DLCT-Toolbox, a Matlab package for the Dictionary Learning Approach to Tomographic Image Reconstruction

Sara Soltani *

Department of Applied Mathematics and Computer Science, Technical University of Denmark, DK-2800 Kgs. Lyngby, Denmark.

August 25, 2015

Contents

*ssol@dtu.dk, sarahsoltani@gmail.com
This work is part of the project HD-Tomo funded by Advanced Grant No. 291405 from the European Research Council.
2010 Mathematics Subject Classification: 65F22, 65K10, 15A69.
Key words and Phrases: Tomography, Dictionary learning, Tensor decomposition, Inverse problem, Regularization, Sparse representation, Image reconstruction.

1 Introduction

1.1 Overall

In our recent works [5, 6] we have developed a two-stage algorithm using a dictionary learning approach in a discrete tomographic reconstruction problem both with a matrix and a tensor formulation. We first build a training data base from given training images and then construct a dictionary purely from the training data by looking for a number of basis elements in terms of the dictionary and the coefficients/representations of the training data in the dictionary. In these two works ([5, 6]) dictionary learning is formulated as a non-negative matrix/tensor formulation with sparsity constraints on the representation, and then the reconstruction problem is formulated as a sparse approximation problem in a convex optimization framework. We optimized the dictionary learning problem locally by using the Alternating Direction Method of Multipliers (ADMM), see e.g., [2]. The tomographic reconstruction problems are solved using the software package TFOCS (Templates for First-Order Conic Solvers) [1].

This documentation describes the implementations details of the proposed algorithms for solving the tomographic image reconstruction problem in Matlab. This package is designed for a discretized computed tomography problem formulation, i.e., we use a matrix formulation of the reconstruction problem ($Ax \approx b$), where $b \in \mathbb{R}^m$ contains the noisy measurement data and A is the system matrix. The vector $x \in \mathbb{R}^n$ represents an $M \times N$ image, with $n = MN$, of absorption coefficients. The total number of tomographic measurement is m. The system matrix $A \in \mathbb{R}^{m \times n}$ is available. We use the AIR Tools [4] to compute the system matrix A. The discretized tomographic problem is a large sparse and often ill-posed system and incorporating a priori information about the solution is a necessity to improve the reconstruction and regularize the solution. This package aims at providing a computational framework for the use of training images as priors for the solution in tomographic image reconstruction in the scheme described above. The algorithms work for any size of the system matrix A; however, we are more concerned with underdetermined problems where $m < n$, because the need for regularization is even more pronounced in that scenarios.

This package of Matlab routines provides the user with easy-to-use routines and demo scripts, based on formulation and numerical algorithms described in [5, 6]. With the demo script files, the user can easily specifies a few values of problem parameters to obtain a solution to the dictionary learning problem,

the tomographic reconstruction problem and compute the best approximation error of a given test image by the obtained dictionary. For an advanced use of the routines, we invite the reader to carefully read [5, 6], as well as this documentation and the descriptions in the `.m` files provided in this package.

1.2 Toolbox and Software Dependencies

The following toolboxes and software routines are required for this package to be run smoothly on a computer system.

- *Matlab*: The package has been implemented in Matlab version 2014a.
- *AIR Tools* [4] is a Matlab software package for tomographic reconstruction consisting of tomographic test problems and a number of algebraic iterative reconstruction methods. The user can download AIR Tools in a zip format file from `http://www2.compute.dtu.dk/~pcha/AIRtools/`.
- *TFOCS* [1] is a suite of programs and routines designed to help constructing of first-order methods for a variety of convex optimization problems arise in compressed sensing, sparse recovery, and low rank matrix completion e.g. (BPDN and Lasso). The user can download TFOCS in a zip format file from `http://cvxr.com/tfocs/download/`.

The user may download and unpack the software routines in a separate folder while the paths to those folders can be added to the Matlab's current path by including `addpath` statements.

1.3 Test Images

A collection of gray scale test images has been provided in the package, located in the folder "*/TestImages*". There are 6 built-in test problems in this version namely: "`Peppers, Matches, Binary, D53, Zirconium` and `Steel`". The high-resolution photos of `Peppers` and `Matches` are provided by professor Samuli Siltanen from University of Helsinki. The Zicronium test image is courtesy of Dr. Hamidreza Abdolvand from University of Oxford. The steel microstructure image is from [8] and the D53 test image is chosen from the normalized brodatz texture database [7]. The binary test image is generated by means of `phantomgallery.m` function from *AIR Tools*. The user should update Matlab's path with links to the subdirectories /TestImage and /TestResults.

