Encrypted Image Folding Software (EIFS)
Manual for MATLAB routines

James Bowley
Aston University, Birmingham, B4 7ET, UK

This software is intended to support the dissemination of techniques developed within the EPSRC funded project.

"Highly nonlinear approximations for sparse signal representation"

`http://www.nonlinear-approx.info`

Contents

Introduction

This software if freely distributed, under GNU license, for noncommercial use. It was developed by James Bowley to implement the method Encrypted Image Folding (EIF) proposed in the paper

[1] *Sparsity and "Something Else": An Approach to Encrypted Image Folding, by James Bowley and Laura Rebollo-Neira.*

All the source files for the routines are available at

http://www.nonlinear-approx.info/EIFS/EIFS.zip,

an example in html can be found at

http://www.nonlinear-approx.info/EIFS/Example/index.html.

To install the EIFS software:

1) Download and extract the file EIFS.zip. This should give you a top directory EIFS and 7 sub directories listed below:
   - Approximation_Routines - required for approximating and folding the images.
   - Common_Routines - required for both folding and unfolding.
   - Example - contains an example showing how to use the EIF software.
   - Expanding_Routines - required for expanding folded images.
   - Folding_Routines - required for folding the image.
   - Images - location of test images.
   - Mex_Files - source code for the mex implementation of OMP2D.
2) Add the directory EIFS and all subdirectories to your MATLAB path.
3) Optional: To decrease the execution time when using the RDCTDB dictionary you can use the RDCTDBMex method which uses a C++ implementation of OMP2D. To do this i) compile the OMP2D.cpp file ii) copy it to the Approximation_Routines subdirectory.
   Note 1: i) and ii) can be done simply running the script CompileOMP in the Mex_Files subdirectoy.
   Note 2: MATLAB has to be configured to compile mex files, please refer to the MATLAB documentation for details of how to do this.

The EIFS software has been tested on MATLAB versions 2009a and 2010a.
Most of this manual from our MATLAB sources was generating with the `M2TEX` script of Andreas Hartmann.

Chapter 1

Approximation Routines

1.1 Function-Summary

ApproxDCT	Approximates a 2D signal by thresholding the smallest DCT coefficients.
ApproxRDCTDB	Approximates a 2D signal using the RDCTDB dictionary and the OMP algorithm.
OMP2D	Orthogonal Matching Pursuit in 2D
ProcessCellImageDCT	Applies ApproxDCT to every cell in a blocked image.
ProcessCellImageGreedy2D	Using the supplied 1D dictionary it applies the greedy algorithm specified to every cell in a blocked image.
ProcessCellImageRDCTDB	Applies ApproxRDCTDB to every cell in a blocked image.

1.2 Function-Description

1.2.1 ApproxDCT

Approximates a 2D signal by thresholding the smallest DCT coefficients.

1.2.2 ApproxRDCTDB

Approximates a 2D signal using the RDCTDB dictionary and the OMP algorithm.

```
Implementation of OMP in 2D using 2 separable 1D dictionaries (for rows and columns)
The dictionaries used here are D_a and D_b where:
  D_a - Normalized redundancy 2 discrete cosine dictionary.
  D_b - Dirac basis.
This implementation uses the fast DCT to calculate the inner products with the residual as
in equations (3)-(5) of [1].

Usage:    [ mImageApprox, vCoefficients, iD1, iV1, iD2, iV2, k, i1D ] =  ApproxRDCTDB( ...
              mImage, mDictionary, tolerance, blockWidth );

Inputs:
  mImage          Matrix representing a block of the image.
  mDictionary     Dictionary of normalized atoms: [D_a, D_b] ⊗ [D_a, D_b];
  tolerance       Stop when norm of the residual < tolerance.
  blockWidth      Width of the square matrix mImage.

Outputs:
  mImageApprox    Approximation of mImage.
  vCoefficients   Coefficients of the decomposition.
  iD1             Indices of the dictionary to which the selected atom belongs (w.r.t
```

```
                columns) iD1 = 1 indicates that the selected atom belongs to dictionary
                $D_a$.
  iV1           Indices of the selected atom in the dictionary iD1.
  iD2           Indices of the dictionary to which the selected atom belongs (w.r.t.
                rows).
  iV2           Indices of the selected atom in the dictionary iD2.
  k             Number of selected atoms.
  i1D           Indices of the selected atoms in mDictionary.

The implementation of OMP method is based on  Gram-Schmidt orthonormalization and adaptive
birothogonalization, as proposed in Ref 2 below.

References:

1-Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, "Orthogonal matching pursuits:
recursive function approximation with applications to wavelet decomposition", in Proceedings of
the 27th Asilomar Conference on Signals, Systems and Computers, 1993.

2-L. Rebollo-Neira and D. Lowe, "Optimized Orthogonal Matching Pursuit Approach",
IEEE Signal Processing Letters, Vol(9,4), 137-140, 2002.
```

