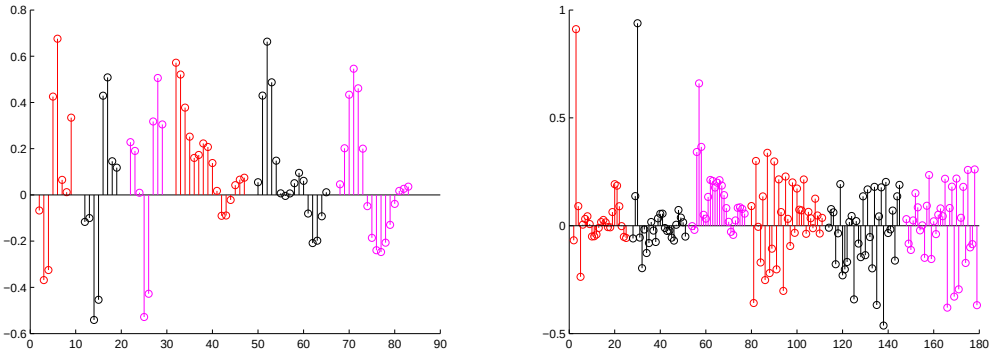

Figure 2.6: Vectors taken from the TS dictionaries calculated with the procedure described in Appendix C. Each dictionary was trained on Q blocks $\mathbf{X}_q$, randomly sampled from 10 of the top 100 images captured by the Hubble Telescope.The Figure on the left shows from left to right, atoms $50, 51$ and 52, taken from a dictionary trained with $N_c = 8$, and a dictionary trained with $N_c = 16$. The Figure on the right shows from left to right, atoms $50, 51$ and 52, taken from, a dictionary trained with $N_c = 24$, and a dictionary trained with $N_c = 32$.

sparse approximation $\mathbf{I}_q^K$ of each of these smaller blocks $\mathbf{I}_q$ is,

$$\mathbf{I}_q^K = \sum_{k=1}^{K_q} \mathbf{D}^c(:, \mathbf{l}_q^c(k)\mathbf{c}_q(k)(\mathbf{D}^r(:, \mathbf{l}_q^r(k)))^T, q = 1, \ldots, Q^N, \tag{2.15}$$

where the notation Q^N is to indicate the dependence of the number of blocks on the variable N. This is the same as equation (2.13) with the addition of the subscript q indicating the image block $\mathbf{I}_q$ being approximated. For simplicity, in this and future discussions it is assumed the original image can be exactly divided into an integer number of blocks.

The sparsity of each approximation will be reported by the SR, defined as

$$\text{SR} = \frac{N_p}{K}, \tag{2.16}$$

with $N_p = Q^N N^2$, the number of pixels in the original image, and

$$K = \sum_{q=1}^{Q^N} K_q, \tag{2.17}$$

the number of coefficients used in the approximation in equation (2.15).

2.5.1 Experiment

The approximations in this Experiment were performed with four separable greedy algorithms resulting in a representation of the form in (2.15), and the non separable greedy algorithm OOMP described in Section 1.5.3. All the algorithms were implemented in C++ mex files to reduce the execution time in the approximation. The source code is available from [84].

The four separable greedy algorithms were:

1) Matching Pursuit (MP2D),

2) Self Projected Matching Pursuit with step length $p = 1$ (SPMP2D_1),

3) Self Projected Matching Pursuit with step length $p = 10$ (SPMP2D_{10}),

4) Orthogonal Matching Pursuit (OMP2D).

Each algorithm above chose atoms from the same column and row dictionaries, denoted respectively by $\mathbf{D}_2^{c,1}$ and $\mathbf{D}_2^{r,1}$, with $\mathbf{D}_2^{c,1} \equiv \mathbf{D}_2^{r,1}$. The column dictionary was constructed from the concatenation of the RDC and RDBS dictionaries described in Section 2.4, as $\mathbf{D}_2^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_3^c]$.

The approximation was also performed by the non separable greedy algorithm OOMP. This algorithm required a large separable dictionary $\mathbf{F}$, which was explicitly calculated as the Kronecker product (defined in Appendix A.4) between the column and row dictionaries, shown below

$$\mathbf{F} = \mathbf{D}_2^{r,1} \otimes \mathbf{D}_2^{c,1}. \tag{2.18}$$

Each column of $\mathbf{F}$ is a separable atom and the equivalent approximation to equation (2.15) is

$$\mathbf{I}_q^K = \sum_{k=1}^{K} \mathbf{c}_q(k) \widehat{\mathbf{Mat}}(\mathbf{F}(:, \mathbf{l}_q(k)), N, N), q = 1, \ldots, Q^N. \tag{2.19}$$

The vectors $\mathbf{l}_q$ contain the indices of the column within $\mathbf{F}$ chosen at each iteration of the OOMP algorithm. The $\widehat{\mathbf{Mat}}(\cdot)$ operation above relabels the elements of an $NN \times 1$ vector to become elements of an $N \times N$ matrix, applying the convention described in Appendix A.2.1.

This Experiment was performed with a test set of 45, 8 bit grey level versions of astronomical images, taken from the top 100 images captured by the Hubble telescope. The images which had an average resolution of 1168×1280 were approximated with four block sizes $N = 8, 16, 24$ and 32. Each image was originally 24 bit true colour, and was converted to 8 bit grey level by applying the weighted sum described in Section 1.3.2. The approximation was to a global PSNR^a, between the original image $\mathbf{I}$ and the approximated image $\mathbf{I}^K$, with the superscript a indicating that the PSNR is the result of an approximation. The calculation of the PSNR is given in equation (1.3).

Converging to a Global PSNR^a

Approximating an image $\mathbf{I}$ to a global PSNR^a, when the processing is performed in blocks and each block is approximated independently, can be achieved in several ways. In this and future Experiments, unless stated otherwise, the following global convergence approach will be applied.

If the error in the approximation of each block is exactly equal to ρ, that is if

$$\|\mathbf{I}_q - \mathbf{I}_q^K\|_F^2 = \rho, \quad q = 1, \ldots, Q^N \tag{2.20}$$

then ρ can be explicitly calculated from the desired PSNR, xdB as

$$\rho = \frac{N^2 \mathrm{MAX}_{\mathbf{I}}^2}{10^{\frac{-x}{10}}}. \tag{2.21}$$

In practice it is not possible for all blocks in equation (2.20) to be approximated to the same error ρ. Therefore the equality in equation (2.20) is replaced with an inequality, shown in equation (2.12). This guarantees that the PSNR$^a \geq x$dB.

	OMP2D		SPMP2D$_1$		SPMP2D$_{10}$		MP2D		OOMP	
N	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\bar{\mathbf{t}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\bar{\mathbf{t}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\bar{\mathbf{t}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\bar{\mathbf{t}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\bar{\mathbf{t}}$
8	12.36	11.26	12.46	13.70	12.18	12.20	11.72	12.37	12.77	235.97
16	14.35	38.11	14.42	45.23	14.13	34.22	13.21	28.52	14.47	6073.05
24	14.94	113.27	14.96	111.44	14.74	91.11	13.59	70.66	NA	NA
32	15.23	326.09	15.22	237.79	15.05	207.47	13.78	134.73	NA	NA

Table 2.1: Average SR ($\bar{x}_{SR}$) and average processing time ($\bar{t}$) in seconds over the set of 45 grey level astronomical images. The approximation of each one was to a PSNR^a of 45dB $\pm$ 4.5 $\times$ 10^{-3}dB, by applying the algorithms shown, with the combined RDC and RDBS dictionary. The average size of the images in the set was 1264 $\times$ 1194. The processing time for each image is the average of 5 independent runs, therefore the average processing time is the average of this over the image set. The $\bar{x}_{SR}$ and $\bar{t}$ are not shown for OOMP with N greater than 16 because the problem size was too large to calculate.

An approximation to a desired PSNR of say, x = 45dB is then carried out by first approximating all image blocks $\mathbf{I}_q, q = 1, \ldots Q$ to an error less than or equal to ρ, calculated with x = 45dB in equation (2.21). Then if necessary the value of x in (2.21) is iteratively reduced until $|45 - \text{PSNR}^a| < 45 \times 10^{-2}\%$.

Each image in the test set was approximated independently 5 times to 45dB$\pm$4.5$\times 10^{-3}$ for block sizes $N = 8, 16, 24$ and 32, by applying MP2D, SPMP2D$_1$, SPMP2D$_{10}$, OMP2D and OOMP. The average SR, $\bar{x}_{SR}$ and execution time, $\bar{t}$ in seconds are shown in Table 2.1. The $\bar{x}_{SR}$ is the mean taken over the image test set. The recorded processing time for each image is the mean of the 5 independent approximations, therefore $\bar{t}$ is the mean of this value taken over the image test set.

2.5.2 Results

SR Increases with N

The results in Table 2.1 show that for all the greedy algorithms tested the $\bar{x}_{SR}$ of the approximations increased with N (the size of the blocks) partitioning the image. As shown in Table 2.1 this increase comes at a price because again for all the greedy algorithms tested the $\bar{t}$ also increased with the block size.

OOMP, Sparser but Slower

Table 2.1 shows that for $N = 8$ and 16, the $\bar{x}_{SR}$ over the 45 astronomical images is highest when atoms are picked with the OOMP algorithm. What is not shown is that for $N = 8$

and 16 the SR for every image when approximated with OOMP was always higher than the SR for the other algorithms in the Table.

For $N = 8$ and $N = 16$ the percentage increase in $\bar{x}_{SR}$ resulting from the application of OMP2D instead of MP2D is greater than the percentage increase in $\bar{x}_{SR}$ resulting from applying OOMP instead of OMP2D. At the same time the increase in execution time from OMP2D to OOMP is $2.00 \times 10^3\%$ and $1.59 \times 10^4\%$ for respectively $N = 8$ and $N = 16$. This is a far greater increase in processing time than going from MP2D to OMP2D, which, for $N = 8$ falls by 9.01% and only increases by 33.65% for $N = 16$.

OMP2D Sparser than MP2D and SPMP2D$_{10}$

For all N the SR for every astronomical images is greater for OMP2D than for MP2D.

The $\bar{x}_{SR}$ for OMP2D, shown in Table 2.1, for each N is larger than for SPMP2D$_{10}$. However OMP2D did not produce a sparser approximation of every image in the test set. Therefore a one tailed paired sample t-test was performed to determine if OMP2D produces significantly sparser approximations than SPMP2D$_{10}$. The results given in Appendix E.1.1, show that for all values of N the average SR made with OMP2D is significantly higher than the average SR of approximating the same images with SPMP2D$_{10}$. This is at the 95% confidence level, and applies to approximations of astronomical images made with the combination of the RDC and RDBS dictionary.

SPMP2D$_1$ Faster for Larger Blocks with $N = 32$

Table 2.1 shows that the $\bar{t}$ for OMP2D is greater than SPMP2D$_1$ for $N \geq 24$. To determine if SPMP2D$_1$ should be used instead of OMP2D to reduce the processing time for $N \geq 24$, two, one tailed paired sample t-tests were performed. The first to determine if the average SR produced by approximations of astronomical images with OMP2D is significantly higher than SPMP2D$_1$. The second to determine if the average execution time of approximations made with SPMP2D$_1$ is significantly smaller than OMP2D for $N \geq 24$.

The result of these two significance tests shown in Appendix E.1.1, indicate that with large $N = 32$, the SPMP2D$_1$ algorithm should be used in place of OMP2D, to reduce the execution time in the approximation. This is for astronomical images and the combination of the RDC and RDBS dictionary.

Discussion

For the astronomical image corpus approximated by choosing atoms from the combined RDC and RDBS dictionary for all algorithms tested the $\bar{x}_{SR}$ increased with the block size

N. This is an important result and needs to be investigated further to determine if it is a general feature, or a result of the specific images or dictionary tested.

When performing sparse approximations of images with greedy algorithms, the choice of algorithm will depend on the importance or trade off between, sparsity and/or processing time. The results of this experiment indicate that OMP2D fulfils both criteria for approximations of astronomical images. This is compared to the other algorithms tested, when the approximation is made to a $\text{PSNR}^a = 45\text{dB} \pm 4.5 \times 10^{-3}$ and the combination of the RDC and RDBS dictionary is applied.

If processing time is the main priority when approximating images, the results indicate OMP2D is the algorithm to choose. This is because OMP2D had the smallest processing time on average of the five algorithms tested, for all block sizes under the above test conditions.

If the sparsity of the approximation is more important, the results indicate that either OMP2D or SPMP2D_1 are suitable, because they produce the sparsest approximations. However OMP2D takes significantly longer than SPMP2D_1 for $N = 32$, and in this experiment took on average 37.13% times longer to process each image with this block size.

The above result for SPMP2D_1 was not unexpected. As mentioned earlier both OMP2D and SPMP2D_1 have complexity of the same order, however SPMP2D_1 has a much smaller memory footprint. This is because the large matrices $\mathbf{B}_k^{(K)} \in \mathbb{R}^{N\times N}, k = 1, \ldots, K$ required by OMP2D in equation (2.10) do not need to be calculated and stored by the SPMP2D_1 algorithm.

If OOMP could be applied to larger image blocks than $N = 16$ the results of this Experiment indicate it would produce the sparsest approximations of the algorithms tested. This increase in sparsity comes at a price because of the massive increase in processing time that the algorithm requires. Therefore OOMP is not considered suitable for approximating astronomical images, both because it cannot be applied to large enough blocks, and because the processing time is much higher than the other algorithms, for the same average SR.

Finally OMP2D produces significantly sparser approximations than both MP2D and SPMP2D_{10} for all N, these algorithms are therefore also not suitable for approximating astronomical images with the combined RDC and RDBS dictionary.

For the above reasons the OMP2D algorithm was chosen to produce the sparse approximations, required by the remaining Experiments of this Chapter.

2.6 Dictionary Selection

The results of the previous Section indicate that the OMP2D algorithm produces sparser representations of blocked astronomical images, with the combined RDC and RDBS dictionary, than the other algorithms tested. In this Section the SR resulting from choosing atoms with OMP2D with 4 more dictionaries of the same redundancy, and 2 smaller dictionaries is examined. This is compared with the SR resulting from the DCT and CDF9/7 Wavelet Transform which are respectively part of the lossy JPEG and JPEG2000 compression algorithms [17].

2.6.1 Experiment

The approximation procedure is the same as the previous Experiment in Section 2.5, with the addition or 6 extra dictionaries.

This Experiment was performed with the 45, 8 bit astronomical images and an additional set of 45, 8 bit grey level images natural images. The additional images were chosen randomly from the Berkeley Segmentation Dataset [85] containing 300 different true colour images. Each image was converted from 24 bit RGB to 8 bit grey level as prescribed by equation (1.1). The additional test set had an average resolution of 360×442, which is smaller average resolution than the astronomical (1168×1280) set.

Each image was first partitioned into blocks, with $N = 8, 16, 24$ and 32, before being approximated with OMP2D to a fixed $\text{PSNR}^a = 45\text{dB} \pm 4.5 \times 10^{-3}$.

Because OMP2D is separable, column and row dictionaries described in Section 2.3 were constructed for this Experiment. The first 5 dictionaries, who's components are described below, are symmetric with $\mathbf{D}_i^{c,1} = \mathbf{D}_i^{r,1}, i = 1, \ldots, 5$, therefore the construction mentioned is for either $\mathbf{D}_i^{c,1}$ or $\mathbf{D}_i^{r,1}$. The remaining two dictionaries are non symmetric with $\mathbf{D}_i^{c,1} \neq \mathbf{D}_i^{r,1}, i = 6, 7$, both containing the same number of vectors $M_c = M_r$, hence the construction below is for both.

2.6.2 Dictionaries

The following is a list of the 7 dictionaries which were used in this Experiment:

$\mathbf{D}_1^{c,1}$ - RDC column dictionary, $\mathbf{D}_1^{c,1} = \mathbf{D}_1^c$ (RDC).

$\mathbf{D}_2^{c,1}$ - combined RDC and RDBS column dictionary, $\mathbf{D}_2^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_3^c]$ (RDC-RDBS$_1$).

$\mathbf{D}_3^{c,1}$ - combined RDC, Euclidean Basis and RDW, column dictionary, $\mathbf{D}_3^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_4^c]$ (RDC-RDW).

N	$\mathbf{M_i}, \mathbf{i = 2, 3, 4, 6, 7}$	$\mathbf{M_5}$	$\mathbf{M_1}$
8	90	67	16
16	170	139	32
24	250	211	48
32	320	283	64

Table 2.2: The number of vectors $\mathbf{M_i}$ in each of the dictionaries $\mathbf{D}_i^{c,1}, \mathbf{D}_i^{r,1}, i = 1, \ldots, 7$ against the size of the block $\mathbf{N}$.

$\mathbf{D}_4^{c,1}$ - combined RDC, Euclidean Basis and RR, column dictionary, $\mathbf{D}_4^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_5^c]$ (RDC-RR). Five different realizations of the prototype pseudorandom prototype were tested in this Experiment, for each support length.

$\mathbf{D}_5^{c,1}$ - combined RDC, Euclidean Basis, RDU and a smaller B-spline $\mathbf{D}_9^c$, column dictionary, $\mathbf{D}_5^{c,1} = [\mathbf{D}_1^c, \mathbf{D}_2^c, \mathbf{D}_3^c, \mathbf{D}_9^c]$ (RDC-RDBS$_2$). The smaller B-spline dictionary $\mathbf{D}_9^c$ is constructed with the set of vectors below as its columns,

$$\{b_i Y_2^s(n - i + 1); i = n, \ldots, N + j - 1\}_{i=1}^{M_s}, \quad s = 2, 3, 4. \tag{2.22}$$

The support lengths for each prototype atom are $j = 3, 5$ and 7 for respectively $s = 2, 3$, and 4.

$\mathbf{D}_6^{c,1}$, $\mathbf{D}_6^{r,1}$ - TS dictionary, trained with an additional 10 astronomical images taken from the top 100 images captured by the Hubble telescope. Five realizations of this, TS$_1$ dictionary were trained. The ILS-DLA algorithm in Appendix C was applied, each realization was calculated by choosing a different set of $Q = 10,000$ image blocks from the training images. This dictionary was constructed to have the same number of atoms as dictionaries $\mathbf{D}_i^{c,1}, i = 2, 3, 4$.

$\mathbf{D}_7^{c,1}$, $\mathbf{D}_7^{r,1}$ - TS dictionary, trained with an additional 10 natural images chosen from the Berkeley Segmentation Dataset. The same procedure described above for the TS$_1$ dictionary was applied to create the five realizations of this TS$_2$ dictionary.

Table 2.2 shows that dictionary pairs $\mathbf{D}_i^{c,1}, \mathbf{D}_i^{r,1}, i = 2, 3, 4, 6, 7$ all contain the same number of vectors for a given N. Dictionary pairs $\mathbf{D}_1^{c,1}, \mathbf{D}_1^{r,1}$ and $\mathbf{D}_5^{c,1}, \mathbf{D}_5^{r,1}$ contain less vectors for each N.

The results for this Experiment are given in the next Section.

	N = 8		N = 16		N = 24		N = 32	
	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$
CDF9/7	5.12	2.90	5.12	2.90	5.12	2.90	5.12	2.90
DCT	7.58	4.07	5.94	4.08	5.57	4.12	5.35	4.09
$\mathbf{D}_1^{c,1}, \mathbf{D}_1^{c,1}$ **(RDC)**	9.04	4.87	8.09	5.59	7.78	5.77	7.59	5.82
$\mathbf{D}_2^{c,1}, \mathbf{D}_2^{c,1}$ **(RDC-RDBS$_1$)**	12.06	5.77	13.93	8.05	14.51	8.80	14.81	9.21
$\mathbf{D}_3^{c,1}, \mathbf{D}_3^{c,1}$ **(RDC-RDW)**	10.99	5.33	12.00	6.80	12.17	7.17	12.23	7.35
$\mathbf{D}_4^{c,1}, \mathbf{D}_4^{c,1}$ **(RDC-RR)**	10.71	5.08	11.69	6.39	11.92	6.73	12.03	6.91
$\mathbf{D}_5^{c,1}, \mathbf{D}_5^{c,1}$ **(RDC-RDBS$_2$)**	11.58	5.82	13.10	7.93	13.58	8.59	13.83	8.94
$\mathbf{D}_6^{c,1}, \mathbf{D}_6^{c,1}$ **(TS$_1$)**	12.95	6.29	14.94	9.01	14.90	9.44	14.54	9.48
$\mathbf{D}_7^{c,1}, \mathbf{D}_7^{c,1}$ **(TS$_2$)**	13.15	6.48	15.48	9.67	15.98	10.70	15.94	11.13

Table 2.3: Average SR ($\bar{x}_{SR}$) and standard deviation of the SR (s_{SR}) over the set of 45 grey level astronomical images, when approximated with OMP2D. Each image was approximated with the blocks sizes N shown and the column and row dictionaries $\mathbf{D}_i^{c,1}, \mathbf{D}_i^{r,1}, i = 1, \ldots, 7$, to a PSNR[a] of $45 \pm 4.5 \times 10^{-3}$dB. The CDF9/7 and DCT were performed by thresholding the smallest coefficients.

2.6.3 Results

The average SR ($\bar{x}_{SR}$) and standard deviation of the SR (s_{SR}) over the 45 grey level astronomical and natural images, are shown respectively, in Tables 2.3 and 2.4. The results are shown against the dictionaries and the block sizes $N = 8, 16, 24, 32$ tested. The $\bar{x}_{SR}$ and s_{SR}, are calculated over all the images in the respective image sets for all dictionary pairs, except $\mathbf{D}_i^{c,1}, \mathbf{D}_i^{r,1}, i = 4, 6, 7$ because they each have 5 realizations. For these dictionaries, the recorded SR for each image is an average over the 5 realizations, therefore for these dictionaries the $\bar{x}_{SR}$ and s_{SR} are calculated from this value over all images in the respective test sets.

The results of approximating both image sets with the DCT and CDF9/7 are also shown in Tables 2.3 and 2.4. The DCT is applied to images partitioned into square blocks of size N and the CDF9/7 is applied to whole images at once. The approximation is then performed by thresholding the smallest transform coefficients to get a non linear approximation.

Dictionaries, or the DCT and CDF9/7

Both Tables 2.3 and 2.4 show that there is a dramatic increase in sparsity when choosing vectors from the redundant dictionaries with OMP2D, a highly nonlinear approximation,

	N = 8		N = 16		N = 24		N = 32	
	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$	$\bar{\mathbf{x}}_{\mathbf{SR}}$	$\mathbf{s}_{\mathbf{SR}}$
CDF9/7	2.79	2.55	2.79	2.55	2.79	2.55	2.79	2.55
DCT	3.10	2.49	2.79	2.24	2.67	2.03	2.56	1.79
$\mathbf{D}_1^{c,1}, \mathbf{D}_1^{c,1}$ **(RDC)**	3.91	2.91	3.80	2.94	3.69	2.75	3.57	2.45
$\mathbf{D}_2^{c,1}, \mathbf{D}_2^{c,1}$ **(RDC-RDBS$_1$)**	5.86	3.92	6.58	5.51	6.73	5.74	6.76	5.82
$\mathbf{D}_3^{c,1}, \mathbf{D}_3^{c,1}$ **(RDC-RDW)**	5.70	3.81	6.20	5.09	6.25	5.08	6.24	5.09
$\mathbf{D}_4^{c,1}, \mathbf{D}_4^{c,1}$ **(RDC-RR)**	5.63	3.73	6.15	4.84	6.23	4.95	6.21	4.88
$\mathbf{D}_5^{c,1}, \mathbf{D}_5^{c,1}$ **(RDC-RDBS$_2$)**	5.44	3.80	6.01	5.11	6.10	5.16	6.09	5.17
$\mathbf{D}_6^{c,1}, \mathbf{D}_6^{c,1}$ **(TS$_1$)**	6.03	3.93	6.49	5.07	6.34	4.77	6.09	4.42
$\mathbf{D}_7^{c,1}, \mathbf{D}_7^{c,1}$ **(TS$_2$)**	6.17	4.03	6.82	5.53	6.85	5.62	6.66	5.35

Table 2.4: Average SR ($\bar{x}_{SR}$) and standard deviation of the SR (s_{SR}) over the set of 45 grey level natural images, when approximated with OMP2D. Each image was approximated with the blocks sizes N shown and the column and row dictionaries $\mathbf{D}_i^{c,1}, \mathbf{D}_i^{r,1}, i = 1, \ldots, 7$, to a PSNR[a] of $45 \pm 4.5 \times 10^{-3}$dB. The CDF9/7 and DCT were performed by thresholding the smallest coefficients.

instead of applying a nonlinear approximation with the orthogonal (DCT) and biorthogonal (CDF9/7) transforms. More specifically when the RDC dictionary, only containing twice as many discrete cosines as the DC basis, is applied to all the astronomical and natural images, the increase in sparsity is at least 12.53%. Simply enriching this dictionary with supported pseudorandom atoms increases the sparsity again by at least another 8.27%, however this comes at the cost of increasing the size of the dictionary more than 5 times.

Effect of Block Size N

Interestingly for both image sets, the $\bar{x}_{SR}$ falls when the block size N is increased if approximations are made with the DCT transform and the RDC dictionary. The approximation of the astronomical images with all other dictionaries excluding the TS ones, resulted in an increase in the $\bar{x}_{SR}$ with the block size N, agreeing with the results from Experiment 2.5.1. This result also held for approximations of natural images, but only up to $N = 24$.

RDC-RDW and RDC-RR

Tables 2.3 and 2.4 show that for the block sizes tested, the $\bar{x}_{SR}$ for the RDC-RDW dictionary is higher than for the RDC-RR dictionary, however this was not true for all

images in the test sets. To determine if this was significant a one tailed paired sample t-test was performed, described in Appendix E.1.2. The results show that approximations with the RDC-RDW dictionary are significantly sparser than those made with the RDC-RR dictionary, for both astronomical and natural image corpus, at a 95% confidence level.

B-Spline Based Dictionaries, RDC-RDBS$_1$ and RDC-RDBS$_2$

The introduction of the discrete B-splines instead of the discrete wavelets in the RDC-RDBS$_1$ dictionary maintained the redundancy of the dictionary and also increased $\bar{x}_{SR}$ for both image sets, shown in Tables 2.3 and 2.4. For all values of N the SR increased by at least 4.05% for every astronomical image tested. The SR did not increase for every natural images tested. Therefore a one tailed paired sample t-test was performed, at the 95% confidence level, on the results from approximating the natural images with OMP2D. The result of this test, discussed in Appendix E.1.2, were, for all N there was a significant increase in the average SR produced by the RDC-RDBS$_1$ dictionary over the average SR produced by the RDC-RDW dictionary.

To further analyse the SR produced by B-spline based dictionaries, an additional one tailed paired sample t-test was performed. The test was to establish if the average SR produced by approximations made with the smaller RDC-RDBS$_2$ dictionary, were also sparser than the larger RDC-RDW dictionary. The results for this test performed at a 95% confidence level are discussed in Appendix E.1.2. They show that, on average the smaller RDC-RDBS$_2$ dictionary, produced significantly sparser approximations than the RDC-RDW dictionary, for astronomical but not for natural images.

TS Dictionaries, the Largest $\bar{x}_{SR}$

The dictionary which had the highest $\bar{x}_{SR}$ over both grey level image sets was TS$_2$, the dictionary trained on the 10 grey level natural images. To determine if this increase was significant a one tailed paired sample t-test, discussed in Appendix E.1.2, was performed at a 95% confidence level. The results of the test were, approximations made by the TS$_2$ dictionary had significantly higher average SR than those made by the RDC-RDBS$_1$ dictionary, for both image sets and all values of N. Interestingly this dictionary also produced higher $\bar{x}_{SR}$ over the astronomical images than the dictionary trained on this corpus.

Discussion

The results above show that for both grey level astronomical and natural images, choosing atoms from redundant dictionaries with OMP2D produces significantly sparser representations than the DCT or CDF9/7 currently used as part of the JPEG and JPEG2000 compression algorithms. This is when the approximations are to a $\text{PSNR}^a = 45\text{dB} \pm 4.5 \times 10^{-3}$. The results suggest the possibility of including the greedy approximation approach as part of an alternative image compression scheme. An idea which is investigated further in the next Section.

For most of the dictionaries tested, the $\bar{x}_{SR}$ over the astronomical and natural images was highest for respectively $N = 32$ and $N = 24$. The smaller average resolution of the natural images is one possible reason for this reduction in $\bar{x}_{SR}$ over the natural images, for the largest block size. This suggests that higher resolution images may benefit to a higher degree from being processed with larger blocks.

The increase in the SR of all images tested from simply approximating with the RDC dictionary instead of the DCT is significant. As a result, with the exception of the TS dictionaries, the RDC vectors were included in the construction of all other dictionaries.

The sparsest approximations of both sets of images were produced by TS dictionaries trained on natural images. Interestingly this dictionary produced a larger $\bar{x}_{SR}$ over the astronomical images than the dictionary trained on this corpus, a result which requires further investigation.

Even though the sparsest results were produced by the TS dictionaries, the results for the B-spline enriched dictionaries were promising, especially on astronomical images.

2.7 Image Coding

From equation (2.19) it is clear that to reproduce the sparse approximation of an image, both the coefficients $\mathbf{c}_q$ and the corresponding atom indices $\mathbf{l}_q$, are required for each of the $q = 1, \ldots, Q^N$ blocks.

The following describes a straight forward method to create a vector of bits (0's and 1's), or bit stream, $\mathbf{b}$ containing all this information. The simulations in Section 2.8 then demonstrate that the length of $\mathbf{b}$ can be smaller than that produced by the JPEG2000 algorithm when the same approximation quality is used.

The procedure for creating this bit stream $\mathbf{b}$ involves:

1) Quantization of the coefficients $\mathbf{c}_q, q = 1, \ldots, Q^N$.

2) Preprocessing of the quantized coefficients and atom indices $\mathbf{l}_q, q = 1, \ldots, Q^N$ to generate symbols suitable for entropy coding.

3) Entropy coding the symbols from step 2).

To motivate the need for quantization, first consider the definition of the CR, which is closely linked to the SR defined in equation (2.16),

$$\text{CR} = \frac{uN_p}{N_b}. \tag{2.23}$$

The variable u is the number of bits used to represent each pixel in the original image, and N_b is the number of bits required in the compressed representation. Instead of considering N_b to be the number of bits required in the approximation, consider it to be just the number of bits required to store the coefficients from equation (2.15).

During the approximation process, all of the calculations and the resulting coefficients in equation (2.15), are carried out in double 64 bit precision. The value of N_b if the coefficients are stored to this precision is $64K$, the resulting CR being

$$\text{CR} = \frac{8N_p}{64K} = \frac{1}{8}\text{SR}.$$

Given that the SR for some of the images in the Experiment in Section 2.6 was as low as 2.86, the CR for storing just the coefficients would be less than 1, that is instead of compressing the image the required storage for just the coefficients is larger than that of the original image. As a result, the first step in this image coding scheme is to quantize the coefficients, the approach adopted here is to apply a simple mid-tread uniform quantizer, described in the next Section.

2.7.1 Mid-tread Uniform Quantization

Given an input vector $\mathbf{v}(n), n = 1, \ldots, N$, the quantization indices $\mathbf{v}^{\triangle}$ are calculated by the forward quantization stage below:

$$\mathbf{v}^{\triangle}(n) = \text{sign}(\mathbf{v}(n)) \left\lfloor \frac{|(\mathbf{v}(n)|}{\triangle} + \frac{1}{2} \right\rfloor, \quad n = 1, \ldots, N, \tag{2.24}$$

where $\triangle$ is the quantization step size and $\lfloor x \rfloor$ indicates the largest integer not greater than x. The sign function in equation (2.24) is defined as

$$\text{sign}(x) = \begin{cases} -1 & \text{if } x < 0, \\ 1 & \text{if } x \geq 0. \end{cases} \tag{2.25}$$

The forward quantization operation will be denoted as $\widehat{\mathbf{FQuant}}(\cdot, \triangle)$, and is applied to a vector $\mathbf{v}$ with the following syntax

$$\mathbf{v}^{\triangle} = \widehat{\mathbf{FQuant}}(\mathbf{v}, \triangle). \tag{2.26}$$

All that is then required to recover a quantized version $\tilde{\mathbf{v}}$ of $\mathbf{v}$, and therefore all that needs to be stored, is the quantization indices $\mathbf{v}^{\triangle}$ and the quantization step size $\triangle$. The construction of the quantized vector can then be performed as

$$\tilde{\mathbf{v}}(n) = \mathbf{v}^{\triangle}(n)\ \triangle \quad n = 1, \ldots, N, \tag{2.27}$$

which is indicated by the following operation

$$\tilde{\mathbf{v}} = \widehat{\mathbf{RQuant}}(\mathbf{v}^{\triangle}, \triangle). \tag{2.28}$$

2.7.2 Convergence through Quantization

The vector of coefficients $\mathbf{c}_q, q = 1, \ldots, Q$ for each block can be quantized by applying the above two steps, first to calculate the quantization indices for each coefficient,

$$\mathbf{c}_q^{\triangle} = \widehat{\mathbf{FQuant}}(\mathbf{c}_q, \triangle)$$

and then calculate the quantized coefficient values

$$\tilde{\mathbf{c}}_q = \widehat{\mathbf{RQuant}}(\mathbf{c}_q^{\triangle}, \triangle). \tag{2.29}$$

A quantized version of the approximated image can then be written as,

$$\tilde{\mathbf{I}}_q^{K_q} = \sum_{k=1}^{K_q} \tilde{\mathbf{c}}_q(k) \widehat{\mathbf{Mat}}(\mathbf{F}(:, \mathbf{l}_q(k))), \quad q = 1, \ldots, Q^N. \tag{2.30}$$

The coding scheme proposed here applies this quantization procedure, as an alternative to the iterative method from Section 2.5, to converge to a global PSNR^a. This operates by first calculating the approximation with the error ρ calculated in equation (2.12), guaranteeing a global PSNR^a greater than or equal to the requested one. The coefficients in this higher quality approximation are then quantized with a $\triangle$, resulting in the desired PSNR^a.

Remark 1. In the Experiment in Section 2.5, when the approximation error ρ was calculated with equation (2.21), the global PSNR^a of the approximations were always far higher than the desired PSNR^a. Therefore the quantization of the coefficients in the Experiment of the next Section always resulted an adequate reduction in the number of bits required to store them. If this were not true then all that would be required is an increase in the value of ρ.

2.7.3 Storage

It is clear from equation (2.30) and (2.29) that the q'th approximated block $\tilde{\mathbf{I}}_q^{K_q}$, can be recovered if the vector $\mathbf{c}_q^{\triangle}$ of quantization indices, $\triangle$ and the corresponding vector $\mathbf{l}_q$ are known. If these values are stored in combination with the index of the block they correspond to, the entire approximated image can be recovered.

The procedure for storing the information above starts with the creation of 4 larger vectors, $\mathbf{c}_1^{\triangle f}$, $\mathbf{c}_1^{\triangle f}$, $\mathbf{q}$ and $\mathbf{l}^f$ each containing K elements with K calculated in (2.17),

The contents of each of these four vectors is described below:

1) The vectors $\mathbf{c}_q^{\triangle}$ for each block are first placed into a larger vector $\mathbf{c}^{\triangle f}$ by applying $\widehat{\mathbf{Flat}}(\cdot)$ operation, defined in Appendix A.2.3,

$$\mathbf{c}^{\triangle f} = \widehat{\mathbf{Flat}}(\mathbf{c}_q, q = 1, \ldots, Q^N).$$

This vector is then decomposed into a two more vectors, $\mathbf{c}_1^{\triangle f}$ containing the absolute value of each element in $\mathbf{c}^{\triangle f}$, and $\mathbf{c}_2^{\triangle f}$ containing the result of applying the sign() function (2.25) to each element in $\mathbf{c}^{\triangle f}$.

2) The vector $\mathbf{q}$ contains the block index information for each element in $\mathbf{c}^{\triangle f}$. That is for each element in $\mathbf{c}^{\triangle f}$, $\mathbf{q}$ contains the block index q which is required in equation (2.30).

3) The vector $\mathbf{l}^f$ contains the atom indices and is created as,

$$\mathbf{l}^f = \widehat{\mathbf{Flat}}(\mathbf{l}_q, q = 1, \ldots, Q^N).$$

The vectors $\mathbf{c}_2^{\triangle f}$, $\mathbf{q}$, $\mathbf{l}^f$ and $\mathbf{c}_1^{\triangle f}$ are stored respectively as the first four columns of a matrix $\mathbf{T}$.

The final bit stream $\mathbf{b}$ is now created by a similar procedure to that presented in [86]. The first step is to sort the rows of $\mathbf{T}$ in lexicographic order. Columns 1 and 2 are then preprocessed. Then columns $2-4$ are separately entropy coded by the algorithm described in Section 1.7.

Before the entropy coding stage the first two columns are preprocessed to reduce the number of symbols contained in them. Column 1 contains K elements and a maximum of two different values, $\{-1, 1\}$ representing the sign of the coefficients, each sorted in ascending numerical order. Therefore the number N_{-1} of -1 symbols and the total number of symbols K can be stored instead of storing the column itself.

Because the table has been sorted in lexicographical order the second column now contains a maximum of two runs of ascending values. The first, $\mathbf{T}(1 : N_{-1}, 2)$ are the

block indices corresponding to negative coefficients, and the second $\mathbf{T}(N_{-1}+1:K,2)$ are the block indices corresponding to the positive coefficients. To reduce the number of different values, the difference between each pair of ascending elements in each of these two runs is calculated. The result is then stored together with the first symbol of each run, in a new vector $\mathbf{t}$ as shown below

$$\begin{aligned} \mathbf{t}(1) &= \mathbf{T}(1,2), \\ \mathbf{t}(N_{-1}+1) &= \mathbf{T}(N_{-1}+1,2), \\ \mathbf{t}(k) &= \mathbf{T}(k,2) - \mathbf{T}(k-1,2), \quad k = 2,\ldots,N_{-1},N_{-1}+2,\ldots,K. \end{aligned} \tag{2.31}$$

The original elements of the second column of $\mathbf{T}$, can then be recovered by separately cumulatively summing the 2 runs of symbols stored in $\mathbf{t}$, as

$$\begin{aligned} \mathbf{T}(k,2) &= \sum_{i=1}^{k} \mathbf{t}(i), \quad k = 1,\ldots,N_{-1}, \\ \mathbf{T}(k,2) &= \sum_{i=N_{-1}+1}^{k} \mathbf{t}(i), \quad k = N_{-1}+1,\ldots,K, \end{aligned} \tag{2.32}$$

The elements contained in $\mathbf{t}$, and columns 3 and 4 of $\mathbf{T}$ are entropy coded to produce a vector of bits $\mathbf{b}$. The values of N_{-1} and K are represented by fixed length binary code words, and appended to the end of $\mathbf{b}$. The vector $\mathbf{b}$ now contains all the information required to recover the matrix $\mathbf{T}$ and therefore the quantized image approximation in equation (2.30).

The recovery of $\mathbf{T}$ from this can be performed in a straightforward manner. The first step is to read the values of N_{-1} and K and decode the vector $\mathbf{b}$, recovering $\mathbf{t}$ and columns 3 and 4 of $\mathbf{T}$. Next column 1 is populated using N_{-1} and K as

$$\mathbf{T}(1:N_{-1},1) = -1, \quad \mathbf{T}(N_{-1}+1:K,1) = 1.$$

Finally column 2 of $\mathbf{T}$ is recovered from $\mathbf{t}$ by applying the procedure shown in equation (2.32).

2.8 Image Compression

In this Section the sparse approximation of a blocked image produced by OMP2D is stored by the proposed image coding method described in Section 2.7. The resulting size of the bit stream $\mathbf{b}$ produced by this is then compared with the size of the files produced by the JPEG and JPEG2000 image compression algorithms.

In the Experiment in Section 2.6 the the largest $\bar{x}_{SR}$ over both image sets was produced by the dictionary trained on the natural image set. In that Experiment, for each N their

were 5 realizations of this dictionary and the SR presented was an average of these 5. In this Experiment approximations will again be performed with a TS dictionary for each N, with $N = 8, 16, 24$ and 32. This will be for each N the realization from Section 2.6 which resulted in the highest $\bar{x}_{SR}$ over the natural images. This collection of 4 TS dictionaries one for each $N = 8, 16, 24$ and 32 will be referred to as the TS_3 dictionary.

The reduction of a sparse approximation, made with OMP2D and the TS_3 dictionary, to a vector of bits $\mathbf{b}$, applying the coding method in Section 2.7, will be referred to as the dictionary coding algorithm.

Experiment

In this Experiment the images from both the astronomical and natural image sets were first partitioned into blocks of $N = 8, 16, 24$ and 32. Then they were approximated with OMP2D and dictionary TS_3 to 4 approximation quality levels. The number of bpp was then calculated from the bit stream $\mathbf{b}$, produced by storing the approximation information with the proposed coding scheme.

The implementation of both the JPEG and JPEG2000 algorithms applied in this Experiment was that provided by MATLAB's `imwrite()` function. The number of bpp required by both the JPEG and JPEG2000, was calculated from the size of their respective files generated by `imwrite()`.

Both JPEG2000 and the dictionary coding algorithm can converge to within $1 \times 10^{-2}\%$ of a desired PSNR^a. The JPEG algorithm's approximation quality is determined by an integer the range $1, \ldots, 100$, with 1 being the lowest and 100 being the highest quality of the approximation. Because of this JPEG is not guaranteed to approximate all images to within $1 \times 10^{-2}\%$ of the desired PSNR^a.

In this Experiment the comparison was performed for 4 approximation quality levels which would ideally have been a PSNR^a within $1 \times 10^{-2}\%$ of 30dB, 35dB, 40dB and 45dB. However as mentioned above, this is not possible with the JPEG algorithm, and, as a result, the JPEG approximation was performed first on all the images, to converge to the desired 4 levels of PSNR^a. The resulting PSNR^a produced by JPEG which was closest in absolute value to 30dB, 35dB, 40dB and 45dB, became the quality level for the other two methods to converge to. The PSNR^a shown in Figures 2.7 and 2.8 for each image set is therefore the average value which JPEG produced over all the images in the respective sets.

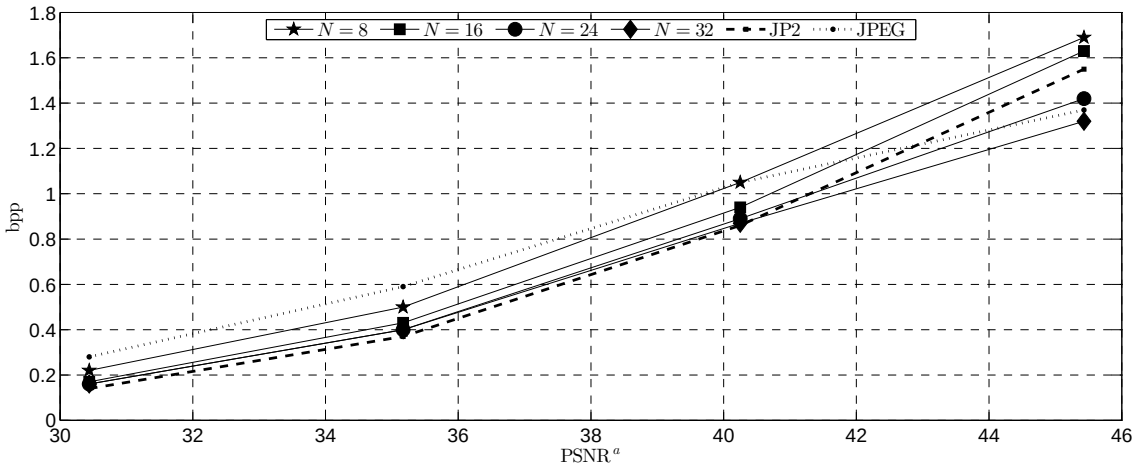

Figure 2.7: The average number of bpp required to store approximations of the astronomical images, made by choosing atoms from the TS_3 dictionary with OMP2D. The average number of bpp is shown for each block size $N = 8, 16, 24$ and 32, and for the JPEG and JPEG2000 compression algorithms. The average number of bpp is shown against the PSNR^a of the approximation for 4 levels of PSNR^a, 30.44dB, 35.17dB, 40.25dB and 45.43dB.