1.4 Notation

The linear tomographic reconstruction problem is formulated as $Ax \approx b$ where the vector $x \in \mathbb{R}^n$ represents the unknown image of size $M \times N$, the vector $b \in \mathbb{R}^m$ is the given noisy data, and the matrix $A \in \mathbb{R}^{m\times n}$ represents the forward model. We use the following notation, where B is an arbitrary matrix:

$$\|B\|_{\mathrm{F}} = \left(\sum_{ij} B_{ij}^2\right)^{1/2}, \quad \|B\|_{\mathrm{sum}} = \sum_{ij} |B_{ij}|,$$

2 Package Details

2.1 Dictionary Learning Problem

`DL_demo.m`	Demo and example script for the dictionary learning (DL) problem in the matrix form.
`DicLear_M.m`	The function routine to learn a matrix dictionary with the given parameters.
`Lea_Algorithm_D2.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the Matrix form such that $D \in Ð_2$.
`Lea_Algorithm_Dinf.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the Matrix form such that $D \in Ð_\infty$.
`dykstra.m`*	Finds a point in the intersection of two convex sets by iteratively projecting onto each of the convex sets.
`montageplot.m`*	This function is a tool to illustrate the dictionary elements as images.

2.1.1 `DL_demo.m`

- **Description**: Demo script for the dictionary learning (DL) problem in the matrix form.
- **Limit**: The test images are predefined in the codes.
- **Dependencies**: `DicLear_M`, `Lea_Algorithm_D2`, `Lea_Algorithm_Dinf`, `montageplot`, `dykstra`
- **Usage:** `DL_demo`

2.1.2 `DicLear_M.m`

- **Description**: The function loads the test images, extract training patches from those images and computes a dictionary with the given parameters by finding a local optimum to:

$$\min_{D,H} \quad \frac{1}{2}\|Y - DH\|_{\mathrm{F}}^2 + \lambda \|H\|_{\mathrm{sum}} \qquad \text{s.t.} \qquad D \in Ð,\ H \in \mathbb{R}_+^{s\times t}, \quad (1)$$

where $\|H\|_{\mathrm{sum}} = \sum_{ij} |H_{ij}|$,

$$Ð_\infty \equiv \{D \in \mathbb{R}_+^{p\times s} \,|\, \|d_j\|_\infty \le 1\} \quad \text{and} \quad Ð_2 \equiv \{D \in \mathbb{R}_+^{p\times s} \,|\, \|d_j\|_2 \le \sqrt{p}\}.$$

The patches of the dictionary are of size $P \times Q$ with $p = PQ$. The code accepts a range of different values for λ.

This function can also run without any input parameter and with the default values defined in the code.

- **Dependencies**: `Lea_Algorithm_D2`, `Lea_Algorithm_Dinf`, `dykstra`

- **Usage**:

  `[D, H, Norm1H, Norm1HCol, Density, NrNZero, t_end, output] = ...`
  `DicLear_M(TImage, patch_size, s, Lambda, DSet, t, Outdisplay);`

- **Inputs**:
  - `TImage`: The name of the test image, the test image options are: `Peppers`, `Matches`, `Binary`, `D53`, `Zirconium`, `Steel`
  - `patch_size`: The training patch size
  - `s`: Number of dictionary elements
  - `Lambda`: The regularization parameter, The code accepts a range of different values for λ.
  - `DSet`: The name of the compact and convex set which the dictionary belongs to, the options are: `D_inf` and `D_2`.
  - `t`: Number of training patches
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default value = 0.

- **Outputs**:
  - `D`: The matrix dictionary(ies) of size p by s for each λ where $p =$`patch_size(1)patch_size(2)`.
  - `H`: The representation matrix(ces) of size s by t for each λ
  - `Norm1H`: $\|H\|_{\text{sum}}$
  - `Norm1HCol`: $\|H\|_{\text{sum}}/t$
  - `Density`: The density percentage(s) of the matrix(ces) H
  - `NrNZero`: The absolute number of nonzeros in the representation matrix(ces) H
  - `t_end`: Total time used by the ADMM algorithm to find a solution for a specific λ
  - `output`: Structured output array of the following fields:
    - `Objective`: The objective function value(s) for each iteration of the ADMM algorithm: $1/2\|Y - DH\|_{\text{F}}^2 + \lambda\|H\|_{\text{sum}}$.
    - `Residual`: The residual value(s) for each iteration of the ADMM algorithm: $\|Y - DH\|_{\text{F}}$.
    - `StopCr1, StopCr1, StopCr1, StopCr1, StopCr1`: The stopping criteria values at each iteration of the ADMM algorithm.