1.2.3 OMP2D

Orthogonal Matching Pursuit in 2D

```
It creates an atomic decomposition of a 2D signal using OMP criterion and assuming
separable dictionaries. You can choose a tolerance, the number of atoms to take in or an
initial subspace to influence the OMP algorithm.

Usage:    [H,Di1,Di2, beta, c,Q ] = OMP2D( f, Dx,Dy, tol, No, ind);
          H = OMP2D(f,Dx,Dy); variables tol, No, can also be []

Inputs:
  f       Analyzing 2D signal (Image).
  Dx      Dictionary of normalized atoms (w.r.t. Image rows).
  Dy      Dictionary of normalized atoms (w.r.t. Image columns).
  tol     Desired distance between $f$ and its approximation $H$.
          Default=6.5.
  No      (optional) maximal number of atoms to choose, if the number of chosen atoms
          equals to No, routine will stop (default numel($f$)).
  indx    (optional) indices for an initial subspace; They operate as indx(k),indy(k).
  indy    (optional) indices determining the initial subspace (as above).

Outputs:
  H       Approximation of $f$
  Di1     Indices of selected atoms w.r.t the original Dx
  Di2     Indices of selected atoms w.r.t the original Dy
  beta    Biorthogonal atoms corresponding to the 2D selected atoms
  c       Coefficients of the atomic decomposition
  Q       Orthogonal 2D basis for the selected subspace

The implementation of OMP method is based on  Gram-Schmidt orthonormalization and adaptive
birothogonalization, as proposed in Ref 2 below.

 References:

 1-Y.C. Pati, R. Rezaiifar, and P.S. Krishnaprasad, "Orthogonal matching pursuits:
```

recursive function approximation with applications to wavelet decomposition", in
Proceedings of the 27th Asilomar Conference on Signals, Systems and Computers, 1993.

2-L. Rebollo-Neira and D. Lowe, "Optimized Orthogonal Matching Pursuit Approach", IEEE
Signal Processing Letters, Vol(9,4), 137-140, (2002).

1.2.4 ProcessCellImageDCT

Applies ApproxDCT to every cell in a blocked image.

1.2.5 ProcessCellImageGreedy2D

Using the supplied 1D dictionary it applies the greedy algorithm specified to every cell in a blocked image.

1.2.6 ProcessCellImageRDCTDB

Applies ApproxRDCTDB to every cell in a blocked image.

Chapter 2

Common Routines

2.1 Function-Summary

ApproxBitdepth	Returns the minimum bit depth that can be used to represent the range of pixel intensities in an image.
BlockImage	Splits a 2D signal into nYBlocks x nXBlocks cell of blockWidth x blockWidth matrices.
CalcPSNR	Calculates the Peak Signal to Noise Ratio (PSNR) between 2 matrices containing pixel intensity values.
DCos	Generates a matrix who's columns are Discrete Cosine vectors.
Dictionary	Constructs and returns the requested dictionary.
GenerateEncryptionOperator	Generates encryption operator for the FoldImage and ExpandImage routines.
KroneckerIndex	Given the indexes in D_a and D_b it returns the index in $D \otimes D$ where $D = [D_a, D_b]$.
LoadImage	Takes the path to an image file as input and returns a matrix representation of the image.
MImage2Grey	Converts indexed and truecolor image matrices to a matrix where each element represents a greyscale pixel intensity value.
MakeFigures	Displays a figure with four subfigures showing the original image, the folded image, the image recovered with the incorrect private key and the image recovered with the correct private key.
NormDict	Normalizes a given dictionary.
SaveImage	Saves a matrix as a lossless JPEG file.

2.2 Function-Description

2.2.1 ApproxBitdepth

Returns the minimum bit depth that can be used to represent the range of pixel intensities in an image.