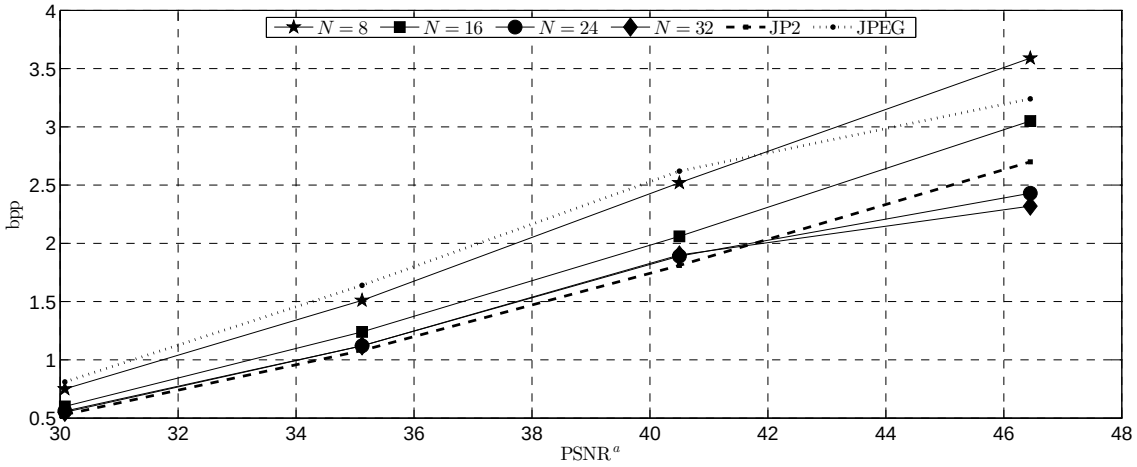

Figure 2.8: The average number of bpp required to store approximations of the natural images, made by choosing atoms from the TS_3 dictionary with OMP2D. The average number of bpp is shown for each block size $N = 8, 16, 24$ and 32, and for the JPEG and JPEG2000 compression algorithms. The average number of bpp is shown against the PSNR^a of the approximation for 4 levels of PSNR^a, 30.08dB, 35.12dB, 40.50dB and 46.45dB.

2.8.1 Results

The results for the 45 astronomical and natural grey level images are shown respectively in Figures 2.7 and 2.8. Both Figures show the average number of bpp produced by JPEG, JPEG2000 and the dictionary coding method with $N = 8, 16, 24$ and 32, against the PSNR^a.

Both Figures show the average number of bpp produced by the dictionary coding method is nearly always lowest for the largest block size $N = 32$. Therefore a one tailed paired sample t-test, described in Appendix E.1.3, was performed to see if this difference was significant. The t-test was performed on the results for both astronomical and natural images to a 95% confidence level. The result of the test for all quality levels is, the average number of bpp produced by the dictionary coding method, with $N = 32$ is significantly smaller than with $N = 8$ or $N = 16$. The result for the highest quality approximations tested is, the average bpp produced by the dictionary coding method, with $N = 32$, is significantly smaller than all other block sizes. Therefore the value of the bpp produced by the dictionary coding method for $N = 32$ was compared with both JPEG and JPEG2000 below.

Comparison with JPEG

The PSNR^a shown on the x axis of Figure 2.7 for the astronomical image set was 30.44dB, 35.17dB, 40.25dB and 45.43dB. For the first three quality levels, the dictionary coding method required less bpp than JPEG for every image, with the JPEG always requiring at least 4.24% bpp more when compressing a single image.

For the highest quality level a one tailed paired sample t-test described in Appendix E.1.3 was performed on the results for the astronomical images. The results of this t-test at a 95% confidence level are shown in the bottom row, of the top of Table E.6. The result is, approximations made to an average PSNR^a of 45.43dB with $N = 32$, require significantly less bpp on average, when coded with the dictionary method, than those compressed by the JPEG algorithm.

The PSNR^a shown on the x axis of Figure 2.8 for the natural image set was 30.08dB, 35.12dB, 40.50dB and 46.45dB. For all quality levels tested the dictionary coding method always required less bpp than JPEG, with the JPEG always requiring at least 5.25% bpp more when compressing a single image.

Figure 2.9: Astronomical test image approximated to $\text{PSNR}^a = 40.19\text{dB}$, compressed with both JPEG2000 and the dictionary coding method, required respectively 0.58 and 0.44bpp. The image contains 1280×1731 pixels.

Comparison with JPEG2000

The average number of bpp required by the dictionary coding method and the JPEG2000 compression algorithm were much closer for the quality levels tested. Therefore a one tailed paired sample t-test was performed on both grey level image sets to determine if they were significantly different. The t-test is described in Appendix E.1.3, and the results shown to a 95% confidence level in Table E.6. The main result is that for the $\text{PSNR}^a > 45\text{dB}$ tested, JPEG2000 requires on average significantly more bpp than the dictionary coding method.

On average, for the lower 3 quality levels, JPEG2000 did not require significantly more bpp than the dictionary coding method. However there were several images where it did, two examples where this occurred are now given. The astronomical image shown in Figure 2.9, approximated to $\text{PSNR}^a = 40.19\text{dB}$, compressed with both JPEG2000 and the dictionary coding method, required respectively 0.58 and 0.44bpp. The natural image shown in Figure 2.10, approximated to $\text{PSNR}^a = 40.00\text{dB}$, compressed with both JPEG2000 and the dictionary coding method, required respectively 1.16 and 1.06bpp.

Figure 2.10: Natural test image approximated to $\text{PSNR}^a = 40.00$dB, compressed with both JPEG2000 and the dictionary coding method, required respectively 1.16 and 1.06bpp. The image contains 321×481 pixels and is displayed at twice the resolution of Figure 2.9.

Discussion

The conclusions of the Experiments in Sections 2.5.1 and 2.6.1 were that there is an increase in sparsity from processing images in larger blocks. The results of this Experiment show that this translates into a reduction in the number of bpp required when the image is compressed. Specifically, the average number of bpp produced by the proposed dictionary coding algorithm, when the approximation is of high quality, is significantly lower for the largest block size ($N = 32$). This is encouraging and supports the use of the proposed SPMP2D_1 algorithm, which was shown to produce equivalent results to OMP2D in less time, as a first step in a dictionary coding algorithm.

The $\bar{x}_{SR}$ over the natural images, approximated with the TS_2 dictionary shown in Table 2.4, was lower for the largest block size $N = 32$, than for $N = 24$. However applying the dictionary coding method with the TS_3 dictionary resulted in the lowest average number of bpp for the largest block size of $N = 32$. The obvious question is now, how can an image with lower SR (when $N = 32$) produce a smaller compressed version than, an image with higher SR (when $N = 24$). The cause of this is now explained below.

When an approximation is performed with a larger block size N, it is possible for the number of different atom indices in the vector $\mathbf{l}^f$ to increase. On its own this can increase the storage requirements for the vector $\mathbf{T}(:, 3)$, because a larger range of integer values need to be entropy coded. However increasing the block size is guaranteed to reduce the number of blocks, and therefore block indices in $\mathbf{q}$. This reduces the number of different values in the vector $\mathbf{t}$ calculated with equation (2.31). Therefore the increase in block size resulted experimentally, in the average number of bpp for $N = 32$ falling below that of $N = 24$, shown in Figure 2.10. The fact that a reduction in the number of bpp can be a result of processing with larger N, and not a larger $\bar{x}_{SR}$, is interesting and demonstrates

another possible advantage to processing the images in larger blocks.

It is encouraging that for the PSNR^a tested, the simple dictionary coding method outperforms the older JPEG algorithm, in terms of CR. It is even more encouraging that for the highest quality approximations, the dictionary coding method also outperforms the newer JPEG2000 algorithm.

The proposed approximation and coding scheme is at an early stage, however it is still competitive with current image compression formats. This is an important result, implying that sparse image approximations produced by OMP2D, could be an important first step in a new image compression scheme.

2.9 Conclusions

Sparse representations of astronomical of images, produced by standard greedy selection algorithms were investigated. The OMP2D was algorithm applied to approximate this image corpus, by selecting atoms from the combined RDC and RDBS dictionary. This resulted in hight quality sparse approximations. The suitability of this algorithm for quickly processing small blocks, was demonstrated by the low average processing time over this class of images. As the block size N was increased, so were both the average SR, and processing time. To reduce the approximation processing time for large N, the SPMP2D$_1$ algorithm was proposed. SPMP2D$_1$ was then shown experimentally to produce equivalent approximations to OMP2D, in a shorter period of time.

It has been shown that for a variety of dictionaries OMP2D results in significantly sparser approximations, of both astronomical and natural images. This is when it is compared to the DCT and CDF9/7 transforms, currently employed as part of the JPEG and JPEG2000 image compression standards.

The SR results produced by OMP2D with the RDC dictionary are encouraging. Furthermore its combination with supported B-splines significantly increase the resulting SR of astronomical image data.

The increase in the SR resulting from approximations made with OMP2D inspired the dictionary coding scheme. This was shown experimentally, in Section 2.8, to compress astronomical and natural images, significantly better than JPEG, for a variety approximation qualities. More importantly, the proposed image coding scheme compressed higher quality approximations, of both astronomical and natural images, requiring significantly less bpp than JPEG2000.

The highest level of compression, for the proposed image coding scheme, resulted from

processing images with the largest block size, $N = 32$. Unfortunately processing images with OMP2D in larger blocks, increases the processing time. An algorithm termed SPMP2D$_1$ which produced equivalent results to OMP2D was then proposed. In the Experiment in Section 2.5, the SPMP2D$_1$ algorithm required significantly less time than OMP2D, to approximate images partitioned into blocks with $N = 32$. The result indicates that SPMP2D$_1$ is a valid alternative to OMP2D, for processing larger blocks to further increase the approximation sparsity.

The proposed approximation and coding scheme is competitive when compared to current image compression formats. This implies that sparse image approximations produced by greedy algorithms, could be an important first step in a new image compression scheme. This result is important, and encouraging, because the proposed coding scheme is not yet at an advanced stage.

3 Encrypted Image Folding

This Chapter describes and analyses a method for hiding information in the null space created by a sparse approximation of an image. The main idea stems from the fact that sparsity entails a projection onto a lower dimensional subspace, therefore creating a null space. Extra information can then be embedded and stably extracted from such a space.

The proposed idea can be applied to images as part of a partial encryption model taking advantage of image sparsity. The method termed image *folding*, takes a sparse approximation of an image and splits it into two sections, a host and embedded section. The embedded section is added to the host section to produce a *folded* image, this can then be stored in any conventional lossless image format. Both sections can then be fully recovered from the *folding* process, by applying an orthogonal projection. The security comes from securing the embedded section before it is *folded*, thus partially encrypting the image.

Two methods are discussed for protecting the embedded image. The first approach, based on a previously outlined method, is successfully applied to this particular image processing application. The second, is the security scheme described in the paper Sparsity and "Something Else": An Approach to Encrypted Image Folding [3].

The procedure can be applied to any sparse image representations including those realized by OMP2D in Chapter 2.

The contents of the Chapter are as follows: the first Section contains a general overview of the information embedding and recovery scheme. The next Section describes a specific application of this known as image *folding*. The following Section describes two methods for securing the *folded* information, including a number of simulations to determine the size of the *keyspace* for each method.

3.1 Information Embedding

Given an approximation $\mathbf{I}^K \in \mathbb{V}_K$ of an image array $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$ and denoting $\mathbb{V}^\perp$ to be the orthogonal complement of $\mathbb{V}_K$ in $\mathbb{R}^{N_r \times N_c}$, the embedding and retrieving principle is simple to describe: Any matrix $\mathbf{E} \in \mathbb{V}^\perp$ can be added and stably extracted from $\mathbf{I}^K \in \mathbb{V}_K$ with an orthogonal projector $\hat{P}_{\mathbb{V}_K}$ which acts by projecting onto $\mathbb{V}_K$ and along $\mathbb{V}^\perp$ in the way that is shown below,

$$\mathbf{I}^{f_1} = \mathbf{I}^K + \mathbf{E},$$

$$\hat{P}_{\mathbb{V}_K}\mathbf{I}^{f_1} = \hat{P}_{\mathbb{V}_K}(\mathbf{I}^K + \mathbf{E}) = \mathbf{I}^K,$$

$$\mathbf{E} = \mathbf{I}^{f_1} - \mathbf{I}^K.$$

This suggests the possibility of using the sparse representation of an image $\mathbf{I}^K \in \mathbb{V}_K \subset \mathbb{R}^{N_r \times N_c}$ as a host for embedding extra information. To achieve this a previously proposed scheme for embedding redundant representations [87], is applied to images as described below:

Embedding Scheme: Consider

$$\mathbf{I}^K = \sum_{k=1}^{K} \mathbf{c}(k)\mathbf{S}_k, \tag{3.1}$$

as the reconstruction of a sparse approximation of an image $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$ in the proper subspace $\mathbb{V}_K = \text{span}\{\mathbf{S}_k\}_{k=1}^K$. If the set of matrices $\{\mathbf{S}_k\}_{k=1}^K$ are linearly independent the dimension of $\mathbb{V}^\perp$, the orthogonal complement of $\mathbb{V}_K$ in $\mathbb{R}^{N_r \times N_c}$, is $N_e = N^2 - K$. Therefore a vector of N_e numbers denoted as $\mathbf{e}(n), n = 1, \ldots, N_e$ can be constructed to store coefficients for constructing a matrix $\mathbf{E} \in \mathbb{V}^\perp$. The numbers $\mathbf{e}(n)$ can be hidden and extracted from this embedded vector as prescribed below:

- Take an orthonormal basis $\mathbf{S}_n^\perp, n = 1, \ldots, N_e$ for $\mathbb{V}^\perp$ and form $\mathbf{E}$ as the linear combination

$$\mathbf{E} = \sum_{n=1}^{N_e} \mathbf{e}(n)\mathbf{S}_n^\perp. \tag{3.2}$$

- Add $\mathbf{E}$ to $\mathbf{I}^K$ to obtain $\mathbf{I}^{f_1} = \mathbf{I}^K + \mathbf{E}$.

Information Retrieval: Given $\mathbf{I}^{f_1}$ retrieve the vector of numbers $\mathbf{e}(n), n = 1, \ldots, N_e$ as follows.

- Construct an orthogonal projection operator $\hat{P}_{\mathbb{V}_K}$ onto the subspace $\mathbb{V}_K = \text{span}\{\mathbf{S}_k\}_{k=1}^K$, and remove the components in $\mathbb{V}^\perp$ from $\mathbf{I}^{f_1}$ as $\mathbf{I}^K = \hat{P}_{\mathbb{V}_K}\mathbf{I}^{f_1}$.
- From $\mathbf{I}^{f_1}$ and the recovered image $\mathbf{I}^K$ obtain $\mathbf{E}$ as $\mathbf{E} = \mathbf{I}^{f_1} - \mathbf{I}^K$.
- Retrieve the vector of numbers $\mathbf{e}(n), n = 1, \ldots, N_e$ from the recovered $\mathbf{E}$ with the orthonormal basis $\mathbf{S}_n^\perp, n = 1, \ldots, N_e$, with the Frobenius inner product

$$\mathbf{e}(n) = \langle \mathbf{S}_n^\perp, \mathbf{E} \rangle_F, \quad n = 1, \ldots, N_e. \tag{3.3}$$

The procedure above can be applied to any sparse approximation in a known subspace $\mathbb{V}_K$. Next is an overview of its application to the blocked image approximation produced in Chapter 2, in a procedure called image *folding*. The term *folding* describes the way a sparse representation of some image blocks, provides space for the coefficients from other blocks, to be embedded or *folded*.

3.2 Image Folding

The embedding procedure outlined above can be applied to sparse representations of images in a procedure known as image *folding*. A high level overview of this procedure is given in the next Section.

3.2.1 Overview

Image *folding* can be described as the process of taking a section of a sparse representation of an image and then *folding* it into the remaining section of the same image. Figure 3.1 shows a high level overview of the procedure when applied to the image of Bertrand Russell.

The procedure starts by taking the original image, shown at the top of Figure 3.1, and splitting it into into two sections. In this example, it is the top and bottom section of the original image, shown respectively to the left and right of the second row of Figure 3.1. It should be noted that this choice is arbitrary, and two other groups of pixels could have been chosen.

The next step is to generate a sparse approximation of the two sections. The sparse approximation in Figure 3.1 was realized by applying the DCT transform to 8×8 blocks

of pixels in each of the two sections. The coefficients were then quantized until the PSNR between the original image and the approximation was 45dB. Row three of Figure 3.1 shows the DCT coefficients remaining for each section following the quantization stage. For demonstration purposes the coefficients have been reshaped into rectangular blocks the same width as the original image, and rescaled to 8 bit grey intensity levels.

An approximation of the top half of the original image is constructed from its DCT coefficients. This is referred to as the host image and is shown on the left of the fourth row of Figure 3.1. The embedded image is shown on the right of the same row. This is constructed by applying equation (3.2) with the DCT coefficients from the sparse approximation of the bottom section of the original image.

The final procedure is to add, or *fold*, the embedded image into the host image. The resulting *folded* image is shown in the fifth row of Figure 3.1.

Clearly from equation (3.2) the *folded* image can be split back into the host and embedded images by, applying an orthogonal projection onto the space $\mathbb{V}_K = \text{span}\{\mathbf{S}_k\}_{k=1}^{K}$. It is also possible to recover the DCT coefficients from the embedded image by applying equation (3.3). The result of these two operations is shown on the left of the bottom row of Figure 3.1.

If the orthonormal basis $\mathbf{S}_n^{\perp}, n = 1, \ldots, N_e$ required in equation (3.3) is not know then it is not possible to recover the DCT coefficients from the embedded image. This prevents recovery of the bottom section of the image, shown on the right of the bottom row of Figure 3.1. In this situation bottom section is secured and the image is said to be partially encrypted.

The focus of this Chapter is therefore, on preventing the recovery of coefficients from the embedded section of an image, unless the correct *private key* is applied. To this end two security schemes are implemented which restrict access to the orthonormal basis $\mathbf{S}_n^{\perp}, n = 1, \ldots, N_e$.

In a similar way to the approaches described in Section 1.6.2, this *folding* procedure can be used to provide partial encryption of digital images. A suitable application of which would be to provide additional security in an online image distribution service. For example the procedure could be applied to all digital images available for sale on a public web site. This would allow customers to get an indication of the images before purchasing, with the full image only available to those in possession of the *private key*.

The next Section contains a more detailed description of the above procedure, with specific application to the blocked image approximations, produced in the last Chapter. The *folding* procedure will be applied to an approximation made by choosing matrices

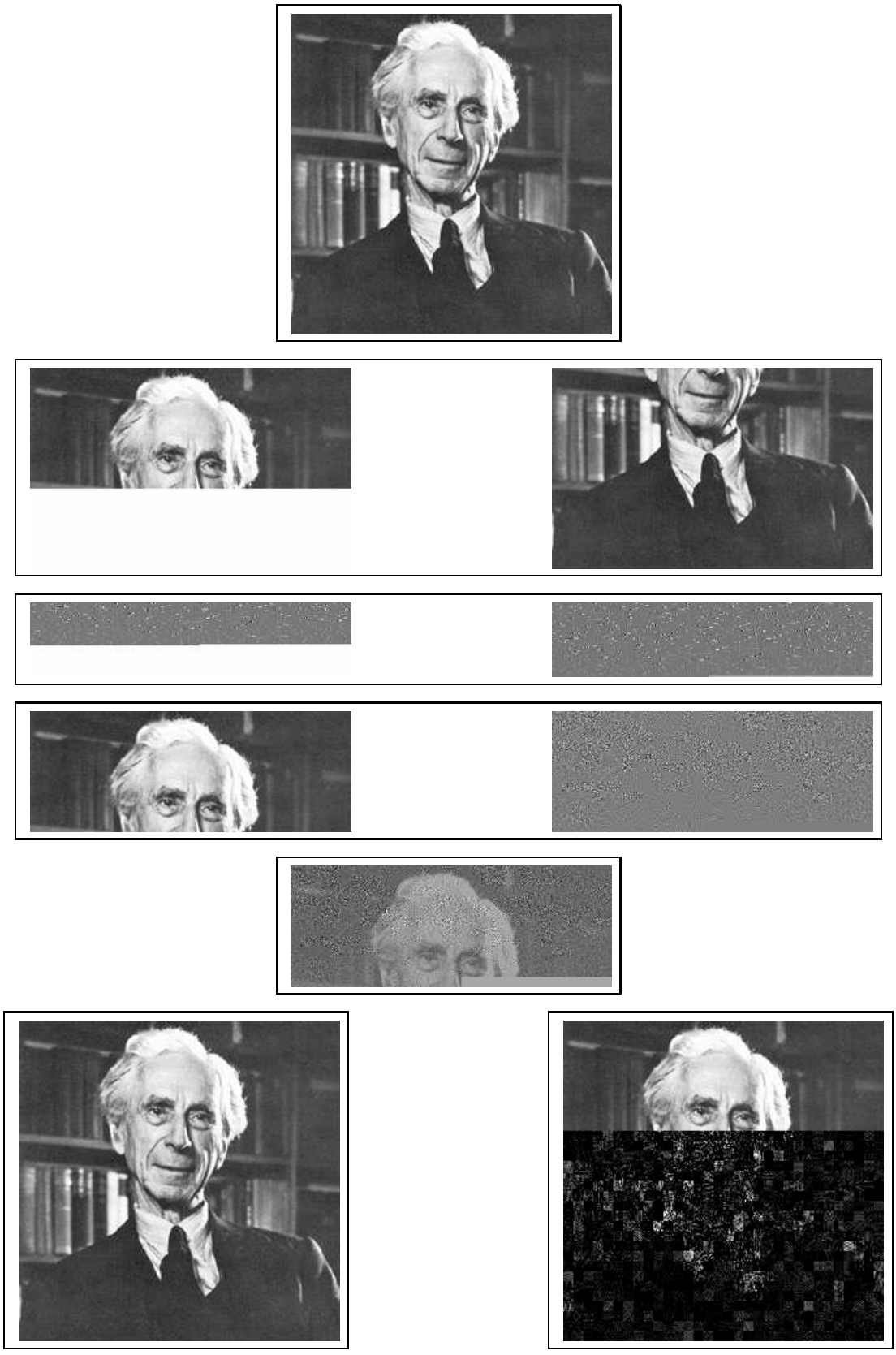

Figure 3.1: The Figure shows the steps involved in applying the *folding* procedure to the image of Bertrand Russell shown at the top. The image is first split into two section shown on the left and right of the second row. The third row shows the DCT coefficients required, to produce a sparse approximation of the images on the second row, to a PSNR of 45dB. The fourth row shows the host and embedded images respectively on the left and right. The fifth row shows the *folded* image. The sixth row contains the recovered images when the correct and incorrect *private key* is applied, displayed respectively on the left and the right of that row.

from the dictionary $\mathcal{D} = \{\mathbf{S}_m\}_{m=1}^{M_r M_c}$. This is equivalent the approximation shown equation (2.19), repeated below for the set $\mathcal{D}$,

$$\mathbf{I}_q^K = \sum_{k=1}^{K_q} \mathbf{c}_q(k)\mathbf{S}_{\mathbf{l}_q(k)}, q = 1, \ldots, Q^N. \tag{3.4}$$

3.2.2 Folding Procedure

The *folding* procedure begins as follows, $H_1 = \lceil \frac{K}{N^2} \rceil$ blocks are kept as *hosts*, where K is calculated with equation (2.17), and $\lceil x \rceil$ indicates the smallest integer not less than x. These H_1 blocks are now *hosts*, for embedding the $(Q^N - H_1)$ vectors of coefficients $\mathbf{c}_q, q = H_1 + 1 \ldots, Q^N$, of the remaining $(Q^N - H_1)$ equations in (3.4).

The vectors of coefficients $\mathbf{c}_q(k), k = 1, \ldots, K_q, q = H_1 + 1 \ldots, Q^N$ are relabelled to become $\mathbf{e}_q(n), n = 1, \ldots, N_{e,q}, q = 1 \ldots, H$, where each $N_{e,q} = N^2 - K_q$. If $N_1 = (H_1 N^2 - K)$ is greater than zero, a vector of padding $\mathbf{p}_1 \in \mathbb{R}^{N_1}$ will also be included with the coefficients. The relabelling procedure is performed with the $\widehat{\mathbf{Set}}(\cdot)$ operation, defined in Appendix A.2.4, as shown below

$$\{\mathbf{e}_q\}_{q=1}^{H} = \widehat{\mathbf{Set}}([\widehat{\mathbf{Flat}}(\{\mathbf{c}_q\}_{q=H_1+1}^{Q^N}); \mathbf{p}_1], \mathbf{n}_e), \tag{3.5}$$

where $\mathbf{n}_e(q) = N_{e,q}, q = 1, \ldots, H_1$, and the elements of the vector $\mathbf{p}_1$ can take on any value.

These coefficients are embedded in the H_1 host blocks, according to the following procedure:

- Build the embedded matrix $\mathbf{E}_q \in \mathbb{R}^{N \times N}$ as

$$\mathbf{E}_q = \sum_{n=1}^{N_e} \mathbf{e}_q(n)\mathbf{S}_{n,q}^{\perp}, q = 1, \ldots, H_1, \tag{3.6}$$

  where $\mathbf{S}_{n,q}^{\perp} \in \mathbb{R}^{N \times N}, n = 1, \ldots, N_e$, is an orthonormal basis for $\mathbb{V}_q^{\perp}$, the orthogonal complement of $\mathbb{V}_{K_q} = \text{span}\{\mathbf{S}_{\mathbf{l}(k)}\}_{k=1}^{K_q}$ in $\mathbb{R}^{N \times N}$. The construction of such a basis is discussed in Section 3.3.

- For $q = 1, \ldots, H_1$ *fold* the image by the superposition $\mathbf{I}_q^{f_1} = \mathbf{I}_q^K + \mathbf{E}_q$ and subsequent composition $\mathbf{I}^{f_1} = \cup_{q=1}^{H_1} \mathbf{I}_q^{f_1}$.

The *folded* image (Figure 3.1 row five) is now ready to be stored and/or transmitted to a third party for recovery.

3.2.3 Recovery Procedure

The following discussion requires a procedure for orthonormalizing a set of vectors. This will be denoted by the operation $\widehat{\mathbf{Orth}}(\cdot)$, which can be applied to any set of vectors to construct an orthonormal basis for the space spanned by those vectors.

The procedure for the recovery of the approximation $\mathbf{I}^K = \cup_{q=1}^{Q^N} \mathbf{I}_q^K$ of the image $\mathbf{I}$ from the folded image $\mathbf{I}^{f_1}$ is as follows:

- Calculate an orthonormal basis for the space $\mathbb{V}_{K_q}$ as

$$\{\mathbf{V}_{k,q}\}_{k=1}^{K_q} = \widehat{\mathbf{Orth}}(\mathbf{S}_{\mathbf{l}_q(k)}, k = 1, \ldots, K_q), \quad q = 1, \ldots, H_1. \tag{3.7}$$

- Remove the components in $\mathbb{V}_q^{\perp}$ from $\mathbf{I}_q^{f_1}$ as

$$\tilde{\mathbf{I}}_q^K = \hat{P}_{\mathbb{V}_{K_q}} \mathbf{I}_q^{f_1}, \quad q = 1, \ldots, H,$$

  where $\hat{P}_{\mathbb{V}_{K_q}} \mathbf{I}_q^{f_1}$ is the orthogonal projection

$$\hat{P}_{\mathbb{V}_{K_q}} \mathbf{I}_q^{f_1} = \sum_{k=1}^{K_q} \mathbf{V}_{k,q} \langle \mathbf{I}_q^{f_1}, \mathbf{V}_{k,q} \rangle_F, \quad q = 1, \ldots, H_1, \tag{3.8}$$

  onto $\mathbb{V}_{K_q}$ along $\mathbb{V}_q^{\perp}$.

- Recover $\tilde{\mathbf{E}}_q$ from $\tilde{\mathbf{I}}_q^K$,

$$\tilde{\mathbf{E}}_q = \mathbf{I}_q^{f_1} - \tilde{\mathbf{I}}_q^K, \quad q = 1, \ldots, H_1.$$

- Recover the coefficients

$$\tilde{\mathbf{e}}_q(n) = \langle \mathbf{S}_{k,q}^{\perp}, \tilde{\mathbf{E}}_q \rangle_F, \quad n = 1, \ldots, N_e, \quad q = 1, \ldots, H_1. \tag{3.9}$$

- Flatten the coefficients to get a temporary vector $\mathbf{t}$

$$\mathbf{t} = \widehat{\mathbf{Flat}}(\{\tilde{\mathbf{e}}_q\}_{q=1}^{H_1}).$$

  Remove any padding $\mathbf{p}_1$ and relabel the recovered coefficients

$$\{\tilde{\mathbf{c}}_q\}_{q=H_1+1}^{Q^N} = \widehat{\mathbf{Set}}(\mathbf{t}(1 : N_K), \mathbf{n}_K)$$

  The vector $\mathbf{n}_K(q - H_1) = K_q, q = H_1 + 1, \ldots, Q^N$ contains the number of coefficients in the remaining blocks, and N_K is the number of hidden coefficients which is equal to

$$N_K = \sum_{q=1}^{Q^N - (H_1+1)} \mathbf{n}_K(q).$$

- The remaining $\tilde{\mathbf{I}}_q^K, q = (H_1+1), \ldots, Q^N$ blocks of the approximated image, are calculated from the recovered coefficients with equation (3.4). The recovered image $\tilde{\mathbf{I}}^K$ is constructed as,

$$\tilde{\mathbf{I}}^K = \cup_{q=1}^{Q^N} \tilde{\mathbf{I}}_q^K. \tag{3.10}$$

From the above implementation it is clear that anyone in possession of the method which generates the matrices $\mathbf{S}_{n,q}^{\perp} \in \mathbb{R}^{N\times N}, n = 1, \ldots, N_{e,q}, q = 1, \ldots, H_1$, spanning the space $\mathbb{V}_q^{\perp}$, can recover the hidden numbers. To prevent recovery of the image $\tilde{\mathbf{I}}^K$ from these hidden numbers, the construction of the matrices $\mathbf{S}_{n,q}^{\perp}$ needs to be hidden, except to those authorized to view the image. Two procedures for achieving this goal are discussed in the next Section.

3.3 Calculating the vectors spanning the space $\mathbb{V}^{\perp}$

Two methods for constructing the matrices $\mathbf{S}_{n,q} \in \mathbb{R}^{N\times N}, n = 1, \ldots, N_{e,q}$ spanning the space $\mathbb{V}_q^{\perp}, q = 1, \ldots, H_1$ are described below. Both methods address the security of the hidden numbers $\mathbf{e}_q$ by employing secret initialization variables, referred to as a *private key*, to change the vectors spanning the space $\mathbb{V}_q^{\perp}, q = 1, \ldots, H_1$. The idea is that each unique secret *key*, will generate a unique set of basis vectors for the spaces $\mathbb{V}_q^{\perp}, q = 1, \ldots, H_1$.

The security of the system relies on the *keyspace* being large enough to prevent brute force attacks on the system, as described in Section 1.6.1.

For simplicity the subscript q is removed from notation in this Section.

3.3.1 The SVD Method

The following describes the procedure for constructing a large matrix $\mathbf{F} \in \mathbb{R}^{N^2\times K}$. The vectors spanning the null space of $\mathbf{F}$ will then form the basis for the required space $\mathbb{V}^{\perp}$.

First the N^2 elements of each $\mathbf{S}_{\mathbf{l}(k)} \in \mathbb{V}_K \subset \mathbb{R}^{N\times N}, k = 1, \ldots, K$ from equation (3.4) are relabelled to become the columns of a larger matrix $\mathbf{F} \in \mathbb{R}^{N^2\times K}$ as

$$\mathbf{F}(:,k) = \widehat{\mathbf{Vec}}(\mathbf{S}_k), k = 1, \ldots, K.$$

A basis $\mathbf{s}_n^{\perp} \in \mathbb{R}^{N^2}, n = 1, \ldots, N_e$ for the null space of $\mathbf{F}^T \in \mathbb{R}^{K\times N^2}$, will by definition satisfy the relationship

$$\langle \mathbf{F}(:,k), \mathbf{s}_n^{\perp} \rangle = 0, \quad k = 1, \ldots, K, \quad n = 1, \ldots, N_e.$$

This is equivalent to the relation required in equation (3.6), between the matrices $\mathbf{S}_k$ and the matrices $\mathbf{S}_n^{\perp}$, shown below,

$$\langle \mathbf{S}_k, \mathbf{S}_n^{\perp} \rangle_F = 0, \quad k = 1, \ldots, K, \quad n = 1, \ldots, N_e.$$

Therefore the matrices $\mathbf{S}_n^{\perp}$ can be generated by relabelling the vectors $\mathbf{s}_n^{\perp}$ as shown below

$$\mathbf{S}_n^{\perp} = \widehat{\mathbf{Mat}}(\mathbf{s}_n^{\perp}, N, N), n = 1, \ldots, N_e.$$

By calculating the basis for the space $\mathbb{V}^{\perp}$ in this way, a previously proposed method for securing information in the null space of a transformation, first outlined in [87] and further discussed in [88], can be applied in a straight forward manner. The method relies on the instability in the calculation of singular vectors corresponding to multiple singular values. This instability results in the calculated vectors spanning the null space of a rank deficient matrix, changing dramatically, if the matrix is initially perturbed by a small amount.

To illustrate this consider a rank deficient matrix $\mathbf{G}$, whose null space is $\mathbb{V}^{\perp}$. The idea is to apply a perturbation ϵ to a single element of $\mathbf{G}$, to get a perturbed matrix $\tilde{\mathbf{G}}$. The first step is to assign $\tilde{\mathbf{G}} = \mathbf{G}$, and then perturb the matrix as

$$\tilde{\mathbf{G}}(n_r, n_c) = \tilde{\mathbf{G}}(n_r, n_c) + \epsilon. \tag{3.11}$$

The orthonormal basis for $\mathbb{V}^{\perp}$ can then be calculated as the singular vectors corresponding to zero singular values, of the perturbed matrix $\tilde{\mathbf{G}}$.

Providing that a suitably sized perturbation has been applied, the vectors calculated from $\tilde{\mathbf{G}}$ will be completely different to the vectors calculated with $\mathbf{G}$, whilst still spanning the space $\mathbb{V}^{\perp}$.

As a result of the instability in the calculation of the singular vectors, a second perturbation added to the matrix $\tilde{\mathbf{G}}$ will result in a different set of singular vectors, to those calculated from $\tilde{\mathbf{G}}$. This second perturbation could be applied to further increase the security of this scheme.

Given that the basis for the null space of the matrix $\mathbf{F}^T$ will generate a basis for $\mathbb{V}^{\perp}$, $\mathbf{F}$ could be the rank deficient matrix $\mathbf{G}$ in the above discussion. Alternatively the matrix

$$\mathbf{G} = \mathbf{F}\mathbf{F}^T, \tag{3.12}$$

which by definition has the same null space as $\mathbf{F}^T$, could be used instead. This matrix has two advantages over $\mathbf{F}$,

1) The number of positions which the perturbation can be applied to, increases to N^4.

2) A square matrix increases the instability in the calculation of the singular vectors, corresponding to zero singular values.

The matrices $\mathbf{S}_n^{\perp}, n = 1, \ldots, N_e$ required in equation (3.6) will therefore be the relabelled singular vectors corresponding to zero singular values of $\mathbf{F}\mathbf{F}^T$.

Remark 2. The above discussion can easily be extended to include more than just a single private perturbation. The secret *key* would then become a combination of, the number of perturbations, the location (n_r, n_c) of each perturbation and the perturbation size ϵ.

Calculation of the Singular Vectors

Experimentally, the calculation of the singular vectors corresponding to the zero singular values of the matrix $\tilde{\mathbf{G}}$, from (3.12), will be performed by the MATLAB, `svd()` function. As a result of the singular vectors sensitivity to the algorithm used in their calculation [89], the results found in Section 3.4 below, are only repeatable on the experimental set up described in Section 1.7. This is because the source code for MATLAB's inbuilt routines, depends on both the computer architecture and the MATLAB version installed.

The following Experiment was designed to check that an image *folded* on one computer, can be recovered successfully on other computers, if the same algorithm for calculating the singular vectors is applied.

The quality of recovery of a *folded* image will be assessed by the PSNR^f. In a similar way to the PSNR^a, introduced in Section 2.5.1, which measures the error introduced by the approximation, the PSNR^f is the global PSNR which measures the error introduced by the *folding* procedure. This is calculated between the original image $\mathbf{I}$ and the recovered image $\tilde{\mathbf{I}}^K$, with $\tilde{\mathbf{I}}^K$ used instead of $\mathbf{I}^K$ in equation (1.4).

For this Experiment, 3 computers were chosen shown below, which differed in CPU, operating system, MATLAB version and compiler.

(1) CPU: Intel Core 2 Duo P8600, Operating System: Linux (64-bit) Kernel 3.5.0, MATLAB version: R2012a, Compiler: gcc 4.7.2.

(2) CPU: AMD Quad-Core Opteron 8380, Operating System: Linux (64-bit) Kernel 2.6.18, MATLAB version: R2012a, Compiler: gcc 4.1.2.

(3) CPU: AMD Dual-Core Athlon 7850, Operating System: Windows 7 Professional, MATLAB version: R2011b, Compiler: Visual Studio 2010 C++ compiler.

The C++ Singular Value Decomposition (SVD) algorithm from Numerical Recipes [90], was implemented in a mex file to allow its integration into the MATLAB programming environment. The routine was compiled on the three separate computers with the different compilers listed above. Any small difference ϵ to the matrix $\mathbf{G}$, calculated in equation

Figure 3.2: From top left to the bottom the recovered images of Lena Söderberg, the "rose made of galaxies" and a plane, all, initially approximated to 45.00dB, *folded* on computer (1) and then recovered on computers (2) and (3)

(3.12), will alter the singular vectors corresponding to zero singular values. Therefore this matrix needs to be identical all machines. To guarantee this, the matrix multiplication in (3.12), was performed on all computers by a simple loop, in a second C++ mex file. The rest of the calculations were performed on all computers with standard MATLAB functions.

This *folding* and recovery procedure was tested on 3, 8 bit grey scale images. The first is the classic image of Lena Söderberg shown in the top left image of Figure 3.2. The second is the "rose made of galaxies" image taken from the astronomical test set, shown in the top right of Figure 3.2. The final image is of a plane, taken from the natural test, set shown at the bottom of Figure 3.2.

The column and row matrices $\mathbf{D}_8^{c,1} \in \mathbb{R}^{N \times M_c}$ and $\mathbf{D}_8^{r,1} \in \mathbb{R}^{N \times M_r}$, are both constructed from the union the RDC dictionary and the Euclidean basis ($\mathbf{D}_8^{c,1} = [\mathbf{D}_1^{c,1}, \mathbf{D}_2^{c,1}]$), described in Section 2.4. These construct the separable matrices $\mathbf{S}_m, m = 1, \ldots, M_c M_r$, required in equation (3.4).

The 3 images were first split into blocks with $N = 8$, and then by applying OMP2D to choose atoms from the combined dictionary, approximated to a $\text{PSNR}^a = 45.00\text{dB}$.

Each image approximation was then *folded* on computer (1), applying the security scheme describe above with a perturbation of $\epsilon = 1 \times 10^{-12}$, added to the matrix $\tilde{\mathbf{G}}$. The *folded* images were then recovered with the same ϵ, on computers (2) and (3), and the difference between the PSNR^a and the PSNR^f calculated for each image.

The results of this Experiment were assessed by examining the additional loss introduced by the *folding* process. The loss was measured by the percentage difference between the PSNR^a and PSNR^f. This value was zero for every image except that of the plane, where a loss of only $1.57 \times 10^{-14}\%$ was introduced into the recovery, made on both computers (2) and (3).

The results of this Experiment show that is is possible to recover an image, *folded* with the SVD method on one computer, on different computers if the routine for calculating the singular vectors, is the same on both machines.

3.3.2 The Random Method

Here the basis for $\mathbb{V}^{\perp}$ is created by projecting a set of N_e pseudorandomly generated matrices in $\mathbb{R}^{N\times N}$ onto the space $\mathbb{V}^{\perp}$. The system is secured by creating a new set of N_e vectors as linear combinations of these random vectors.

The procedure for calculating the basis for $\mathbb{V}^{\perp}$ is as follows [3]:

- Initialize a PRNG with a seed s_1, and generate the pseudorandom matrices $\mathbf{Z}_n \in \mathbb{R}^{N\times N}, n = 1, \ldots, N_e$. With the orthogonal projection $\hat{P}_{\mathbb{V}_K}$ operator from (3.8) compute the matrices, $\mathbf{Z}_n^{\perp}$ as

$$\mathbf{Z}_n^{\perp} = \mathbf{Z}_n - \hat{P}_{\mathbb{V}_K}\mathbf{Z}_n \in \mathbb{V}^{\perp}, \quad n = 1, \ldots, N_e. \tag{3.13}$$

- Initialize a second PRNG with a seed s_2 and generate a matrix $\mathbf{U} \in \mathbb{R}^{N_e\times N_e}$, containing $(N_e)^2$ pseudorandom numbers. These number act as coefficients for creating new matrices $\mathbf{Y}_n^{\perp} \in \mathbb{R}^{N\times N}, n = 1, \ldots, N_e$, shown below

$$\mathbf{Y}_n^{\perp} = \sum_{n_r=1}^{N_e} \mathbf{U}(n_r, n)\mathbf{Z}_{n_r}^{\perp}, \quad n = 1, \ldots, N_e. \tag{3.14}$$

- Orthonormalize the matrices $\mathbf{Y}_n^{\perp}$ to have the orthonormal basis for $\mathbb{V}^{\perp}$

$$\{\mathbf{S}_n^{\perp}\}_{n=1}^{N_e} = \widehat{\mathbf{Orth}}(\mathbf{Y}_n^{\perp}, n = 1, \ldots, N_e) \tag{3.15}$$

  required in equation (3.6).

If the pseudorandom matrices $\mathbf{Z}_n \in \mathbb{R}^{N\times N}, n = 1, \ldots, N_e$ or $\mathbf{U}$ in the above prescription, are not known, it will be impossible to construct the vectors $\mathbf{S}_n^{\perp}$. Therefore the vector of hidden coefficients $\tilde{\mathbf{e}}$ cannot be recovered with equation (3.9), thus securing the *folding* procedure.

Pseudorandom Number Generation

The pseudorandom matrices above, securing each block of the *folded* image can be generated with a Cryptographically Secure Pseudorandom Number Generator (CSPRNG) [53].

A CSPRNG generates a stream of cryptographically secure pseudorandom numbers, once it has been initialized with a seed, or *key*. The seed will exactly determine the content of the stream, therefore anyone in possession of it will be able to generate the exact same stream of numbers. Each seed should generate a different set of cryptographically secure pseudorandom numbers, with the total number of seeds denoted by s_{max}.

If s_{max} is small then this security scheme will be susceptible to brute force attacks (Section 1.6.1), where an attacker simply has to try all s_{max} possible seeds to recover the *folded* image. Therefore a suitable CSPRNG for this security scheme is one which has a large number of initialization seeds s_{max}. An example of this would be the RC4 stream cipher which has an $s_{max} = 2^{256}$ [91].

Experimentally this was realized without a CSPRNG, by MATLAB's default PRNG.

3.4 Finding the Keyspace for the SVD Method

There are 3 parameters to consider when applying perturbations ϵ to the matrix $\tilde{\mathbf{G}} \in \mathbb{R}^{N^2\times N^2}$, from equation (3.11):

1) The magnitude $\epsilon \in \mathbb{R}$ of each perturbation.

2) The location within $\tilde{\mathbf{G}}(n_r, n_c), n_r, n_c \in \{1 \ldots N^2\}$ of each perturbation.

3) The number of perturbations N_ϵ. To make the *key* unique each of the N^4 positions in $\tilde{\mathbf{G}}$ can contain at most 1 perturbation, therefore $N_\epsilon \in \{1 \ldots N^4\}$.

The combination of all these parameters form the *private key* which is required to correctly unfold an image $\mathbf{I}^{f_1}$.

The focus of this Section is to find *keyspace* described in Section 1.6 which will fulfil the following two requirements:

a) Without the correct *key* it should not be possible to recover the vectors of hidden numbers $\tilde{\mathbf{e}}_q, q = 1, \ldots, H_1$ with equation (3.9).