- **Example**:

```
TImage = 'Peppers';
patch_size = [5 5];
t = 10000;
s = 100;
Lambda = 1;
DSet = 'D_2';
Outdisplay = 1;
[D,H] = DicLear_M(TImage, patch_size, s, Lambda, DSet, t, Outdisplay);
```

2.1.3 `Lea_Algorithm_D2.m`

- **Description**: This function performs the ADMM algorithm in k iterations for the dictionary learning problem in the Matrix form, for one value of λ, where the dictionary D belongs to the set $Ð_2$, `DSet = 'D_2'`;
- **Usage:**

```
[W, H, Rs1, Rs2, Rs3, Rs4, Rs5, fobjec, ResF] = ...
Lea_Algorithm_D2(X, tol, maxiter, s, Cof_Lambda, U, V, ...
Cof_rho, Outdisplay);
```

- **Input:**
  - `X`: Training matrix
  - `tol`: The tolerance for the ADMM convergence
  - `maxiter`: Total number of ADMM iterations
  - `s`: The number of dictionary elements
  - `Cof_Lambda`: the regularization parameter λ in the dictionary learning problem formulation
  - `U,V`: The initial value for the auxiliary variables in the ADMM algorithm
  - `Cof_rho`: The augmented Lagrangian parameter
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default value = 0.
- **Output:**
  - `W`: The dictionary matrix for the corresponding λ
  - `H`: The representation matrix for the corresponding λ
  - `fobjec`: The objective function value for each iteration of the ADMM algorithm: $\frac{1}{2}\|Y - DH\|_{\mathrm{F}}^2 + \lambda\|H\|_{\mathrm{sum}}$
  - `ResF`: The residual value for each iteration of the ADMM algorithm: $\|Y - DH\|_{\mathrm{F}}$
  - `Rs1, Rs2, Rs3, Rs4, Rs5`: The stopping criteria values at each iteration of the ADMM algorithm

2.1.4 `Lea_Algorithm_Dinf.m`

- **Description**: This function performs the ADMM algorithm in k iterations for the dictionary learning problem in the Matrix form„ for one value of λ, where the dictionary D belongs to the set $Ð_\infty$, `DSet = 'D_inf'`;
- **Usage**:

```
[W, H, Rs1, Rs2, Rs3, Rs4, Rs5, fobjec, ResF] = ...
Lea_Algorithm_Dinf(X, tol, maxiter, s, Cof_Lambda, U, V, ...
Cof_rho, Outdisplay);
```

- **Input, Output**: **See** `Lea_Algorithm_D2.m`

2.2 Mean Approximation Error

`MAEM_demo.m`	Demo script file illustrating how to compute the approximation error of a test problem in the cone defined by the given dictionary.
`MAE_M.m`	Computes the approximation error for the matrix formulation.

2.2.1 `MAEM_demo.m`

- **Description**: This demo script file illustrates how to compute the approximation error of a test problem in the cone defined by the dictionary. This code handle various dictionaries for a range of values for λ
- **Dependencies**: TFOCS Toolbox, `MAE_M.m`
- **Limit**: This codes uses TFOCS optimization solver (`http://cvxr.com/tfocs/`). Make sure TFOCS is properly installed on the computer.
- **Usage:** `MAEM_demo`

2.2.2 `MAE_M.m`

- **Description**: This function computes the approximation error for the matrix formulation i.e., how well we can represent the exact image in the cone defined by the dictionary. To evaluate the approximation error, i.e., the distance of the exact image x^{exact} to its projection on the cone $\mathcal{C} = \{Dz | z \in \mathbb{R}^s_+\} \subseteq \mathbb{R}^p_+$, we compute the solutions $\alpha_j^\star$ to the q approximation problems for all blocks $j = 1, 2, \ldots, q$ in x^{exact}. This code handle various dictionaries for a range of values for λ.
- **Dependencies**: TFOCS Toolbox
- **Limit:** The test images are predefined, the user should modify the test images in the code for other test problem or image sizes. Make sure TFOCS is properly installed on the computer.
- **Usage:**

```
[Appr_Error_Mean, Appr_Error, Alph, Norm1Alph, SparsityAlph] =
MAE_M(TrImage, D, Lambda, patch_size, Outdisplay);
```