2.2.2 BlockImage

Splits a 2D signal into nYBlocks x nXBlocks cell of blockWidth x blockWidth matrices.

```
If the matrix cannot be split exactly into blocks of blockWidth x blockWidth then it is
reduced in size so it can be.
```

2.2.3 CalcPSNR

Calculates the Peak Signal to Noise Ratio (PSNR) between 2 matrices containing pixel intensity values.

2.2.4 DCos

Generates a matrix who's columns are Discrete Cosine vectors.

```
Returns discrete cosine vectors that belong to the Euclidean space of size szSpace.  The
default is to return a basis for the space.

Usage             mCosines = DCos( szSpace, nFrequencies, redundancy );
                  mCosines = DCos( szSpace, nFrequencies );
                  mCosines = DCos( szSpace );

Inputs:
  szSpace         Size of the Euclidean space the vectors should belong to.
  nFrequencies    Number of frequencies to use starting from 0.  If not specified will be
                  the same as the size of the space.
  redundancy      Redundancy of the dictionary, the default is 1 (basis).

Outputs:
  mCosines        Matrix who's columns are discrete cosine vectors.
```

2.2.5 Dictionary

Constructs and returns the requested dictionary.

```
Returns the Kronecker product of either:
1) The discrete cosine atoms spanning R^sz1DSpace with themselves or,
2) The combined dictionary [D_a, D_b] with itself where:
  D_a - Normalized redundancy 2 discrete cosine dictionary.
  D_b - Dirac basis.

Usage:    [mDictionaryKron, mDictionary, vSize] = Dictionary( type, sz1DSpace );

Inputs:
  type        DCT - Basis of discrete cosines for the space R^sz1DSpace.
              RDCTDB - Dictionary [D_a, D_b].
  sz1DSpace   Dimension of atoms before applying the Kronecker product.

Output:
  mDictionaryKron Matrix containing the 2D dictionary.
  mDictionary     Matrix containing the 1D dictionary.
  vSize           Vector containing the sizes of each 1D dictionary.
```

2.2.6 GenerateEncryptionOperator

Generates encryption operator for the FoldImage and ExpandImage routines.

```
First the routine calculates the orthogonal complement of the approximated image space
using a random matrix generated with a public key (c.f. eq. 11 and 12 in [1]) to avoid
plain text attacks.
Then it rotates (or permutes) these vectors using a a random matrix generated with a
private key to obtain the encryption operator (c.f. eq. 13 in [1]).

Usage:    mEncryptionOperator = GenerateEncryptionOperator( mAtoms, x, sKey, mPOrthSpace );

Inputs:
  mAtoms          Atoms spanning the approximated image space.
  x               Number of used to construct the encryption operator.
  sKey            Structure containing encryption variables.
  mPOrthSpace     Orthogonal projection matrix onto the mAtoms.
```

```
Outputs:
  mEncryptionOperator     Encryption operator.
```

2.2.7 KroneckerIndex

Given the indexes in D_a and D_b it returns the index in $D \otimes D$ where $D = [D_a, D_b]$.

```
Usage:    ind = KroneckerIndex( iD1, iD2, iV1, iV2, vSize );

Inputs
  iD1             Indices of the dictionary to which the selected atom belongs (w.r.t
                  columns) iD1 = 1 indicates that the selected atom belongs to dictionary
                  Da.
  iV1             Indices of the selected atom in the dictionary iD1.
  iD2             Indices of the dictionary to which the selected atom belongs (w.r.t.
                  rows).
  iV2             Indices of the selected atom in the dictionary iD2.
  vSize           Vector containing the sizes of each 1D dictionary.

Output
  ind             Index in the larger dictionary D.
```

2.2.8 LoadImage

Takes the path to an image file as input and returns a matrix representation of the image.

2.2.9 MImage2Grey

Converts indexed and truecolor image matrices to a matrix where each element represents a greyscale pixel intensity value.

2.2.10 MakeFigures

Displays a figure with four subfigures showing the original image, the folded image, the image recovered with the incorrect private key and the image recovered with the correct private key.

2.2.11 NormDict

Normalizes a given dictionary.

```
Usage:    D = NormDict( D, delta );
          D = NormDict( D );

Inputs:
  D       Non-normalized dictionary.
  delta   Parameter, the discrete norm of D is multiplied by √(delta).
          Default value is 1.
Outputs:
  D       Normalized dictionary.

Remark: It normalizes the columns of matrix D.
```

2.2.12 SaveImage

Saves a matrix as a lossless JPEG file.