This will be assessed by examining the error between the hidden and recovered coefficients over a given image set, applying the following measure

$$\overline{\text{Err}} = \frac{100}{N_I} \sum_{n=1}^{N_I} \text{Err}_n, \tag{3.16}$$

where N_I is the number of images in the set being tested, and

$$\text{Err}_n = \sum_{q=1}^{H_1} \left(\frac{\|\mathbf{e}_q - \tilde{\mathbf{e}}_q\|}{\|\mathbf{e}_q\|} \right)^2 \tag{3.17}$$

measures the error in the coefficients hidden in image n.

b) With the correct *key* the recovered image $\tilde{\mathbf{I}}^K$, should not be significantly different to the approximated image $\mathbf{I}^K$. The will be assessed by the δPSNR between the the original $\mathbf{I}$, and the recovered image $\tilde{\mathbf{I}}^K$. Given that there is a percentage difference $x\%$ tolerated between the desired PSNR ydB and the actual PSNR^a in the approximation of an image. The convention which will be adopted, is that the δPSNR, defined next, should also be less than $x\%$. That is for test image n,

$$\delta\text{PSNR}_n = 100 \frac{|\text{PSNR}_n^a - \text{PSNR}_n^f|}{\text{PSNR}_n^a} < x\%, \tag{3.18}$$

For the Experiments involving more than one image this will be taken in mean value,

$$\overline{\delta\text{PSNR}} = \frac{1}{N_I} \sum_{n=1}^{N_I} \delta\text{PSNR}_n < x\%, \tag{3.19}$$

where N_I is again the number of images in the set.

3.4.1 Minimum Perturbation ϵ_{min}

The minimum perturbation ϵ_{min} is the minimum value which, when added to any element of the matrix $\tilde{\mathbf{G}}$, in equation (3.11), is guaranteed to change its value. This value will depend on the largest element in absolute value of $\tilde{\mathbf{G}}$, calculated from $\mathbf{F}$ in (3.12).

Whilst examining the sparsity produced by OMP2D in the Experiments of Section 2.6, the largest absolute value of the matrix $\mathbf{G}$ was calculated. The result for all Experiments was, the largest absolute value of $\mathbf{G}$ did not exceed 3. The Experiments in this Section are all calculated in double precision, as defined by the IEEE 754 standard [92]. Therefore ϵ_{min} is the minimum value which when added to 3 is guaranteed to increase its value, when the calculations are performed in double precision. This is $\epsilon_{min} \approx 4.44 \times 10^{-16}$.

Remark 3. ϵ_{min} is guaranteed to perturb the matrix $\tilde{\mathbf{G}}$. It is not guaranteed to induce a different set of vectors for the null space, than would be produced by $\mathbf{G}$. This is an important distinction, and the focus of the investigation in the next Section.

3.4.2 Experimental Overview

The *folding* and recovery procedure in Section 3.2 can be applied to any sparse approximation of any image $\mathbf{I}$, expressed by equation (3.4). For the Experiments in this Section, the images $\mathbf{I}$ have been restricted to be the 55 astronomical images, contained in both the training and test sets introduced in Chapter 2. The algorithm producing the sparse approximations, is fixed to be OMP2D, and the column and row dictionaries are also fixed. These are the matrices $\mathbf{D}_8^{c,1}$ and $\mathbf{D}_8^{r,1}$ who's columns are the union of the RDC and Euclidean basis from the previous Experiment.

For all Experiments the sparse approximations of each 55 images in the test set is *folded* with the SVD procedure outlined in Section 3.2.2. Two simulations are then performed to examine incorrect and correct recovery, from the *folding* procedure. The first aims to simulate a third party attempting to recover the hidden portion of the image, without knowledge of the perturbation applied at the *folding* stage. The success of this is measured by the resulting $\overline{\text{Err}}$. The second is to simulate the recovery obtained by a third party who knows the value of the perturbation. The quality of this recovery is examined by the PSNR^a.

An additional third simulation is performed in Experiment 1, which is described in that Section.

The first three Experiments investigate the *keyspace* parameters, $1) - 3)$ above, with a fixed block size $N = 8$. A PSNR$^a = 45 \pm x\%$dB, such that $x = 1 \times 10^{-2}$, was chosen because it results in high quality approximations. The small block size of $N = 8$ was chosen to reduce the processing time, and increase the number of Experiments which could realistically be performed. The final Experiment then examined the effect on this *keyspace* of changing the values of N and the PSNR^a.

The Experiments in this Section are again run inside the MATLAB programming environment with the same configuration described in Section 1.7. The $\widehat{\mathbf{Orth}}(\cdot)$ operation involved in equation (3.7) is performed by the MATLAB `qr()` routine, which calculates the orthogonal-triangular decomposition of matrix. The singular vectors corresponding to zero singular values in Section 3.3.1, are left singular vectors corresponding to zero singular values returned from the MATLAB `svd()` routine.

3.4.3 Experiment 1 - Range of Perturbations

The range of perturbations which can be applied to the matrices $\tilde{\mathbf{G}}$ while fulfilling requirements a) and b) above are examined.

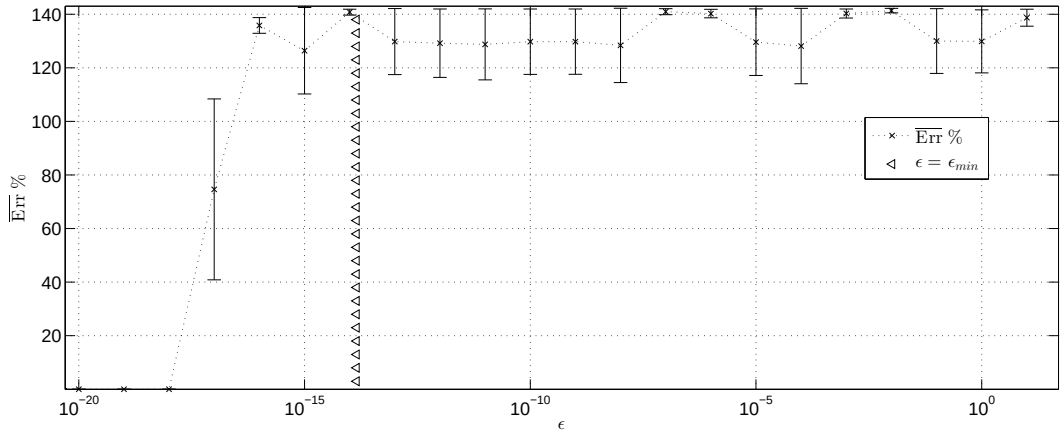

Figure 3.3: $\overline{\text{Err}}$ over the astronomical image set against the value of ϵ in equation (3.12), applied at the folding stage. The hidden coefficients are securely hidden by adding the perturbation ϵ to the first element of $\tilde{\mathbf{G}}$ in (3.12) and then recovered without applying a perturbation. A single standard deviation from this mean is shown above and below the $\overline{\text{Err}}$ by the error bars. The dashes $\triangleleft$ indicate where perturbations smaller than ϵ_{min} occur.

The Experiments described in this Section were performed with a single perturbation ϵ added to element $\tilde{\mathbf{G}}(1,1)$, when *folding* each image in the test set.

To examine the effect of perturbations smaller than $\epsilon_{min} \approx 4.44 \times 10^{-16}$, the first Experiment will apply a minimum perturbation of 10^{-20}. The maximum perturbation size for this Experiment will be 10 (chosen as it is far in excess of what could be considered to be a perturbation of the $\tilde{\mathbf{G}}$). The set $\mathcal{E}$ of perturbations chosen for Experiment 1 is therefore

$$\mathcal{E} = \{10^i\}_{i=-20}^{1}.$$

In addition to the two standard simulations designed to investigate incorrect and correct recovery from the *folding* procedure, an additional simulation is performed. This is to simulate a third party trying to guess the correct perturbation, by examining how close their guess has to be.

Simulation 1

Simulation of a third party attempting to recover the hidden portion of an image, without knowledge of the private *key*. For each $\epsilon \in \mathcal{E}$, added with equation (3.11), the images were *folded*. The perturbation, applied with equation (3.11) at the recovery stage, is denoted by ϵ_R, and for this Experiment is $\epsilon_R = 0$.

Figure 3.3 shows the $\overline{\text{Err}}$ in the recovery of the hidden coefficients, for each $\epsilon \in \mathcal{E}$ applied at the *folding* stage. For $\epsilon > \epsilon_{min}$, displayed on Figure 3.3 to the right of the $\triangleleft$ symbol, the $\overline{\text{Err}} > 120\%$.

An $\overline{\text{Err}}$ greater than 100% for all $\epsilon > \epsilon_{min}$ indicates that, as expected the instability in the calculation of the vectors spanning the space $\mathbb{V}^{\perp}$, is sensitive to all perturbations greater than ϵ_{min}. Thus under these test conditions, any perturbation greater than ϵ_{min} can prevent unauthorized access to the hidden coefficients $\mathbf{e}$.

Figure 3.3 shows that for $\epsilon < 10^{-18}$ the $\overline{\text{Err}} < 10^{-12}\%$. Thus $\epsilon < 10^{-18}$ applied at the *folding* stage, does not prevent a third party, without knowledge of the correct perturbation, from recovering the hidden coefficients.

This simulation indicates that, all $\epsilon > \epsilon_{min}$ in equation (3.11), fulfil requirement a) for the astronomical image set. Because of this and because $\epsilon \leq 10^{-18}$ did not prevent recovery of the hidden coefficients $\tilde{\mathbf{e}}$, the set of test perturbations was restricted further to

$$\mathcal{E}_1 = \{10^i\}_{i=-16}^{1}. \tag{3.20}$$

The main focus now being on perturbations greater than 10^{-15}.

Remark 4. Notice that in Figure 3.3 there are three data points to the left of the $\triangleleft$ symbol, where the $\overline{\text{Err}} > 120\%$. This indicates that correct recovery of the hidden coefficients $\tilde{\mathbf{e}}$, is also prevented by perturbations less than ϵ_{min}. This is because the maximum value of any element of the matrices $\mathbf{G}$ can be smaller than 3, reducing the size of the minimum perturbation ϵ_{min}, which is guaranteed to perturb the matrix.

Simulation 2

To examine which perturbation sizes in $\mathcal{E}$ fulfil requirement b), all the images were *folded* and recovered with the same perturbation $\epsilon \in \mathcal{E}_1$, applied to $\tilde{\mathbf{G}}(1,1)$ in equation (3.12).

The results plotted in Figure 3.4 show that for perturbations $\epsilon = 10^i, i = -16, \ldots, -12$, the recovered images $\tilde{\mathbf{I}}^K$ are identical to the approximated images $\mathbf{I}^K$. In other words the approximated image $\mathbf{I}^K$ is not numerically altered by *folding* and recovery procedure.

All perturbations $\epsilon > 10^{-11}$ introduce error to $\tilde{\mathbf{I}}^K$, which increases with the perturbation size ϵ. This is shown in Figure 3.4 by the $\overline{\delta\text{PSNR}}$. More importantly for all $\epsilon \leq 10^{-3}$ the $\overline{\delta\text{PSNR}} < 2 \times 10^{-4}\%$, less than the maximum of $x = 1 \times 10^{-2}\%$. Additionally at this level their is no visual difference between the approximated image $\mathbf{I}^K$ and the recovered image $\tilde{\mathbf{I}}^K$. Therefore the results of this simulation indicate that $\epsilon \leq 10^{-3}$ in equation (3.12) is sufficient to fulfil requirement b) for the astronomical image set.

Simulation 3

Simulation 1 demonstrated that any perturbation $\epsilon \geq \epsilon_{min}$ applied at the *folding* stage, can prevent a third party accessing the vectors of hidden coefficients $\mathbf{e}$, if $\epsilon_R = 0$ at the

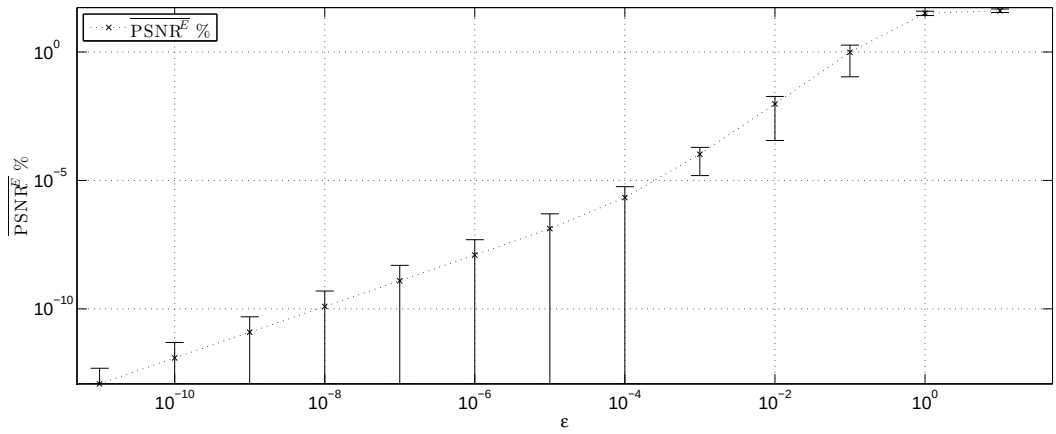

Figure 3.4: $\overline{\delta\text{PSNR}}$ over the astronomical image set against the value of ϵ in equation (3.12), applied at the folding and recovery stage. The images are *folded* and recovered by adding ϵ to element $\tilde{\mathbf{G}}(1,1)$ in (3.12). $\epsilon < 10^{-11}$ is not shown on the Figure because this resulted in a $\overline{\delta\text{PSNR}} = 0$. A single standard deviation is shown above and below the $\overline{\delta\text{PSNR}}$ by the error bars, where the end of the error bar is not shown, if this single standard deviation below the mean is less than or equal to zero.

recovery stage. The aim of this simulation is to determine if a difference of ϵ_{min}, between the perturbation applied at the *folding* and recovery stages, is sufficient to prevent recovery of the hidden coefficients.

To examine this for each $\epsilon \in \mathcal{E}_1$ from (3.20), the images were folded with ϵ, and then recovered with $\epsilon_R = \epsilon + \epsilon_{min}$.

For each $\epsilon \in \mathcal{E}_1$, the resulting $\overline{\text{Err}}$ over the image set was greater than 123.42% with a standard deviation less than 17.62%. This indicates that for all $\epsilon \in \mathcal{E}_1$, a perturbation differing by as little as ϵ_{min} from the correct one, can prevent correct recovery of the hidden coefficients.

Discussion

The results of simulation 1 in Figure 3.3 show, requirement a) is satisfied by all discrete perturbations

$$\mathcal{E}_1 = \{10^i\}_{i=-16}^{1}. \tag{3.21}$$

That is, any $\epsilon \in \mathcal{E}_1$ applied in equation (3.12) when the images were *folded*, prevented recovery of the hidden coefficients $\tilde{\mathbf{e}}_q, q = 1, \ldots, Q^N$, when $\epsilon_R = 0$.

Further to this, simulation 3 showed this result to hold, even when the recovery perturbation was only ϵ_{min} different, to the perturbation applied at the *folding* stage. To verify this result, in the remaining Experiments all simulations designed to verify requirement a), will apply a perturbation of $\epsilon_R = \epsilon + \epsilon_{min}$, when performing the recovery.

The results of simulation 2 in Figure 3.4 show requirement b) is satisfied by $\epsilon =$

$10^i, i = -16, \ldots, -3$. Therefore a set of perturbations $\mathcal{E}_2$, fulfilling requirements a) and b) can be proposed. For the astronomical images processed in blocks with $N = 8$, initially approximated to a $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-3}$dB, this set is,

$$\mathcal{E}_2 = \{10^i\}_{i=-16}^{-3}. \tag{3.22}$$

3.4.4 Experiment 2 - Location of ϵ

The effect of changing the location of the perturbation within the matrix $\tilde{\mathbf{G}}$, on requirements a) and b) above, is investigated.

In the preceding Experiment a single perturbation was added to $\tilde{\mathbf{G}}(1,1)$. In this Experiment the perturbation will be applied to different locations within $\tilde{\mathbf{G}}$. The first two locations are fixed to be the middle $\tilde{\mathbf{G}}(4,8)$, and the end $\tilde{\mathbf{G}}(8,8)$, of the matrix. The other location $\tilde{\mathbf{G}}(r_r^1, r_c^1)$, will be determined for each block by pseudorandom numbers drawn from MATLAB's PRNG.

The additional locations of the middle and the end are to investigate whether any particular fixed location in $\tilde{\mathbf{G}}$, makes a difference to the instability, in the calculation of the singular vectors. The random location, is to investigate the effect of applying the perturbation, to many different locations within $\tilde{\mathbf{G}}$.

The range of perturbations tested is again $\mathcal{E}_1$ from equation (3.20).

Simulation 1

For each $\epsilon \in \mathcal{E}_1$, the astronomical images were *folded* with ϵ, and recovered with $\epsilon_R = \epsilon + \epsilon_{min}$, both applied to $\tilde{\mathbf{G}}(4,4)$. This was then repeated, first applying the perturbation to a random location in each block, and then to the fixed location $\tilde{\mathbf{G}}(4,8)$. The $\overline{\text{Err}}$ was then compared with the results from Experiment 1, Simulation 3, where the perturbation was applied to $\tilde{\mathbf{G}}(1,1)$.

Figure 3.5 shows the $\overline{\text{Err}}$ for each location against each $\epsilon \in \mathcal{E}_1$, applied when *folding* the images. In Figure 3.5 none of the chosen positions, result in a greater $\overline{\text{Err}}$ for all the perturbations ϵ tested. Additionally the result of applying the perturbation to a random location in each block, had the most consistent $\overline{\text{Err}}$ for all $\epsilon \in \mathcal{E}_1$.

Simulation 2

The procedure for this simulation is the same as in simulation 1, except the images are recovered with $\epsilon_R = \epsilon$ instead of $\epsilon_R = \epsilon + \epsilon_{min}$.

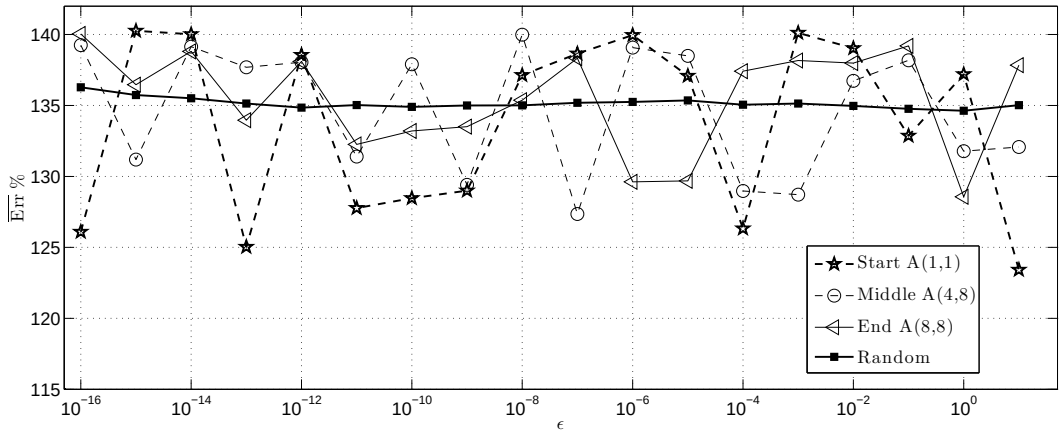

Figure 3.5: $\overline{\text{Err}}$ over the astronomical image set, against the value of ϵ applied at the *folding* stage, for four different locations within $\tilde{\mathbf{G}}$. The coefficients are securely hidden by perturbing $\tilde{\mathbf{G}}$ by ϵ, and then recovered with $\epsilon + \epsilon_{min}$. Each line shows the $\overline{\text{Err}}$ when adding the perturbation to a different location within $\tilde{\mathbf{G}}$. There are three fixed locations, $\tilde{\mathbf{G}}(1,1)$, $\tilde{\mathbf{G}}(4,8)$, $\tilde{\mathbf{G}}(8,8)$ and a location which is randomly assigned for each $\tilde{\mathbf{G}}$.

The results in Figure 3.6 show the $\overline{\delta\text{PSNR}}$ for each $\epsilon \in \mathcal{E}_1$ applied at the *folding* and recovery stages. Figure 3.6 shows that the $\overline{\delta\text{PSNR}}$ is not significantly effected by the locations chosen for this Experiment. Numerically the standard deviation between the four locations is less than $10^{-5}\%$ for all $\epsilon < 10^{-3}$.

Discussion

The results of simulation 1 and 2 are respectively shown in Figure 3.5 and Figure 3.6. They show that over the astronomical test set, requirements a) and b) are satisfied, by the two fixed test locations.

An additional location within $\tilde{\mathbf{G}}$, was determined for each block by numbers drawn from a PRNG, initialized with a seed. This pseudorandom location also satisfied requirements a) and b). If this seed is kept secret, it could be incorporated as an additional part of the *private key*. Because of this possibility, in the remaining experiments, the location of the perturbation is chosen randomly. This is to further verify that changing the location fulfils requirements a) and b).

3.4.5 Experiment 3 - More than One ϵ

In this experiment an extra perturbation is applied to the matrix $\tilde{\mathbf{G}}$, to see the effect on requirements a) and b). Equation (3.12) is changed to accommodate two perturbations ϵ^1 and ϵ^2 as shown below

$$\begin{aligned} \tilde{\mathbf{G}}(r_r^1, r_c^1) &= \tilde{\mathbf{G}}(r_r^1, r_c^1)) + \epsilon^1, \\ \tilde{\mathbf{G}}(r_r^2, r_c^2) &= \tilde{\mathbf{G}}(r_r^2, r_c^2) + \epsilon^2, \end{aligned} \tag{3.23}$$

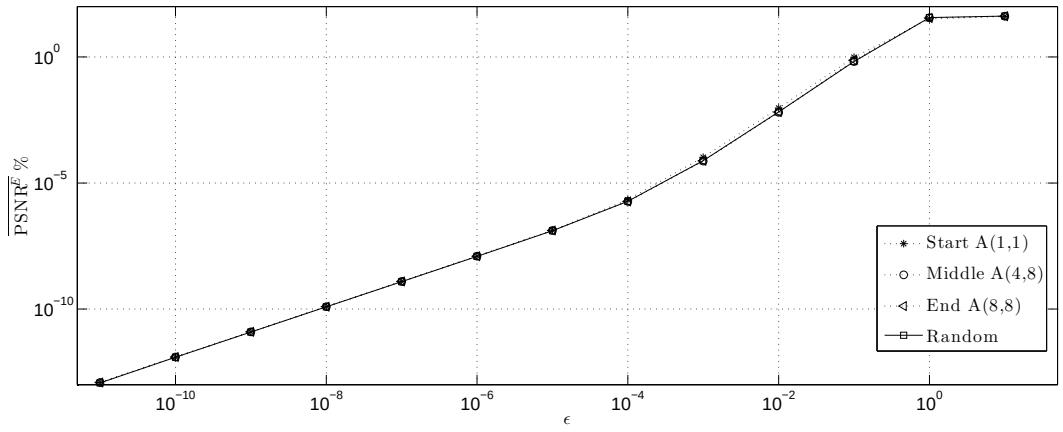

Figure 3.6: $\overline{\delta\text{PSNR}}$ over the astronomical image set, against the value of ϵ applied at the *folding* stage. The $\overline{\delta\text{PSNR}}$ is shown for four different locations within $\tilde{\mathbf{G}}$ from equation (3.12). For each ϵ the images are *folded* and expanded, with the resulting $\overline{\delta\text{PSNR}}$ shown on the Figure. Each line shows the $\overline{\delta\text{PSNR}}$ when adding the perturbation to a different location within $\tilde{\mathbf{G}}$. There are three fixed locations, $\tilde{\mathbf{G}}(1,1)$, $\tilde{\mathbf{G}}(4,8)$, $\tilde{\mathbf{G}}(8,8)$ and a location which is randomly assigned for each $\tilde{\mathbf{G}}$.

where the row column index pairs (r_r^1, r_c^1) and (r_r^2, r_c^2) are different.

The first simulation examines the effect of; a)recovering the images with a single incorrect perturbation; and b)recovering the images with two incorrect perturbations. The second simulation examines the error introduced into the recovered image $\tilde{\mathbf{I}}^K$, by this second perturbation, when the same perturbations are applied, at the *folding* and recovery stages.

Simulation 1

For each combination of $\epsilon^1 \in \mathcal{E}_1$ and $\epsilon^2 \in \mathcal{E}_1$ the astronomical images were *folded*. The perturbations ϵ^1 and ϵ^2 were applied to the random locations within the matrix $\tilde{\mathbf{G}}$ from equation (3.23).

The images were then recovered by applying $\epsilon_R^1 = \epsilon^1$ and $\epsilon_R^2 = \epsilon^2 + \epsilon_{min}$ and the $\overline{\text{Err}}$ measured.

The result for all combinations of perturbations ϵ^1 and ϵ^2 was, the resulting $\overline{\text{Err}}$ did not fall below 134.22%.

Next the images were recovered, applying $\epsilon_R^1 = \epsilon^1 + \epsilon_{min}$ and $\epsilon_R^2 = \epsilon^2 + \epsilon_{min}$. For all combinations of perturbations ϵ^1 and ϵ^2 the resulting $\overline{\text{Err}}$ did not fall below 134.21%.

Therefore the introduction of the extra perturbation $\epsilon^2 \in \mathcal{E}_1$ did not reduce the security of the hidden numbers, under these test conditions.

Simulation 2

The images were folded with the same procedure as simulation 1. They were then recovered by applying the same perturbations added at the *folding* stage, $\epsilon_R^1 = \epsilon^1$ and $\epsilon_R^2 = \epsilon^2$.

The results showed that the $\overline{\delta\text{PSNR}}$ is dominated by the largest perturbation applied to $\tilde{\mathbf{G}}$. Two examples of this are shown in Figure 3.7, the top half of the Figure showing the result for $\epsilon^2 = 10^{-8}$ and the bottom half showing the result for $\epsilon^2 = 10^{-3}$.

The top half of Figure 3.7 shows the $\overline{\delta\text{PSNR}}$ against each $\epsilon^1 \in \mathcal{E}_1$ when $\epsilon^2 = 10^{-8}$. The $\overline{\delta\text{PSNR}}$ from Experiment 2, Simulation 1 for a single perturbation of $\epsilon = 10^{-8}$ is included in the Figure. This was a $\overline{\delta\text{PSNR}} = 1.24 \times 10^{-10}$ and is indicated by the dashed line in the top of the Figure. As can be seen in the top of Figure 3.7 the $\overline{\delta\text{PSNR}}$ for the combination of all $\epsilon^1 \leq 10^{-8}$ and $\epsilon^2 = 10^{-8}$ is of the same order as that for a single perturbation of 10^{-8}. The bottom half of the figure shows the same result for $\epsilon^2 = 10^{-3}$.

Discussion

Simulation 1 showed that the hidden coefficients $\mathbf{e}$, cannot be recovered when an extra perturbation $\epsilon^2 \in \mathcal{E}_1$ is applied at the *folding* stage. This is if either ϵ_R^1 or ϵ_R^2, differ by ϵ_{min} from ϵ^1 or ϵ^2, applied when the images are *folded*. Therefore applying a second perturbation fulfils requirement a).

The second simulation again *folded* and recovered the images with the same perturbations, this time applying two, ϵ^1 and ϵ^2. The result being, the numerical error introduced into the recovered image $\tilde{\mathbf{I}}^K$, was of the same order, as the error introduced by the largest single perturbation. This implies that applying more perturbations, may keep the error below that introduced by the largest perturbation.

3.4.6 Experiment 4 - Effect of the PSNR^a on ϵ

The effect of changing the PSNR^a of the approximated image $\mathbf{I}^K$ before it is *folded* on the range of perturbations fulfilling requirements a) and b) above is examined.

All the images in the preceding Experiments were approximated to a PSNR$^a = 45 \pm x$%dB, with $x = 1 \times 10^{-2}$, resulting in approximations which are visibly identical to the original. In this Experiment the images were approximated in blocks with $N = 8$, to two additional quality levels, PSNR$^a = 55 \pm x$%dB and PSNR$^a = 65 \pm x$%dB, again with $x = 1 \times 10^{-2}$. This is to investigate the effect of the *folding* and recovery procedure on higher quality approximations.

The images were then *folded* with two perturbations ϵ^1 and ϵ^2 applied to two different

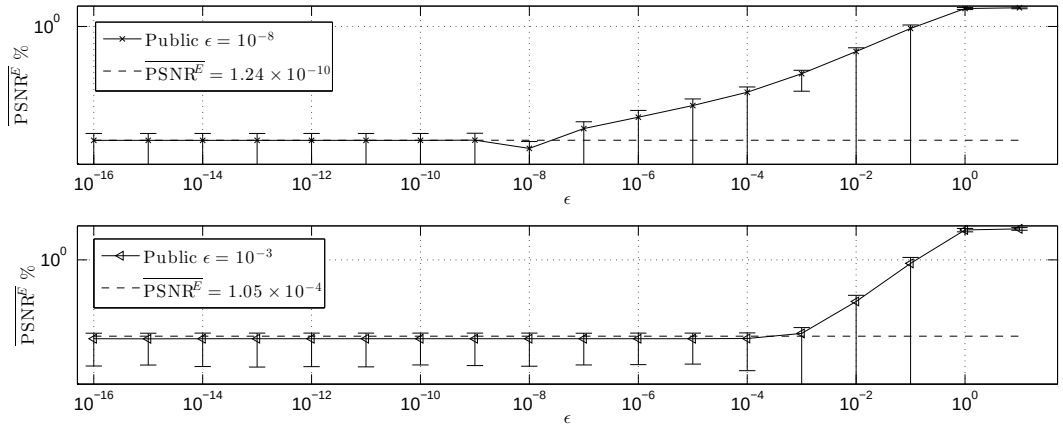

Figure 3.7: $\overline{\delta\text{PSNR}}$ over the astronomical image set against the value of ϵ^1 applied at the folding stage. The top half of the figure displays the $\overline{\delta\text{PSNR}}$ for $\epsilon^2 = 10^{-8}$. The dashed line shows the value of the $\overline{\delta\text{PSNR}}$ for a single perturbation of $\epsilon = 10^{-8}$, applied to a random location $\tilde{\mathbf{G}}$. The bottom half shows the same result for $\epsilon^2 = 10^{-3}$.

random locations within $\tilde{\mathbf{G}}$, shown in equation (3.23). The first perturbation was fixed as $\epsilon^1 = \epsilon_{min}$, and the second $\epsilon^2 \in \mathcal{E}_1$.

Simulation 1

Each *folded* image was recovered with the correct value for the first perturbation $\epsilon^1_R = \epsilon_{min}$, and the incorrect value for the second perturbation, $\epsilon^2_R = \epsilon^2 + \epsilon_{min}$.

The result for the additional two values of PSNR^a, and all perturbation sizes $\epsilon^2 \in \mathcal{E}_1$ applied at the *folding* stage, was an $\overline{\text{Err}}$ of at least 134% for every image.

Simulation 2

Each *folded* image was recovered with the correct value for the first and second perturbations, that is $\epsilon^1_R = \epsilon_{min}$ and the $\epsilon^2_R = \epsilon^2$.

The resulting $\overline{\delta\text{PSNR}}$ for the two levels of PSNR^a tested, did not exceed x%, for all perturbation sizes $\epsilon^2 \in \mathcal{E}_1$ applied at the *folding* stage.

Discussion

The PSNR^a in the initial approximation is increased to both $55 \pm x$%dB and $65 \pm x$%dB, before the *folding* procedure is applied. This increase in approximation quality is shown experimentally, not to affect on the range of perturbations $\mathcal{E}_1$, which can be applied in equation (3.23).

3.4.7 Experiment 5 - Effect of the Block Size N on ϵ

The effect that the size of the square $N \times N$ blocks which the original image is split into, on requirements a) and b) is investigated. Two additional block sizes of $N = 16$ and $N = 24$ are tested, and the results compared with those for $N = 8$, from Experiment 3.4.5.

Applying the same experimental set up described in Experiment 3.4.6, the images were first split into square blocks of either 16×16 and 24×24, before being approximated to a fixed $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-2}\%\text{dB}$.

Simulation 1

Each *folded* image was recovered with the correct value for the first perturbation $\epsilon_R^1 = \epsilon_{min}$, and the incorrect value for the second perturbation, $\epsilon_R^2 = \epsilon^2 + \epsilon_{min}$.

The result for the larger block sizes with $N = 16$ and $N = 24$ was, the $\overline{\text{Err}}$ did not fall below that of blocks with $N = 8$.

Simulation 2

Each *folded* image was recovered with the correct value for the first and second perturbations, that is $\epsilon_R^1 = \epsilon_{min}$ and the $\epsilon_R^2 = \epsilon^2$.

The resulting $\overline{\delta\text{PSNR}}$ did not exceed $x\%$ for the two larger block sizes, $N = 16$ and $N = 24$.

Discussion

The size of the partitions which the image is processed in was increased from $N = 8$ to $N = 16$, or $N = 24$. The results of simulation 1 and 2, show that this increase still fulfils requirements a) and b), for the range of perturbations in equation (3.22).

3.4.8 The Keyspace for the SVD Method

Before discussing the size of the *keyspace*, a quick review of the main results of the preceding Experiments is given below, in Experiment order. The Experiments undertaken above were performed on astronomical images, initially approximated by applying OMP2D, to select vectors from the column and row dictionaries $\mathbf{D}_8^{c,1}$ and $\mathbf{D}_8^{r,1}$. The dictionary for all Experiments was constructed as the union of the RDC and Euclidean basis. The results below apply to Experiments undertaken under the above conditions:

- With a fixed block size $N = 8$, approximating all images to a $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-2}\%\text{dB}$ before applying the *folding* procedure:

1) The range of perturbations, which both protect the vectors of hidden coefficients $\mathbf{e}_q, q = 1, \ldots, Q^N$, and did not introduce an unacceptable level of distortion, into the recovered image $\tilde{\mathbf{I}}^K$, are

$$\mathcal{E}_1 = \{10^x\}_{x=-16}^{-3}.$$

The set $\mathcal{E}_1$ was determined for a single perturbation ϵ, applied to a fixed location within each matrix $\tilde{\mathbf{G}}$, as shown in equation (3.22).

These perturbations also prevented recovery of the hidden coefficients $\mathbf{e}$, when the recovery was performed with an incorrect perturbation. This was realized experimentally by a perturbation ϵ_R at the recovery stage, differing by ϵ_{min} to that used in the *folding* stage.

2) Requirements a) and b) were satisfied, when the perturbation in $\mathcal{E}_1$ were applied to two additional fixed locations, and a pseudorandom location in $\tilde{\mathbf{G}}$.

3) Requirements a) and b) were again satisfied for two perturbations from $\mathcal{E}_1$. Realized experimentally by applying all combinations of two perturbations from $\mathcal{E}_1$, to two different pseudorandom locations within the matrix $\tilde{\mathbf{G}}$. The second perturbation was found not to increase the $\overline{\delta\text{PSNR}}$ of the recovered image, beyond that of a single perturbation of the same size.

- Applying two perturbations to pseudorandom locations within the matrix $\tilde{\mathbf{G}}$, the first being fixed as ϵ_{min}, and the second taking on all the values from $\mathcal{E}_1$:

4) With a fixed block size $N = 8$, increasing in the approximation quality to $\text{PSNR}^a = 55 \pm 5.5 \times 10^{-2}$%dB and $\text{PSNR}^a = 65 \pm 6.5 \times 10^{-2}$%dB, the *folding* and recovery procedure still fulfilled requirements a) and b).

5) With a fixed $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-2}$%dB, increasing the block size to $N = 16$ and $N = 24$, the *folding* and recovery procedure again fulfilled requirements a) and b).

1) # of perturbation sizes $\#_{pert}$

As mentioned earlier perturbations less than ϵ_{min} are not guaranteed to perturb the matrix $\tilde{\mathbf{G}}$. Therefore only perturbations greater than or equal to this amount will be considered in this discussion.

The conclusions below assume that the results for the discrete set $\mathcal{E}_1$ hold for all ϵ_{min} precision perturbations between ϵ_{min} and 10^{-3}. This assumption results in a number of unique perturbation sizes $\#_{pert}$, of $\#_{pert} = \frac{10^{-3}}{\epsilon_{min}} \approx 2.25 \times 10^{12}$.

A difference of ϵ_{min} between the perturbation used to *fold* an image, and the perturbation used recover the image fulfilled requirement b). Therefore for this discussion it is assumed that each of the $\#_{pert}$ perturbations, result in a different *folded* image $\mathbf{I}^{f_1}$.

2) Location of the perturbation

First assume a single perturbation is applied to the same fixed location within each of the H_1 matrices $\tilde{\mathbf{G}}$, where H_1 is the number of *folded* image blocks $\mathbf{I}_q^{f_1}, q = 1, \ldots, Q^N$. Because the location is fixed, the total number of permutations of location over the H_1 blocks, is only N^4, the same number of positions in a single matrix $\tilde{\mathbf{G}}$. Alternatively if the location with in each block is allowed to be different, the result would be, $(N^4)^{H_1}$ permutations of the location, over all the H_1 image blocks. For this to be applied in practice, the H_1 locations used at the *folding* stage, would have to be be passed to the recipient of the *folded* image, to allow it to be recovered.

An alternative to this, applied in Experiment 3.4.4, is to generate the locations within each $\tilde{\mathbf{G}}$ with a PRNG, initialized with a known seed. This would require only one parameter, the seed, to be passed to the recipient of the *folded* image. The total number of permutations over all image blocks would then be, the minimum of $(N^4)^{H_1}$ and the total number of seeds s_{max}, which can initialize the PRNG.

The pseudorandom numbers could be generated by the RC4 stream cipher which has an $s_{max} = 2^{256} \approx 1.16 \times 10^{77}$. In this case any image with $H_1 \geq \lceil \frac{256}{\log_2(N^4)} \rceil$, has s_{max} possible combinations of location, over the H_1 image blocks.

3) Number of perturbations N_ϵ

If the results of Experiment 3.4.5 hold for more perturbations, then a fixed combination of N_ϵ perturbations, $N_\epsilon < N^4$, could be applied. The SVD method only fulfils requirement a) and b) for perturbations smaller than 10^{-3}. Therefore to prevent the situation where combinations of two or more perturbations exceed this value, a maximum of one perturbation can be applied to each location.

If the perturbation applied to each of the N_ϵ locations is allowed to be different, the order the perturbations are applied to the N_ϵ locations, has to be the same at both the *folding* and recovery stage. Because the order is important, this is the number of permutations

$$P(N^4, N_\epsilon) = \frac{N^4!}{(N^4 - N_\epsilon)}. \tag{3.24}$$

If the same perturbation is applied to each of the N_ϵ locations, the order the perturbations are applied to the N_ϵ locations does not make any difference. Because the order

is not important this is the number of combinations

$$C(N^4, N_\epsilon) = \frac{P(N^4, N_\epsilon)}{N_\epsilon!}. \tag{3.25}$$

From the above it is clear that applying a different perturbation to each location results in a larger N_ϵ shown in equation (3.24). Therefore the number of permutations for $N_\epsilon = 2, 3$ and 4 different perturbations, is shown in the against the block size N in the Table below.

N	8	16	24
$P(N^4, 2)$	1.67×10^7	4.29×10^9	1.10×10^{11}
$P(N^4, 3)$	6.87×10^{10}	2.81×10^{14}	3.65×10^{16}
$P(N^4, 4)$	2.81×10^{14}	1.84×10^{19}	1.21×10^{22}

Choice of Private Key

Under the assumption that each perturbation size and location combination result in a different *folded* image, the *private key* can be, either:

1) A single perturbation. In this case the only two parameters which can be changed, are the perturbation size and its location. Therefore the choice for a *private key* can be, either one or the other, or a combination of the two.

   The number of locations resulting from choosing a pseudorandom location within each block depends on H_1. In extreme cases such as $H_1 = 1$, this results in only N^4 locations. Therefore the location on its own is not suitable as a *private key*. The location could however be applied in combination with the size of the perturbation. This would increase the *keyspace*, from the $\#_{pert}$ to a minimum of $N^4\#_{pert}$, show below for the block sizes $N = 8, 16$ and 24.

N	8	16	24
keyspace	9.22×10^{15}	1.48×10^{17}	7.47×10^{17}

2) N_ϵ perturbations. Combining the perturbation size with location increases the number of permutations, from those shown in equation (3.24). Each location N_ϵ can have all the $\#_{pert}$ perturbations applied to it, however, once a perturbation has been applied the number of locations is decreased by one. Therefore for l perturbations the size of the *keyspace* is calculated from

$$keyspace = \frac{(N^4)!}{(N^4 - N_\epsilon)!}(\#_{pert})^{N_\epsilon}. \tag{3.26}$$

N	8	16	24
$N_\epsilon = 2$	8.51×10^{31}	2.18×10^{34}	5.58×10^{35}
$N_\epsilon = 3$	7.85×10^{47}	3.21×10^{51}	4.17×10^{53}
$N_\epsilon = 4$	7.23×10^{63}	4.74×10^{68}	3.12×10^{71}
$\vdots$	$\vdots$	$\vdots$	$\vdots$
$N_\epsilon = 10$	4.40×10^{159}	4.90×10^{171}	5.42×10^{178}

Table 3.1: The size of the *keyspace* for the SVD method is shown against the number of perturbations $N_\epsilon = 2, 3, 4$ and 10, for each block size $N = 8, 16$ and 24.

The *keyspace* is shown in Table 3.1 against the number of perturbations $l = 2, 3, 4$ and 10, for each block size $N = 8, 16$ and 24.

As shown in Table 3.1 the size of the *keyspace* rapidly increases with the number of perturbations N_ϵ. The result for $N_\epsilon = 10$ is shown to illustrate that, even with the smallest block size $N = 8$, this system can have a *keyspace* more than double the s_{max} of the RC4 stream cipher.

The *keysize* for the N_ϵ perturbations is much larger than for a single perturbation. Therefore the combinations of N_ϵ different perturbations will be the *private key* for the SVD method.

3.5 Examining the Random Method

Unlike the SVD security scheme, this method has only 2 possible parameters, both of which are used as the seeds used for initializing a CSPRNG. The first parameter s_1, initializes a CSPRNG which generates the matrices $\mathbf{Z}_n \in \mathbb{R}^{N \times N}, n = 1, \ldots, N_e$, required in equation (3.13). The second parameter s_2, initializes a second CSPRNG which produces the matrix of coefficients $\mathbf{U}$, required in equation (3.14)

The Experiments in this Section again apply the *folding* and recovery procedure described in Section 3.2, to sparse approximations of the 55 astronomical images. The sparse approximations are calculated with OMP2D, by selecting vectors from the column and row dictionaries, $\mathbf{D}_8^{c,1}$ and $\mathbf{D}_8^{r,1}$. The approximations are therefore identical to those from Section 3.4. The only difference in this Experiment, is the application of the random security scheme to prevent unauthorized access, to the hidden vectors of coefficients $\mathbf{e}_q, q = 1, \ldots, Q^N$.

The PRNG's were each initialized with 10 different seeds. The different seeds, s_1 and

s_2, were stored respectively in the vectors $\mathbf{s}_1$ and $\mathbf{s}_2$. The *folded* images were secured by the random method with each combination of seeds $\mathbf{s}_1(i), i = 1, \ldots, 10$ and $\mathbf{s}_2(l), l = 1, \ldots, 10$, totalling 100 different combinations.

In a similar way to the Experiments in Section 3.4, two simulations were performed to examine incorrect and correct recovery, from the *folding* procedure. The first simulates a third party attempting to recover the hidden portion of the image, without knowledge of the correct seed. This was realized by examining the $\overline{\text{Err}}$ at the recovery stage, for each of the 100 combination of i and l. The procedure was to apply the correct value of the first seed $s_1 = \mathbf{s}_2(i)$, and the incorrect value of the second seed $s_2 = \mathbf{s}_2(l) + 1$.

The second simulates the recovery obtained by a third party who knows the correct recovery seed. This is realized by examining the PSNR^a at the recovery stage, for each of the 100 combination of i and l. The procedure this time was to apply the correct value of both the first and second seed, that is $s_1 = \mathbf{s}_2(i)$, and $s_2 = \mathbf{s}_2(l)$.

The first Experiment in this Section examines requirements a) and b), described in Section 3.4, with a fixed block size $N = 8$, and PSNR$^a = 45 \pm 4.5 \times 10^{-3}$%dB. The final two Experiments examine the effect of changing the values of N, and the PSNR^a on these requirements.

The Experiments in this Section are again run inside the MATLAB programming environment with the same configuration described in Section 1.7. All pseudorandom numbers will be generated by MATLAB's default Mersenne twister [73] PRNG, simulating a CSPRNG.

3.5.1 Experiment 1 - Requirements a) and b)

The effect of the seeds s_1 and s_2, which initialize the PRNG, on requirements a) and b) is investigated. Each image is initially partitioned into blocks with $N = 8$ and approximated to a PSNR$^a = 45 \pm 4.5 \times 10^{-3}$%dB.

The results given in the left half of the top row of Table 3.2, show the average $\overline{\text{Err}}$ for all combinations of seeds was 140.00%. More importantly the standard deviation between the $\overline{\text{Err}}$ resulting from all combinations of seeds in $\mathbf{s}_1$ and $\mathbf{s}_2$, was only 6.66×10^{-2}%. This Indicates that the actual value of the seed may have very little influence on the security of the method.

The results given in the right half of the top row of Table 3.2, show an average $\overline{\delta\text{PSNR}}$ of 2.46×10^{-14} in the recovery. This level of distortion, nearly the same order as the machines precision, is negligible.

	$\overline{\text{Err}}$		$\overline{\delta\text{PSNR}}$	
PSNR$^{\mathbf{a}}$	$\bar{\mathbf{x}}$	$\mathbf{s}$	$\bar{\mathbf{x}}$	$\mathbf{s}$
45	140.00%	6.66×10^{-2}%	2.46×10^{-14}	1.42×10^{-14}
55	139.64%	8.81×10^{-2}%	0	0
65	138.95%	1.39×10^{-1}%	0	0

Table 3.2: $\overline{\text{Err}}$ and $\overline{\delta\text{PSNR}}$ against each approximation PSNR^a. $\bar{x}$ is the mean over the 100 different combinations of the 10 seeds in both $\mathbf{s}_1$ and $\mathbf{s}_2$, and s is the standard deviation of the $\overline{\text{Err}}$ or $\overline{\delta\text{PSNR}}$ between the seed combinations.

3.5.2 Experiment 2 - Effect of the PSNR^a

The effect of the PSNR^a of the approximated image $\mathbf{I}^K$, on requirements a) and b) above is examined. In this Experiment the images are approximated in blocks with $N = 8$, to two quality levels of PSNR$^a = 55 \pm 5.5 \times 10^{-3}$%dB and PSNR$^a = 65 \pm 6.5 \times 10^{-3}$%dB.

The results are shown in the bottom 2 rows of Table 3.2, for these two additional quality levels. The average $\overline{\text{Err}}$ shown in the first column of Table 3.2 decreases marginally when the PSNR^a is increased. However this value is still far in excess of 100%, indicating a failure to recover the hidden coefficients $\mathbf{e}$.

The standard deviation between the $\overline{\text{Err}}$ for each of the 100 seed pairs is shown in the second column of Table 3.2. For all quality levels it is below 2×10^{-2}, indicating that the seed has very little influence on the security of the method, for all tested levels of PSNR^a.

For both additional levels of PSNR^a, when the correct seed s_2 is applied at the recovery stage, the recovered images $\tilde{\mathbf{I}}^K$, were exactly the same as the original approximated ones, $\mathbf{I}^K$.

3.5.3 Experiment 3 - Effect of the Block Size N

The effect of increasing the block size N which the images are processed in, on the range of perturbations fulfilling requirements a) and b) above, is examined. In this Experiment the images are first partitioned into blocks with $N = 16$ and 24, and then approximated to a PSNR$^a = 45 \pm 4.5 \times 10^{-3}$%dB.

The results are shown in the bottom 2 rows of Table 3.3, for these two larger block sizes. The average $\overline{\text{Err}}$ shown in the first column of Table 3.2 increases marginally when the block size N, is increased. Again this value is still far in excess of 100%, indicating a failure to recover the hidden coefficients $\mathbf{e}$. The standard deviation between the $\overline{\text{Err}}$ for each of the 100 seed pairs shown in the second column of Table 3.3. It is below 7×10^{-3}

	$\overline{\text{Err}}$		$\overline{\delta\text{PSNR}}$	
N	$\bar{\mathbf{x}}$	**s**	$\bar{\mathbf{x}}$	**s**
8	140.00%	$6.66 \times 10^{-2}\%$	2.46×10^{-14}	1.42×10^{-14}
16	141.09%	$5.09 \times 10^{-2}\%$	0	0
24	141.24%	$4.77 \times 10^{-2}\%$	1.61×10^{-15}	3.39×10^{-15}

Table 3.3: $\overline{\text{Err}}$ and $\overline{\delta\text{PSNR}}$ for each block size N. $\bar{x}$ is the mean over the 100 different combinations of the 10 seeds in both $\mathbf{s}_1$ and $\mathbf{s}_2$, and s is the standard deviation of the $\overline{\text{Err}}$ or $\overline{\delta\text{PSNR}}$ between the seed combinations.

for all block sizes N, indicating that the actual value of the seed has little influence on the security of the method, for these larger blocks.

The results with the correct seed s_2 at the recovery stage, are shown in columns 3 and 4 of Table 3.3. The results show that for the larger block sizes $N = 16$ and 24, the images $\tilde{\mathbf{I}}^K$ were recovered with less distortion than the smaller block size $N = 8$. However the level of distortion, close to machine precision, is again negligible.

3.5.4 Discussion

Images were initially split into blocks with $N = 8$, and approximated to a $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-3}\%\text{dB}$. For this block size and quality level, requirements a) and b) were satisfied, for the 100 different combinations of seeds which initialized the PRNG. This result was not affected by increasing the block size to $N = 16$ and 24. It was also not affected by with $N = 8$ when increasing the PSNR^a to either $\text{PSNR}^a = 55 \pm 5.5 \times 10^{-3}\%\text{dB}$ or $\text{PSNR}^a = 65 \pm 6.5 \times 10^{-3}\%\text{dB}$. That is requirements a) and b) were still satisfied, for $N = 16$ and 24, and $\text{PSNR}^a = 55 \pm 5.5 \times 10^{-3}\%\text{dB}$ or $\text{PSNR}^a = 65 \pm 6.5 \times 10^{-3}\%\text{dB}$.

The standard deviation between the $\overline{\text{Err}}$ for each seed combination was always less than 2×10^{-1}. The average $\overline{\text{Err}}$ over all seed combinations was in also in excess of 138%. That is for all block sizes $N = 8, 16$ and 24, and all quality levels $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-3}\%\text{dB}$, $55 \pm 5.5 \times 10^{-3}\%$ and $65 \pm 6.5 \times 10^{-3}\%\text{dB}$. Both these results imply that the actual value of the seed does not influence the security of the random method.

The average $\overline{\delta\text{PSNR}}$ over all seed combinations was less than $1 \times 10^{-13}\%$. This is for all block sizes $N = 8, 16$ and 24, and quality levels $\text{PSNR}^a = 45 \pm 4.5 \times 10^{-3}\%\text{dB}$, $55 \pm 5.5 \times 10^{-3}\%$ and $65 \pm 6.5 \times 10^{-3}\%\text{dB}$. This level of distortion, close the the machines precision is much smaller than $x\%$, required to fulfil requirement a).

3.5.5 The Keysize for the Random Method

As previously mentioned, the random method requires only two parameters, both of which are seeds for initializing a CSPRNG. Therefore the combination of the two seeds, initializing the CSPRNG's, will be the *private key* for the random method. The maximum number of seeds are s^1_{max} and s^2_{max}, respectively for initializing the CSPRNG's. Therefore the number of permutations and hence the *keysize* for the random method is $s^1_{max}s^2_{max}$.

To try and increase the security a different CSPRNG could be applied to generate the matrices $\mathbf{Z}$ and $\mathbf{U}$. If instead of this the RC4 stream cipher is applied to generate both matrices or alternatively two generators with the same $s_{max} = 2^{256}$ are used, the *keysize* for the random method would be $2^{512} \approx 1.34^{154}$.

3.6 Comparison of the Random and SVD Security Schemes

Procedure

The SVD security scheme relies on the instability in the calculation of singular vectors of a rank deficient matrix, corresponding to the zero singular values.

The random security scheme relies entirely on the security of a CSPRNG. This could be an advantage because the specific CSPRNG can easily be changed, for either one with a larger s_{max}, or one providing a higher level of security. The example CSPRNG given in the discussion above, was the RC4 stream cipher. Unlike the SVD method, which is based on a recently introduced idea [87], both stream ciphers and CSPRNG have been widely studied, in the field of cryptography [53, 93].

Keysize

The *keysize* for the SVD method increases with the size of the blocks N, and the number of perturbations N_ϵ, applied to each block. Only $N_\epsilon = 2$ perturbations were tested and found to fulfil requirements a) and b), in the Experiments of Section 3.4. Therefore the adopted *keysize* for the SVD method can only be calculated based on $N_\epsilon = 2$. The resulting *keysize* is shown in the first row of Table 3.1.

In contrast to the SVD method, the *keysize* for the random method relies entirely on the value s_{max}, for a CSPRNG.

To compare the *keysize* for the two methods consider the following example, again involving the RC4 stream cipher. As discussed in Section 3.5.5 this has a *keyspace* of $2^{512} \approx 1.34^{154}$. This is much larger than the *keysize* for the SVD method, when $N_\epsilon = 2$.

Now consider the possibility of $N_\epsilon > 2$ perturbations also fulfilling requirements a) and b). For this example with only $N_\epsilon = 10$, the *keysize* for the SVD scheme would be greater than the random method. The *keysize* for the SVD scheme for $N_\epsilon = 10$ is shown in the bottom of Table 3.1.

Effect on Recovery

Both the SVD and Random methods, prevented access to the hidden vectors $\mathbf{e}_q, q = 1, \ldots, Q^N$ when recovery is attempted with the incorrect *private key*.

The images were recovered, without an unacceptable level of distortion, by both SVD and Random methods and the correct *private key*. However the level of distortion introduced in to the recovered image $\tilde{\mathbf{I}}^K$, by the SVD method, was always larger than that introduced by the random method. This level of distortion also increased with the size of the perturbation.

3.7 Conclusions

Two methods have been proposed for securing information in a sparse approximation of an image. The SVD method relies on the instability in the calculation of singular vectors, corresponding to multiple zero singular values. The random method relies on a random transformation and the security of a CSPRNG.

The procedure called image *folding*, utilizes a *private key* to control access to the secured information. A range of *private keys* were proposed and tested under two simple criteria. The first, secured information should not be recoverable by someone who is not in possession of the correct *private key*. The second, when the correct *private key* is applied, the recovered image $\tilde{\mathbf{I}}^K$, should not be visibly different to the approximated image $\mathbf{I}^K$.

Both methods satisfied criteria b). However the amount of distortion, introduced into the recovered image $\tilde{\mathbf{I}}^K$ by the random method, was always below 2.46×10^{-14}. On the other hand the level of distortion introduced by the SVD method, was dependant on the size of the perturbation, and exceeded this amount for many perturbations tested.

The number of *keys* which satisfied criteria a) and b) for each scheme was then calculated to establish the *keysize* for each security scheme.

Under the given test conditions, the random method was shown to have a much larger *keysize* than the SVD method. However the SVD method opened the possibility of having a dramatically larger *keysize*, if more than two perturbations could be applied to the matrix $\tilde{\mathbf{G}}$. This was not tested experimentally to verify that it would still fulfil requirements a)

and b), but should be investigated in future work.

Because the level of distortion did not appear to depend on the *private key*, and because of the larger *keysize*, the random method will be applied to secure *folded* images, in all remaining Chapters.

Remark 5. As discussed in Section 1.6.1 the *keysize* is not a measure of the cryptographic security of a method. It is simply an enumeration of the number of *private keys* which can be applied to secure the information. Therefore the random method with the largest *keysize* may not in practice be the most secure.

4 Self Contained Encrypted Image Folding

The Chapter extends image *folding* procedure described in Chapter 3 to make it self contained. Recall that image *folding* involves embedding coefficients from sections of a sparse representation of an image, into other sections of the same sparse approximation. This procedure is not self contained, because, in addition to the image coefficients, other image dependent information is also required at the recovery stage.

The extension to image *folding* considered here, enlarges the *folded* image to includes all the information required for recovery. Thus all that is needed to recover the original sparse approximation, is the *folded* image and the *private key*.

Two methods for securing this additional information are described in this Chapter. The first which will be referred to as the "pixel" scheme stores the additional information as pixel intensities. The second referred to as the "ad hoc" scheme is a variation on the folding procedure from Chapter 3.

This procedure is then extended from grey level intensity images to RGB colour images. Two methods are considered for finding sparse approximations of RGB images, before the *folding* stage is applied. The first repeats the procedure for grey level images on each of the three planes of the colour image. The second describes an adaptation of the selection

criteria in equation (2.4), which chooses common atoms for each colour plane.

The contents of the Chapter are as follows: the first Section contains an overview of the additional information required to make the procedure self contained. The next two Sections describe the two methods for securing this additional information. Experiments then follow investigating the suitability of each method.

The remainder of the Chapter is then devoted to RGB images by first, describing the two methods for approximating, and *folding* colour images. Then comparing the CR resulting from these two methods, and finally presenting an illustration of the application of both schemes to a colour image.

4.1 Information Required For Recovery

The recovery procedure in Section 3.2.3 requires the set of matrices $\{\mathbf{S}_{\mathbf{l}_q(k)}\}_{k=1}^{K_q}, q = 1, \ldots, Q^N$. These are used to both recover the hidden coefficients, and to reconstruct the missing sections of the original image. Therefore without knowledge of the matrices, the image $\tilde{\mathbf{I}}^K$, in equation (3.10), cannot be recovered.

Because the dictionary is fixed, only the vectors of indices $\mathbf{l}_q, q = 1, \ldots, Q^N$ and not the sets of matrices for each block are required. Hence, the inclusion of these indices is the focus of the methods detailed in the next two Sections.

In addition to the *folded* image, *private key* and vectors of indices $\mathbf{l}_q, q = 1, \ldots, Q^N$, other parameters are also required to construct the image $\tilde{\mathbf{I}}^K$. These are:

1 Details of the column and row dictionaries $\mathbf{D}^c$ and $\mathbf{D}^r$, and dimension N that the image was processed with.

2 The dimensions of the original image $\mathbf{I}$, the number of host blocks H_1, and any padding $\mathbf{p}_1$ added to the hidden coefficients.

3 Quantization parameters $min_{p,1}$ and $\triangle_1$, required to recover the original values from the quantization indexes, discussed in Section 4.2.

The additional variables $1 - 3$ above will be stored without security as image header information. Such variables can be extracted before recovery takes place. This information will require a fixed amount of space and will not be considered in the procedures detailed in the next Sections.

4.2 Quantization and Storage issues

The *folded* image will now contain all the information required to construct the recovered image $\tilde{\mathbf{I}}^K$ from equation (3.10). Therefore the storage of this image needs to be considered. A simple quantization, and entropy coding approach is outlined below. The aim of this approach is to create a self contained *folded* image, requiring less storage space than the original image $\mathbf{I}$. That is applying the definition of the CR from equation (2.23), a *folded* image with CR > 1. The method described next is different to that discussed in Section 2.7, which stored the coefficients of the approximation. Here the *folded* image itself contains the coefficients, and it is that which needs to be stored.

Given an image $\mathbf{I}^{f_1} \in \mathbb{R}^{N_r \times N_c}$ *folded* by the procedure described in Chapter 3, the coded version $\mathbf{I}^{f_1,\triangle}$ (where each pixel is represented with u bits), is realized by means of the following steps:

- Calculate the vector of quantization indices $\mathbf{f}_1^{\triangle}$, representing the pixels in $\mathbf{I}^{f_1}$, by applying the mid-tread uniform quantizer from Section 2.7.1,

$$\mathbf{f}_1^{\triangle} = \widehat{\mathbf{FQuant}}(\widehat{\mathbf{Vec}}(\mathbf{I}^{f_1}) - min_{p,1}, \triangle_1). \tag{4.1}$$

  The minimum pixel intensity $min_{p,1}$, in $\mathbf{I}^{f_1}$, is subtracted to make each value in $\mathbf{f}_1^{\triangle}$ positive. This procedure is to avoid having to store the sign information separately.

- Choose a suitable entropy coding algorithm and convert the vector $\mathbf{f}_1^{\triangle}$, to a vector of bits $\mathbf{b}^{f_1}$ containing $N_b^{f_1}$ values.

- Take u bits at a time from $\mathbf{b}^{f_1}$ to create a temporary vector $\mathbf{t}$, containing $\lceil \frac{N_b^{f_1}}{u} \rceil$ pixels. The vector $\mathbf{t}$ is constructed as:

$$\mathbf{t}(n) = \widehat{\mathbf{UInt}}(\mathbf{b}^{f_1}(i, \ldots, i+u-1)), \quad i = nu, n = 1, \ldots, \left\lceil \frac{N_b^{f_1}}{u} \right\rceil. \tag{4.2}$$

  The operation denoted as $\widehat{\mathbf{UInt}}(\cdot)$, takes a vector of u bits $\mathbf{b}(i), i = 1, \ldots, u$, and converts them to an unsigned integer by the transformation $\sum_{i=1}^{u} \mathbf{b}(i)2^i$. The value of u in equation (4.13) can be any number greater than 1. However a suitable choice is the number of bits representing each pixel in the original image. Choosing this value guarantees *folded* images, constructed with the preceding steps, will be in the same format as the original images.

- The vector of pixels $\mathbf{t}$ can now be reshaped to become an image $\mathbf{I}^{f_1,\triangle} = \widehat{\mathbf{Mat}}(\mathbf{t}, N_r^1, N_c^1)$. The choice of N_r^1 or N_c^1 can be left up to the user. The image $\mathbf{I}^{f_1,\triangle}$ can now be stored in any standard lossless image format.

The quantized representation $\tilde{\mathbf{I}}^{f_1}$, of the *folded* image $\mathbf{I}^{f_1}$, can be recovered by simply reversing the procedure above by following the steps below:

- First relabel the elements of the image $\mathbf{I}^{f_1,\triangle}$ back to become the vector $\mathbf{t} = \widehat{\mathbf{Vec}}(\mathbf{I}^{f_1,\triangle})$.
- Convert each unsigned integer in $\mathbf{t}$ to its binary representation to populate the bit stream $\mathbf{b}^{f_1}$,

$$\mathbf{b}^{f_1}(i,\ldots,i+u-1) = \widehat{\mathbf{Bin}}(\mathbf{t}(n),u), \quad i = nu, n = 1,\ldots,\left\lceil \frac{N_b^{f_1}}{u} \right\rceil.$$

  The operation $\widehat{\mathbf{Bin}}(\cdot,u)$ takes an unsigned integer and converts it to a u bit representation.

- Apply the decoding stage of entropy coding algorithm chosen above to, decode the the bit stream $\mathbf{b}^{f_1}$, and recover the original vector of quantization indices $\mathbf{f}_1^{\triangle}$.
- Reverse the quantization applied to $\mathbf{f}_1^{\triangle}$. Then relabel the elements of $\mathbf{f}_1^{\triangle}$ to recover the quantized version of $\mathbf{I}^{f_1} \in \mathbb{R}^{N_r \times N_c}$, with,

$$\tilde{\mathbf{I}}^{f_1} = \widehat{\mathbf{Mat}}(\widehat{\mathbf{RQuant}}(\mathbf{f}_1^{\triangle},\triangle) + min_{p,1}, N_r, N_c).$$

4.3 Examining the Storage Requirements of the $\mathbf{I}^{f_1,\triangle}$

In this Section the quantization step size $\triangle$ is investigated. A suitable $\triangle$ should reduce the maximum value in $\mathbf{I}^{f_1,\triangle}$, without introducing an unacceptable level of distortion, into the recovered image $\tilde{\mathbf{I}}^{f_1}$.

Before proceeding the quantization step size is reformulated in terms of the maximum number of quantization indices, in the representation of $\mathbf{I}^{f_1,\triangle}$. That is

$$\triangle_1 = \frac{2^{u_1-1}}{max_{p,1} - min_{p,1}}. \tag{4.3}$$

The value 2^{u_1-1} is the maximum number of quantization indices and, $max_{p,1}$ and $min_{p,1}$ are respectively, the minimum and maximum values of the pixels in the *folded* image $\mathbf{I}^{f_1}$.

The number of bits required to store the *folded* image $\mathbf{I}^{f_1} \in \mathbb{R}^{N_r \times N_c}$, is $N_b = uN_rN_c$, with u the number of bits required to store each pixel of $\mathbf{I}^{f_1}$. The CR of the *folded* image can be calculated by substituting this into equation (2.23). Quantizing $\mathbf{I}^{f_1}$ so that each pixel can be represented by u_1 bits with $u_1 < u$, will reduce the size of N_b and hence increase the CR.

As discussed in Section 4.2, quantization will also introduce distortion into the recovered image $\tilde{\mathbf{I}}^{f_1}$, the question now becomes how much distortion is acceptable. The

images in the following Experiments will again be initially approximated to a $\text{PSNR}^a = 45 \pm 1 \times 10^{-2}\%$ before applying the *folding* procedure. A variance of $\pm 1 \times 10^{-2}\%$ is allowed in the approximation PSNR. Therefore a suitable amount of variance for the *folding* procedure, which now includes a quantization stage, will be one order of magnitude greater than this. This is realized by allowing a $\overline{\delta\text{PSNR}} \leq 1 \times 10^{-1}\%$ between, the approximated images $\mathbf{I}^K$ and the recovered images $\tilde{\mathbf{I}}^K$.

4.3.1 Experimental Overview

In this Experiment and the remaining experiments in this Chapter, two sets of 55 grey level images, were approximated and then *folded*. The final image was quantized by applying the additional quantization step described above, with range of values of u_1. The images were then recovered, and the smallest number of bits u_1, which resulted in a $\overline{\delta\text{PSNR}} \leq 1 \times 10^{-2}\%$ between, the approximated and the recovered images found.

A range of dictionaries of increasing size will be required in the next Section, to test the *folded* images containing the index information. Therefore in this Experiment, a range of block sizes $N = 8, 16, 24$ were tested, together with a range of 8 dictionaries of increasing size.

The B-spline based dictionaries showed promising approximation performance, over the astronomical images in Experiment 2.6.1. Therefore in this Experiment, both the RDBS and RDC components present in these dictionaries, and described in Section 2.4, will be included again.

Each of the 8 dictionaries in this Experiment were separable, with $\mathbf{D}_i^{c,2} \equiv \mathbf{D}_i^{r,2}, i = 1, \ldots, 8$, therefore the construction of the column dictionary $\mathbf{D}_i^{c,2}$ for each is given below:

1) $\mathbf{D}_1^{c,2} = \mathbf{D}_1^c$, (RDC).

2) $\mathbf{D}_2^{c,2} = \mathbf{D}_8^{c,1} = [\mathbf{D}_1^{c,2}, \mathbf{D}_2^c]$.

3) $\mathbf{D}_3^{c,2} = [\mathbf{D}_2^{c,2}, \{U_2(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_2}]$.

4) $\mathbf{D}_4^{c,2} = [\mathbf{D}_3^{c,2}, \{b_i Y_2^2(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_2}]$.

5) $\mathbf{D}_5^{c,2} = [\mathbf{D}_4^{c,2}, \{U_4(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_4}]$.

6) $\mathbf{D}_6^{c,2} = [\mathbf{D}_5^{c,2}, \{b_i Y_2^3; (n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_3}]$.

7) $\mathbf{D}_7^{c,2} = [\mathbf{D}_6^{c,2}, \{U_6(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_6}]$.

8) $\mathbf{D}_8^{c,2} = \mathbf{D}_5^{c,1} = [\mathbf{D}_7^{c,2}, \{b_i Y_2^4(n-i+1); n = j, \ldots, N+j-1\}_{i=1}^{M_4}]$, (RDC-RDBS$_2$).

	$\mathbf{M_i}$							
N	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**
8	16	24	31	39	46	54	59	67
16	320	48	63	79	94	110	123	139
24	48	72	95	119	142	166	187	211

Table 4.1: The number of vectors $\mathbf{M_i}$, in each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ against the size of the block $\mathbf{N}$.

The number of vectors M_i, in each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ is shown in Table 4.1, against the size of the block N.

The Experiment was performed with two sets of 55 images. The first is the astronomical images from Chapter 3, the second is a combination of the training and test set, of natural images, from in Chapter 2

As mentioned above all the test images were first approximated to a $\text{PSNR}^a = 45 \pm 1 \times 10^{-2}\%$, before being *folded*. The *folded* images $\mathbf{I}^{f_1}$ were all quantized to $u_1 = 8, \ldots, 16$ bits with equation (4.1). The quantization step size $\triangle$ determined by the values of u_1, with equation (4.3). In this Experiment the vectors of indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ for each image, are stored exactly. They are then used at the recovery stage, to ensure that the only distortion from the quantization procedure, is introduced into the recovered image $\tilde{\mathbf{I}}^K$.

The security was considered applying the random method from Section 3.3.2, with a *private key* composed of two seeds, $s_1 = 1.43398 \times 10^6$, and $=2.365658978 \times 10^9$. For this Experiment both the *folding* and recovery stages were performed with the same *private key*.

Experiment

The Experiment described above was first performed on the set of 55, 8 bit grey level astronomical images.

Figure 4.1 shows the $\overline{\delta\text{PSNR}}$ from $\mathbf{D}_1^{c,2}$ for each N, against the number of bits u_1, $\mathbf{I}^{f_1}$ was quantize to. As discussed earlier the maximum allowable distortion, is one order of magnitude greater, than that introduced at the approximation stage. This is indicated in Figure 4.1 by the dashed $\triangledown$ line, which equates to a PSNR between the original and recovered image, of $44.995\text{dB} < \text{PSNR} < 45.045\text{dB}$. The Figure shows that for $\mathbf{D}_1^{c,2}$, a value of $u_1 \geq 12$ results in an acceptable $\overline{\delta\text{PSNR}}$, for all block sizes N.

The result for $\mathbf{D}_1^{c,2}$ was the same for all other dictionaries $\mathbf{D}_i^{c,2}, i = 2, \ldots, 8$, that is a

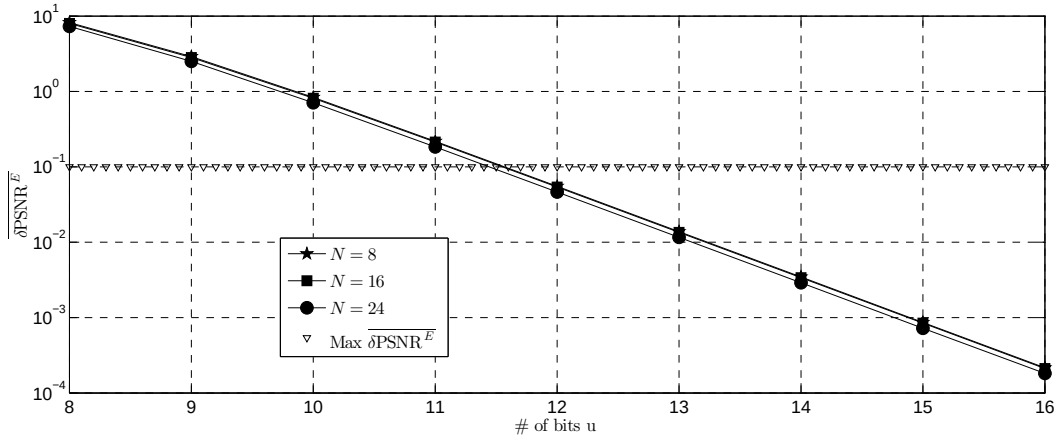

Figure 4.1: $\overline{\delta\text{PSNR}}$ over the 55, 8 bit grey astronomical image set, when approximated with dictionary $\mathbf{D}_1^{c,2}$ and stored with $u_1 = 8, \ldots, 16$ by applying the method in Section 4.2. The $\overline{\delta\text{PSNR}}$ is shown for each block size N against the number of bits u_1 the images were quantized to. The standard deviation between the 55 images was of the same order or less than the $\overline{\delta\text{PSNR}}$, for all block sizes N and # of bits u_1. The dashes $\triangledown$ indicate the maximum tolerable $\overline{\delta\text{PSNR}}$.

value of $u_1 \geq 12$, results in an acceptable $\overline{\delta\text{PSNR}}$ for all block sizes N. This is demonstrated in Figure 4.1, which shows the $\overline{\delta\text{PSNR}}$ for each block N against the dictionary, for both $u_1 = 11$ and $u_1 = 12$.

The same Experiment described above was performed again on the set of 55, 8 bit grey level natural images.

The result for the natural image set was the same as the result for the astronomical image set. That is $u_1 \geq 12$ bits were required for the images to be recovered to an acceptable level of $\overline{\delta\text{PSNR}}$, for all block sizes N.

4.4 The "Ad Hoc" Scheme

A method for storing the indexes in an additional image $\mathbf{I}^{f_2}$ has been proposed in [4]. The method involves creating some "ad hoc" *host* blocks, for embedding the vectors of indices $\mathbf{l}_q, q = 1 \ldots, Q^N$, with a similar method to that proposed in Chapter 3 for the coefficients $\mathbf{e}_q, q = 1 \ldots, H_1$. These indices can then be secured, by either of the security schemes proposed in Chapter 3.

The method presented in this Section extends that proposed in [4] to apply the simple quantization and entropy coding stage from Section 4.2, to reduce the storage requirements of the *folded* image.

The self contained *folded* image, denoted as $\mathbf{I}^f$, will be the union of two *folded* images. The first is $\mathbf{I}^{f_1}$ from Chapter 3 containing the coefficient information, the second is the newly created "ad hoc" *folded* image $\mathbf{I}^{f_2}$, containing the index information.

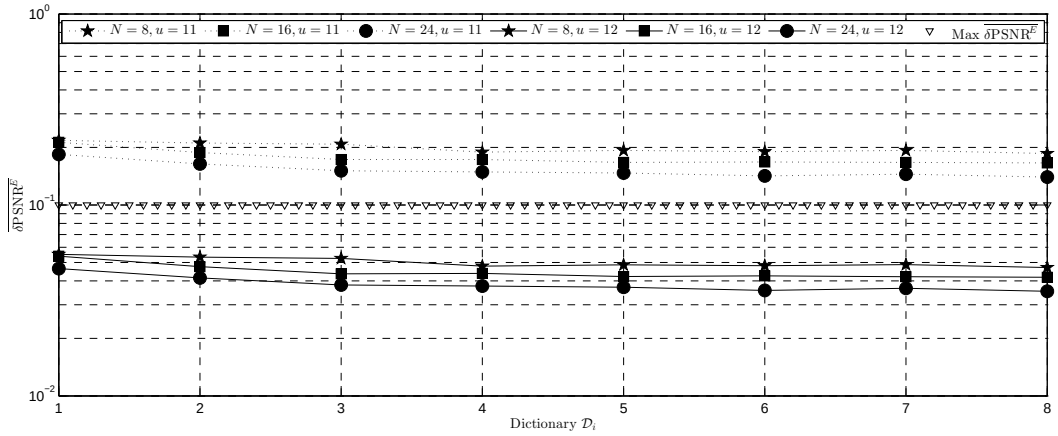

Figure 4.2: $\overline{\delta\text{PSNR}}$ over the 55, 8 bit grey astronomical image set, when approximated with each dictionary $\mathbf{D}_i^{c,2}, i = 2, 3, 4, 5, 6, 7, 8$ and stored to $u_1 = 11$ and $u_1 = 12$ bits by applying the method in Section 4.2. The $\overline{\delta\text{PSNR}}$ is shown for each block size N against the index of the dictionary applied in the approximation. The standard deviation between the 55 images was of the same order or less than the $\overline{\delta\text{PSNR}}$, for all block sizes N and dictionaries. The dashes $\triangledown$ indicate the maximum tolerable $\overline{\delta\text{PSNR}}$.

4.4.1 Folding Procedure

In the procedure below, the vectors of indices $\{\mathbf{l}_q(k), k = 1, \ldots, K_q\}_{q=1}^{Q^N}$ are placed into a larger vector, through the $\widehat{\mathbf{Flat}}(\cdot)$ operation. To recover the vectors of indices from this operation the numbers $K_q, q = 1, \ldots, Q^N$ need to be known. Therefore in the description below, these numbers are also stored in the "ad hoc" *folded* image $\mathbf{I}^{f_2}$.

- Unlike the *host* blocks required in Chapter 3, the "ad hoc" *host* blocks are artificially created square $N \times N$ constant intensity arrays. The intensity of which is dictated by the values $K_q, q = 1, \ldots, H_2$ where $H_2 = \lceil \frac{K+Q^N}{N^2} \rceil$.

  With any normalized to unity matrix, say $\mathbf{J} \in \mathbb{W} \subset \mathbb{R}^{N \times N}$, the "ad hoc" intensity arrays $\ddot{\mathbf{J}}_q \in \mathbb{R}^{N \times N}, q = 1, \ldots, H_2$ are created as

  $$\ddot{\mathbf{J}}_q = K_q \mathbf{J}_q, q = 1, \ldots, H_2. \tag{4.4}$$

  This leaves space for $N_h = N^2 - 1$ indices, to be embedded in the orthogonal complement $\mathbb{W}^{\perp}$ (with respect to $\mathbb{R}^{N \times N}$), of the subspace spanned by the single matrix $\mathbf{J}$.

- The vectors of indices $\{\mathbf{l}_q(k), k = 1, \ldots, K_q\}_{q=1}^{Q^N}$ and the remaining $K_q, q = H_2 + 1, \ldots, Q^N$ are relabelled, to become the vectors of coefficients $\{\mathbf{h}(n), n = 1, \ldots, N_h\}_{q=1}^{H_2}$ by applying the $\widehat{\mathbf{Flat}}(\cdot)$ and $\widehat{\mathbf{Set}}(\cdot)$ operations defined in Appendix A. If $N_b = (H_2 N^2 - (K + Q^N))$ is greater than zero, then a vector of padding $\mathbf{p}_2(n) \in \mathbb{R}^{N_b}$ will

be included as shown below:

$$\{\mathbf{h}\}_{q=1}^{H_2} = \widehat{\mathbf{Set}}([\widehat{\mathbf{Flat}}(\{\mathbf{l}_q\}_{q=1}^{Q^N}); K_{H_2+1}; \ldots; K_{Q^N}; \mathbf{p}_2], \mathbf{n}_h), \tag{4.5}$$

where each element of the vector $\mathbf{n}_h \in \mathbb{R}^{H_2}$ is N_h, and the elements of the vector $\mathbf{p}_2$ can take on any value.

- The procedure for embedding these vectors of coefficients in each $\ddot{\mathbf{J}}_q, q = 1, \ldots H_2$ is then equivalent to that for embedding the vectors of coefficients $\mathbf{e}_q$ in the host blocks in Section 3.2.2, that is,
  - Using either the SVD or random security scheme described in Chapter 3, generate the sets of orthonormal basis for the space $\mathbb{W}^{\perp} = \text{span}\{\mathbf{W}_{q,n}^{\perp}\}_{n=1}^{N_h}, q = 1, \ldots, Q^N$ orthogonal to the single matrix $\mathbf{J}$.
  - Build each embedded matrix $\mathbf{H}_q$ as

$$\mathbf{H}_q = \sum_{n=1}^{N_h} \mathbf{h}_q(n)\mathbf{W}_{q,n}^{\perp}, q = 1, \ldots, H_2. \tag{4.6}$$

  - For $q = 1, \ldots, H_2$ *fold* the blocks by the superposition $\mathbf{I}_q^{f_2} = \ddot{\mathbf{J}}_q + \mathbf{H}_q$ and subsequent composition $\mathbf{I}^{f_2} = \cup_{q=1}^{H_2} \mathbf{I}_q^{f_2}$.

The procedure described in Section 4.2 for quantizing and coding the *folded* image $\mathbf{I}^{f_1}$ can then be applied to $\mathbf{I}^{f_2}$ as shown below:

- Calculate the vector of quantization indices $\mathbf{f}_2^{\triangle}$ representing the pixels in $\mathbf{I}^{f_2}$ by the mid-tread uniform quantizer from Section 2.7.1 as

$$\mathbf{f}_2^{\triangle} = \widehat{\mathbf{FQuant}}(\widehat{\mathbf{Vec}}(\mathbf{I}^{f_2}) - min_{p,2}, \triangle_2).$$

- Applying a suitable entropy coding algorithm convert the vector $\mathbf{f}_2^{\triangle}$ to a vector $\mathbf{b}^{f_2,\triangle}$ containing $N_b^{f_2,\triangle}$ bits.

- Take u bits at a time from $\mathbf{b}^{f_2,\triangle}$ to create a temporary vector $\mathbf{t}$ containing $\lceil \frac{N_b^{f_2,\triangle}}{u} \rceil$ pixels as,

$$\mathbf{t}(n) = \widehat{\mathbf{UInt}}(\mathbf{b}^{f_2,\triangle}(i, \ldots, i+u-1)), \quad i = nu, n = 1, \ldots, \left\lceil \frac{N_b^{f_2,\triangle}}{u} \right\rceil. \tag{4.7}$$

  Again the value of u in equation (4.7) can be any number greater than 1 however a suitable choice is the number of bits to represent each pixel in the original image.

- The temporary vector of pixels $\mathbf{t}$ can now be reshaped to become an image $\mathbf{I}^{f_2,\triangle} = \widehat{\mathbf{Mat}}(\mathbf{t}, N_r^2, N_c^2)$, where the choice of N_r^2 or N_c^2 can be left up to the user. The image $\mathbf{I}^{f_2,\triangle}$ can now be stored in any standard lossless image format.

The self contained *folded* image $\mathbf{I}^f$ is then the union of the two images, $\mathbf{I}^{f_1,\triangle}$ containing the vectors of coefficients $\mathbf{c}_q$, and $\mathbf{I}^{f_2,\triangle}$ containing the vectors of indices $\mathbf{l}_q$. These are both required in equation (3.4) to calculate the recovered image blocks $\tilde{\mathbf{I}}_q^K$, with $q = 1, \ldots, Q^N$.

4.4.2 Recovery Procedure

The procedure for recovering the vectors of indices $\mathbf{l}_q, q = 1, \ldots, Q^N$ from the self contained *folded* image $\mathbf{I}^f$ is as follows.

First the self contained *folded* image $\mathbf{I}^f$ is split into $\mathbf{I}^{f_1,\triangle}$ and $\mathbf{I}^{f_2,\triangle}$, before reversing the quantization and coding procedure in Section 4.2 to recover the *folded* image $\tilde{\mathbf{I}}^{f_2}$.

The image $\tilde{\mathbf{I}}^{f_2}$ is then divided back into square $N \times N$ blocks $\tilde{\mathbf{I}}_q^{f_2}, q = 1, \ldots, H_2$ and the vectors of indices $\mathbf{l}_q, q = 1, \ldots, Q^N$ for each block are recovered by the following steps:

- The numbers $\tilde{K}_q, q = 1, \ldots, H_2$ are obtained as

$$\tilde{K}_q = \langle \mathbf{J}, \tilde{\mathbf{I}}_q^{f_2} \rangle_F, \quad q = 1, \ldots, H_2$$

- Obtain $\tilde{\mathbf{H}}_q$ as $\tilde{\mathbf{H}}_q = \tilde{\mathbf{I}}_q^{f_2} - \tilde{K}_q \mathbf{J}, q = 1, \ldots, H_2$.

- By the same security scheme as in the *folding* stage, generate the sets of matrices $\{\mathbf{W}_{q,n}^{\perp}, n = 1, \ldots, N_h\}_{q=1}^{H_2}$, which are an orthonormal basis for the space $\mathbb{W}^{\perp}$.

- Recover the coefficients $\{\tilde{\mathbf{h}}_q(n), n = 1, \ldots, N_h\}_{q=1}^{H_2}$ from $\tilde{\mathbf{H}}_q$,

$$\tilde{\mathbf{h}}_q(n) = \langle \tilde{\mathbf{H}}_q, \mathbf{W}_{q,n}^{\perp} \rangle_F, \quad n = 1, \ldots, N_e, q = 1, \ldots, H_2. \tag{4.8}$$

- Flatten the vectors of coefficients $\{\tilde{\mathbf{h}}_q(n), n = 1, \ldots, N_e\}_{q=1}^{H_2}$ to get a temporary vector,

$$\mathbf{t} = \widehat{\mathbf{Flat}}(\{\tilde{\mathbf{h}}_q\}_{q=1}^{H_2}),$$

  remove any padding $\mathbf{p}_2$ and recover the remaining $\tilde{K}_q, q = H_2 + 1, \ldots, Q^N$ from the last $Q^N - H_2 - 1$ remaining elements of $\mathbf{t}$.

- Relabel the first K elements of the temporary vector $\mathbf{t}$, to obtain the recovered vectors of indices

$$\{\tilde{\mathbf{l}}_q\}_{q=1}^{Q^N} = \widehat{\mathbf{Set}}(\mathbf{t}(1:K), \mathbf{n}_{\tilde{K}}),$$

  with K is the number of coefficients chosen for all blocks in the whole image calculated in equation (2.17), and the vector $\mathbf{n}_{\tilde{K}}(q) = \tilde{K}_q, q = 1 \ldots, Q^N$.

4.5 Examining the Storage Requirements of the $\mathbf{I}^{f_2}$

In this Section the number of bits u_2 which the pixels of the *folded* image $\mathbf{I}^{f_2}$ can be quantized with the mid-tread uniform quantizer described in Section 4.2 is investigated.

When $\mathbf{I}^{f_1}$ was quantized in Section 4.8.1 the constraint on the value of u_1 was that the $\overline{\delta\text{PSNR}} \leq 1 \times 10^{-1}\%$. This criteria cannot be applied to the index information stored in $\mathbf{I}^{f_2}$, because it needs to be recovered exactly, to enable the *folded* image to be recovered. Therefore the aim of the Experiment below is to find the smallest value of u_2, which for all sample images, can be used without effecting the recovery of the index information.

The size of each element of the "ad hoc" folded image $\mathbf{I}^{f_2,\triangle}$ is dependent on the indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$. If on average larger indices are chosen the expectation would be that on average the size of the elements of $\mathbf{I}^{f_2,\triangle}$ and hence the number of bits u_2 required to quantize each element, may also increase.

Table 4.1 shows that the number of vectors M_i in $\mathbf{D}_i^{c,2}, i = 1, \ldots 8$ increases both with the index of the dictionary and the size of the blocks N, partitioning the image. Therefore as either the dictionary index is increased or the block size N the number of vectors in $\mathbf{D}_i^{c,2}$ will also increase. With $\mathbf{D}_i^{c,2} \equiv \mathbf{D}_i^{r,2}$ the number of separable indices stored in the "ad hoc" image blocks, $M_i^2, i = 1, \ldots, 8$, will also increase with both the dictionary index and block size N.

From the above it is clear that larger dictionaries or blocks means that on average larger indices can be chosen, however it does not mean that this will happen. Therefore the Experiment below is to determine if increasing the dictionary size or the size of the blocks N requires an increase in the number of bits u_2, and if it does what level of increase is required.

4.5.1 Experimental Overview

The 55 astronomical and natural images were again used as the sample for this Experiment. The vectors of indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ resulting from approximating these images to a $\text{PSNR}^a = 45 \pm 1 \times 10^{-2}\%$ were used to create the "ad hoc" *folded* images with the procedure described in Section 4.4.

Each experiment was performed with the 8 dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ described in Section 4.7.5 and 3 block sizes $N = 8, 16$ and 24.

For $u_2 = 8, \ldots, 20$ the resulting *folded* images $\mathbf{I}^{f_2}$ were quantized to u_2 bits with the mid-tread uniform quantizer described in Section 4.2 before applying the recovery scheme from Section 4.4.2. The smallest value of u_2 which allowed the index information for all

images in the sample to be recovered exactly was stored.

The security was again realized by the random method from Section 3.3.2 with a *private key* composed of two seeds for initializing the PRNG's, $s_1 = 1.43398 \times 10^6$ and $s_2 = 2.365658978 \times 10^9$. For this Experiment the same *private key* was applied at both the *folding* and recovery stages.

The normalized to unity matrix $\mathbf{J} \in \mathbb{R}^{N \times N}$ in (4.4) was a the constant matrix $\mathbf{J}(n_r, n_c) = \frac{1}{N}, n_r, n_c = 1, \ldots, N$.

Results

The minimum number of bits u_2 required to perfectly recover the vectors of indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ resulting from the approximation of the astronomical and natural images are shown respectively in the left and right of Table 4.2 against the dictionary used in the approximation.