Chapter 3

Example

3.1 Function-Summary

EIFSExample	This example demonstrates the use of the Encrypted Image Folding Software (EIFS) by reproducing the example in the paper:

3.2 Function-Description

3.2.1 EIFSExample

This example demonstrates the use of the Encrypted Image Folding Software (EIFS) by reproducing the example in the paper:

```
  [1]  Sparsity and ‘something else’: An Approach to  Encrypted Image Folding, by James
       Bowley and Laura Rebollo-Neira (pdf)

The image is a 256 x 256 pixel grey level intensity photo of Bertrand Russell. The EIF
method is applied twice, using two different approaches for the representation:
 1) The standard DCT in 2D by disregarding the smallest coefficients.
 2) The RDCDB dictionary - composed of Discrete Cosine (redundancy 2) plus Dirac Basis
    and a dedicated implementation of the Orthogonal Matching Pursuit (OMP) method for
    these dictionaries.
For a much faster implementation of 2) please use the C++ implementation of OMP in 2D
(OMP2D).
For this:
  i) compile the OMP2.cpp file and move it to EIFS/Approximation_Routines folder, all this
  can be done simply by running the script CompileOMP in the EIFS/Mex_Files
  subdirectory
  ii) replace ’RDCTDB’ below (line 58) by ’RDCTDBMex’.
For each method, the unfolding operation is performed twice, using the correct and
incorrect private keys. The corresponding results shown in Figures 1 and 2.
```

Chapter 4

Expanding Routines

4.1 Function-Summary

ExpandImage	Expands an image from folded image by 1) splitting the host and embedded image and 2) using the embedded coefficients to reconstruct the original.
RRescaleCoefficients	Used by ExpandImage to rescale coefficients.
SeparateCellImage	Used by ExpandImage to 1) separate the host and embedded image using SplitOrthComp and 2) retrieve the embedded coefficients using GenerateEncryptionOperator.

4.2 Function-Description

4.2.1 ExpandImage

Expands an image from folded image by 1) splitting the host and embedded image and 2) using the embedded coefficients to reconstruct the original.

```
The procedure is as follows:
I) Load the image from the imagePath or matrix supplied and split it into Q blocks I_q of
blockWidth^2 pixels each.
II) Split the host and embedded image.
III) Recover the embedded coefficients from the embedded image using the encryption
operator
IV) Reconstruct the rest of the original image from embedded coefficients.

Usage:   mRecoveredImage = ExpandImage( foldedImage, cIndex, method, blockWidth, mImage,...
              publicKey, privateKey, rotate );

Inputs:
MANDATORY
  foldedImage     Either the path to where the folded image is saved as a lossless JPEG
                  file or a matrix representing the folded image.
  cIndex          A cell structure containing the index's of the atoms in the sparse
                  representation.
  mImageGrey      A matrix representation of the original image converted to grey level
                  and altered in size so that it can be blocked exactly.
  method          Approximation method either:
                      DCT - Discrete Cosine Transform.
                      RDCTDB - Implementation of OMP in 2D using 2 separable 1D
                      dictionaries (for rows and columns).
                      The dictionaries used here are D_a and D_b where:
```

D_a: Normalized redundancy 2 discrete cosine dictionary.
D_b: Dirac basis.
RDCTDBMex - Mex file for implementing the above RDCTDB method.

blockWidth	The image is independently processed in Q square blocks of width blockWidth. Default is 8.
publicKey	Key used to generate a the random matrix to calculate the vectors spanning the orthogonal subspace $S^{\perp}_{K_q}$ (c.f. eq. 11 in [1]). This can be changed publicly to prevent plain text attacks. Default is 7.
privateKey	Key used to prevent unauthorized unfolding of the image (c.f. eq. 13 in [1]). Default 123456789.
rotate	Determines the method of preventing unauthorized recovery: 1 - the orthogonal basis is randomly rotated (c.f. eq. 13 in [1]). 0 - the orthogonal basis is randomly permuted. Default is to rotate (1).

Outputs:

mRecoveredImage	A matrix containing the recovered image.

4.2.2 RRescaleCoefficients

Used by ExpandImage to rescale coefficients.

4.2.3 SeparateCellImage

Used by ExpandImage to 1) separate the host and embedded image using SplitOrthComp and 2) retrieve the embedded coefficients using GenerateEncryptionOperator.