Table 4.2 shows that as the number of atoms is increased, either by increasing the dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ or the block size N, the number of bits u_2 required for the index information to be successfully recovered increases.

4.6 The CR of the "ad hoc" Scheme

Ignoring the entropy coding stage the CR of the self contained *folded* image $\mathbf{I}^f$ can be expressed as

$$\mathrm{CR} = \frac{uN_p}{N^2 H_1 u_1 + N^2 H_2 u_2}.$$

Given that $H_1 = \lceil \frac{K}{N^2} \rceil$ and $H_2 = \lceil \frac{K+Q^N}{N^2} \rceil$, $N^2 H_1 \approx K$ and $N^2 H_2 \approx K$. Therefore with $u_1 = 12$ and u, N_p fixed,

$$\mathrm{CR} \lessapprox \frac{1}{K(12 + u_2)}. \tag{4.9}$$

This equation shows the CR will fall when the number of bits u_2 increases, unless the number of coefficients K is reduced at the same time. This is equivalent to saying that the SR needs to increase to compensate for any increase in u_2.

Table 4.2 shows that larger values of u_2 are required as the dictionary index and/or the block size N increases. Therefore, for the dictionaries tested the CR will fall, as the number of vectors in the dictionary increases unless there is a suitable increase in SR in the initial approximation.

The results of the Experiments in Chapter 2 indicated that increasing the number of vectors in a dictionary increases the resulting SR. From the above it is clear that this is not necessarily going to increase the CR of an "ad hoc" self contained *folded* image. This is also without including the entropy coding stage. Therefore an investigation is undertaken

in the next Section into the effect on the CR produced by applying the "ad hoc" *folded* method with 8 dictionaries $\mathbf{D}_i^{c2}, i = 1, \ldots, 8$ of increasing size, in combination with the 3 block sizes $N = 8, 16$ and 24.

4.6.1 Experimental Overview

The 55 astronomical and natural images were used as the sample for this Experiment, with each image being first approximated to a $\text{PSNR}^a = 45 \pm 1 \times 10^{-2}\%$. The coefficients were then *folded* to produce the vector $\mathbf{b}^{f_1}$ containing $N_b^{f_1}$ bits with $u_1 = 12$. The indices were then *folded* by the "ad hoc" method to produce the vector of bits $\mathbf{b}^{f_2,\triangle}$ containing $N_b^{f_2,\triangle}$ bits, using the values of u_2 shown in Table 4.2. The CR for the self contained *folded* image was then calculated as

$$\text{CR} = \frac{uN_p}{N_b^{f_1} + N_b^{f_2,\triangle}}. \tag{4.10}$$

The remaining experimental set up used is the same as that in the previous Experiment in Section 4.5.1.

Results

Figure 4.3 shows the average SR ($\bar{x}_{SR}$) over both the astronomical and natural image. The results are shown for for each block size $N = 8, 16$ and 24 partitioning the image, against the dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$, the atoms providing the approximation were chosen from. The $\bar{x}_{SR}$ in the Figure confirms two observations from the Experiments in Chapter 2,

1) Approximations made with the RDC dictionary are sparsest when $N = 8$ (the smallest block size).

2) The B-spline based dictionaries produce much sparser approximations of the astronomical images as the size of the block N is increased.

An additional observation from Figure 4.3 is that the $\bar{x}_{SR}$ for the astronomical images appears to increase more rapidly, than for the natural images when the dictionary index increases. The significance of this effect on the CR is now investigated.

The average CR ($\bar{x}_{CR}$) over the 55 astronomical and natural images, calculated with equation (4.10) is shown by the solid lines respectively in Figures 4.4 and 4.5. As expected the $\bar{x}_{CR}$ for $\mathbf{D}_1^{c,2}$ was highest for $N = 8$, mirroring the result for the $\bar{x}_{SR}$.

Figure 4.4 shows the $\bar{x}_{CR}$ over the astronomical images, increases with the dictionary index for the smaller dictionaries, but not for the larger ones. Therefore a one tailed paired sample t-test was performed, the results of which are shown in Appendix E.2.1. The

purpose of the test was to investigate whether, the average CR produced by *folding* images with the largest dictionary $\mathbf{D}_8^{c,2}$, was significantly higher than, the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$. The results to a 95% confidence level show when $N = 16$, the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than that produced by all of the the smaller dictionaries. When $N = 24$ the results show the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ was significantly higher than that produced by the five smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 5$.

Figure 4.5 shows that the $\bar{x}_{CR}$ over the natural images, increases with the dictionary index, but again not for every dictionary. Therefore an identical one tailed paired sample t-test to the one for the astronomical images above was performed, with results given in Appendix E.2.1. The conclusion of the test was, the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is only significantly higher than the two smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$, for a 95% confidence level.

4.6.2 Discussion

The results above show that, as expected the average CR of a self contained *folded* image $\mathbf{I}^f$ constructed with the "ad hoc" scheme to store the indices is highly dependent on the SR resulting from the initial approximation. Also as expected the SR is dependent on the suitability of the dictionary approximating the particular image corpus.

The suitability of the B-spline based dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ used in this Experiment for approximations of astronomical images can be seen in Figure 4.3 by the rapid increase in $\bar{x}_{SR}$ for larger dictionaries. This then translates into a significant increase in the average CR when larger dictionaries are used to produce the self contained *folded* image $\mathbf{I}^{f_1}$.

The same is not true for approximations made with the same B-spline dictionaries to calculate sparse approximations of natural images, shown in Figure 4.3. For this image corpus the $\bar{x}_{SR}$ did not increase as rapidly for larger dictionaries. This then translates into the average CR for the largest dictionary only being significantly higher than average CR produced by the two smallest dictionaries, for all block sizes.

4.7 The "pixel" Scheme

An alternative method for storing the vectors of indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ which is discussed below is to apply a pseudorandom permutation to the index information.

The "ad hoc" method applies a transformation to the indices and then reduces the

	Astronomical Images			Natural Images		
$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$N = 8$	$N = 16$	$N = 24$	$N = 8$	$N = 16$	$N = 24$
1	12	14	16	12	14	15
2	14	16	17	13	16	17
3	14	17	18	14	16	17
4	15	17	18	15	17	18
5	15	18	19	15	17	18
6	16	18	19	16	18	19
7	16	18	19	16	18	19
8	16	18	20	16	19	19

Table 4.2: Minimum number of bits u_2 which $\mathbf{I}^{f_2}$ can be quantized to without distorting the index information for any image in the 8 bit astronomical and natural image sets. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 8$ indicated by the index given in the first column.

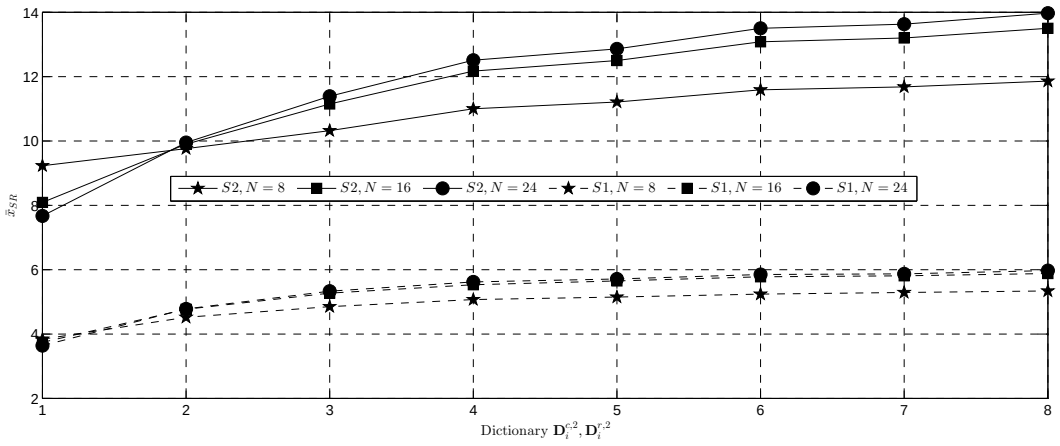

Figure 4.3: Average SR ($\bar{x}_{SR}$) over the 55, 8 bit grey level astronomical and natural image sets when atoms are picked from the dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ by the OMP2D algorithm. The $\bar{x}_{SR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines indicates the $\bar{x}_{SR}$ for the astronomical images and the dashed lines indicate the average $\bar{x}_{SR}$ for the natural images.

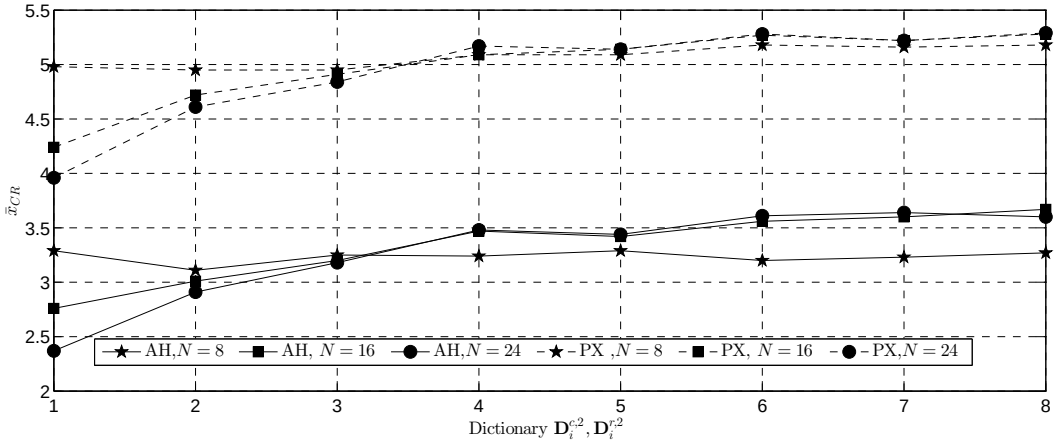

Figure 4.4: Average CR ($\bar{x}_{CR}$) over the 55, 8 bit grey level astronomical image set when applying the "ad hoc" and "pixel" *folding* procedures. The $\bar{x}_{CR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines and dashed lines indicate the $\bar{x}_{CR}$ for respectively the "ad hoc" and "pixel" *folding* procedures.

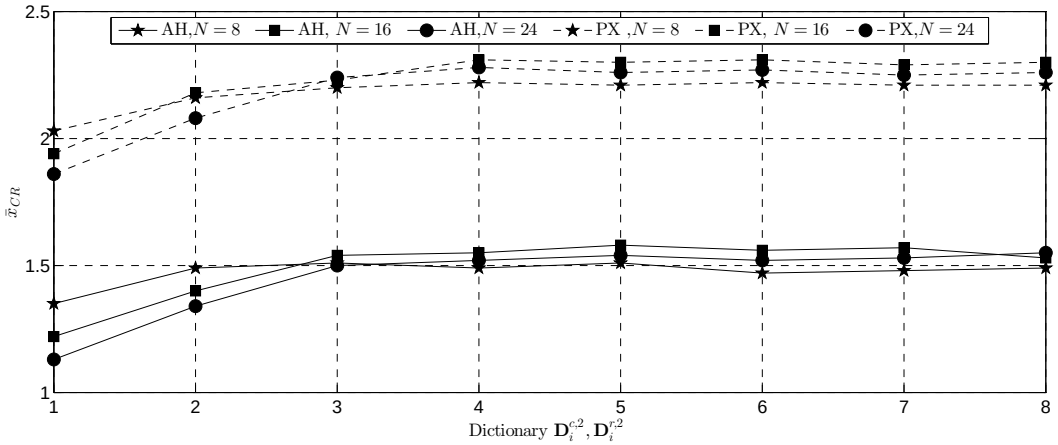

Figure 4.5: Average CR ($\bar{x}_{CR}$) over the 55, 8 bit grey level natural image set when applying the "ad hoc" and "pixel" *folding* procedures. The $\bar{x}_{CR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines and dashed lines indicate the $\bar{x}_{CR}$ for respectively the "ad hoc" and "pixel" *folding* procedures.

storage requirement by applying a simple quantization and entropy coding stage. The approach in this Section is instead to first code the indices to reduce the storage requirement of the indices and then control the access to this information.

This method can then be secured by means of a *private key* to initialize the pseudo-random permutation, preventing a third party not in possession of the *private key* from recovering the indices. The linear nature of a simple permutation without other security measures will makes this approach susceptible to plain text attacks [50], and therefore less secure than the "ad hoc" method.

4.7.1 Storing the Separable Indices (m_1)

This approach flattens the vectors of indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ required in equation (3.4), and then entropy codes the resulting vector. Before flattening the indices two modifications are made to them which were found to reduce the size of the resulting bit stream.

Complement coding

The aim is to find out if any index, which will be referred to as the complementary index i_c, has been chosen in over half the image blocks. If it has then the index i_c is removed from each vector $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ which contains it and included in each vector $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ which does not. To recover the original sets of indices the process is simply repeated with i_c.

If the index i_c is chosen in the approximation of $x\%$ of the image blocks this will reduce the number of indices by $(\frac{x}{50} - 1)Q^N$.

Index Difference

Storing the indices in independent vectors for each block means there is no restriction on the order that the indices are in, as long as the order of the corresponding coefficient vectors is altered so that the coefficients still correspond to the correct indices. Therefore the indices for each image block can be sorted into ascending numerical order without effecting the recovery. If the indices are sorted in this way then the difference between each index for each block $q = 1, \ldots, Q^N$ could be stored along with the first (smallest) index, instead of the just the indices. This is performed in 2 steps:

1) First the vectors of indices are sorted into ascending numerical order to get the vectors $\{\mathbf{l}_q^a\}_{q=1}^{Q^N}$, where

$$\mathbf{l}_q^a(k) \leq \mathbf{l}_q^a(k+1), k = 1, \ldots, K_q - 1, q = 1, \ldots, Q^N.$$

Then storing in the vectors $\mathbf{l}_q^i$, the original position of each element of $\mathbf{l}_q^a$ in $\mathbf{l}_q$, that is $\mathbf{l}_q^i$ is such that $\mathbf{l}_q^a(k) = \mathbf{l}_q(\mathbf{l}_q^i(k)), k = 1, \ldots, K_q - 1, q = 1, \ldots, Q^N$.

The coefficients which are used to create the embedded image can then be reordered by the following relation

$$\mathbf{c}_q^a(k) = \mathbf{c}_q(\mathbf{l}_q^i(k)), k = 1, \ldots, K_q, q = H_1 + 1, \ldots, Q^N,$$

so that their position in the sorted vectors $\mathbf{c}_q^a$ corresponds to the position of their index in the corresponding vectors $\mathbf{l}_q^a$, with $q = H_1 + 1, \ldots, Q^N$. These vectors can now be used in equation (3.5).

2) The difference between each index in the sorted vectors, is taken in order, and included with the first element to get the vectors $\{\mathbf{l}_q^d\}_{q=1}^{Q^N}$, where

$$\mathbf{l}_q^d(1) = \mathbf{l}_q^a(1),$$

and

$$\mathbf{l}_q^d(k) = \mathbf{l}_q^a(k+1) - \mathbf{l}_q^a(k), \quad k = 2, \ldots, K_q, q = 1, \ldots, Q^N.$$

To recover the sorted sets of indices $\{\mathbf{l}_q^a\}_{q=1}^{Q^N}$ the procedure is simply reversed by cumulatively summing each set of indices, that is

$$\mathbf{l}_q^a = \sum_{i=1}^{k} \mathbf{l}_q^d(i), \quad k = 1, \ldots, K_q, q = 1, \ldots, Q^N.$$

Storing the Indices

The vectors of indices will now be stored in a large temporary vector $\mathbf{t}$. To enable them to be assigned back from $\mathbf{t}$ to the vectors of indices for each block $\{\mathbf{l}_q^d\}_{q=1}^{Q^N}$, either the numbers $K_q, q = 1, \ldots, Q^N$ can be stored, or the indices for each block split into separate partitions. The first approach is used when creating the "ad hoc" blocks for storing the separable indices, and the second approach is used here, as described below.

First a reserved symbol r is chosen which is not a valid index, this will be used to separate the vectors of indices for each image block when they are placed into new larger vector. The symbol r is first included at the end of each vector $\{\mathbf{l}_q^d\}_{q=1}^{Q^N}$ as

$$\mathbf{l}_q^d = [\mathbf{l}_q^d; r], q = 1, \ldots, Q^N.$$

These vectors are then placed into the larger temporary vector

$$\mathbf{t} = \widehat{\mathbf{Flat}}(\{\mathbf{l}_q^d\}_{q=1}^{Q^N}),$$

which contains the separable indices for each block separated from each other by r as shown below.

$$\mathbf{t} = [\mathbf{l}_1^d(1); \ldots; \mathbf{l}_1^d(K_1); r; \mathbf{l}_2^d(1); \ldots; \mathbf{l}_2^d(K_2); r; \ldots\ldots; r; \mathbf{l}_{Q^N}^d(1); \ldots; \mathbf{l}_{Q^N}^d(K_{Q^N})]. \quad (4.11)$$

Using a suitable entropy coding algorithm the vector of indices $\mathbf{t}$ can be converted to a bit stream $\mathbf{b}^{m_1} \in \mathbb{R}^{N_b^{m_1}}$.

4.7.2 Restricting access

Access can be restricted to the indices stored in $\mathbf{b}^{m_1}$ by applying a pseudorandom permutation to the order of its elements to get a reorder vector $\bar{\mathbf{b}}^{m_1}$, which is generated in the following way.

A vector $\mathbf{m} \in \mathbb{R}^{N_b^{m_1}}$ of unique pseudorandom natural numbers is first generated by the pseudorandom number generator initialized with a seed. These vectors are then used to reorder the elements of the vectors $\mathbf{b}^{m_1}$ by the relation

$$\bar{\mathbf{b}}^{m_1}(i) = \mathbf{b}^{m_1}(\mathbf{m}(i)), i = 1, \ldots, N_b^{m_1}. \quad (4.12)$$

Taking u bits at a time from $\mathbf{b}^{m_1}$ a new vector f containing $\lceil \frac{N_b^{m_1}}{u} \rceil$ pixels is created as,

$$\mathbf{g}(n) = \widehat{\mathbf{UInt}}(\bar{\mathbf{b}}^{m_1}(i, \ldots, i+u-1)), \quad i = nu, n = 1, \ldots, \left\lceil \frac{N_b^{m_1}}{u} \right\rceil. \quad (4.13)$$

The vector of pixels $\mathbf{g}$ can now be reshaped to become the image $\mathbf{I}^{f_2} = \widehat{\mathbf{Mat}}(\mathbf{g}, N_r^3, N_c^3)$, where the choice of N_r^3 or N_c^3 can be left to the user.

The self contained *folded* image $\mathbf{I}^f$ is then the union of the two images, $\mathbf{I}^{f_1,\Delta}$ containing the vectors of coefficients $\mathbf{c}_q$, and $\mathbf{I}^{f_2}$ containing the vectors of indices $\mathbf{l}_q$ which are both required to recover image blocks $\tilde{\mathbf{I}}_q^K$ in equation (3.4), with $q = 1, \ldots, Q^N$.

As discussed in Section 3.3.2 the pseudorandom vector $\mathbf{m} \in \mathbb{R}^{N_b^{m_1}}$ should be generated with a CSPRNG.

4.7.3 Recovery procedure

The self contained *folded* image $\mathbf{I}^f$ is first split into $\mathbf{I}^{f_1,\Delta}$ and $\mathbf{I}^{f_2}$. The indices are then recovered from $\mathbf{I}^{f_2}$ by:

- Relabelling the elements of the image array $\mathbf{I}^{f_2}$ back to become the vector $\mathbf{g} = \widehat{\mathbf{Vec}}(\mathbf{I}^{f_2})$.

- Converting each unsigned integer in $\mathbf{g}$ back to its binary representation to create the bit stream $\bar{\mathbf{b}}^{m_1}$ as

$$\bar{\mathbf{b}}^{m_1}(i,\ldots,i+u-1) = \widehat{\mathbf{Bin}}(\mathbf{g},u), \quad i = nu, n = 1,\ldots,\left\lceil \frac{N_b^{m_1}}{u} \right\rceil.$$

- Reverse the permutation with the same vector of pseudorandom numbers $\mathbf{m} \in \mathbb{R}^{N_b^{m_1}}$ from equation (4.12) with the following procedure,

$$\mathbf{b}^{m_1}(\mathbf{m}(i)) = \bar{\mathbf{b}}^{m_1}(i), i = 1,\ldots,N_b^{m_1}.$$

- Apply the decoding phase of the entropy coding algorithm which created the original vector $\mathbf{b}^{m_1}$ torecover the vector of indices $\mathbf{t}$.

- The large vector $\mathbf{t}$ is then split into Q^N smaller vectors each containing the index information for a single block. This is achieved by placing the elements in between the reserved symbols r shown in equation (4.11) back into the sets $\{\mathbf{l}_q^d\}_{q=1}^{Q^N}$.

- The original sets of indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ can then be recovered from $\{\mathbf{l}_q^d\}_{q=1}^{Q^N}$, by reversing the procedure outlined in Section 4.7.1.

4.7.4 Storing the row and column indices (m_2)

An alternative to storing the vectors of separable indices $\{\mathbf{l}_q\}_{q=1}^{Q^N}$ intervening in (3.4) is to store both vectors of indices $\{\mathbf{l}_q^c\}_{q=1}^{Q^N}$ and $\{\mathbf{l}_q^r\}_{q=1}^{Q^N}$ corresponding. This can be performed in a similar way to the procedure above for the vectors of separable indices. The main difference is that now the indices come in pairs, that is each separable index $\mathbf{l}_q(k)$ now corresponds to the index pair $(\mathbf{l}_q^c(k), \mathbf{l}_q^r(k))$. This means that for each block $q = 1,\ldots,Q^N$ only one of the vectors of indices $\mathbf{l}_q^c(k), k = 1,\ldots,K_q$ or $\mathbf{l}_q^r(k), k = 1,\ldots,K_q$ can be sorted in ascending numerical order, the other vector must then be reordered so that the index pairs are still consistent. In the following discussion the convention will be to sort the elements of the vectors $\{\mathbf{l}_q^c\}_{q=1}^{Q^N}$ into ascending numerical order.

The procedure for coding the indices is given below. The access restriction procedure is exactly the same as described in Section 4.7.2 and is not repeated below.

Complement coding

The aim now is to find out the most common index pair $(\mathbf{l}_q^c(k), \mathbf{l}_q^r(k)), k = 1\ldots, K_q, q = 1,\ldots,Q^N$ defined as (i_c^c, i_c^r). If this index pair, (i_c^c, i_c^r) occurs in over half the pairs of vectors $(\mathbf{l}_q^c, \mathbf{l}_q^r), q = 1,\ldots,Q^N$ then it is removed from each pair where is occurs and included in each pair where it does not.

That is given the pair of index vectors $\mathbf{l}_q^c = [1; i_c^c; 2; i_c^r; 4; i_c^c]$ and $\mathbf{l}_q^r = [3; 1; 2; i_c^c; 7; i_c^r]$ for a block q containing the common index pair (i_c^c, i_c^r), the new vectors with the compliment removed are $\mathbf{l}_q^c = [1; i_c^c; 2; i_c^r; 4]$ and $\mathbf{l}_q^r = [3; 1; 2; i_c^c; 7]$. Then given the pair of index vectors $\mathbf{l}_q^c = [1; i_c^c; 2; i_c^r; 4]$ and $\mathbf{l}_q^r = [3; 1; 2; i_c^c; 7]$ for a block q which do not contain the common index pair (i_c^c, i_c^r), the new vectors with the compliment added are $\mathbf{l}_q^c = [1; i_c^c; 2; i_c^r; 4; i_c^c]$ and $\mathbf{l}_q^r = [3; 1; 2; i_c^c; 7; i_c^r]$.

It is clear from the above that to recover the original vectors of indices the process is simply repeated with (i_c^c, i_c^r).

Index difference

Storing the indices in independent vectors $\{\mathbf{l}_q^c\}_{q=1}^{Q^N}$ and $\{\mathbf{l}_q^r\}_{q=1}^{Q^N}$ for each block again means there is no restriction on the order that the indices are in, as long as any change to the order of the elements in $\{\mathbf{l}_q^c\}_{q=1}^{Q^N}$ is also made to the elements of $\{\mathbf{l}_q^r\}_{q=1}^{Q^N}$ and $\{\mathbf{c}_q\}_{q=1}^{Q^N}$. This order needs to be maintained to ensure that for $q = 1, \ldots, Q^N$, each coefficient $\mathbf{c}_q(k), k = 1, \ldots, K_q$ still corresponds to the same column and row vectors $\mathbf{D}^c(:, \mathbf{l}_q^c(k))$ and $\mathbf{D}^r(:, \mathbf{l}_q^r(k))$ used to generate the sparse approximation in equation (2.15). Therefore the vectors of indices $\{\mathbf{l}_q^c\}_{q=1}^{Q^N}$ can be sorted into ascending numerical order without effecting the recovery. If the indices are sorted in this way the difference between each index in each vector $\{\mathbf{l}_q^c\}_{q=1}^{Q^N}$ could be stored along with the first (smallest) index, instead of the just the indices. This is performed in 2 steps:

1) First the vectors of indices are sorted into ascending numerical order to get the vectors $\{\mathbf{l}_q^{c,a}\}_{q=1}^{Q^N}$, where

$$\mathbf{l}_q^{c,a}(k) \leq \mathbf{l}_q^{c,a}(k+1), k = 1, \ldots, K_q - 1, q = 1, \ldots, Q^N.$$

Then storing in the vectors $\mathbf{l}_q^i$, the original position of each element of $\mathbf{l}_q^{c,a}$ in $\mathbf{l}_q^c$, with $q = 1, \ldots, Q^N$, that is $\mathbf{l}_q^i$ is such that $\mathbf{l}_q^{c,a}(k) = \mathbf{l}_q^c(\mathbf{l}_q^i(k))$.

The vectors of indices $\{\mathbf{l}_q^r\}_{q=1}^{Q^N}$ and coefficients $\{\mathbf{c}_q\}_{q=1}^{Q^N}$, which create the embedded image, can then be reordered by the following

$$\mathbf{l}_q^{r,a}(k) = \mathbf{l}_q^r(\mathbf{l}_q^i(k)), k = 1, \ldots, K_q, q = 1, \ldots, Q^N,$$

and

$$\mathbf{c}_q^a(k) = \mathbf{c}_q(\mathbf{l}_q^i(k)), k = 1, \ldots, K_q, q = H_1 + 1, \ldots, Q^N.$$

The vectors of coefficients can now be used in equation (3.5).

2) The difference between each index in the sorted vectors $\{\mathbf{l}_q^{c,a}\}_{q=1}^{Q^N}$, is taken in order, and included with the first element to get the vectors $\{\mathbf{l}_q^{c,d}\}_{q=1}^{Q^N}$, where

$$\mathbf{l}_q^{c,d}(1) = \mathbf{l}_q^{c,a}(1),$$

and

$$\mathbf{l}_q^{c,d}(k) = \mathbf{l}_q^{c,a}(k+1) - \mathbf{l}_q^{c,a}(k), \quad k = 2, \ldots, K_q, q = 1, \ldots, Q^N.$$

To recover the sorted sets of indices $\{\mathbf{l}_q^{c,a}\}_{q=1}^{Q^N}$ the procedure is simply reversed by cumulatively summing each set of indices, that is

$$\mathbf{l}_q^{c,a} = \sum_{i=1}^{k} \mathbf{l}_q^{c,d}(k), \quad k = 1, \ldots, K_q, q = 1, \ldots, Q^N.$$

Storing the Indices

The same procedure which was applied to the vectors in Section 4.7.1 to produce the temporary vector $\mathbf{t}$, is applied here. The new temporary vector contains each $\{\mathbf{l}_q^{c,d}\}_{q=1}^{Q^N}$, separated by a reserved symbol r. This vector is then enlarged to include the sets of vectors $\{\mathbf{l}_q^r\}_{q=1}^{Q^N}$ as

$$\mathbf{t} = [\mathbf{t}; \widehat{\mathbf{Flat}}(\{\mathbf{l}_q^r\}_{q=1}^{Q^N})].$$

Through a suitable entropy coding algorithm the vector of indices $\mathbf{t}$ can be converted to a bit stream $\mathbf{b}^{m_2} \in \mathbb{R}^{N_b^{m_2}}$.

The same procedure as in Section 4.7.2 is then applied to secure the information in $\mathbf{b}^{m_2}$ and produce the image $\mathbf{I}^{f_2}$. The index information can be recovered from $\mathbf{I}^{f_2}$ by simply reversing the procedure, applying the method for the separable indices described in Section 4.7.3.

4.7.5 Comparison of index storage methods

To examine which index storage method (m_1 or m_2) requires the smallest of bit stream the following Experiment was realized. The entropy coding in this Experiment was again performed with the same adaptive arithmetic coding algorithm in Section 2.7.

Given that the range of possible indices is dependent on the both the number of atoms in the dictionary and the size N of the square blocks, three block sizes $N = 8, 16, 24$ were tested together with the 8 dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$, of increasing size described in Section 4.8.1.

For 55 astronomical and natural images and each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ the images were processed in blocks of $N = 8, 16, 24$ and approximated to a PSNR^a of $45 \pm$

4.5×10^{-3}dB. The indices were then coded first using m_1 and then m_2 and the size of the respective bit streams $N_b^{m_1}$ and $N_b^{m_2}$ recorded.

A paired sample t-test was realized on the results of the 55 sample images to a 95% confidence level. The aim of the test was to investigate whether the index coding methods produce bit streams of the same size. The results for each dictionary and block size $N = 8, 16, 24$ are shown in Appendix E.2.

The results for astronomical images showed that when $N = 8$ and 16, the bit stream produced by m_2 was significantly smaller than that produced by m_1, for the dictionaries, $\mathbf{D}_i^{c,2}, i = 5, \ldots, 8$. When $N = 24$, the bit stream produced by m_2 was significantly smaller than that produced by m_1, for the dictionaries, $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$.

The results for the natural images were similar. For $N = 8$ and 16, the bit stream produced by m_2 was significantly smaller than that produced by m_1, for the dictionaries, $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$. When $N = 24$, the bit stream produced by m_2 was significantly smaller than that produced by m_1, for dictionaries, $\mathbf{D}_i^{c,2}, i = 3, \ldots, 8$.

Both image corpus produce the same conclusion, that is, m_2 produces a significantly smaller bit stream when the number of vectors in the dictionary used increases.

In the next Experiments involving the "pixel" method, the approach used for coding the indices will be to used the results of this Experiment to determine when to use m_1 and m_2. That is when the pixel method is used, m_1 will code the indices in all Experiments for both astronomical and natural images except for experiments involving:

- astronomical images approximated by dictionaries $\mathbf{D}_i^{c,2}, i = 5, \ldots, 8$ for $N = 8, 16$, and $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$ for $N = 24$, or experiments involving
- natural images approximated by dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for $N = 8, 16$, and $\mathbf{D}_i^{c,2}, i = 3, \ldots, 8$ for $N = 24$,

when m_2 will be used.

The self contained *folded* image resulting from applying this will represented with $N_b^{f_2}$ bits and denoted as $\mathbf{I}^{f_2} \in \mathbb{R}^{N_r^3 \times N_c^3}$.

4.8 The CR of the "pixel" scheme

In this Section the CR resulting of the self contained *folded* images, $\mathbf{I}^f$ constructed by the "pixel" scheme is established.

Although the "pixel" scheme stores the vectors of indices with a different procedure to the "ad hoc" method, the increase in the range of indices resulting with larger dictionaries

or blocks is still expected to increase the number of bits $N_b^{f_2}$ required to store the indices.

The Experiment below calculates the size of the CR of the self contained *folded* image $\mathbf{I}^f$ when the "pixel" scheme is used to *fold* the indices from approximations of both the astronomical and natural images, made with the 8 dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ in combination with the 3 block sizes $N = 8, 16$ and 24. This CR is then compared with that produces when the "ad hoc" method was used over the same set images, dictionaries and block sizes N in Experiment 4.6.

The "pixel" scheme directly codes the indices instead of first transforming and then quantizing them. Therefore it is expected that the number of bits $N_b^{f_2}$ required to store the image $\mathbf{I}^{f_2} \in \mathbb{R}^{N_r^3 \times N_c^3}$ produced by the "pixel" scheme will be less than the number of bits $N_b^{f_2,\Delta}$ required to store $\mathbf{I}^{f_2,\Delta} \in \mathbb{R}^{N_r^2 \times N_c^2}$ produced by the "ad hoc" method. This will result in a larger overall CR for the self contained *folded* image built by the "pixel" scheme than the self contained *folded* image made with the "ad hod" method, which is investigated in the Experiment below.

4.8.1 Experimental Overview

The Experiment in this Section applies the same procedure as the Experiment in Section 4.6 except that instead of applying the "ad hoc" method for *folding* the indices in the image $\mathbf{I}^{f_1}$ the indices are stored by the "pixel" scheme, to produce the image $\mathbf{I}^{f_2}$. Therefore the CR for the self contained *folded* image constructed by the "pixel" scheme to store the indices is calculated as $\text{CR} = \frac{uN_p}{N_b^{f_1}+N_b^{f_2}}$.

The pseudorandom number genereator required by this method was initialized by the seed $s_3 = 5.8453 \times 10^7$.

Results

The average CR ($\bar{x}_{CR}$) over the 55 astronomical and natural images, when processed to produce the self contained *folded* image with the "pixel" method to store the indices, is shown by the solid lines respectively in Figures 4.4 and 4.5. These results for the CR can be summarized as follows:

1) The $\bar{x}_{CR}$ shown in Figures 4.4 and 4.5 for the self contained *folded* images within the "pixel" scheme is significantly higher than the $\bar{x}_{CR}$ for the self contained *folded* images with the "ad hoc" method for all combinations of dictionary and block size N.

2) Figure 4.4 shows that the $\bar{x}_{CR}$, over the astronomical images, for all N increase with

the dictionary size. Therefore a one tailed paired sample t-test was performed to see if the $\bar{x}_{CR}$ produced by *folding* with the largest dictionary $\mathbf{D}_8^{c,2}$ was significantly higher than the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$. The results of this test are shown in Appendix E.2.3, where it is shown that the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the average CR produced by all of the the smaller dictionaries with $N = 16$ and significantly higher than the average CR produced by all of the dictionaries except $\mathbf{D}_6^{c,2}$ when $N = 8$ and 24, for a 95% confidence level.

3) Figure 4.5 shows that the $\bar{x}_{CR}$, over the natural images again increases with the dictionary size. Therefore a one tailed paired sample t-test was performed to see if the $\bar{x}_{CR}$ produced by *folding* with the largest dictionary $\mathbf{D}_8^{c,2}$ was significantly higher than the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$. The results of this test are shown in Appendix E.2.3 where it is shown that the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the two smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$ when $N = 8$, and significantly higher than the three smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ when $N = 16$ and 24, for a 95% confidence level.

The general results above concerning the "pixel" method is used are identical to the results of the "ad hoc" method is used, with the difference being that the "pixel" method produced a higher $\bar{x}_{CR}$ over both image sets for all dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ and block sizes $N = 8, 16$ and 24.

Discussion

Self contained *folded* images were constructed by both the "ad hoc" and "pixel" methods, resulting in a CR > 1 for all the images tested. The highest $\bar{x}_{CR}$ occured for images which applied the "pixel" method for storing the indices.

To get an idea of whether through the process of *folding*, the storage requirements of the image coefficients and indices are increased, over simply coding them, the results for $N = 24$ from this Experiment were compared with those from the Experiment in Section 2.8. It is important to note that the results compared in the discussion below were produced by two different dictionaries, however as the comparisson is just to get and indication, the difference in the $\bar{x}_{SR}$ can be ignored.

Both experiments were performed over the 45 astronomical and natural images which were initially approximated to a $\text{PSNR}^a = 45 + 4.5 \times 10^{-3}$. The $\bar{x}_{CR}$ (and $\bar{x}_{SR}$) results for the astronomical and natural images, constructed with the "ad hoc" method were

respectively 8.15 ($\bar{x}_{SR} = 15.41$) and 4.10 ($\bar{x}_{SR} = 6.32$). The corresponding $\bar{x}_{CR}$ (and $\bar{x}_{SR}$) for the "pixel" method, were respectively, 5.15 ($\bar{x}_{SR} = 13.58$) and 2.31 ($\bar{x}_{SR} = 6.10$). Ignoring the reduction in the $\bar{x}_{SR}$ in the initial approximations, resulting from approximating with the RDC-RDBS$_2$ instead of the TS$_3$ dictionary, it is clear that the *folding* procedure introduces some storage overhead.

4.9 RGB Colour Images

4.9.1 Single Channel Method

Colour images are often given as three intensity planes, representing each of the three primary colours Red (R), Green (G) and Blue (B). Each colour plane is a grey level intensity array of pixels which can be approximated by the selection criteria in equation (2.4) without any modification.

This results in the sparse approximation of RGB image $\mathbf{I}_z \in \mathbb{R}^{N_r \times N_c}, z = 1, 2, 3$ processed using square blocks containing N row and column pixels

$$\mathbf{I}_{z,q}^{K} = \sum_{k=1}^{K_{z,q}} \mathbf{c}_{z,q}(k)\mathbf{S}_{\mathbf{l}_{z,q}(k)}, q = 1, \ldots, Q^N, z = 1, \ldots, 3. \tag{4.14}$$

The self contained *folded* image is then constructed out of the coefficients and indices of this approximation according to the steps below.

1) Apply the *folding* procedure from Chapter 3 to each set of coefficients $\{\mathbf{c}_{z,q}, q = 1, \ldots, Q^N\}_{z=1}^{3}$ from equation (4.14) to produce a *folded* image $\mathbf{I}_z^{f_1}, z = 1, \ldots, 3$. This image can be secured by either the SVD or random security procedure also discussed in Chapter 3.

2) Each *folded* image $\mathbf{I}_z^{f_1}, z = 1, \ldots, 3$ is then quantized and coded with the procedure given in Section 4.2 to produce the *folded* images $\mathbf{I}_z^{f_1,\triangle}$. Each z component containing the coefficients from the sparse approximation (4.14) of $\mathbf{I}_z$, with $z = 1, \ldots, 3$.

3) Each set of indices $\{\mathbf{l}_{z,q}, q = 1, \ldots, Q^N\}_{z=1}^{3}$ from equation (4.14) is then stored by either the "ad hoc" or "pixel" methods described receptively in Sections 4.4 and 4.7 to produce the *folded* images $\mathbf{I}_z^{f_2}$, each containing the indices from the sparse approximation (4.14) of $\mathbf{I}_z$, with $z = 1, \ldots, 3$.

4) Each plane of the self contained *folded* image $\mathbf{I}_z^{f}$ is then formed as the union of the two images $\mathbf{I}_z^{f_1,\triangle}$ and $\mathbf{I}_z^{f_2}$ with $z = 1, \ldots, 3$.

4.9.2 Multi Channel Method

Given and image $\mathbf{I}_z \in \mathbb{R}^{N_r \times N_c}, z = 1, 2, 3$ the Single Channel method mentioned above works by independently approximating each of the 3 colour planes. This means that each of the vectors of indices $\mathbf{l}_{z,q}, q = 1, \ldots, Q^N, z = 1, \ldots, 3$ can be different.

The assumption behind the Multi Channel method is, there is strong correlation between the Red Green and Blue channels of a digital image. If this assumption is true then, approximating each colour plane with the same set of atoms, should still result in a sparse representation. In addition unlike the single channel method only one set of indices $\{\mathbf{l}_q, q = 1, \ldots, Q^N\}$ needs to be stored in the self contained *folded* image. Although this method is applied to RGB images where each array of pixels represents a single colour, it will also be applicable to other types of colour image providing correlation between the arrays is preserved.

The approximation is achieved by modifying the selection criteria shown in equation (2.4) to become

$$\mathbf{l}(k+1) = \underset{m=1,\ldots,M_r M_c}{\operatorname{argmax}} \sum_{z=1}^{3} |\langle \mathbf{S}_m, \mathbf{R}_z^K \rangle_F| \quad \text{with,} \quad \mathbf{R}_z^K = \mathbf{I}_z - \sum_{k=1}^{K} \mathbf{c}_z(k) \mathbf{S}_{\mathbf{l}(k)}. \tag{4.15}$$

The resulting multi channel sparse approximation of RGB image $\mathbf{I}_z^K \in \mathbb{R}^{N \times N}, z = 1, 2, 3$ processed in square blocks containing N row and column pixels is then given as

$$\mathbf{I}_{z,q}^K = \sum_{k=1}^{K_q} \mathbf{c}_{z,q}(k) \mathbf{S}_{\mathbf{l}_q(k)}, q = 1, \ldots, Q^N, z = 1, \ldots, 3. \tag{4.16}$$

The self contained *folded* image is then constructed in a similar way to the single channel method, except that only one set of indices $\{\mathbf{l}_q, q = 1, \ldots, Q^N\}$ is stored by either the "ad hoc" or "pixel" method through the steps described below.

1) Apply the *folding* procedure from Chapter 3 to each set of coefficients $\{\mathbf{c}_{z,q}, q = 1, \ldots, Q^N\}_{z=1}^3$ from equation (4.16) to produce a *folded* image $\mathbf{I}_z^{f_1}, z = 1, \ldots, 3$. This image can be secured by either the SVD or random security procedure also discussed in Chapter 3.

2) Each *folded* image $\mathbf{I}_z^{f_1}, z = 1, \ldots, 3$ is then quantized and coded applying the procedure given in Section 4.2 to produce the *folded* images $\mathbf{I}_z^{f_1,\Delta}$, each containing the coefficients from the sparse approximation (4.14) of $\mathbf{I}_z$, with $z = 1, \ldots, 3$.

3) The set of indices $\{\mathbf{l}_q, q = 1, \ldots, Q^N\}$ from equation (4.16) are then stored using either the "ad hoc" or "pixel" methods described receptively in Sections 4.4 and 4.7 to produce the *folded* image $\mathbf{I}^{f_2}$.

4) The self contained *folded* image is then formed by the union of the two images $\mathbf{I}_z^{f_1,\triangle}, z = 1, \ldots, 3$ and $\mathbf{I}^{f_2}$.

Remark 6. The self contained *folded* image $\mathbf{I}_z^f$ resulting from the multi channel method can be smaller than that for the single channel method but this will depend on how the SR of the approximation is affected by the restriction of using only one set of indices for all the colour planes.

4.9.3 Examining the CR

In this Section the CR resulting from the RGB *folding* approaches, is examined. To this end an Experiment was performed over two sets of 55 true colour images. These were the original RGB versions of the astronomical and natural images from Chapters 3 and 4.

Experimental Set Up

The astronomical and natural images were first partitioned into blocks and approximated to a PSNR^a of $45 \pm 1 \times 10^{-2}\%$.

The coefficients were secured by the random method given in Section 3.3.2 with two seeds, $s_1 = 1.43398 \times 10^6$ and $s_2 = 2.365658978 \times 10^9$, which initialize the pseudorandom number generators. The *folded* images containing these coefficients $\mathbf{I}_z^{f_1}, z = 1, \ldots, 3$, were then quantized with $u_1 = 12$ bits. This ensured the error $\overline{\delta\text{PSNR}}$, between the approximated and the recovered images was acceptable, that is a $\overline{\delta\text{PSNR}} \leq 1 \times 10^{-1}$.

The indices were secured using the "pixel" method given in Section 4.7. The pseudorandom number genereator required by this method was initialized by the seed $s_3 = 5.8453 \times 10^7$.

The *private key* for this Experiment was then the combination of these 3 seeds, with the same *key* being used at both the *folding* and recovery stages.

This procedure was performed over the test images for three block sizes, $N = 8, 16$ and 24, and 8 dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$.

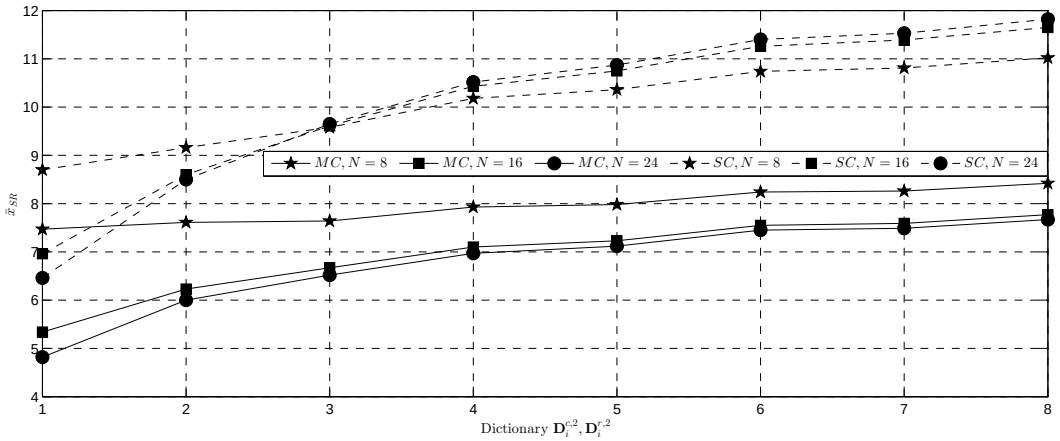

Figure 4.6: Average SR ($\bar{x}_{SR}$) over the 55, RGB astronomical images when atoms are picked from the dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ using the single and multi channel methods. The $\bar{x}_{SR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines indicates the $\bar{x}_{SR}$ for the single channel method and the dashed lines indicate the average $\bar{x}_{SR}$ for the multi channel method.

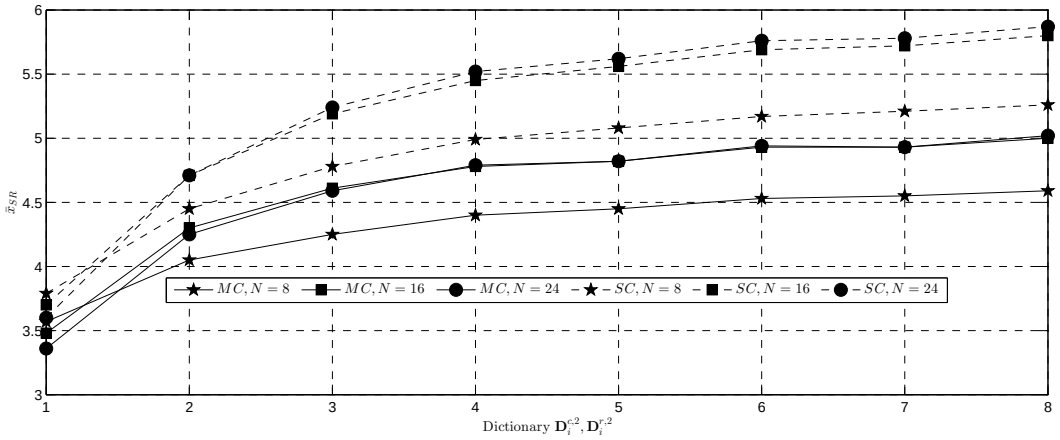

Figure 4.7: Average SR ($\bar{x}_{SR}$) over the 55, RGB natural images when atoms are picked from the dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ using the single and multi channel methods. The $\bar{x}_{SR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines indicates the $\bar{x}_{SR}$ for the single channel method and the dashed lines indicate the average $\bar{x}_{SR}$ for the multi channel method.

4.9.4 Results

Average SR, $\bar{x}_{SR}$

Figures 4.6 and 4.7 show the $\bar{x}_{SR}$ for respectively the astronomical and natural image sets. Each Figure shows the $\bar{x}_{SR}$ resulting from the single and multi channel approximations when the images were processed in blocks $N = 8, 16$ and 24, against the dictionary $\mathbf{D}_i^{c,2}, \imath = 1, \ldots, 8$, used to make the approximation. Three general observations can be made regarding the $\bar{x}_{SR}$:

1) The same general results regarding the $\bar{x}_{SR}$ for the grey level astronomical and natural images are seen for the RGB versions in Figures 4.6 and 4.7. That is the $\bar{x}_{SR}$ is greater for the astronomical images shown in Figure 4.6 than the natural images shown in Figure 4.7, and the increase in $\bar{x}_{SR}$ resulting from using larger dictionaries is greater for the astronomical images than the natural ones.

2) The $\bar{x}_{SR}$ for each block size $N = 8, 16$ and 24 for both the astronomical and natural image sets is always higher for the single channel approximation.

3) For the astronomical images the $\bar{x}_{SR}$ from multi channel approximation is highest for the smallest block size $N = 8$. This is the opposite of the result for the single channel approximation for dictionaries $\mathbf{D}_i^{c,2}, i = 3, \ldots, 8$ where the largest $\bar{x}_{SR}$ is produced when images are processed in the partition involving the largest block size $N = 24$.

Average CR, $\bar{x}_{CR}$

Figures 4.8 and 4.9 show the $\bar{x}_{CR}$ for respectively the astronomical and natural image sets. Each Figure shows the $\bar{x}_{CR}$ for the single and multi channel methods against the dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$, for the 3 block sizes $N = 8, 16$ and 24. The CR results are given below:

1) Figures 4.8 and 4.9 show that for all dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$, and block sizes $N = 8, 16$ and 24, the $\bar{x}_{CR} > 1$ over both image sets, for both the single and multi channel methods.

2) When either the single or multi channel method is used the $\bar{x}_{CR}$ over the astronomical images is higher than the $\bar{x}_{CR}$ over the natural images. This is due to the $\bar{x}_{SR}$ for the astronomical images being higher than the $\bar{x}_{SR}$ for the natural images and is identical to the result for the grey level versions used in Experiment 4.8.

3) The multi channel method always produced a higher $\bar{x}_{CR}$ over the astronomical and natrual image sets, for each block size tested. This is a consequence of the $\bar{x}_{SR}$ from the initial approximation. The $\bar{x}_{SR}$ for the multi channel approximation is close to that of the single channel approximation. Therefore the inclusion of only a single set of indices $\{\mathbf{l}_q, q = 1, \ldots, Q^N\}$ in self contained *folded* image, increases the $\bar{x}_{CR}$ over that of the single channel method.

4) The $\bar{x}_{CR}$ over the astronomical images when a multi channel approximation is performed, is higher for the smallest block size $N = 8$. This is because the $\bar{x}_{SR}$ is higher for $N = 8$, than for $N = 16$ and $N = 24$, in the results for the multi channel approximation.

5) The $\bar{x}_{CR}$ over the RGB astronomical images using the multi channel method with $N = 8$ was higher for every dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ than the $\bar{x}_{CR}$ over the grey level versions of the same astronomical images, shown in Figure 4.8 by the thicker dashed line. The $\bar{x}_{CR}$ over the RGB natural images using the multi channel method was higher for every dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ than the $\bar{x}_{CR}$ and block size $N = 8, 16$ and 24 than the $\bar{x}_{CR}$ over the grey level versions of the same natural images, shown in Figure 4.9 by the thicker dashed line.

6) To establish if the $\bar{x}_{CR}$ over the astronomical images for the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than for the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ a one tailed paired sample t-test was performed. The results of this test are shown in Appendix E.3.1 for the single and multi channel methods.

   The result for the single channel method with $N = 8$ was: the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the average CR produced by using all of the the smaller dictionaries except for $\mathbf{D}_6^{c,2}$, for a 95% confidence level. For blocks with $N = 16$ and 24 the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ was significantly higher than the average CR produced by all of the the smaller dictionaries, for a 95% confidence level.

   The result for the multi channel method was: for all $N = 8, 16$ and 24, the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the average CR produced by using all of the the smaller dictionaries for a 95% confidence level.

7) To establish if the $\bar{x}_{CR}$ over the natural images for the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than for the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ a one tailed paired sample t-test was performed. The results of this test are shown in Appendix

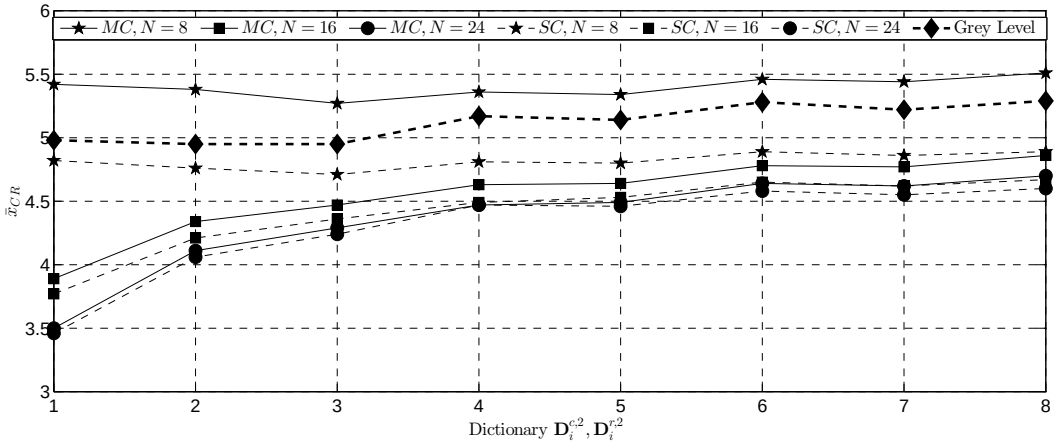

Figure 4.8: Average CR ($\bar{x}_{CR}$) over the 55, RGB astronomical images when atoms are picked from the dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ using the single and multi channel methods. The $\bar{x}_{CR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines indicates the $\bar{x}_{CR}$ for the single channel method and the dashed lines indicate the average $\bar{x}_{CR}$ for the multi channel method. The thick dashed line is included to indicate the highest $\bar{x}_{CR}$ result for each dictionary over the grey level versions of the astronomical images.

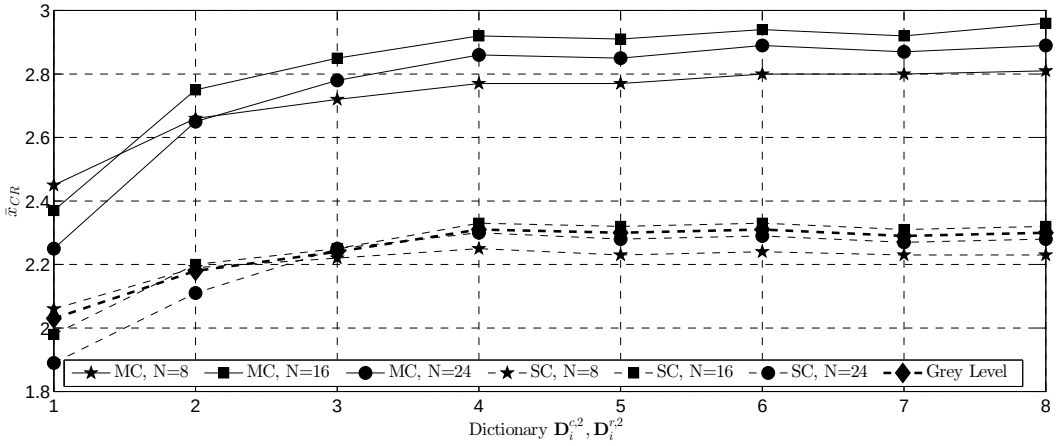

Figure 4.9: Average CR ($\bar{x}_{CR}$) over the 55, RGB natural images when atoms are picked from the dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ using the single and multi channel methods. The $\bar{x}_{CR}$ is shown against the index i of each dictionary $\mathbf{D}_i^{c,2}, i = 1, \ldots, 8$ for each block size $N = 8, 16, 24$. The solid lines indicates the $\bar{x}_{CR}$ for the single channel method and the dashed lines indicate the average $\bar{x}_{CR}$ for the multi channel method. The thick dashed line is included to indicate the highest $\bar{x}_{CR}$ result for each dictionary over the grey level versions of the natural images.

E.3.1 for the single and multi channel methods.

The result for the single channel method with $N = 8$ was: the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the average CR produced by using the two smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$, for a 95% confidence level. For blocks with $N = 16$ and 24 the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ was significantly higher than the average CR produced by the three smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ and dictionary $\mathbf{D}_7^{c,2}$, for a 95% confidence level.

The result for the multi channel method with $N = 8$ and 16 was: the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the average CR produced by using all of the the smaller dictionaries, for a 95% confidence level. For $N = 24$ was, the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the average CR produced by using all of the the smaller dictionaries except for $\mathbf{D}_6^{c,2}$, for a 95% confidence level.

Discussion

Both of the proposed methods produced compressed self contained *folded* versions of all the RGB test images. That is all *folded* versions of the astronomical and natural images, produced by either the single and multi channel methods resulted in $\bar{x}_{CR} > 1$.

The $\bar{x}_{SR}$ over both sets of images when using the single channel method is always greater than the multi channel because only one set of indices is used in the approximation of the Red, Green and Blue colour plane. Conversely the $\bar{x}_{CR}$ for the multi channel method is always higher than the single channel method because only one set of indices needs to be stored in the self contained *folded* image $\mathbf{I}^f$.

The $\bar{x}_{CR}$ was maximal for the astronomical images when processed using the smallest block size, $N = 8$. This is the opposite of the result for the grey level images where the $\bar{x}_{CR}$ was greatest for the largest block size $N = 24$. The advantage of using this block size, demonstrated in the Experiments in Chapter 2, is that it significantly reduces the approximation processing time.

The multi channel method benefits from using larger dictionaries to a greater extent than the single channel method. This is demonstrated by the significant increase in average CR over both image sets, resulting from using the largest dictionary instead of the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 5$, when using the multi channel method.

Finally the multi channel method allowed RGB images to be compressed to a higher $\bar{x}_{CR}$ than their grey level versions.

4.9.5 Folding Example

In this Section an example is given demonstrating the application of the single and multi channel *folding* methods respectively from Sections 4.9.1 and 4.9.2.

Both *folding* procedures were applied to the "NGC 2440" image shown in the top centre of Figure 4.10 from the Hubble Top 100 Images [83]. The image is composed of a Red, Green and Blue channel with each channel composed of 1280×1280, 8 bit pixels.

The experimental set up used in the example is exactly the same as that used in the previous two Experiments. Setting $N = 8$ each plane of the image was initially split into square 8×8 blocks before being approximated to a $\text{PSNR}^a = 45 \pm 1 \times 10^{-2}\%$ by atoms picked from dictionary $\mathbf{D}_8^{c,2}$.

The *private key* for this example was composed of the three seeds $s_1 = 1.43398 \times 10^6$ $s_2 = 2.365658978 \times 10^9$ and $s_3 = 5.8453 \times 10^7$ considered in the Experiments in Section 4.9.3. Each example demonstrates the recovery procedure using the correct *private key*, and an incorrect *private key* with the seeds $s_1 = 1.43398 \times 10^6 + 1$ and $s_2 = 2.365658978 \times 10^9 + 1$.

Single Channel

The single channel method was applied to create the $241 \times 1280 \times 3$ self contained *folded* image $\mathbf{I}^f$ displayed in the centre of the third row of Figure 4.10.

The recovery was then performed, by the same *private key* as in the *folding* stage to correctly recover the image. Due to the quantization of $\mathbf{I}_z^{f_1}, z = 1, 2, 3$ with $u_1 = 12$ bits there was a difference between the PSNR of the approximated image $\mathbf{I}^K$ and the recovered image $\tilde{\mathbf{I}}^K$ of $7.49 \times 10^{-2}\%$dB.

The recovery was then performed, using the incorrect *private key* which resulted in the recovered image $\tilde{\mathbf{I}}^K$ shown at the bottom of Figure 4.10 which is completely different to the original image shown at the top of Figure 4.10.

Multi Channel

The multi channel method was applied to create the $196 \times 1296 \times 3$ self contained *folded* image displayed in the centre of the third row of Figure 4.11.

The recovery was then performed, by the same *private key* as in the *folding* stage to correctly recover the image. Again due to the quantization of $\mathbf{I}_z^{f_1}, z = 1, 2, 3$ with $u_1 = 12$ bits there was a difference between the PSNR of the approximated image $\mathbf{I}^K$ and the recovered image $\tilde{\mathbf{I}}^K$ of $6 \times 10^{-2}\%$dB.

The recovery was then performed, with the incorrect *private key* which resulted in the recovered image $\tilde{\mathbf{I}}^K$ shown at the bottom of Figure 4.11 which is completely different to the original image shown at the top of Figure 4.11.

Discussion

This example demonstrates visually that the multi channel method can result in a *folded* image requiring less storage than the single channel method. This size difference can be observed in the third row of Figure 4.11, where the black border surrounding the multi channel self contained *folded* image represents the size of the single channel self contained *folded* image.

Both the single and multi channel method approximations result in images which appear identical to each other when the approximation is up to the same PSNR. This can be seen by comparing the top image in Figure 4.10 with the top image from Figure 4.11.

The single and multi channel methods successfully *fold* and recover the images with a negligible loss in PSNR of respectively 7.49×10^{-2}%dB and 6×10^{-2}%dB

4.10 Conclusions

Two methods have been proposed for storing the indices resulting from sparse approximations of an image. The first, referred to as the "ad hoc" method *folds* the vectors of indices with the procedure outlined in Chapter 3. This approach can therefore be secured with a *private key*, and either the SVD or random method, proposed in that Chapter. Thus preventing a third party not in possession of the *private key* from recovering them.

An alternative, termed the "pixel" method applies the *private key* to initialize a simple pseudorandom permutation of the indices. This prevents a third party not in possession of the *private key*, from recovering them.

Both methods result in compressed versions of the original images, however it is clear that the *folding* stage introduces some storage overhead into the compressed representation.

Experiments 4.5 and 4.8 calculate the average CR ($\bar{x}_{CR}$), for both variants of the self contained *folding* procedure. The results of these Experiments show the $\bar{x}_{CR}$ when the "pixel" scheme is applied, is significantly higher than the $\bar{x}_{CR}$, for images *folded* with the "ad hoc" method.

The result for both the "ad hoc" and "pixel" methods indicate, larger dictionaries only offer an improvement in the *folded* CR, if the type of dictionary is suitable for making

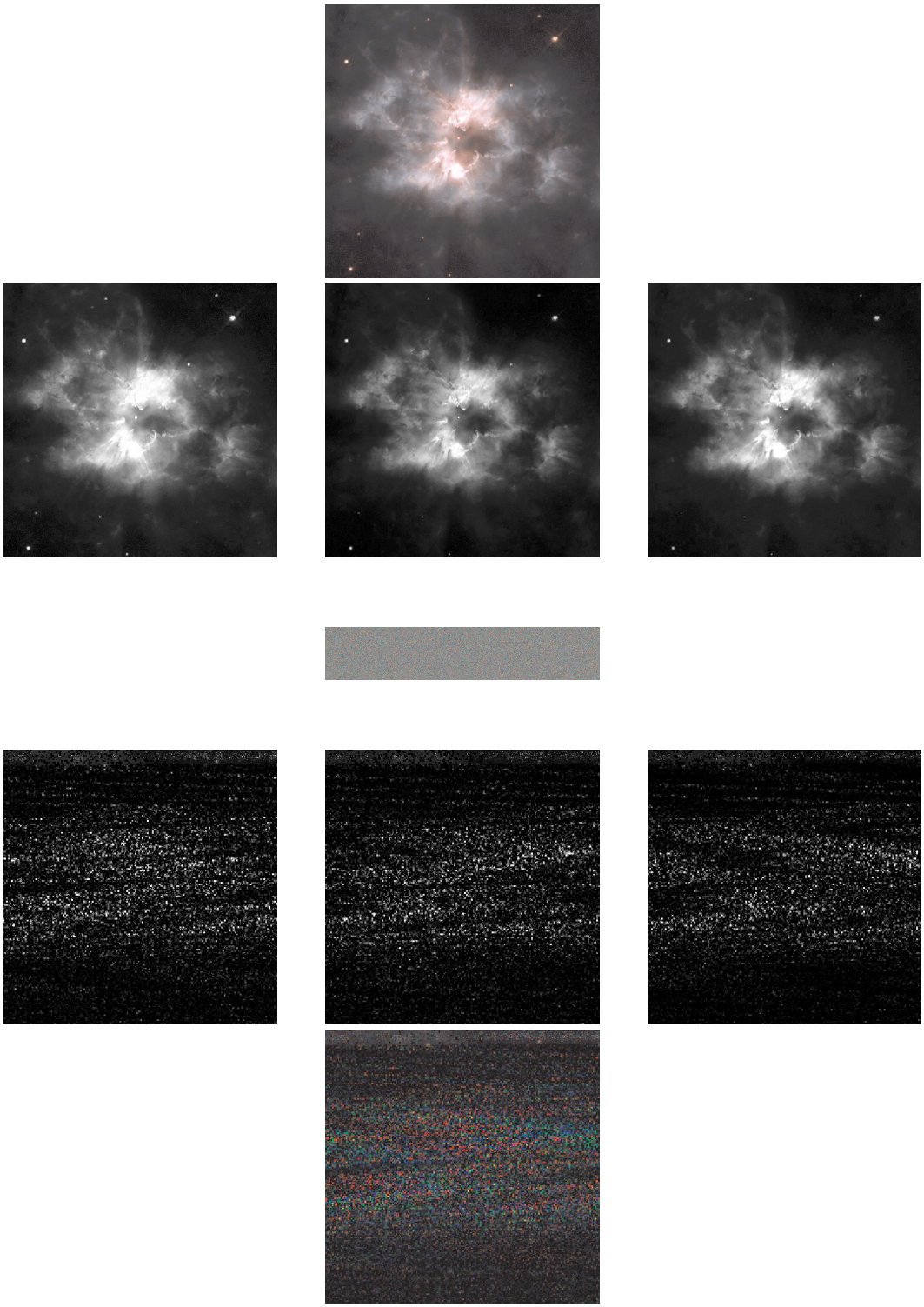

Figure 4.10: The image in the centre of the first row is the approximation $\mathbf{I}_z^K, z = 1, 2, 3$ to 45.0039dB of the original RGB image of "NGC 2440" using a single channel approximation. The three images from left to right on the second row show respectively the Red, Green and Blue colour planes of this approximation. The small image in the centre of the third row is the single channel self contained *folded* version of the top image. The three images from left to right on the fourth row show respectively the Red, Green and Blue colour planes of an attempt at recovery from the *folded* image using a *private key* different by 1 to the *private key* used at the *folding* stage. The image in the centre of the bottom row is the recovered image RGB image using this incorrect *private key*.

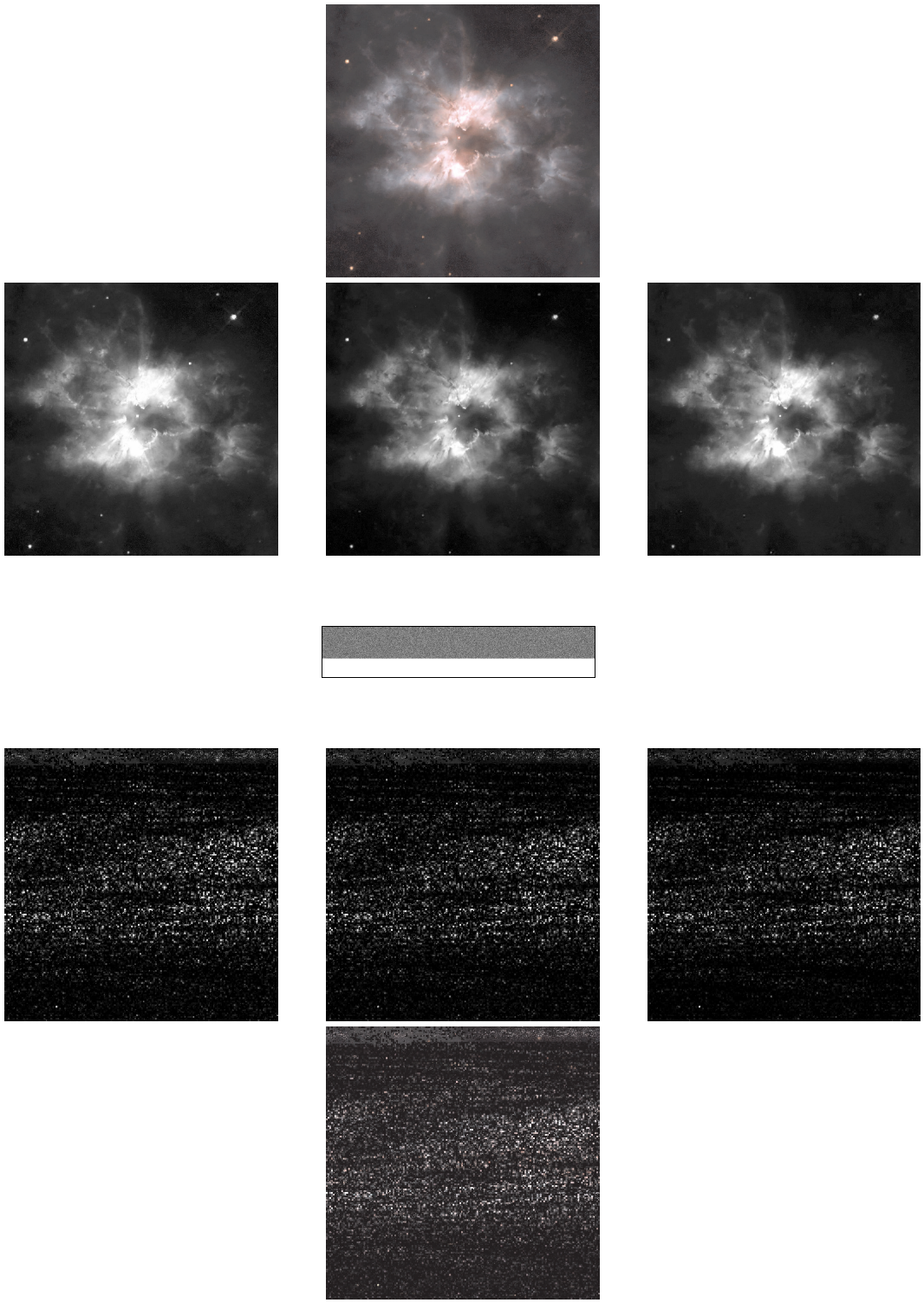

Figure 4.11: The image in the centre of the first row is the approximation $\mathbf{I}_z^K, z = 1, 2, 3$ to 45.0021dB of the original RGB image of "NGC 2440" using a multi channel approximation. The three images from left to right on the second row show respectively the Red, Green and Blue colour planes of this approximation. The small image in the centre of the third row is the multi channel self contained *folded* version of the top image. The black frame around this image is set to the size of the single channel self contained *folded* version of the top image to demonstrate the reduction in size resulting from the multi channel method. The three images from left to right on the fourth row show respectively the Red, Green and Blue colour planes of an attempt at recovery from the *folded* image using a *private key* different by 1 to the *private key* used at the *folding* stage. The image in the centre of the bottom row is the recovered image RGB image using this incorrect *private key*.

sparse approximations, of the images being *folded*. If the dictionary is not suitable then as the dictionary size increases the CR will fall.

Two methods have been proposed, and shown experimentally to produce self contained *folded* RGB images with a CR greater than 1. The first a single channel method, simply repeats the procedure proposed at the beginning of this Chapter, on each colour plane of an RGB image. The second a multi channel method, operates by applying a modified selection criteria, indicated in equation (4.15), during the approximation process. This modification produces only one set of atom indices $\{\mathbf{l}_q, q = 1, \ldots, Q^N\}$, instead of the three $\{\mathbf{l}_{z,q}, q = 1, \ldots, Q^N\}_{z=1}^{3}$, produced by the single channel approximation.

The multi channel modification was shown experimentally to reduce the average SR from that of the single channel approximation. However the multi channel self contained *folded* image, only requires one third of the indices stored by the single channel method. Therefore with recourse to two test sets, each containing 55 images, the multi channel method produced *folded* version of every image with a larger CR, than that produced by the single channel method. Additionally this allowed the RGB images to be *folded* to a higher average CR, than their corresponding grey level versions.

The multi channel method produced a significantly higher average SR with the largest dictionary. This was over both the image corpus tested, implying the possibility that even larger dictionaries will continue to increase the CR.

The security of both the proposed methods can be secured with *private key* to prevent access to the;

- coefficients from the initial approximation. Applying either the SVD or random methods proposed in Chapter 3, and
- the indices. Securing them with either the "ad hoc" or "pixel" scheme..

Finally an example was given applying both the single and multi channel approaches to the same image. The results in Figure 4.11 show visually how much smaller the multi channel self contained *folded* image is, than the corresponding single channel version.

5 Conclusions and Directions for Future Work

Sparse signal representation can be realized by many techniques. Generally the specific application or type of solution, dictates the approach which is undertaken. In this work the specific application was to performing sparse approximations of images. That is finding a method to attack the minimization problem shown in equation (1.12), repeated again below,

$$\min \|\mathbf{c}\|_0 \quad \text{subject to} \quad \|\mathbf{f} - \mathbf{D}\mathbf{c}\| \leq \rho.$$

In this work the combinatorial problem above was addressed by greedy pursuit strategies. An approach whose suitability was ratified, by producing approximations sparse enough to rival current image compression standards.

The investigation into sparse image approximations began in Chapter 2. The initial investigation concentrated on sparse approximations of astronomical grey level images, processed in square blocks, of $N \times N$ pixels. These approximations were made by the greedy algorithms, MP2D, OMP2D and OOMP. Of these algorithms OMP2D was established as the most suitable, both in terms of sparsity and processing time. This identifies OMP2D as an appropriate algorithm for making sparse approximations of astronomical images, but more importantly it implies that it can be successfully applied to other classes of images

as well.

When the size N, of the blocks partitioning the images, was increased, OMP2D produced even sparser representations. The only issue was the resulting processing time, which for this algorithm, became prohibitive for larger blocks, especially when $N = 32$. Therefore the choice of block size should be determined from the users priorities. A simple rule of thumb which can be applied, is to process images in blocks with $N = 32$ if sparsity is more important, if not use blocks with $N = 8$, because this smaller size still produces very sparse representations.

The processing time for larger blocks was reduced, with the application of the newly proposed SPMP2D$_1$ algorithm. This algorithm is theoretically equivalent to OMP2D, however it has a smaller memory footprint. It was shown experimentally, for $N = 32$, to reduce the processing time over OMP2D, whilst maintaining an equivalent level of sparsity. This result is encouraging and suggests that when $N = 8, 16$ and 24 images should be approximated with OMP2D, and for $N = 32$, SPMP2D$_1$ should be chosen.

The level of sparsity was highly dependant on the dictionary, or transform applied to the image blocks. In the experiments in Chapter 2 images were approximated by both the DCT and CDF9/7 wavelet transform as well as several dictionaries including, trained, and, B-spline and wavelet based dictionaries. Of these, the trained dictionaries produced the sparsest approximations, of the original test images. Therefore the prescription for generating sparse image approximations should be, to process images in blocks with $N = 32$, choosing atoms from trained dictionaries, with SPMP2D$_1$.

Applying the above prescription for producing sparse image approximations, made it possible to implement a successful dictionary coding scheme. Success being a measure of the schemes ability to produce compressed images, requiring less storage than those produced by JPEG2000. The result is encouraging and implies that sparse approximations should be considered as an important first step in a future image compression algorithm.

Chapter 3 proposed an image security approach, with possible applications in an online image distribution service. It operates by securing some of the sparse coefficients produced by OMP2D, in a procedure termed image *folding*. Both the SVD and random methods were proposed for securing these coefficients. The SVD method relies on instability in the calculation of singular vectors, corresponding to multiple zero singular values, and the random method relies on a random transformation. Of the two the random security scheme offered a larger *keyspace*, and therefore greater resistance to brute force attacks. This method was therefore applied in the remaining chapter, for securing the hidden coefficients.

The *folded* image produced in Chapter 3 did not contain all the information required to recover the original sparse approximation of an image. Therefore in Chapter 4 the additional information was included and secured, by two alternative methods, producing a so called, self contained *folded* image. Of the two security methods tested, the "pixel" scheme was found to produce the smallest compressed images. Because of the promising results this procedure was then extended from grey level to RGB true colour images.

Two alternative approaches were enlisted to approximate RGB colour images, made with OMP2D. The first known as the single channel method simply repeated the procedure for a grey level images on the, Red, Green, and Blue, planes of a colour image. The second, multi channel method, instead operated on all three colour planes at once. Of the two, the multi channel method was shown to produce the smallest compressed images, which were, in all cases smaller than their compressed grey level counterparts. This method was also shown to benefit, in terms of resulting CR from using the largest dictionary. This suggests that the CR for the multi channel method may increase further, when approximations are made with even larger dictionaries.

The resulting suggested path for producing self contained *folded* images, is therefore to first produce the sparsest approximation possible, then to apply the "pixel" scheme as described in Chapter 4. In addition if the original images are RGB colour images then the multi channel procedure should be applied increase the CR of the *folded* image.

5.1 Future Directions

The results and experience gained through the research undertaken, suggest the following directions for future research:

- Applying MP implementations to SPMP

  The SPMP algorithm proposed operates on image blocks, recalculating all the inner products at every iteration, thus making it impractical to apply to whole images at once. Implementations of the MP algorithm with small support atoms, operate over the whole image at once. This is achieved by, at each iteration, only recalculating the inner products for regions of an image, which have been effected by the atom chosen. Essentially the SPMP algorithm is two separate rounds of MP. When applied to image blocks the largest an atom in a block can be is N, these atoms therefore have small support as $N \leq 32$. It is therefore possible to apply the SPMP algorithm in the same way as MP, to process an entire image at once, instead of in blocks. This would allow the sparse approximation performance of the OMP algorithm to

be investigated, both with, and without blocking.

- Process images in blocks larger than 32×32

  In Chapter 2 the sparsity of the image approximations increased up until the largest block size N tested. The largest block size tested was $N = 32$, because blocks containing 32×32 pixels required a dramatic increase in processing time over blocks of 24×24, when processed with OMP2D. The newly proposed SPMP2D$_1$ algorithm was found to significantly reduce the processing time for larger blocks. Therefore images could to be processed with $N > 32$ to investigate the effect this has on the sparsity of the resulting approximations.

- Image coding

  The proposed dictionary coding method produced promising compression results, however it is not at an advance stage, suggesting that improvements can be made. Therefore additional research should be undertaken into the possible ways of increasing the compression performance.

- Increasing the number of perturbations applied in the SVD method

  The size of the *keyspace* for the SVD method increases dramatically with the number of perturbations. Experimentally only two perturbations were applied, therefore the possibility of increasing the number of perturbations without effecting the quality of the recovered images should be investigated.

- Larger dictionaries

  The CR produced by the multi channel method on RGB images was highest, for each N, for the largest dictionary. Therefore larger dictionary constructions should be investigated to see if the CR can be increased further.

- Wavelet Domain

  In both [48] and [86], images were first transformed into the CDF9/7 wavelet domain, before performing sparse approximations. The authors of both papers found that the results produced by this approach were superior, in terms of sparsity, to the same approximations, when carried out in the pixel domain. This initial first step should be applied within the framework of the experiments in Chapter 2, to investigate whether both the SR and CR reported their can be increased.

Bibliography

[1] I. Daubechies. *Ten Lectures on Wavelets*. Society for Industrial and Applied Mathematics, Philadelphia, 1992.

[2] L. Rebollo-neira and J. Bowley. Sparse representation of astronomical images. *Journal of the Optical Society of America A*, 30(3):758–768, 2013.

[3] J. Bowley and L. Rebollo-Neira. Sparsity and something else: an approach to encrypted image folding. *IEEE Signal Processing Letters*, 18(3):189–192, March 2011.

[4] L. Rebollo-Neira, J. Bowley, A. G. Constantinides, and A. Plastino. Self-contained encrypted image folding. *Physica A: Statistical Mechanics and its Applications*, 391(23):5858–5870, December 2012.

[5] M. Petrou and P. Bosdogianni. *Image Processing The Fundamentals*. John Wiley & Sons, Chichester, 2000.

[6] N. Efford. *Digital Image Processing a practical introduction using Java*. Pearson Education, Harlow, 2000.

[7] W. Burger and M. J. Burge. *Digital Image Processing*. Springer, New York, 2008.

[8] R. F. Graf. *Modern Dictionary of Electronics*. Butterworth-Heineman, Massachusetts, seventh edition, 1999.

[9] R. C. Gonzalez and R. E. Woods. *Digital Image Processing*. Pearson Education, New Jersey, third edition, 2008.

[10] A. K. Jain. *Fundamentals of Digital Image Processing*. Prentice-Hall, New Jersey, 1989.

[11] K. N. Plataniotis and A. N. Venetsanopoulos. *Colour Image Processing and Applications*. Springer-Verlang, Berlin, 2000.

[12] T. Acharya and A. K. Ray. *Image Processing*. John Wiley & Sons, New Jersey, 2005.

[13] R. C. Gonzalez, R. E. Woods, and S. L. Eddids. *Digital Image Processing using MATLAB*. Pearson Education, New Jersey, 2004.

[14] D. Salomon. *Data Compression The Complete Reference*. Springer-Verlang, London, fourth edition, 2007.

[15] Y. Q. Shi and H. Sun. *Image and Video Compression for Multimedia Engineering*. Taylor & Francis Group, Boca Raton, 2008.

[16] A. Baskurt. JPEG standard. In A. Naït-Ali and C. Cavaro-Ménard, editors, *Compression of Biomedical Images and Signals*, chapter 2, pages 22–26. ISTE, London, 2008.

[17] D. S. Taubman and M. W. Marcellin. *JPEG2000 Image Compression Fundamentals, Standards and Practice*. Kluwer Academic Publishers, Norwell, 2002.

[18] N. Ahmed, T. Natarajan, and K. R. Rao. Discrete Cosine Transform. *IEEE Transactions on Computers*, 23(1):90–93, 1974.

[19] Y. Zeng, G. Bi, and A. R. Leyman. New algorithm for r-dimensional DCT-II. *IEE Procedings Vision, Image & Signal Processing*, 148(1):1–8, 2001.

[20] T. J. Van Fleet. *Discrete Wavelet Transformations*. John Wiley & Sons, New Jersey, 2008.

[21] H.R. Rabiee, R.L. Kashyap, and S.R. Safavian. Adaptive image representation with segmented orthogonal matching pursuit. In *Proceedings 1998 International Conference on Image Processing. ICIP98*, volume 2, pages 233–236. IEEE Comput. Soc, 1998.

[22] Felipe Marti-Lopez and Thomas Koenig. Approximating method of frames. *Digital Signal Processing*, 13(3):519–529, July 2003.

[23] S. S. Chen, D. L. Donoho, and M. A. Saunders. Atomic Decomposition by Basis Pursuit. *SIAM Review*, 43(1):129–159, 2001.

[24] D. L. Donoho and M. Elad. Optimally sparse representation in general (nonorthogonal) dictionaries via l1 minimization. In *Proceedings of the National Academy of Sciences of the United States of America*, volume 100, pages 2197–202, March 2003.

[25] J. Portilla and L. Mancera. L0-based sparse approximation: Two alternative methods and some applications. In D. Van De Ville, V. K. Goyal, and M. Papadakis, editors, *SPIE*, volume 6701, page 67011Z, San Diago, September 2007.

[26] I.F. Gorodnitsky and B.D. Rao. A new iterative weighted norm minimization algorithm and its applications. In *[1992] IEEE Sixth SP Workshop on Statistical Signal and Array Processing*, pages 412–415, 1992.

[27] P. Rodriguez and B. Wohlberg. A Comparisson of the Computational Performance of Iteratively Reweighted Least Squares and Alternating Minimization Algorithms for l1 Inverse Problems. In *19th IEEE International Conference on Image Processing*, pages 3069–3072, Orlando, 2012.

[28] D. Angelosante and G. B. Giannakis. RLS-Weighted LASSO For Adaptive Estimation of Sparse Signals. In *IEEE International Conference on Acoustics, Speech and Signal Processing, 2009*, volume 1, pages 3245–3248, Taipei, 2009.

[29] J. A. Tropp. Greed is Good : Algorithmic Results for Sparse Approximation. *IEEE Transactions on Information Theory*, 50(10):2231–2242, 2004.

[30] S. G. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, December 1993.

[31] Y. Pati and R. Rezaiifar. Orthogonal Matching Pursuit : Recursive Function Approximat ion with Applications to Wavelet Decomposition. In *Asilomar Conference on Signals Systems and Computers*, pages 40–44, 1993.

[32] L. Rebollo-Neira and D. Lowe. Optimized orthogonal matching pursuit approach. *IEEE Signal Processing Letters*, 9(4):137–140, April 2002.

[33] M. Andrle, L. Rebollo-Neira, and E. Sagianos. Backward-Optimized Orthogonal Matching Pursuit Approach. *IEEE Signal Processing Letters*, 11(9):705–708, September 2004.

[34] M. Andrle and L. Rebollo-Neira. A swapping-based refinement of orthogonal matching pursuit strategies. *Signal Processing*, 86(3):480–495, March 2006.

[35] S. Chen, S. A. Billings, and W. Luo. Orthogonal least squares methods and their application to non-linear system identification. *International Journal of Control*, 50(5):1873–1896, November 1989.

[36] K. Skretting. *Sparse Signal Representation using Overlapping Frames*. PhD thesis, 2002.

[37] M. Elad. *Sparse and Redundant Representations*. Springer Science, New York, 2010.

[38] W. Dong, G. Shi, and X. Li. Image deblurring with low-rank approximation structured sparse representation. In *Signal & Information Processing ...*, pages 1–5, Hollywood, CA, 2012.

[39] S. K. Sahoo and W. Lu. Image inpainting using sparse approximation with adaptive window selection. In *IEEE 7th International Symposium on Intelligent Signal Processing*, pages 1–4, Floriana, September 2011. Ieee.

[40] M. Thilagavathi and P. Deepa. An efficient dictionary learning algorithm for 3d Medical Image Denoising based on Sadct. In *International Conference on Information Communication and Embedded Systems*, pages 442–447, Chennai, 2013.

[41] S. Wang, Z. Shu, L. Zhang, G. Liu, and L. Gan. Iterative Image Coding with Overcomplete Curvelet Transform. In *Congress on Image and Signal Processing*, pages 666–670. Ieee, 2008.

[42] O. Bryt and M. Elad. Compression of facial images using the K-SVD algorithm. *Journal of Visual Communication and Image Representation*, 19(4):270–282, May 2008.

[43] M. Aharon, M. Elad, and A. Bruckstein. K -SVD : An Algorithm for Designing Overcomplete Dictionaries for Sparse Representation. *IEEE Transactions on Signal Processing*, 54(11):4311–4322, 2006.

[44] O. Bryt and M. Elad. Improving the k-svd facial image compression using a linear deblocking method. In *IEEE 25th Convention of Electrical and Electronics Engineers in Israel*, pages 533–537, Eilat, March 2008. Ieee.

[45] Q. Kun and X. Quan. Low Bit Rate Compression of Facial Images Based on Adaptive Over-Complete Sparse Representation. In *Congress on Image and Signal Processing, 2009.*, number 2008, pages 1–3, 2009.

[46] O. G. Sezer, O. Harmanci, and O. G. Guleryuz. Sparse orthonormal transforms for image compression. In *IEEE International Conference on Image Processing*, pages 149–152, 2008.

[47] A. Said and W. A. Pearlman. A New, Fast, and Efficient Image Codec Based on Set Partitioning in Hierarchical Trees. *IEEE Transactions on Circuits and Systems for Video Technology*, 6(3):243–250, June 1996.

[48] K. Skretting and K. Engan. Image Compression Using Learned Dictionaries by RLS-DLA and compared with K-SVD. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, number 2, pages 1517 – 1520, Prague, 2011.

[49] O. Goldreich. *Foundations of Cryptography A Primer*, volume 1. now Publishers Inc., Hanover, 2005.

[50] B. Furht, D. Socek, and A. M. Eskicioglu. Fundamentals of Multimedia Encryption Techniques. In B. Furht and D. Kirovski, editors, *Multimedia Security Handbook*, chapter 3, pages 95–132. CRC Press, Boca Raton, 2005.

[51] Q. Liu and J. Ying. Grayscale image digital watermarking technology based on wavelet analysis. In *IEEE Symposium on Electrical & Electronics Engineering*, pages 618–621, June 2012.

[52] I. E. Ziedan, M. M. Fouad, and D. H. Salem. Application of Data Encryption Standard to Bitmap and JPEG Images. In *Twentieth National Radio Science Conference*, volume 5, pages 1–8, 2003.

[53] A. J. Menezes, P. C. V. Oorschot, and S. A. Vanstone. *Applied Cryptography*. CRC Press, Boca Raton, 1996.

[54] J. Katz and Y. Lindell. *Introduction to Modern Cryptography*. Taylor & Francis Group, Boca Raton, 2008.

[55] H. Delfs and H. Knebl. *Introduction to Cryptography*. Springer-Verlang, Berlin, second edition, 2007.