Chapter 5

Folding Routines

5.1 Function-Summary

FoldImage	Loads an image and folds it using a public and private key.
GenerateEmbeddedImage	Used by FoldImage to embed coefficients in a generated image (c.f. eq. 8 in [1]).
RescaleCoefficients	Used by FoldImage to rescale coefficients.

5.2 Function-Description

5.2.1 FoldImage

Loads an image and folds it using a public and private key.

When viewing an image from top to bottom this routine folds the bottom section into a sparse representation of the top section. The procedure is as follows:
I) Load the image from the supplied imagePath and split it into Q blocks I_q containing N_q x N_q = blockWidth^2 pixels each.
II) Find a sparse representation of each block $I_q^{K_q}$ in a subspace S_{K_q} (c.f. eq. 10 in [1]).
III) The number of host blocks H depends on the sparsity in the approximation of the whole image I. For each of these H blocks we calculate a basis for $S_{K_q}^{\perp}$ (the orthogonal complement of S_{K_q} in R^{N_q} x R^{N_q}).
IV) We map the coefficients of the sparse representation of the remaining $Q - H$ blocks onto the subspace $S_{K_q}^{\perp}$ constructing in that way the matrices F_q in $S_{K_q}^{\perp}$ (c.f. eq. 8 in [1]).
V) We add each matrix F_q to the corresponding host block $I_q^{K_q}$ to obtain $G_q = F_q + I_q^{K_q}$, with $I_q^{K_q}$ in S_{K_q} and F_q in $S_{K_q}^{\perp}$.

```
Usage:        [ foldedImage, cIndex, mImageGrey, maxIntensity] = FoldImage( imagePath,...
                  approxMethod, approxCriteria, blockWidth, publicKey, privateKey,...
                  rotate, saveImage );

Inputs:
MANDATORY
  imagePath   Relative or absolute path to the location of the image file including the
              image file name.
OPTIONAL
  method      Approximation method either:
                  DCT - Discrete Cosine Transform.
                  RDCTDB - Implementation of OMP in 2D using 2 separable 1D dictionaries
                  (for rows and columns).
```

```
                The dictionaries used here are D_a and D_b where:
                    D_a: Normalized redundancy 2 discrete cosine dictionary.
                    D_b: Dirac basis.
                RDCTDBMex - Mex file for implementing the above RDCTDB method.
            Default is RDCTDB.
  PSNR      Desired PSNR of image approximation.  The result will be a greater PSNR for
            the whole image as each block must individually satisfy this criteria.
            Default is 40dB.
  blockWidth The image is independently processed in Q square blocks of width blockWidth.
            Default is 8.
  publicKey Key used to generate a the random matrix to calculate the vectors spanning
            the orthogonal subspace S^⊥_{K_q} (c.f. eq. 11 in [1]). This can be changed
            publicly to prevent plain text attacks.
            Default is 7.
  privateKey Key used to prevent unauthorized unfolding of the image (c.f. eq. 13 in [1]).
            Default 123456789.
  rotate    Determines the method of preventing unauthorized recovery:
                1 - the orthogonal basis is randomly rotated (c.f. eq. 13 in [1]).
                0 - the orthogonal basis is randomly permuted.
            Default is to rotate (1).
  saveImage Determines what is returned to the variable foldedImage.
                1 - path to where the folded image is saved as a lossless JPEG.
                0 - matrix representation of the folded image.
            Default is to save (1).

Outputs:
  foldedImage   Either the path to where the folded image is saved as a lossless JPEG
                file or a matrix representing the folded image.
  cIndex        A cell structure containing the index's of the atoms in the sparse
                representation.
  mImageGrey    A matrix representation of the original image converted to grey level
                and altered in size so it can be blocked exactly.
  maxIntensity  Maximum size of pixel intensity that can be used.
```

5.2.2 GenerateEmbeddedImage

Used by FoldImage to embed coefficients in a generated image (c.f. eq. 8 in [1]).

5.2.3 RescaleCoefficients

Used by FoldImage to rescale coefficients.

Chapter 6

Additional Scripts

6.1 Function-Summary

CompileOMP	Script to compile the OMP2D.cpp file and copy it to the Approximation_Routines directory.

6.2 Function-Description

6.2.1 CompileOMP

Script to compile the OMP2D.cpp file and copy it to the Approximation_Routines directory.