[56] C. Wang and S. Ju. Book Cipher with Infinite Key Space. In *2008 International Symposium on Information Science and Engineering*, number 2, pages 456–459. Ieee, December 2008.

[57] A. Uhl and A. Pommer. *Image and Video Encryption from Digital Rights Management to Secured Personal Communication*. Springer Science, New York, 2005.

[58] T. Kunkelmann and R. Reinema. A scalable security architecture for multimedia communication standards. In *Proceedings of IEEE International Conference on Multimedia Computing and Systems*, pages 660–661. IEEE Comput. Soc, 1997.

[59] M . Van Droogenbroeck and R. Benedett. Techniques for a Selective Encryption of Uncompressed and Compressed Images. In *Proceedings of Advanced Concepts for Intelligent Vision Systems (ACIVS)*, number September, pages 90–97, 2002.

[60] Wenjun Zeng and Shawmin Lei. Efficient frequency domain video scrambling for content access control. In *Proceedings of the seventh ACM international conference on Multimedia (Part 1)*, MULTIMEDIA '99, pages 285–294, New York, NY, USA, 1999. ACM.

[61] W. Zeng and S. Lei. Efficient Frequency Domain Selective Scrambling of Digital Video. *IEEE Transactions on Multimedia*, 5(1):118–129, 2003.

[62] M. Podesser, H. Schmidt, and A. Uhl. Selective Bitplane Encryption for Secure Transmission of Image Data in Mobile Environments. In *5th Nordic Signal Processing Symposium*, page 1037, 2002.

[63] H. Cheng and X. Li. Partial Encryption of Compressed Images and Videos. *IEEE Transactions on Signal Processing*, 48(8):2439–2451, 2000.

[64] S. Dey, S. A. Sriam, S. B. Subin, and P. K. A. Asis. SD-IES : An Advanced Image Encryption Standard Application of Different Cryptographic Modules in a New Image Encryption System. In *2013 7th International Conference on Intelligent Systems and Control (ISCO)*, pages 0–4, 2012.

[65] C. Li, Y. Chen, T. Chang, L. Deng, and K. To. Period Extension and Randomness Enhancement using High-Throughput Reseeding-Mixing PRNG. *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, 20(2):385–389, 2012.

[66] Y. Zhang, W. Liu, S. Cao, Z. Zhai, X. Nie, and W. Dai. Digital Image Encryption Algorithm Based on Chaos and Improved DES. In *IEEE International Conference on Systems, Man, and Cybernetics*, pages 474–479, 2009.

[67] Z. Zhe, Y. Haibing, Z. Yu, P. Wenjie, and Z. Yunpeng. A Block Encryption Scheme Based on 3D Chaotic Arnold Maps. In *International Asia Symposium on Intelligent Interaction and Affective Computing*, pages 15–20. Ieee, December 2009.

[68] Z. Yu, Z. Zhe, Y. Haibing, P. Wenjie, and Z. Yunpeng. A chaos-based image encryption algorithm using wavelet transform. In *2nd International Conference on Advanced Computer Control*, pages 217–222, 2010.

[69] D. Engel, T. Stutz, and A. Uhl. Efficient transparent JPEG2000 encryption with format-compliant header protection. In *IEEE International Conference on Signal Processing and Communications*, number November, pages 24–27, 2007.

[70] M. Pazarci and V. Dipçin. A MPEG2-transparent scrambling technique. *IEEE Transactions on Consumer Electronics*, 48(2):345–355, 2002.

[71] C. Wang, H. Yu, and M. Zheng. A DCT-based MPEG-2 transparent scrambling algorithm. *IEEE Transactions on Consumer Electronics*, 49(4):1208–1213, November 2003.

[72] Mathworks. MATLAB R2012a. `http://www.mathworks.co.uk/products/matlab/`, 2012.

[73] M. Matsumoto and T. Nishimura. Mersenne twister: a 623-dimensionally equidistributed uniform pseudo-random number generator. *ACM Transactions on Modeling and Computer Simulation*, 8(1):3–30, January 1998.

[74] A. Said. Introduction to Arithmetic Coding - Theory and Practice. Technical report, Palo Alto, CA, 2004.

[75] A. Said. Fast Arithmetic Coding (FastAC) Implementations. `http://www.nonlinear-approx.info/soft_other/FastAC.zip`, 2004.

[76] L. Jones, K. On a conjecture of Huber concerning the convergence of projection pursuit regression. *The Annals of Statistics*, 15(2):880–882, 1987.

[77] R. Young. *An Introduction to Nonharmonic Fourier Series*. Academic Press, New York, 1980.

[78] D. S. Bernstein. *Matrix Mathematics*. Princeton University Press, New Jersey, 2005.

[79] M. Andrle and L. Rebollo-Neira. Cardinal B-spline dictionaries on a compact interval. *Applied and Computational Harmonic Analysis*, 18(3):336–346, May 2005.

[80] L. Rebollo-Neira and Z. Xu. Sparse signal representation by adaptive non-uniform B-spline dictionaries on a compact interval. *Signal Processing*, 90(7):2308–2313, July 2010.

[81] K. Engan, K. Skretting, and J. H. Husøy. Family of iterative LS-based dictionary learning algorithms, ILS-DLA, for sparse signal representation. *Digital Signal Processing*, 17(1):32–49, January 2007.

[82] K. Skretting. Dictionary Learning Tools for Matlab. `http://www.ux.uis.no/~karlsk/dle/`, 2013.

[83] European Space Agency. Hubble Top 100 Images. `http://www.spacetelescope.org/images/archive/top100/`.

[84] J. Bowley and L. Rebollo-Neira. Highly Nonlinear Approximations for Sparse Signal Representation. `http://www.nonlinear-approx.info/`, 2013.

[85] D. Martin, C. Fowlkes, D. Tal, and J. Malik. Berkeley Segmentation Dataset and Benchmark. `http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/`, 2007.

[86] R. Maciol, Y. Yuan, and I. T. Nabney. Colour image coding with matching pursuit in the spatio-frequency domain. *Lecture Notes in Computer Science*, 6978(1):306–317, 2011.

[87] J.R. Miotke and L. Rebollo-Neira. Oversampling of Fourier coefficients for hiding messages. *Applied and Computational Harmonic Analysis*, 16(3):203–207, May 2004.

[88] L. Rebollo-Neira and A. Plastino. Statistical distribution, host for encrypted information. *Physica A: Statistical Mechanics and its Applications*, 359:213–221, January 2006.

[89] G. E. Forsythe, M. A. Malcolm, and C. E. Moler. *Computer Methods for Mathematical Computations*. Prentice-Hall, Englewood Cliffs N.J, 1977.

[90] W. H. Press, S. A. Teukolsky, W.T. Vetterling, and B. P. Flannery. Numerical Recipes: The Art of Scientific Computing. In *Preliminaries*, chapter 1, page 10. Cambridge University Press, New York, third edition, 2007.

[91] A. Mousa and A. Hamad. Evaluation of the RC4 algorithm for data encryption. *International Journal of Computer Science and Applications*, 3(2):44–56, 2006.

[92] IEEE Std 754-2008 (Revision of IEEE Std 754-1985), IEEE Standard for Floating-Point Arithmetic. Technical report, IEEE Computer Society, 2008.

[93] A. Konheim. Stream Ciphers. In *Computer Security and Cryptography*, chapter 8, pages 244–282. Wiley-Interscience, 2007.

[94] R. Bhatia. *Matrix Analysis*. Springer-Verlang, Pennsylvania, 1997.

[95] D. S. Watkins. *Fundamentals of Matrix Computations*. John Wiley & Sons, New York, second edition, 2002.

[96] C. De Boor. *A Practical Guide to Splines: Applied Mathematical Sciences 27.* Springer-Verlang, New York, 1978.

Appendices

A Matrix and Vector Operations

A.1 Defining and Accessing Elements

A column vector $\mathbf{v} \in \mathbb{R}^N$ will always be represented using lower case bold font, the notation for accessing each element of of $\mathbf{v}$ is $\mathbf{v}(n), n = 1, \ldots, N_r$. A row vector will be represented as the transpose of this, that is $\mathbf{v}^T$ where the superscript indicates the transpose operation, is a row vector containing the same elements as $\mathbf{v}$.

Accessing a range of elements from index i, inclusive to index n of a vector $\mathbf{v} \in \mathbb{R}^{N_r}$ will be indicate by the expression $\mathbf{v}(i : n)$.

The notation for constructing a column vector $\mathbf{v} \in \mathbb{R}^{N_r}$ from the elements $x_n \in \mathbb{R}, n = 1, \ldots, N_r$ is

$$\mathbf{v} = [x_1; x_2; \ldots; x_{N_r}]$$

Alternatively a column vector $\mathbf{v} \in \mathbb{R}^{N_r}$ can be constructed from two or more other column vectors $\mathbf{u}_i \in \mathbb{R}^{N_{r,i}}, i = 1, \ldots, l$, with $N_r = \sum_{i=1}^{l} N_{r,i}$ as,

$$\mathbf{v} = [\mathbf{u}_1; \mathbf{u}_2; \ldots; \mathbf{u}_l].$$

A matrix $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ will always be represented using upper case bold font. Each

element in a matrix $\mathbf{M}$ can be accessed using the elements row n_r, and column n_c, indices, as $\mathbf{M}(n_r, n_c), n_r = 1, \ldots, N_r, n_c = 1, \ldots, N_c$.

The transpose of the matrix $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ is denoted using the superscript T, as $\mathbf{M}^T$ where

$$\mathbf{M}^T(n_c, n_r) = \mathbf{M}(n_r, n_c), n_r = 1, \ldots, N_r, n_c = 1, \ldots, N_c.$$

Each row or column of the matrix $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ is accessed respectively using the following notation $\mathbf{M}(n_r, :), n_r = 1, \ldots, N_r$ or $\mathbf{M}(:, n_c), n_c = 1, \ldots, N_c$.

A new matrix $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ can be constructed from two or more other matrices $\mathbf{N}_i \in \mathbb{R}^{N_r \times N_{c,i}}, i = 1, \ldots, l$, with $N_c = \sum_{i=1}^{l} N_{c,i}$ as,

$$\mathbf{M} = [\mathbf{N}_1, \mathbf{N}_2, \ldots, \mathbf{N}_l]. \tag{A.1}$$

If one or more of the matrices $\mathbf{N}_i \in \mathbb{R}^{N_r \times N_{c,i}}, i = 1, \ldots, l$ above is a set of $N_{c,i}$ column vectors $\{\mathbf{u}_n\}_{n=1}^{N_{c,i}}$ the construction is equivalent. That is if $\mathbf{N}_2 = \{\mathbf{u}_n\}_{n=1}^{N_{c,2}}$ then $\mathbf{M}$ below

$$\mathbf{M} = [\mathbf{N}_1, \{\mathbf{u}_n\}_{n=1}^{N_{c,2}}, \ldots, \mathbf{N}_l],$$

is equivalent to that in equation (A.1).

A.2 Reshaping

A.2.1 From $\mathbf{v} \in \mathbb{R}^{N_r N_c}$ to $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$

The elements of a vector $\mathbf{v} \in \mathbb{R}^{N_r N_c}$ can be relabelled using column major order to become the elements of a matrix $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$, this procedure will be indicated using the $\widehat{\mathbf{Mat}}(\cdot)$ operation as

$$\mathbf{M} = \widehat{\mathbf{Mat}}(\mathbf{v}, N_r, N_c).$$

The convention used for determining the pair of indices in $\mathbf{M}(n_r, n_c), n_r = 1, \ldots, N_r, n_c = 1, \ldots, N_c$ from the linear index in $\mathbf{v}(n), n = 1, \ldots, N_r N_c$ is shown below

$$n_r = n - (n_c - 1)N_r, \quad n_c = \left\lceil \frac{n}{N_r} \right\rceil, \quad n = 1, \ldots, N_r N_c. \tag{A.2}$$

The resulting matrix is then

$$\mathbf{M} = \widehat{\mathbf{Mat}}(\mathbf{v}) = \begin{bmatrix} \mathbf{v}(1) & \mathbf{v}(N_r + 1) & \cdots & \mathbf{v}(N_r(N_c - 1) + 1) \\ \mathbf{v}(2) & \mathbf{v}(N_r + 2) & \cdots & \mathbf{v}(N_r(N_c - 1) + 2) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}(N_r) & \mathbf{v}(2N_r) & \cdots & \mathbf{v}(N_r N_c) \end{bmatrix}.$$

A.2.2 From $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ to $\mathbf{v} \in \mathbb{R}^{N_r N_c}$

The $\widehat{\mathbf{Mat}}(\cdot)$ operation can be reversed to relabel all the elements of $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ to become the elements of a vector $\mathbf{v} \in \mathbb{R}^{N_r N_c}$. This will be indicated by the $\widehat{\mathbf{Vec}}(\cdot)$ operation as

$$\mathbf{v} = \widehat{\mathbf{Vec}}(\mathbf{M}).$$

The convention for determining the linear index in $\mathbf{v}(n), n = 1, \ldots, N_r M_c$ from the index pair used in $\mathbf{M}(n_r, n_c), n_r = 1, \ldots, N_r, n_c = 1, \ldots, N_c$ is

$$n = (n_c - 1)N_r + n_r, \quad n_r = 1, \ldots, N_r, n_c = 1, \ldots, N_c. \tag{A.3}$$

The resulting vector in then,

$$\begin{aligned}\mathbf{v} =&[\mathbf{M}(1,1); \mathbf{M}(2,1); \ldots; \mathbf{M}(N_r,1); \mathbf{M}(1,2); \mathbf{M}(2,2); \ldots; \mathbf{M}(N_r,2); \ldots\ldots; \\ &\mathbf{M}(1,N_c); \mathbf{M}(2,N_c); \ldots; \mathbf{M}(N_r,N_c)].\end{aligned}$$

A.2.3 From $\mathcal{V}$ to $\mathbf{v}$

A set of l vectors $\mathcal{V} = \{\mathbf{u}_i \in \mathbb{R}^{N_{r,i}}\}_{i=1}^{l}$ can be placed one after the other in a larger vector $\mathbf{v} \in \mathbb{R}^{N_r}$, with

$$N_r = \sum_{i=1}^{l} N_{r,i}.$$

This will be denoted using the $\widehat{\mathbf{Flat}}(\cdot)$ operation as

$$\mathbf{v} = \widehat{\mathbf{Flat}}(\mathcal{V}). \tag{A.4}$$

The convention for assigning elements from the vectors in $\mathcal{V}$ to become elements of the larger vector $\mathbf{v}$ is

$$\mathbf{v}(n) = \mathbf{u}_i(n_r), \quad n = \sum_{m=1}^{i-1} N_{r,m} + n_r \quad n_r = 1, \ldots, N_{r,i}, i = 1, \ldots, l,$$

that is

$$\mathbf{v} =[\mathbf{u}_1(1); \ldots; \mathbf{u}_1(N_{r,1}); \mathbf{u}_2(1); \ldots; \mathbf{u}_2(N_{r,2}); \ldots\ldots; \mathbf{u}_l(1); \ldots; \mathbf{u}_l(N_l)].$$

A.2.4 From $\mathbf{v}$ to $\mathcal{V}$

The $\widehat{\mathbf{Flat}}(\cdot)$ operation can be reversed to split a larger vector into a set of l smaller vectors, where the size of each vector is stored in an additional vector $\mathbf{n} \in \mathbb{R}^l$, with

$$\mathbf{n}(i) = N_{r,i}, i = 1, \ldots, l.$$

This is denoted using the $\widehat{\mathbf{Set}}(\cdot)$ operation as

$$\mathcal{V} = \widehat{\mathbf{Set}}(\mathbf{v}, \mathbf{n}). \tag{A.5}$$

The convention for assigning elements from the the large vector $\mathbf{w}$ to become elements of the vectors in $\mathcal{V}$ is

$$\mathbf{u}_i(n_r) = \mathbf{v}(n), \quad n_r = n - \sum_{m=1}^{i-1} N_{r,m}, \quad i = \underset{h}{\operatorname{argmin}} \left(\sum_{m=1}^{h} N_{r,m} \geq n \right), \quad n = 1, \ldots, N_r,$$

that is

$$\begin{aligned}
\mathbf{u}_1 =& \mathbf{v}(1, \ldots, N_{r,1}), \\
\mathbf{u}_2 =& \mathbf{v}(N_{r,1} + 1, \ldots, N_{r,1} + N_{r,2}), \\
& \ldots \\
\mathbf{u}_l =& \mathbf{v}(N_{r,l-1} + 1, \ldots, N_r).
\end{aligned}$$

A.3 Inner Products and Norms

The following definitions have been taken from [78, 94, 95].

A.3.1 Vector Inner Product

Definition 3. *The inner product between two real matrices* $\mathbf{v} \in \mathbb{R}^{N_r}$ *and* $\mathbf{u} \in \mathbb{R}^{N_r}$ *denoted* $\langle \mathbf{v}, \mathbf{u} \rangle$ *is defined as:*

$$\langle \mathbf{v}, \mathbf{u} \rangle = \sum_{n_r=1}^{N_r} \mathbf{v}(n_r)\mathbf{u}(n_r) \tag{A.6}$$

A.3.2 Euclidean Norm

Definition 4. *The Euclidean norm of a vector* $\mathbf{v} \in \mathbb{R}^{N_r}$ *denoted* $\|\mathbf{v}\|$ *is defined as:*

$$\|\mathbf{v}\| = (\langle \mathbf{v}, \mathbf{v} \rangle)^{\frac{1}{2}} \tag{A.7}$$

A.3.3 Frobenius Inner Product

Definition 5. *The Frobenius inner product between two real matrices* $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ *and* $\mathbf{N} \in \mathbb{R}^{N_r \times N_c}$*, denoted* $\langle \mathbf{M}, \mathbf{N} \rangle_F$ *is defined as:*

$$\langle \mathbf{M}, \mathbf{N} \rangle_F = \sum_{n_r=1}^{N_r} \sum_{n_c=1}^{N_c} \mathbf{M}(n_r, n_c)\mathbf{N}(n_r, n_c). \tag{A.8}$$

A.3.4 Frobenius Norm

Definition 6. *The Frobenius norm of a matrix* $\mathbf{Z} \in \mathbb{R}^{N_r \times N_c}$*, denoted* $\|\mathbf{M}\|_F$ *is defined as:*

$$\|\mathbf{M}\|_F = (\langle \mathbf{M}, \mathbf{M} \rangle_F)^{\frac{1}{2}}. \tag{A.9}$$

A.4 Kronecker Product

Definition 7. *The Kronecker product between two real vectors* $\mathbf{v} \in \mathbb{R}^{N_r}$ *and* $\mathbf{u} \in \mathbb{R}^{N_c}$*, denoted* $\mathbf{v} \otimes \mathbf{u}$ *is defined as:*

$$\mathbf{v} \otimes \mathbf{u} = \begin{bmatrix} \mathbf{v}(1)\mathbf{u}(1) & \mathbf{v}(1)\mathbf{u}(2) & \cdots & \mathbf{v}(1)\mathbf{u}(N_c) \\ \mathbf{v}(2)\mathbf{u}(1) & \mathbf{v}(2)\mathbf{u}(2) & \cdots & \mathbf{v}(2)\mathbf{u}(N_c) \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{v}(N_r)\mathbf{u}(1) & \mathbf{v}(N_r)\mathbf{u}(2) & \cdots & \mathbf{v}(N_r)\mathbf{u}(N_c) \end{bmatrix}.$$

Definition 8. *The Kronecker product between two real matrices* $\mathbf{M} \in \mathbb{R}^{N_r \times N_c}$ *and* $\mathbf{N}$*, denoted* $\mathbf{M} \otimes \mathbf{N}$ *is defined as:*

$$\mathbf{M} \otimes \mathbf{N} = \begin{bmatrix} \mathbf{M}(1,1)\mathbf{N} & \mathbf{M}(1,2)\mathbf{N} & \cdots & \mathbf{M}(1,N_c)\mathbf{N} \\ \mathbf{M}(2,1)\mathbf{N} & \mathbf{M}(2,2)\mathbf{N} & \cdots & \mathbf{M}(2,N_c)\mathbf{N} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{M}(N_r,1)\mathbf{N} & \mathbf{M}(N_r,2)\mathbf{N} & \cdots & \mathbf{M}(N_r,N_c)\mathbf{N} \end{bmatrix}.$$

B B-Spline Dictionary Construction

The B-spline atoms are constructed using the procedure detailed in [2] repeated below for completeness.

We consider equally spaced knots so that the corresponding B-splines are called cardinal. All the cardinal B-splines of order m can be obtained from one cardinal B-spline $B(x)$ associated with the uniform simple knot sequence $\delta = 0, 1, \ldots, m$. Such a function is given as [96]

$$B_m(x) = \frac{1}{m!} \sum_{i=0}^{m} (-1)^i \binom{m}{i} (x-i)_+^{m-1}, \tag{B.1}$$

where $(x-i)_+^{m-1}$ is equal to $(x-i)^{m-1}$ if $x-i > 0$ and 0 otherwise. We shall consider only B-Splines of order $m = 2$ and $m = 4$ and include associated derivatives. For $m = 2$ the corresponding space is the space of piece wise linear functions and can be spanned by a linear B-spline basis, or dictionaries of broader support, arising by translating a prototype 'hat' function. Equivalently, the cubic spline space corresponding to $m = 4$ is spanned by the usual cubic B-spline basis, or dictionaries of cubic B-spline functions of broader support. Details on how to build B-spline dictionaries are given in [79, 80]. The numerical construction of the cases $m = 2$ and $m = 4$ considered here is very simple and arises by translations of the prototype functions given below:

$$B_2^l(x) = \begin{cases} \dfrac{x}{l} & \text{if} \quad 0 \leq x < l & \text{(B.2a)} \\ 2 - \dfrac{x}{l} & \text{if} \quad l \leq x < 2l & \text{(B.2b)} \\ 0 & \text{otherwise.} & \text{(B.2c)} \end{cases}$$

$$B_4^l(x) = \begin{cases} \dfrac{x^3}{6l^3} & \text{if} \quad 0 \leq x < l & \text{(B.3a)} \\ -\dfrac{x^3}{2l^3} + 2\dfrac{x^2}{l^2} - 2\dfrac{x}{l} + \dfrac{2}{3} & \text{if} \quad l \leq x < 2l & \text{(B.3b)} \\ \dfrac{x^3}{2l^3} - 4\dfrac{x^2}{l^2} + 10\dfrac{x}{l} - \dfrac{22}{3} & \text{if} \quad 2l \leq x < 3l & \text{(B.3c)} \\ -\dfrac{x^3}{6l^3} + 2\dfrac{x^2}{l^2} - 8\dfrac{x}{l} + \dfrac{32}{3} & \text{if} \quad 3l \leq x < 4l & \text{(B.3d)} \\ 0 & \text{otherwise.} & \text{(B.3e)} \end{cases}$$

The B-spline basis for the cardinal spline space corresponding to $m = 2$ is constructed by considering $l = 1$ in (B.2) and translating the prototype every knot. Dictionaries for the identical space of functions of broader support arise by setting $l \in \mathbb{N}$ in order to fix the desired support. The B-spline basis for the cubic cardinal spline space, corresponding to $m = 4$, requires to set $l = 1$ in (B.3) and translate the concomitant prototype. Dictionaries are obtained by taking larger values of l.

As discussed below, derivatives of the above functions also provide suitable prototypes to achieve higher levels of sparsity in the representation of a signal. Now, for constructing dictionaries for digital image processing we need to

a) Discretize the functions to obtain adequate Euclidean vectors.

b) Restrict the functions to intervals which allows images to be approximated in small blocks.

We carry out the discretization by taking the value of a prototype function only at the knots (c.f. small circles in graphs Fig. 2.2) and translating the prototype one sampling point at each translation step. At the boundaries we apply the 'cut off' approach and keep all the vectors whose support has nonzero intersection with the interval being considered.

Remark 7. It is worth mentioning that by the proposed discretization the hat B-spline basis for the corresponding interval becomes the standard set of vectors forming a Euclidean basis for the space $\mathbb{R}^N$. The matrix $\mathbf{D}_2^c$ is constructed from these Euclidean vectors as shown below

$$\mathbf{D}_2^c(n, m) = Y_2^2(n - m + 1), n = 1, \ldots, N, m = 1 \ldots, M. \tag{B.4}$$

By discretizing the hats of broader support the samples preserve the hat shape.

As mentioned earlier for a finite dimension Euclidean space one can construct arbitrary dictionaries. In particular, redundant B-spline based dictionaries with prototypes of different support and shapes arising from the functions (B.2) and (B.3) and their corresponding derivatives.

Indicating as $\mathrm{d}^1 B_m^l(x)$ the derivative of $B_m^l(x)$ and as $\mathrm{d}^2 B_m^l(x)$ its second derivative, in addition to linear and cubic B-splines we shall consider the additional prototypes $\mathrm{d}^1 B_2^l(x)$, $\mathrm{d}^1 B_4^l(x)$ and $\mathrm{d}^2 B_4^l(x)$. The union of the following 8 sets of vectors,

$$\{b_i Y_m^s(n-i+1); n=j,\ldots,N+j-1\}_{i=1}^{M_s}, \quad m=2,4,\ s=1,2,3,5,6,7,8,9,$$

with b_i, $i=1,\ldots,M_s$ the normalization constants, forms a set of different supports j. The matrix $\mathbf{D}_3^c$ referred to as the RDBS dictionary is constructed from this set of vectors, with each one forming a column of $\mathbf{D}_3^c$. Each column of $\mathbf{D}_3^c$ will therefore contain N discrete values, with j representing the length of the supports, which are, respectively $1,3,5,2,4,7,7,7$ for $s=1,2,3,5,6,7,8,9$.

The arrays Y_2^s, $s=1,2,3$, Y_4^7, shown consecutively in the left graph of Fig. 2.2, and Y_2^s, $s=5,6$, $Y_4^s, s=8,9$ shown consecutively in the right graph of the same figure, are defined as follows:

$$Y_2^s = \begin{cases} B_2^l,\ l=1,2,3,4 & \text{for} \quad \text{s=1,2,3,4 (respectively)} \quad \text{(B.5a)} \\ \mathrm{d}^1 B_2^l,\ l=2,3 & \text{for} \quad \text{s=5,6 (respectively).} \quad \text{(B.5b)} \end{cases}$$

$$Y_4^s = \begin{cases} B_4^2 & \text{for} \quad \text{s=7} \quad \text{(B.6a)} \\ \mathrm{d}^1 B_4^2 & \text{for} \quad \text{s=8} \quad \text{(B.6b)} \\ \mathrm{d}^2 B_4^2 & \text{for} \quad \text{s=9.} \quad \text{(B.6c)} \end{cases}$$

The cut off approach applied to the boundaries implies that the numbers M_s of total atoms in the sth-dictionary varies according to the atom's support.

C Separable ILS-DLA

The separable ILS-DLA with OMP2D is applied up to iteration J to find the row and column dictionaries $\mathbf{D}^{c,(J)} \in \mathbb{R}^{N_c \times M_c}$ and $\mathbf{D}^{r,(J)} \in \mathbb{R}^{N_r \times M_r}$ using the following steps.

Begin by choosing, a tolerance ρ for OMP2D, a dictionary convergence parameter τ, initial dictionaries $\mathbf{D}^{c,(0)} \in \mathbb{R}^{N_c \times M_c}$ and $\mathbf{D}^{r,(0)} \in \mathbb{R}^{N_r \times M_r}$, and a set of Q image blocks $\mathbf{X}_q \in \mathbb{R}^{N_r \times N_c}, q = 1, \ldots, Q$, taken from a set of training images. The algorithm at iteration $i+1$ then evolves as follows:

a) Using OMP2D with dictionaries $\mathbf{D}^{c,(i)}$ and $\mathbf{D}^{r,(i)}$, find a sparse representation of each image block $\mathbf{X}_q \in \mathbb{R}^{N_r \times N_c}, q = 1, \ldots, Q$, as

$$\mathbf{X}_q^{K_q} = \sum_{k=1}^{K_q} \mathbf{D}^{c,(i)}(:, \mathbf{l}_q^{c,(i)}(k)) \mathbf{c}_q^{(i)}(k) (\mathbf{D}^{r,(i)}(:, \mathbf{l}_q^{r,(i)}(k))^T,$$

subject to $\|\mathbf{X}_q^{K_q} - \mathbf{X}_q\| < \rho$, for $q = 1, \ldots, Q$.

b) Fixing the vectors of coefficients and indices respectively to be $\mathbf{c}_q^{(i)}$ and $(\mathbf{l}_q^{c,(i)}, \mathbf{l}_q^{r,(i)})$ the idea is to find a solution, $\mathbf{D}^{c,(i+1)}$ and $\mathbf{D}^{r,(i+1)}$, to the following sets of equations

$$\mathbf{X}_q = \sum_{k=1}^{K_q} \mathbf{D}^{c,(i+1)}(:, \mathbf{l}_q^{c,(i)}(k)) \mathbf{c}_q^{(i)}(k) (\mathbf{D}^{r,(i+1)}(:, \mathbf{l}_q^{r,(i)}(k)))^T, q = 1, \ldots, Q. \quad \text{(C.1)}$$

Because these equations cannot be reduced to a linear problem they are reduced to two linear problems.

First equation (C.1) is simplified to become

$$\mathbf{X}_q = \mathbf{D}^{c,(i+1)}\mathbf{K}_q^{(i)}(\mathbf{D}^{r,(i+1)})^T, q = 1, \ldots, Q, \tag{C.2}$$

where the matrices $\mathbf{K}_q^{(i)} \in \mathbb{R}^{M_c \times M_r}$ contain zeros, except at positions

$$\mathbf{K}_q^{(i)}(\mathbf{l}_q^{c,(i)}(k), \mathbf{l}_q^{r,(i)}(k)) = \mathbf{c}_q^{(i)}(k), k = 1, \ldots, K_q.$$

A solution to equation (C.2) is then found by recursively solving the two least squares problems below, until both $\|\mathbf{D}^{r,(i+1)} - \mathbf{D}^{r,(i)}\| < \tau$, and $\|\mathbf{D}^{c,(i+1)} - \mathbf{D}^{c,(i)}\| < \tau$, are satisfied.

1) Fix the row dictionary to be $\mathbf{D}^{r,(i)}$, and find the least squares solution $\mathbf{D}^{c,(i+1)}$, to the sets of equations

$$\mathbf{X}_q = \mathbf{D}^{c,(i+1)}\mathbf{K}_q^{r,(i)}, q = 1, \ldots, Q, \tag{C.3}$$

with

$$\mathbf{K}_q^{r,(i)} = \mathbf{K}_q^{(i)}(\mathbf{D}^{r,(i)})^T.$$

2) Use the solution to equation (C.3) to find the least squares solution, $\mathbf{D}^{r,(i+1)}$ to

$$\mathbf{X}_q = \mathbf{K}_q^{c,(i)}(\mathbf{D}^{r,(i+1)})^T, q = 1, \ldots, Q,$$

with

$$\mathbf{K}_q^{c,(i)} = \mathbf{D}^{c,(i+1)}\mathbf{K}_q^{(i)}.$$

D SPMP2D Pseudo Code

Algorithm 1 $[\mathbf{I}^K, \mathbf{c}] \leftarrow \textbf{SPMP2D}(\mathbf{I}, \mathbf{D}^r, \mathbf{D}^c, \rho, \tau)$

Input: $\mathbf{I} \in \mathbb{R}^{N_r \times N_c}$; $\mathbf{D}^c \in \mathbb{R}^{N_c \times M_c}$ with $\|\mathbf{D}^c(:, m_c)\| = 1, m_c = 1, \ldots, M_c$;
$\mathbf{D}^r \in \mathbb{R}^{N_r \times M_r}$ with $\|\mathbf{D}^r(:, m_r)\| = 1, m_r = 1, \ldots, M_r$; $\rho > 0$; $\tau > 0$; $p \geq 1$.

Output: $\mathbf{I}^K \in \mathbb{R}^{N_r \times N_c}$; $\mathbf{c} \in \mathbb{R}^K$.

Initialize: $\Gamma \leftarrow \{\emptyset\}$, $\mathbf{R} \leftarrow \mathbf{I}$, $k \leftarrow 1$, $\text{Error}_1 \leftarrow 2\rho$.

while $\text{Error}_1 > \rho$ **do**

  $i \leftarrow 1$

  while $\text{Error}_1 > \rho$ **and** $i \leq p$ **do**

    $[\{\mathbf{l}^c(k), \mathbf{l}^r(k)\}, c] \leftarrow \textbf{Select\_Atom\_MP}(\mathbf{R}, \mathbf{D}^c, \mathbf{D}^r)$

    $[\mathbf{R}, \text{Error}_1] \leftarrow \textbf{Update\_Residual}(\mathbf{R}, c, \mathbf{D}^c(:, k), \mathbf{D}^r(:, k))$

    $[\mathbf{c}, k, \Gamma] \leftarrow \textbf{Update\_Coefficient}(\mathbf{c}, c, \{\mathbf{l}^c(k), \mathbf{l}^r(k)\}, \Gamma, k)$

    $i \leftarrow i + 1$

  end while

  $[\mathbf{R}, \mathbf{c}, \text{Error}_1] \leftarrow \textbf{ProjMP2D}(\mathbf{R}, \mathbf{D}^c(:, \mathbf{l}^c), \mathbf{D}^r(:, \mathbf{l}^r), \Gamma, \mathbf{c}, \rho)$

end while

$\mathbf{I}^K \leftarrow \mathbf{I} - \mathbf{R}$

Algorithm 2 $[\{\mathbf{l}^c(k), \mathbf{l}^r(k)\}, c] \leftarrow$ **Select_Atom_MP**$(\mathbf{R}, \mathbf{D}^c, \mathbf{D}^r)$

$\mathbf{A} = |\mathbf{D}^c\mathbf{R}(\mathbf{D}^r)^T|$

$\{\mathbf{l}^c(k), \mathbf{l}^r(k)\} \leftarrow \max(\mathbf{A})$ // row and column index of the maximum element in $\mathbf{A}$

$c \leftarrow \mathbf{A}(\mathbf{l}^c(k), \mathbf{l}^r(k))$

Algorithm 3 $[\mathbf{c}, k, \Gamma] \leftarrow$ **Update_Coefficient**$(\mathbf{c}, c, \{\mathbf{l}^c(k), \mathbf{l}^r(k)\}, \Gamma, k)$

if Γ contains $\{\mathbf{l}^c(k), \mathbf{l}^r(k)\}$ **then**

$\mathbf{c}(i) \leftarrow \mathbf{c}(i) + c$, with i such that $\Gamma(i) = \{\mathbf{l}^c(k), \mathbf{l}^r(k)\}$

else

$\mathbf{c}(k) \leftarrow c$

$\Gamma \leftarrow \Gamma \cup \{\mathbf{l}^c(k), \mathbf{l}^r(k)\}$

$k \leftarrow k + 1$ // a new atom has been chosen

end if

Algorithm 4 $[\mathbf{R}, \text{Error}_1] \leftarrow$ **Update_Residule**$(\mathbf{R}, c, \mathbf{D}^c(:, k), \mathbf{D}^r(:, k))$

$\mathbf{R} \leftarrow \mathbf{R} - \mathbf{D}^c(:, k)\mathbf{D}^r(:, k)^T c$

$\text{Error}_1 \leftarrow \|\mathbf{R}\|_F$

Algorithm 5 $[\mathbf{R}, \mathbf{c}] \leftarrow$ **ProjMP2D**$(\mathbf{R}, \mathbf{D}^c(:, \mathbf{l}^c), \mathbf{D}^r(:, \mathbf{l}^r), \Gamma, \mathbf{c}, \rho)$

Initialize: $c \leftarrow 2\rho$

while $c > \rho$ **do**

$[l, c] \leftarrow$ **Select_Atom**$(\mathbf{R}, \mathbf{D}^c(:, \mathbf{l}^c), \mathbf{D}^r(:, \mathbf{l}^r), k)$

$[\mathbf{R}, \text{Error}_1] \leftarrow$ **Update_Residual**$(\mathbf{R}, c, \mathbf{D}^c(:, \mathbf{l}^c(l)), \mathbf{D}^r(:, \mathbf{l}^r(l)))$

$\mathbf{c}(l) = \mathbf{c}(l) + c$

end while

Algorithm 6 $[l, c] \leftarrow$ **Select_Atom**$(\mathbf{R}, \mathbf{D}^c(:, \mathbf{l}^c), \mathbf{D}^r(:, \mathbf{l}^r), k)$

for $i = 1$ **to** k **do**

$\mathbf{a}(i) \leftarrow \mathbf{D}^c(:, \mathbf{l}^c(i))^T \mathbf{R} \mathbf{D}^r(:, \mathbf{l}^r(i))$

end for

$l \leftarrow \max(\mathbf{a})$ // index of the maximum element in $\mathbf{a}$

$c \leftarrow \mathbf{a}(l)$

E Paired Sample t-tests

E.1 Sparse Image Representation with Greedy Algorithms

E.1.1 Sparsity of Greedy Algorithms

SR produced by OMP2D and SPMP2D$_{10}$

A one tailed paired sample t-test was performed over the 45 astronomical sample images to determine if the SR produced by choosing atoms from the RDC-RDBS dictionary using OMP2D was significantly higher than using SPMP2D$_{10}$. The test was performed for each block size $N = 8, 16, 24, 32$. The mean sample SR for OMP2D and SPMP2D$_{10}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the SR for both OMP2D and SPMP2D$_{10}$ is the same, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the SR produced by using OMP2D is greater than the that produced by SPMP2D$_{10}$, $\mu_1 > \mu_2$.

The results of the paired sample t-test are shown in Table 2.1. Table 2.1 shows the difference between the sample mean for OMP2D and SPMP2D$_{10}$, $\bar{x}_d$, the sample standard deviation, the t-statistic and the p-value for $N = 8, 16, 24, 32$. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.68.

N	$\bar{\mathbf{x}}_d$	$\mathbf{s}_d$	t-statistic	p-value
8	1.57×10^{-1}	5.45×10^{-1}	1.93	0.03
16	1.84×10^{-1}	5.02×10^{-1}	2.46	< 0.01
24	1.59×10^{-1}	3.83×10^{-1}	2.79	< 0.01
32	1.49×10^{-1}	2.97×10^{-1}	3.37	< 0.01

Table E.1: Results of one tailed paired t-test undertaken on the 45 grey level astronomical images to determine if the SR produced by choosing atoms from the RDC-RDBS dictionary using OMP2D is significantly higher than using SPMP2D$_{10}$. The results are shown for $N = 8, 16, 24, 32$. The mean sample SR for OMP2D and SPMP2D$_{10}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

Table 2.1 shows that for all N there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at the 95% confidence level. That is the SR resulting from approximating using OMP2D is significantly higher than the SR resulting from approximating using SPMP2D$_{10}$.

SR produced by OMP2D and SPMP2D$_1$ for $N = 24, 32$

A one tailed paired sample t-test was performed over the 45 astronomical sample images to determine if the SR produced by choosing atoms from the RDC-RDBS dictionary with $N = 24, 32$ is significantly higher for OMP2D than for SPMP2D$_1$. The mean sample SR for OMP2D and SPMP2D$_1$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the SR for both OMP2D and SPMP2D$_1$ is the same $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the SR produced by using OMP2D is larger than the that produced by SPMP2D$_1$, $\mu_1 > \mu_2$.

For $N = 24$ the difference between the sample means shown in column 3 rows 1 and 3 of Table 2.1 respectively for OMP2D and SPMP2D$_1$ is, $\bar{x}_d = -0.04$. Therefore for $N = 24$ the SR for OMP2D is not significantly higher than for SPMP2D$_1$.

For $N = 32$ the difference between the sample means shown in column 3 rows 1 and 3 of Table 2.1 respectively for OMP2D and SPMP2D$_1$ is, $\bar{\mathbf{x}}_d = 6.32 \times 10^{-3}$, the sample standard deviation is $\mathbf{s}_d = 3.04 \times 10^{-1}$, the t-statistic is 1.40×10^{-1} and the p-value is 0.44. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.68. Therefore for $N = 32$ there is not enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for $N = 32$ the

SR for OMP2D is not significantly greater than SPMP2D$_1$.

Processing time for OMP2D and SPMP2D$_1$ with $N = 24, 32$

A one tailed paired sample t-test was performed over the 45 astronomical sample images to determine if the processing time resulting from choosing atoms from the RDC-RDBS dictionary with $N = 24, 32$ is significantly higher for OMP2D than for SPMP2D$_1$. The mean sample processing time for OMP2D and SPMP2D$_1$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the processing time for both OMP2D and SPMP2D$_1$ is the same $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the processing time when using OMP2D is larger than by SPMP2D$_1$, $\mu_1 > \mu_2$.

For $N = 24$ the difference between the sample means shown in column 3 rows 1 and 3 of Table 2.1 respectively for OMP2D and SPMP2D$_1$ is, $\bar{x}_d = 1.83$, the sample standard deviation is $s_d = 11.28$, the t-statistic is 1.09 and the p-value is 0.14. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.68. Therefore for $N = 24$ there is not enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for $N = 24$ the processing time for OMP2D is not significantly greater than SPMP2D$_1$.

For $N = 32$ the difference between the sample means shown in row 4 columns 1 and 3 of Table 2.1 respectively for OMP2D and SPMP2D$_1$ is, $\bar{x}_d = 88.31$, the sample standard deviation is $s_d = 88.00$, the t-statistic is 6.73 and the p-value is < 0.01. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.68. Therefore for $N = 32$ there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for $N = 32$ the processing time for OMP2D is significantly greater than SPMP2D$_1$.

E.1.2 Dictionary Selection

$\mathbf{D}_3^{c,1}$ (RDCT-RDW) and $\mathbf{D}_4^{c,1}$ (RDCT-RR)

A one tailed paired sample t-test was performed over the 45 astronomical and natural sample images to determine if the SR produced by approximating using OMP2D with either the the $\mathbf{D}_3^{c,1}$ dictionary or the $\mathbf{D}_4^{c,1}$ dictionary is the same for these two image corpus. The test was performed for each block size $N = 8, 16, 24, 32$. The mean sample SR for $\mathbf{D}_3^{c,1}$ and $\mathbf{D}_4^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the SR produced by OMP2D using each dictionary

45 **Grey Level Astronomical Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	2.89×10^{-1}	2.24×10^{-1}	8.53	< 0.01
16	3.16×10^{-1}	4.29×10^{-1}	4.87	< 0.01
24	2.51×10^{-1}	5.18×10^{-1}	3.21	< 0.01
32	2.00×10^{-1}	5.60×10^{-1}	2.37	0.01
45 **Grey Level Natural Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	7.07×10^{-2}	5.82×10^{-2}	8.07	< 0.01
16	5.74×10^{-1}	1.36	2.81	< 0.01
24	6.22×10^{-1}	1.36	3.05	< 0.01
32	6.13×10^{-1}	1.37	2.97	< 0.01

Table E.2: Results of one tailed paired t-test undertaken on 45, grey level astronomical and natural images to determine if the SR produced by the approximation with OMP2D using either dictionary $\mathbf{D}_3^{c,1}$ or dictionary $\mathbf{D}_4^{c,1}$ is equivalent. The results are shown for $N = 8, 16, 24, 32$. The mean sample SR for $\mathbf{D}_3^{c,1}$ and $\mathbf{D}_4^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

is equivalent, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the SR produced by the $\mathbf{D}_3^{c,1}$ dictionary is greater, $\mu_1 > \mu_2$.

Table E.2 shows the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For both image corpus shown in Table E.2 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level all block sizes $N = 8, 16, 24, 32$. That is the average SR produced by dictionary $\mathbf{D}_3^{c,1}$ is significantly higher than that produced by dictionary $\mathbf{D}_4^{c,1}$.

$\mathbf{D}_2^{c,1}$ (RDCT-RDBS) and $\mathbf{D}_3^{c,1}$ (RDCT-RDW)

A one tailed paired sample t-test was performed over the 45 natural sample images to determine if the SR produced by approximating using OMP2D with either the the $\mathbf{D}_2^{c,1}$ dictionary or the $\mathbf{D}_3^{c,1}$ dictionary is the same for these two image corpus. The test was

45 **Grey Level Natural Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	1.55×10^{-1}	1.61×10^{-1}	6.41	< 0.01
16	3.76×10^{-1}	4.47×10^{-1}	5.57	< 0.01
24	4.82×10^{-1}	6.77×10^{-1}	4.72	< 0.01
32	5.21×10^{-1}	7.54×10^{-1}	4.58	< 0.01

Table E.3: Results of one tailed paired t-test undertaken on 45, grey level natural images to determine if the SR produced by the approximation with OMP2D using either dictionary $\mathbf{D}_2^{c,1}$ or dictionary $\mathbf{D}_3^{c,1}$ is equivalent. The results are shown for $N = 8, 16, 24, 32$. The mean sample SR for $\mathbf{D}_2^{c,1}$ and $\mathbf{D}_3^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

performed for each block size $N = 8, 16, 24, 32$. The mean sample SR for $\mathbf{D}_2^{c,1}$ and $\mathbf{D}_3^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the SR produced by OMP2D using each dictionary is equivalent, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the SR produced by the $\mathbf{D}_2^{c,1}$ dictionary is greater, $\mu_1 > \mu_2$.

Table E.3 shows the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the natural image set. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For the natural image corpus shown in Table E.3 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level for all block sizes $N = 8, 16, 24, 32$. That is the average SR produced by the RDC-RDBS dictionary, $\mathbf{D}_2^{c,1}$ is significantly higher than that produced by the RDCT-RDWdictionary $\mathbf{D}_3^{c,1}$.

$\mathbf{D}_5^{c,1}$ (RDCT-RDW) and $\mathbf{D}_3^{c,1}$ (RDCT-SM)

A one tailed paired sample t-test was performed over the 45 astronomical and natural sample images to determine if the SR produced by approximating using OMP2D with either the the $\mathbf{D}_5^{c,1}$ dictionary or the $\mathbf{D}_3^{c,1}$ dictionary is the same for these two image corpus. The test was performed for each block size $N = 8, 16, 24, 32$. The mean sample SR for $\mathbf{D}_5^{c,1}$ and $\mathbf{D}_3^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the SR produced by OMP2D using each dictionary

45 **Grey Level Astronomical Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	5.85×10^{-1}	6.11×10^{-1}	6.35	< 0.01
16	1.10	1.29	5.65	< 0.01
24	1.42	1.62	5.81	< 0.01
32	1.60	1.80	5.90	< 0.01
45 **Grey Level Natural Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	2.63×10^{-1}	1.31×10^{-1}	-13.26	> 0.99
16	1.92×10^{-1}	1.46×10^{-1}	-8.73	> 0.99
24	-1.53×10^{-1}	1.66×10^{-1}	-6.12	> 0.99
32	-1.50×10^{-1}	1.65×10^{-1}	-6.03	> 0.99

Table E.4: Results of one tailed paired t-test undertaken on 45, grey level astronomical and natural images to determine if the SR produced by the approximation with OMP2D using either dictionary $\mathbf{D}_5^{c,1}$ or dictionary $\mathbf{D}_3^{c,1}$ is equivalent. The results are shown for $N = 8, 16, 24, 32$. The mean sample SR for $\mathbf{D}_5^{c,1}$ and $\mathbf{D}_3^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

is equivalent, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the SR produced by the $\mathbf{D}_5^{c,1}$ dictionary is greater, $\mu_1 > \mu_2$.

Table E.4 shows the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For the astronomical image corpus shown in the top of Table E.4 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level all block sizes $N = 8, 16, 24, 32$. That is the average SR produced by the smaller dictionary, $\mathbf{D}_5^{c,1}$ is significantly higher than that produced by dictionary $\mathbf{D}_3^{c,1}$.

For the natural image corpus shown in the bottom of Table E.4 there is not enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level all block sizes $N = 8, 16, 24, 32$. That is the average SR produced by the smaller dictionary, $\mathbf{D}_5^{c,1}$ is not significantly higher than that produced by dictionary $\mathbf{D}_3^{c,1}$.

$\mathbf{D}_7^{c,1}$ (TS$_2$) and $\mathbf{D}_2^{c,1}$ (RDCT-RDBS)

A one tailed paired sample t-test was performed over the 45 astronomical and natural sample images to determine if the SR produced by approximating using OMP2D with either the the $\mathbf{D}_7^{c,1}$ dictionary or the $\mathbf{D}_2^{c,1}$ dictionary is the same for these two image corpus. The test was performed for each block size $N = 8, 16, 24, 32$, including an additional test for $N = 24$ using dictionary $\mathbf{D}_7^{c,1}$ against $N = 32$ for dictionary $\mathbf{D}_2^{c,1}$ shown in the bottom of Table E.5. The mean sample SR for $\mathbf{D}_7^{c,1}$ and $\mathbf{D}_2^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that the SR produced by OMP2D using each dictionary is equivalent, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the SR produced by the $\mathbf{D}_7^{c,1}$ dictionary is greater, $\mu_1 > \mu_2$.

Table E.5 shows the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For the astronomical image corpus shown in the top of Table E.5 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level all block sizes $N = 8, 16, 24, 32$. That is the average SR produced by the trained dictionary, $\mathbf{D}_7^{c,1}$ is significantly higher than that produced by dictionary $\mathbf{D}_2^{c,1}$.

For the natural image corpus shown in the bottom of Table E.5 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis for block sizes $N = 8, 16, 24$ and not enough evidence for $N = 32$, both at a 95% confidence level. That is the average SR produced by the trained dictionary, $\mathbf{D}_7^{c,1}$ is significantly higher than that produced by dictionary $\mathbf{D}_2^{c,1}$ for blocks when $N = 8, 16, 24$ and the average SR produced by the trained dictionary, $\mathbf{D}_7^{c,1}$ is not significantly higher than that produced by dictionary $\mathbf{D}_2^{c,1}$ for blocks when $N = 32$.

Even though the SR produced by $\mathbf{D}_7^{c,1}$ for $N = 32$ is not significantly higher than dictionary $\mathbf{D}_2^{c,1}$, the SR for $\mathbf{D}_7^{c,1}$ with $N = 24$ is , shown in the last row of Table E.5.

E.1.3 Image Compression

TS$_3$, JPEG, JPEG2000

A one tailed paired sample t-test was performed over the 45 astronomical and natural sample images to determine if the number of bpp required when approximating using OMP2D with $N = 32$ using the the trained dictionary $\mathbf{D}_8^{c,1}$ is significantly different to

45 **Grey Level Astronomical Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	1.09	8.91×10^{-1}	8.12	< 0.01
16	1.4	2.02	5.08	< 0.01
24	1.47	2.45	3.97	< 0.01
32	1.12	2.64	2.83	< 0.01
45 **Grey Level Natural Images**				
N	$\bar{x}_d$	s_d	t-statistic	p-value
8	3.29×10^{-1}	2.23×10^{-1}	9.50	< 0.01
16	2.44×10^{-1}	2.93×10^{-1}	5.53	< 0.01
24	1.30×10^{-1}	3.28×10^{-1}	2.43	< 0.01
32	-1.06×10^{-1}	5.61×10^{-1}	-1.26	0.89
24	9.03×10^{-2}	3.40×10^{-1}	1.76	0.04

Table E.5: Results of one tailed paired t-test undertaken on 45, grey level astronomical and natural images to determine if the SR produced by the approximation with OMP2D using either dictionary $\mathbf{D}_7^{c,1}$ or dictionary $\mathbf{D}_2^{c,1}$ is equivalent. The results are shown for $N = 8, 16, 24, 32$ for both image sets. An additional result is shown for the natural images set, comparing the SR for $\mathbf{D}_7^{c,1}$ with $N = 24$ to that of $\mathbf{D}_2^{c,1}$ with $N = 32$. The mean sample SR for $\mathbf{D}_7^{c,1}$ and $\mathbf{D}_2^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

the that produced by the JPEG2000 algorithm. The mean sample number of bpp for JPEG2000 and $\mathbf{D}_8^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that number of bpp produced equivalent for JPEG2000 and OMP2D with $N = 32$ using dictionary $\mathbf{D}_8^{c,1}$, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the number of bpp produced when using OMP2D with $N = 32$ and dictionary $\mathbf{D}_8^{c,1}$ is greater, $\mu_1 > \mu_2$.

Table E.6 shows the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For both the astronomical and natural image corpus shown in Table E.6 there is not enough evidence to reject the null hypothesis in favour of the alternative hypothesis for lower 3 levels of PSNR, and enough evidence to reject the null hypothesis for the highest level of PSNR at a 95% confidence level. That is for astronomical and natural images the average number of bpp produced by JPEG2000 is significantly higher than that produced by the trained dictionary, $\mathbf{D}_8^{c,1}$ for the highest quality approximation tested, of respectively 45.43dB and 46.45dB.

An additional test was performed using the same parameters on the astronomical image set to determine if the number of bpp required when approximating using OMP2D with $N = 32$ using the the trained dictionary $\mathbf{D}_8^{c,1}$ is significantly higher than produced by the JPEG algorithm for PSNR of 45.43dB. The mean sample number of bpp for JPEG and $\mathbf{D}_8^{c,1}$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that number of bpp produced equivalent for JPEG and OMP2D with $N = 32$ using dictionary $\mathbf{D}_8^{c,1}$, $\mu_1 = \mu_2$. The alternative hypothesis (H_1) is that the number of bpp produced when using JPEG is greater, $\mu_1 > \mu_2$.

The results for this test, displayed in the last row of the top of Table E.6, show there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for the astronomical image corpus the average number of bpp produced by JPEG is significantly higher than that produced by the trained dictionary, $\mathbf{D}_8^{c,1}$ for a PSNR of 45.43dB.

$\mathbf{TS}_3$, $\mathbf{N = 32}$

A one tailed paired sample t-test was performed over the 45 astronomical and natural sample images to determine if the number of bpp required by the dictionary coding method, when approximating using the largest block size with $N = 32$ is significantly lower than

45 **Grey Level Astronomical Images**				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.44dB	-1.91×10^{-2}	2.86×10^{-2}	-4.45	> 0.99
35.17dB	-2.86×10^{-2}	4.98×10^{-2}	-3.81	> 0.99
40.25dB	-1.34×10^{-2}	1.38×10^{-1}	-0.65	0.74
45.43dB	2.30×10^{-1}	2.12×10^{-1}	7.20	< 0.01
45.43dB	4.95×10^{-2}	1.70×10^{-1}	1.93	0.03
45 **Grey Level Natural Images**				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.08dB	-2.00×10^{-2}	6.23×10^{-2}	-2.13	0.98
35.12dB	-3.95×10^{-2}	8.33×10^{-2}	-3.15	> 0.99
40.50dB	-8.79×10^{-2}	1.03×10^{-1}	-5.67	> 0.99
46.45dB	3.86×10^{-1}	2.41×10^{-1}	10.60	< 0.01

Table E.6: Results of one tailed paired t-test undertaken on 45, grey level astronomical and natural images to determine if the number of bpp produced the approximation with OMP2D with $N = 32$ using dictionary $\mathbf{D}_8^{c,1}$ is equivalent to JPEG or JPEG2000. The results are shown for against the average PSNR over the image set of the approximation. The mean number of bpp for JPEG or JPEG2000 and OMP2D are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

using the smaller block sizes, $N = 8, 16$ and 24. The mean sample number of bpp for $N = 8, 16$ and 24, and for the largest block size $N = 32$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$.

The null hypothesis (H_0) is that number of bpp produced when using the largest block size $N = 32$ is equivalent to that of the smaller block sizes, $N = 8, 16$ and 24. The alternative hypothesis (H_1) is that the number of bpp produced when $N = 32$ is smaller, $\mu_1 > \mu_2$.

Table E.7 and E.8 show respectively the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 44 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For both the astronomical and natural image corpus, for all PSNR^a, for both $N = 8$ and 16 shown in the top two Tables of respectively E.7 and E.8 there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical and natural images the average number of bpp produced by the dictionary coding method for all PSNR^a tested, when images are initially approximated with either $N = 8$ or $N = 16$ is significantly higher than when images are initially approximated with $N = 32$.

For the astronomical image corpus shown in the bottom Table of E.7 when images were initially approximated in blocks with $N = 24$ to a PSNR^a of 40.25 and 45.43 with $N = 24$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images the average number of bpp produced by the dictionary coding method for the higher quality approximations tested (PSNR^a = 40.25dB and 45.43dB) when the images are initially approximated using blocks with $N = 24$ is significantly higher than the number of bpp produced when the images are initially approximated using blocks with $N = 32$.

For the natural image corpus shown in the bottom Table of E.8 when images were initially approximated in blocks with $N = 24$ to a PSNR^a of 30.08 and 46.45 with $N = 24$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images the average number of bpp produced by the dictionary coding method for the lowest and highest quality approximations tested (PSNR^a = 30.08dB and 46.45dB) when the images are initially approximated using blocks with $N = 24$ is significantly higher than the number of bpp produced when the images are initially approximated using blocks with $N = 32$.

$N = 8$				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.44dB	6.401×10^{-2}	4.37×10^{-2}	9.72	< 0.01
35.17dB	1.02×10^{-1}	8.27×10^{-2}	8.15	< 0.01
40.25dB	1.76×10^{-1}	1.36×10^{-1}	8.60	< 0.01
45.43dB	3.61×10^{-1}	2.36×10^{-1}	10.14	< 0.01
$N = 16$				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.44dB	1.07×10^{-2}	2.52×10^{-2}	2.82	< 0.01
35.17dB	2.87×10^{-2}	5.89×10^{-2}	3.23	< 0.01
40.25dB	6.61×10^{-1}	6.46×10^{-2}	6.79	< 0.01
45.43dB	3.06×10^{-1}	2.31×10^{-1}	8.81	< 0.01
$N = 24$				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.44dB	-1.32×10^{-3}	2.08×10^{-2}	-0.42	0.66
35.17dB	7.71×10^{-3}	3.39×10^{-2}	1.51	0.07
40.25dB	2.51×10^{-2}	3.89×10^{-2}	4.28	< 0.01
45.43dB	9.86×10^{-2}	1.30×10^{-1}	5.01	< 0.01

Table E.7: Results of one tailed paired t-test undertaken on 45, grey level astronomical images to determine if the number of bpp produced the dictionary coding method using $N = 32$ is significantly lower than for $N = 8, 16$ and 24. The results are shown for against the average PSNR^a over the image set. The mean number of bpp for the smaller block sizes $N = 8, 16$ and 24, and $N = 32$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

N=8				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.08dB	2.01×10^{-1}	1.27×10^{-1}	10.46	< 0.01
35.12dB	3.90×10^{-1}	2.40×10^{-1}	10.79	< 0.01
40.50dB	6.22×10^{-1}	2.82×10^{-1}	14.61	< 0.01
46.45dB	1.28	4.82×10^{-1}	17.45	< 0.01
$N = 16$				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.08dB	4.99×10^{-2}	7.03×10^{-2}	4.71	< 0.01
35.12dB	1.11×10^{-1}	1.05×10^{-1}	6.98	< 0.01
40.50dB	1.56×10^{-1}	1.29×10^{-1}	8.07	< 0.01
46.45dB	7.30×10^{-1}	3.92×10^{-1}	12.36	< 0.01
$N = 24$				
PSNR^a	$\bar{x}_d$	s_d	t-statistic	p-value
30.08dB	1.52×10^{-2}	5.89×10^{-2}	1.72	0.04
35.12dB	-9.84×10^{-4}	8.46×10^{-2}	-0.08	0.53
40.50dB	-6.43×10^{-3}	8.02×10^{-2}	-0.53	0.70
46.45dB	1.07×10^{-1}	2.90×10^{-1}	2.45	< 0.01

Table E.8: Results of one tailed paired t-test undertaken on 45, grey level natural images to determine if the number of bpp produced the dictionary coding method using $N = 32$ is significantly lower than for $N = 8, 16$ and 24. The results are shown for against the average PSNR^a over the image set. The mean number of bpp for the smaller block sizes $N = 8, 16$ and 24, and $N = 32$ are respectively denoted as $\bar{x}_1$ and $\bar{x}_2$ with the sample mean and standard deviation being respectively $\bar{x}_d = \bar{x}_1 - \bar{x}_2$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 44 degrees of freedom and the critical t-value is 1.68.

E.2 Self Contained Encrypted Image Folding

E.2.1 The CR of the "ad hoc" Scheme

A one tailed paired sample t-test was performed over the 55 astronomical and natural sample images, *folded* using the "ad hoc" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The test was performed for each block size $N = 8, 16$ and 24. The mean sample CR for the largest dictionary $\mathbf{D}_8^{c,2}$ is denoted by $\bar{x}_1$ and the mean sample CR for the smaller dictionaries will be denoted by $\bar{x}_2$.

The null hypothesis (H_0) is that CR produced by using the largest dictionary $\mathbf{D}_8^{c,2}$ is equivalent to that of the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$. The alternative hypothesis (H_1) is that the CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is higher, $\mu_1 > \mu_2$.

Table E.9 and E.10 show respectively the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 54 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

The results for the astronomical images processed with $N = 8$ shown in the top of Table E.9 show that for dictionaries $\mathbf{D}_i^{c,2}, 2, 4, 6, 7$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images processed with $N = 8$ the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary$\mathbf{D}_8^{c,2}$ is significantly higher than dictionaries $\mathbf{D}_i^{c,2}, 2, 4, 6, 7$.

The results for the astronomical images processed with $N = 16$ shown in the middle of Table E.9 show that for all dictionaries there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images processed with $N = 16$ the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than all the smaller dictionaries.

The results for the astronomical images processed with $N = 24$ shown in the top of Table E.9 show that for dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 5$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images processed with $N = 24$ the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary$\mathbf{D}_8^{c,2}$ is significantly higher than the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 5$.

The results for the natural images processed with $N = 8$ shown in the top of Table E.10 show that for the two smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images processed with $N = 8$ the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the two smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$.

The results for the natural images processed with $N = 16$ shown in the middle of Table E.10 show that for the three smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images processed with $N = 16$ the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the three smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$.

The results for the natural images processed with $N = 24$ shown in the top of Table E.10 show that for the three smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ and dictionary $\mathbf{D}_7^{c,2}$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images processed with $N = 24$ the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than three smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ and dictionary $\mathbf{D}_7^{c,2}$.

E.2.2 Comparison of Index Storage Methods

A one tailed paired sample t-test was performed over the 55 astronomical and natural sample images to determine if the two index coding methods m_1 and m_2 described in Section 4.7 produce bit streams of equivalent size. The test was performed for each dictionary $\mathcal{D}_i^2, i = 1, \ldots, 8$ and for each block size $N = 8, 16, 24$.

The null hypothesis (H_0) is that the size of the bit streams $N_b^{m_1}$ and $N_b^{m_2}$ produced by each method are equal, that is $N_b^{m_1} = N_b^{m_2}$. The alternative hypothesis (H_1) is that $N_b^{m_1} > N_b^{m_2}$.

Tables E.11 and E.12 show respectively the difference between the sample means $x_d = x_{m_1} - x_{m_2}$, the sample standard deviation, the t-statistic and the p-value for the astronomical and natural image sets. The test has 54 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

For the astronomical corpus blocked using $N = 8, 16$ shown in the first two parts of Table E.11 there is enough evidence to reject the null hypothesis in favour of the

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	-2.52×10^{-2}	2.62×10^{-1}	-0.71	0.76
	2	1.52×10^{-1}	1.10×10^{-1}	10.30	< 0.01
	3	1.74×10^{-2}	9.85×10^{-2}	1.31	0.10
	4	2.27×10^{-2}	7.00×10^{-1}	2.40	0.01
	5	-2.87×10^{-2}	6.17×10^{-2}	-3.46	> 0.99
	6	6.50×10^{-2}	3.91×10^{-2}	12.34	< 0.01
	7	3.91×10^{-2}	3.49×10^{-2}	8.30	< 0.01
16	**1**	9.09×10^{-1}	6.71×10^{-1}	10.05	< 0.01
	2	6.64×10^{-1}	4.42×10^{-1}	11.13	< 0.01
	3	4.70×10^{-1}	3.60×10^{-1}	9.68	< 0.01
	4	2.00×10^{-1}	2.03×10^{-1}	7.31	< 0.01
	5	2.53×10^{-1}	2.30×10^{-1}	8.17	< 0.01
	6	1.06×10^{-1}	9.48×10^{-2}	8.33	< 0.01
	7	7.50×10^{-2}	8.27×10^{-2}	6.72	< 0.01
24	**1**	1.23	9.51×10^{-1}	9.58	< 0.01
	2	6.96×10^{-1}	5.42×10^{-1}	9.53	< 0.01
	3	4.24×10^{-1}	3.90×10^{-1}	8.06	< 0.01
	4	1.24×10^{-1}	1.94×10^{-1}	4.74	< 0.01
	5	1.61×10^{-1}	2.15×10^{-1}	5.54	< 0.01
	6	-3.49×10^{-1}	5.71×10^{-2}	-0.45	0.67
	7	-3.36×10^{-2}	5.91×10^{-2}	-4.22	> 0.99

Table E.9: Results of a one tailed paired sample t-test performed over the 55 astronomical sample images, *folded* using the "ad hoc" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	8.16×10^{-2}	7.00×10^{-2}	8.64	< 0.01
	2	2.40×10^{-2}	5.33×10^{-2}	3.34	< 0.01
	3	-6.93×10^{-2}	6.59×10^{-2}	-7.80	> 0.99
	4	-5.45×10^{-2}	4.43×10^{-1}	-9.13	> 0.99
	5	-7.87×10^{-2}	4.90×10^{-2}	-11.90	> 0.99
	6	-3.58×10^{-2}	2.36×10^{-2}	-11.27	> 0.99
	7	-4.56×10^{-2}	3.01×10^{-2}	-11.24	> 0.99
16	**1**	3.68×10^{-1}	3.24×10^{-1}	8.43	< 0.01
	2	1.30×10^{-1}	1.21×10^{-1}	7.98	< 0.01
	3	4.88×10^{-2}	4.43×10^{-2}	8.17	< 0.01
	4	-1.80×10^{-2}	2.18×10^{-2}	-6.10	> 0.99
	5	4.50×10^{-3}	2.11×10^{-2}	1.58	0.06
	6	-2.94×10^{-2}	4.12×10^{-2}	-5.28	> 0.99
	7	-3.92×10^{-2}	4.54×10^{-2}	-6.40	> 0.99
24	**1**	4.21×10^{-1}	3.70×10^{-1}	8.42	< 0.01
	2	1.54×10^{-1}	1.32×10^{-1}	8.66	< 0.01
	3	5.52×10^{-2}	4.53×10^{-2}	9.05	< 0.01
	4	-2.40×10^{-2}	7.29×10^{-2}	-2.43	> 0.99
	5	7.71×10^{-3}	4.88×10^{-2}	1.17	0.12
	6	-2.41×10^{-2}	3.48×10^{-2}	-5.14	> 0.99
	7	1.90×10^{-2}	2.39×10^{-2}	5.90	< 0.01

Table E.10: Results of a one tailed paired sample t-test performed over the 55 natural sample images, *folded* using the "ad hoc" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

alternative hypothesis at a 95% confidence level for dictionaries $\mathbf{D}_i^{c,2}, i = 5, \ldots, 8$. That is for the astronomical corpus, blocked with $N = 8, 16$, the average size of the bit stream produced by m_2 is significantly smaller than that produced by m_1 for the larger dictionaries $\mathbf{D}_i^{c,2}, i = 5, \ldots, 8$. For $N = 24$ shown at the bottom of Table E.11 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level for dictionaries $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$. That is for the astronomical corpus, blocked with $N = 24$, the average size of the bit stream produced by m_2 is significantly smaller than that produced by m_1 for the larger dictionaries $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$.

For the natural corpus blocked using $N = 8, 16$ shown in the first two parts of Table E.12 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level for dictionaries $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$. That is for the natural corpus, blocked with $N = 8, 16$, the average size of the bit stream produced by m_2 is significantly smaller than that produced by m_1 for the larger dictionaries $\mathbf{D}_i^{c,2}, i = 4, \ldots, 8$. For $N = 24$ shown at the bottom of Table E.12 there is enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level for dictionaries $\mathbf{D}_i^{c,2}, i = 3, \ldots, 8$. That is for the natural corpus, blocked with $N = 24$, the average size of the bit stream produced by m_2 is significantly smaller than that produced by m_1 for the larger dictionaries $\mathbf{D}_i^{c,2}, i = 3, \ldots, 8$.

E.2.3 Examining the CR of the Pixel Scheme

A one tailed paired sample t-test was performed over the 55 astronomical and natural sample images, *folded* using the "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The test was performed for each block size $N = 8, 16$ and 24. The mean sample CR for the largest dictionary $\mathbf{D}_8^{c,2}$ is denoted by $\bar{x}_1$ and the mean sample CR for the smaller dictionaries will be denoted by $\bar{x}_2$.

The null hypothesis (H_0) is that CR produced by using the largest dictionary $\mathbf{D}_8^{c,2}$ is equivalent to that of the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$. The alternative hypothesis (H_1) is that the CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is higher, $\mu_1 > \mu_2$.

Table E.13 and E.14 show respectively the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the astronomical and natural image sets. The test has 54 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

The results for the astronomical images processed with $N = 8$ and 24 shown in the top

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	-12.24	18.87	-4.81	> 0.99
	2	-4.46	8.72	-3.79	> 0.99
	3	-1.60	4.76	-2.50	0.99
	4	0.35	3.37	0.76	0.23
	5	1.66	2.53	4.88	< 0.01
	6	3.32	1.88	13.08	< 0.01
	7	4.19	1.77	17.53	< 0.01
	8	5.98	2.20	20.19	< 0.01
16	**1**	-36.29	39.23	-6.86	> 0.99
	2	-11.49	13.85	-6.15	> 0.99
	3	-3.43	7.48	-3.39	> 0.99
	4	0.64	5.01	0.95	0.17
	5	3.23	3.43	6.98	< 0.01
	6	6.53	1.92	25.29	< 0.01
	7	8.64	1.85	34.63	< 0.01
	8	11.97	2.86	31.00	< 0.01
24	**1**	-50.58	50.06	-7.49	> 0.99
	2	-15.01	18.08	-6.16	> 0.99
	3	-3.46	10.34	-2.48	0.99
	4	2.90	6.94	3.10	< 0.01
	5	7.17	5.58	9.53	< 0.01
	6	12.07	5.13	17.44	< 0.01
	7	15.35	6.26	18.19	< 0.01
	8	20.13	6.98	21.40	< 0.01

Table E.11: Results of one tailed paired sample t-test undertaken on the 55 astronomical test images to determine if the two index coding methods m_1 and m_2 described in Section 4.7 produce bit streams of equivalent size. The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 8$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

$N = 8$					
N	$\mathcal{D}_i^2$	μ_d	s_d	t-statistic	p-value
8	**1**	-5.13	2.52	-15.10	> 0.99
	2	-1.52	0.90	-12.49	> 0.99
	3	-0.59	0.56	-7.76	> 0.99
	4	0.15	0.35	3.04	< 0.01
	5	0.73	0.20	27.85	< 0.01
	6	1.46	0.16	65.90	< 0.01
	7	1.82	0.21	64.21	< 0.01
	8	2.61	0.39	49.79	< 0.01
16	**1**	-8.45	3.90	-16.05	> 0.99
	2	-2.12	1.63	-9.67	> 0.99
	3	-0.07	1.24	-0.42	0.66
	4	1.43	1.18	9.02	< 0.01
	5	2.61	1.31	14.81	< 0.01
	6	4.01	1.45	20.55	< 0.01
	7	5.08	1.62	23.17	< 0.01
	8	6.53	1.91	25.38	< 0.01
24	**1**	-10.51	4.73	-16.49	> 0.99
	2	-2.02	2.47	-6.05	> 0.99
	3	0.99	2.25	3.27	< 0.01
	4	3.24	2.67	9.01	< 0.01
	5	5.05	3.11	12.06	< 0.01
	6	7.06	3.79	13.82	< 0.01
	7	8.44	4.30	14.54	< 0.01
	8	10.48	5.03	15.44	< 0.01

Table E.12: Results of one tailed paired sample t-test undertaken on the 55 natural test images to determine if the two index coding methods m_1 and m_2 described in Section 4.7 produce bit streams of equivalent size. The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 8$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

and bottom of Table E.13 show that for all of the smaller dictionaries except $\mathbf{D}_6^{c,2}$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images processed with $N = 8$ and 24 the average CR produced by using the "ad hoc" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than all the smaller dictionaries, with the exception of $\mathbf{D}_6^{c,2}$.

The results for the astronomical images processed with $N = 16$ shown in the middle of Table E.13 show that for all of the smaller dictionaries there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images processed with $N = 16$ the average CR produced by using the "pixel" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than all the smaller dictionaries.

The results for the natural images processed with $N = 8$ shown in the top of Table E.14 show that for the two smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images processed with $N = 8$ the average CR produced by using the "pixel" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the two smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$.

The results for the natural images processed with $N = 16$ shown in the middle of Table E.14 show that for the three smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images processed with $N = 16$ the average CR produced by using the "pixel" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the three smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$.

The results for the natural images processed with $N = 24$ shown in the top of Table E.14 show that for the three smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ and dictionary $\mathbf{D}_7^{c,2}$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images processed with $N = 24$ the average CR produced by using the "pixel" *folding* method with image approximations made using the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than three smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 3$ and dictionary $\mathbf{D}_7^{c,2}$.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	1.96×10^{-1}	3.21×10^{-1}	4.53	> 0.01
	2	2.31×10^{-1}	2.26×10^{-1}	7.56	< 0.01
	3	2.27×10^{-1}	2.04×10^{-1}	8.23	< 0.01
	4	9.18×10^{-2}	1.54×10^{-1}	4.42	< 0.01
	5	9.16×10^{-2}	1.34×10^{-1}	5.08	< 0.01
	6	2.58×10^{-3}	4.96×10^{-2}	0.38	0.35
	7	1.72×10^{-2}	5.93×10^{-2}	2.15	0.02
16	**1**	1.04	9.66×10^{-1}	7.95	< 0.01
	2	5.62×10^{-1}	4.79×10^{-1}	8.71	< 0.01
	3	3.68×10^{-1}	3.90×10^{-1}	7.00	< 0.01
	4	1.90×10^{-1}	2.71×10^{-1}	5.21	< 0.01
	5	1.37×10^{-1}	1.99×10^{-1}	5.12	< 0.01
	6	1.17×10^{-2}	4.98×10^{-2}	1.75	0.04
	7	5.60×10^{-2}	9.18×10^{-2}	4.52	< 0.01
24	**1**	1.33	1.24	7.96	< 0.01
	2	6.76×10^{-1}	6.03×10^{-1}	8.32	< 0.01
	3	4.52×10^{-1}	5.06×10^{-1}	6.61	< 0.01
	4	1.19×10^{-1}	1.74×10^{-1}	5.08	< 0.01
	5	1.52×10^{-1}	2.38×10^{-1}	4.74	< 0.01
	6	1.07×10^{-2}	5.25×10^{-2}	1.51	0.07
	7	6.63×10^{-2}	1.08×10^{-1}	4.54	< 0.01

Table E.13: Results of a one tailed paired sample t-test performed over the 55 astronomical sample images, *folded* using the "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	1.85×10^{-1}	1.96×10^{-1}	7.02	< 0.01
	2	4.98×10^{-2}	6.21×10^{-2}	5.95	< 0.01
	3	1.55×10^{-2}	7.80×10^{-2}	1.48	0.07
	4	-1.09×10^{-2}	2.64×10^{-2}	-3.06	> 0.99
	5	1.72×10^{-5}	2.22×10^{-2}	0.01	0.50
	6	-7.37×10^{-3}	1.11×10^{-2}	-4.93	> 0.99
	7	1.25×10^{-3}	1.51×10^{-2}	0.61	0.27
16	**1**	3.55×10^{-1}	4.57×10^{-1}	5.75	< 0.01
	2	1.21×10^{-1}	2.48×10^{-1}	3.63	< 0.01
	3	6.94×10^{-2}	2.37×10^{-1}	2.17	0.02
	4	-8.65×10^{-3}	2.55×10^{-2}	-2.51	> 0.99
	5	5.07×10^{-3}	4.64×10^{-2}	-0.81	0.79
	6	-1.41×10^{-2}	4.09×10^{-2}	-2.55	> 0.99
	7	3.96×10^{-3}	2.35×10^{-2}	1.25	0.11
24	**1**	4.04×10^{-1}	4.98×10^{-1}	6.01	< 0.01
	2	1.79×10^{-1}	3.35×10^{-1}	3.95	< 0.01
	3	2.23×10^{-2}	6.54×10^{-2}	2.53	0.01
	4	-1.74×10^{-2}	7.07×10^{-2}	-1.82	0.96
	5	5.08×10^{-3}	5.01×10^{-2}	0.75	0.23
	6	-7.78×10^{-3}	3.15×10^{-2}	-1.83	0.96
	7	1.05×10^{-2}	3.03×10^{-2}	2.58	0.01

Table E.14: Results of a one tailed paired sample t-test performed over the 55 natural sample images, *folded* using the "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

E.3 Folding Colour Images

E.3.1 Examining the CR

A one tailed paired sample t-test was performed over the 55 RGB astronomical and natural sample images, *folded* using the single and multi channel "pixel" methods, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The test was performed for each block size $N = 8, 16$ and 24. The mean sample CR for the largest dictionary $\mathbf{D}_8^{c,2}$ is denoted by $\bar{x}_1$ and the mean sample CR for the smaller dictionaries will be denoted by $\bar{x}_2$.

The null hypothesis (H_0) is that CR produced by using the largest dictionary $\mathbf{D}_8^{c,2}$ is equivalent to that of the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$. The alternative hypothesis (H_1) is that the CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is higher, $\mu_1 > \mu_2$.

Tables E.15 and E.16, and Tables E.17 and E.18 show respectively the difference between the sample means $\bar{x}_d = \bar{x}_1 - \bar{x}_2$, the sample standard deviation s_d, the t-statistic and the p-value for the RGB astronomical and natural image sets. The test has 54 degrees of freedom and the critical t-value for the test with $\alpha = 0.05$ is 1.67.

Astronomical Images

The results for the astronomical images *folded* using the single channel "pixel" method in blocks with $N = 8, 16$ and 24 shown in of Table E.15 show that for all of the dictionaries except $\mathbf{D}_6^{c,2}$ when $N = 16$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images *folded* using the single channel "pixel" method in blocks with $N = 8, 16$ and 24 for all of the dictionaries except $\mathbf{D}_6^{c,2}$, when processing is performed blocks with $N = 16$, the average CR produced by largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than all the smaller dictionaries.

The results for the astronomical images *folded* using the multi channel "pixel" method in blocks with $N = 8, 16$ and 24 shown in of Table E.16 show that for all of the dictionaries there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for astronomical images *folded* using the multi channel "pixel" method in blocks with $N = 8, 16$ and 24 for all of the dictionaries, the average CR produced by the largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than all the smaller dictionaries.

The results for the natural images *folded* using the single channel "pixel" method

with $N = 8$ shown in the top of Table E.17 show that for the two smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images *folded* using the single channel "pixel" method in blocks with $N = 8$ the average CR produced by largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than the two smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$.

Natural Images

The results for the natural images *folded* using the single channel "pixel" method with $N = 16$ and 24 shown in the middle and bottom of Table E.17 show that for the three smallest dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$ and dictionary $\mathbf{D}_7^{c,2}$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images *folded* using the single channel "pixel" method in blocks with $N = 8$ and 24 the average CR produced by largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than three smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, 2$, and dictionary $\mathbf{D}_7^{c,2}$.

The results for the natural images *folded* using the multi channel "pixel" method in blocks with $N = 8, 16$ and 24 shown in of Table E.18 show that for all of the dictionaries except $\mathbf{D}_6^{c,2}$ when $N = 24$ there was enough evidence to reject the null hypothesis in favour of the alternative hypothesis at a 95% confidence level. That is for natural images *folded* using the multi channel "pixel" method in blocks with $N = 8, 16$ and 24 for all of the dictionaries except $\mathbf{D}_6^{c,2}$, when processing is performed blocks with $N = 24$, the average CR produced by largest dictionary $\mathbf{D}_8^{c,2}$ is significantly higher than all the smaller dictionaries.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	7.22×10^{-2}	2.59×10^{-1}	2.07	0.02
	2	1.30×10^{-1}	1.73×10^{-1}	5.58	< 0.01
	3	1.85×10^{-1}	1.84×10^{-1}	7.46	< 0.01
	4	7.79×10^{-2}	1.46×10^{-1}	3.97	< 0.01
	5	8.97×10^{-2}	1.32×10^{-1}	5.06	< 0.01
	6	-1.35×10^{-3}	4.95×10^{-2}	-0.20	0.58
	7	2.85×10^{-2}	6.18×10^{-2}	3.42	< 0.01
16	**1**	9.02×10^{-1}	7.41×10^{-1}	9.02	< 0.01
	2	4.65×10^{-1}	4.16×10^{-1}	8.29	< 0.01
	3	3.17×10^{-1}	3.35×10^{-1}	7.02	< 0.01
	4	1.86×10^{-1}	2.64×10^{-1}	5.23	< 0.01
	5	1.38×10^{-1}	1.82×10^{-1}	5.61	< 0.01
	6	2.18×10^{-2}	5.56×10^{-2}	2.91	< 0.01
	7	5.54×10^{-2}	8.36×10^{-2}	4.92	< 0.01
24	**1**	1.14	9.84×10^{-1}	8.60	< 0.01
	2	5.41×10^{-1}	4.88×10^{-1}	8.21	< 0.01
	3	3.67×10^{-1}	4.03×10^{-1}	6.76	< 0.01
	4	1.31×10^{-1}	1.67×10^{-1}	5.82	< 0.01
	5	1.43×10^{-1}	1.95×10^{-1}	5.43	< 0.01
	6	2.49×10^{-2}	4.85×10^{-2}	3.80	< 0.01
	7	5.53×10^{-2}	8.81×10^{-2}	4.65	< 0.01

Table E.15: Results of a one tailed paired sample t-test performed over the 55 RGB astronomical sample images, *folded* using the single channel "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	8.93×10^{-2}	3.48×10^{-1}	1.90	0.03
	2	1.35×10^{-1}	2.48×10^{-1}	4.05	< 0.01
	3	2.45×10^{-1}	2.17×10^{-1}	8.37	< 0.01
	4	1.53×10^{-1}	1.69×10^{-1}	6.70	< 0.01
	5	1.76×10^{-1}	1.58×10^{-2}	8.24	< 0.01
	6	5.51×10^{-2}	1.07×10^{-1}	3.81	< 0.01
	7	7.38×10^{-2}	9.36×10^{-2}	5.85	< 0.01
16	**1**	9.65×10^{-1}	8.45×10^{-1}	8.46	< 0.01
	2	5.19×10^{-1}	5.07×10^{-1}	7.60	< 0.01
	3	3.84×10^{-1}	3.75×10^{-1}	7.61	< 0.01
	4	2.26×10^{-1}	2.89×10^{-1}	5.81	< 0.01
	5	2.16×10^{-1}	2.24×10^{-1}	7.16	< 0.01
	6	7.24×10^{-2}	8.94×10^{-2}	6.01	< 0.01
	7	8.30×10^{-1}	9.79×10^{-2}	6.29	< 0.01
24	**1**	1.20	1.06	8.37	< 0.01
	2	5.89×10^{-1}	5.51×10^{-1}	7.93	< 0.01
	3	4.12×10^{-1}	4.10×10^{-1}	7.46	< 0.01
	4	2.28×10^{-1}	2.25×10^{-1}	7.54	< 0.01
	5	2.03×10^{-1}	2.09×10^{-1}	7.22	< 0.01
	6	5.73×10^{-2}	7.95×10^{-2}	5.34	< 0.01
	7	8.25×10^{-2}	1.18×10^{-1}	5.16	< 0.01

Table E.16: Results of a one tailed paired sample t-test performed over the 55 RGB astronomical sample images, *folded* using the multi channel "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	1.73×10^{-1}	1.80×10^{-1}	7.12	< 0.01
	2	4.38×10^{-2}	6.15×10^{-2}	5.28	< 0.01
	3	1.20×10^{-2}	6.90×10^{-2}	1.29	0.10
	4	-1.51×10^{-2}	3.59×10^{-2}	-3.11	> 0.99
	5	-1.25×10^{-3}	2.98×10^{-2}	-0.31	0.62
	6	-7.24×10^{-3}	1.73×10^{-2}	-3.11	> 0.99
	7	-3.73×10^{-4}	2.13×10^{-2}	-0.13	0.55
16	**1**	3.42×10^{-1}	4.17×10^{-1}	6.08	< 0.01
	2	1.19×10^{-1}	2.62×10^{-1}	3.37	< 0.01
	3	7.23×10^{-2}	2.60×10^{-1}	2.06	0.02
	4	-1.08×10^{-2}	4.64×10^{-2}	-1.73	0.96
	5	4.57×10^{-3}	2.79×10^{-2}	1.21	0.11
	6	-9.79×10^{-3}	2.07×10^{-2}	-3.50	> 0.99
	7	8.71×10^{-3}	2.03×10^{-2}	3.19	< 0.01
24	**1**	3.90×10^{-1}	4.52×10^{-1}	6.39	< 0.01
	2	1.71×10^{-1}	3.25×10^{-1}	3.89	< 0.01
	3	3.31×10^{-2}	4.80×10^{-2}	5.10	< 0.01
	4	-1.37×10^{-2}	2.95×10^{-2}	-3.44	> 0.99
	5	-7.71×10^{-4}	5.80×10^{-2}	-0.10	0.54
	6	-1.17×10^{-2}	2.70×10^{-2}	-3.21	> 0.99
	7	1.10×10^{-2}	2.63×10^{-2}	3.11	< 0.01

Table E.17: Results of a one tailed paired sample t-test performed over the 55 RGB natural sample images, *folded* using the single channel "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.

N	$\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}$	$\bar{x}_d$	s_d	t-statistic	p-value
8	**1**	3.61×10^{-1}	2.60×10^{-1}	10.31	< 0.01
	2	1.48×10^{-1}	1.05×10^{-1}	10.42	< 0.01
	3	9.40×10^{-2}	8.64×10^{-2}	8.07	< 0.01
	4	4.16×10^{-2}	4.01×10^{-2}	7.69	< 0.01
	5	4.11×10^{-2}	4.61×10^{-2}	6.62	< 0.01
	6	1.24×10^{-2}	3.48×10^{-2}	2.65	0.01
	7	1.42×10^{-2}	3.56×10^{-2}	2.96	< 0.01
16	**1**	5.93×10^{-1}	6.46×10^{-1}	6.81	< 0.01
	2	2.11×10^{-1}	3.46×10^{-1}	4.52	< 0.01
	3	1.12×10^{-1}	2.11×10^{-1}	3.93	< 0.01
	4	4.25×10^{-2}	1.04×10^{-1}	3.04	< 0.01
	5	4.87×10^{-2}	5.18×10^{-2}	6.97	< 0.01
	6	1.92×10^{-2}	5.50×10^{-2}	2.59	0.01
	7	4.12×10^{-2}	1.03×10^{-1}	2.98	< 0.01
24	**1**	6.45×10^{-1}	6.78×10^{-1}	7.05	< 0.01
	2	2.40×10^{-1}	2.54×10^{-1}	7.01	< 0.01
	3	1.11×10^{-1}	1.48×10^{-1}	5.59	< 0.01
	4	3.16×10^{-2}	7.64×10^{-2}	3.07	< 0.01
	5	3.98×10^{-2}	4.84×10^{-2}	6.10	< 0.01
	6	6.16×10^{-3}	4.92×10^{-2}	0.99	0.18
	7	2.55×10^{-2}	4.89×10^{-2}	3.87	< 0.01

Table E.18: Results of a one tailed paired sample t-test performed over the 55 RGB natural sample images, *folded* using the multi channel "pixel" method, to determine if the average CR resulting from *folding* the images with the largest dictionary $\mathbf{D}_8^{c,2}$ (μ_1) was significantly higher than the average CR produced by the smaller dictionaries $\mathbf{D}_i^{c,2}, i = 1, \ldots, 7$ (μ_2). The results are split into 3 sub tables, one for each block size $N = 8, 16$ and 24 which the images were processed in. The result is shown for each column and row dictionary pair $\mathbf{D}_i^{c,2}, \mathbf{D}_i^{r,2}, i = 1, \ldots, 7$ indicated by the index given in the second column. The sample mean and standard deviation are respectively $\bar{x}_d$ and s_d. The null hypothesis (H_0) is that the population mean is zero and alternative hypothesis (H_a) is that population mean is greater than zero, the test has 54 degrees of freedom and the critical t-value is 1.67.