- **Inputs:**
  - `TrImage`: The test image, options are: `'Peppers'`, `'Matches'`, `'Binary'`, `'Zirconium'`, `'Steel'`, `'D53'`
  - `D`: The given(learned) dictionary, note that the dictionary should be obtained from a training image similar to the test image
  - `Lambda`: The regularization parameter in the dictionary learning problem formulation, note that it should be consistence with the dictionary D
  - `patch_size`: The patch sizes of the dictionary elements, should be provided if the dictionaries patch sizes are rectangular, default = `[ceil(sqrt(size(D,1))),ceil(sqrt(size(D,1)))]`

  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default value = 0.

- **Outputs:**
  - `Appr_Error_Mean`: Mean approximation error
  - `Appr_Error`: The vector of approximation errors for each block in the image
  - `Alph`: The best representation/approximation of the jth block in the cone ($\alpha_j^\star$).
  - `Norm1Alph`: $\|\alpha_j^\star\|_1$ for each block j
  - `SparsityAlph`: The sparsity percentage of the representation vector $\alpha_j^\star$ for each block

- **Example:**

```
TImage = 'Peppers';
load('Dic.mat','D');
Lambda = 1;
patch_size = [5 5];
Outdisplay = 1;
[Appr_Error_Mean, Appr_Error] = ...
MAE_M(TrImage, D, Lambda, patch_size, Outdisplay);
```

2.3 The Reconstruction Problem

`RecM_demo`	Demo script for the tomographic reconstruction problem in the matrix form.
`RecM_Algorithm`	Solves the tomographic reconstruction problem in the matrix form for the asked test image by TFOCS.
`Perm_Vec`*	Returns the permutation vector which gives a permutation to the solution.
`L_Matrix`*	Returns the matrix L, used to penalize the block artifacts in the reconstruction formulation.
`Linear_Opr`*	Defines a linear operator for the tomographic reconstruction matrix formulation used by the TFOCS.

2.3.1 `RecM_demo.m`

- **Description**: Demo script for the tomographic reconstruction problem in the matrix form. This script illustrates the use of the dictionary learning approach in the discrete tomographic reconstruction problem. Note that this script solves a large scale sparse approximation problem and many iterations are needed to converge to the solution and this is not an bug/error of the code.

- **Dependencies**: TFOCS, `RecM_Algorithm`, `Perm_Vec`, `L_Matrix`, `Linear_Opr`

- **Limit:** The test images are predefined, the user should modify the test images in the code for other test problem or image size. This code handles one dictionary for a specific λ and s. Make sure TFOCS is properly installed on the computer.
- **Usage:** RecM_demo

2.3.2 `RecM_Algorithm.m`

- **Description**: This function solves the tomographic reconstruction problem for the given test problem by TFOCS. The problem is given by:

$$\begin{array}{ll} \min_{\alpha\in\mathbb{R}^{sq}} & \frac{1}{2m}\|A\Pi^{\mathrm{T}}(I\otimes D)\alpha - b\|_2^2 + \mu\,\frac{1}{q}\|\alpha\|_1 + \delta^2\,\psi(\Pi^{\mathrm{T}}(I\otimes D)\alpha) \\ \text{s.t.} & \alpha \geq 0 \end{array} \tag{2}$$

  with regularization parameters $\mu, \delta > 0$, where

$$\psi(z) = \frac{1}{M(M/P-1)+N(N/Q-1)}\,\frac{1}{2}\,\|Lz\|_2^2.$$

  x is an $M \times N$ image such that $n = MN$ and q is the total number of non-overlapping patches in the image x. Let $\tau = \mu/q$.

  The code can handle a range of values for both τ and δ. Note that this function solves a large scale sparse approximation problem and many iterations are needed to converge to the solution and this is not an bug/error of the code.

- **Dependencies**: TFOCS, `Perm_Vec`, `L_Matrix`, `Linear_Opr`
- **Limit:** The test images are predefined, the user should modify the test images in the code for other test problem or image size. This code handles one dictionary for a specific λ and s. The regularization parameters $\tau, \delta > 0$ are predefined in the code. The user can easily modify the code for other values of regularization parameter.
- **Usage:**

```
[TrI, X_sol, Alph, Error_Sol, tau, delta, Sparsity, ...
Resudial_Fit, AlphaNorm1, AlphaNorm1Avr, t_end] = ...
RecM_Algorithm(TrImage, D, patch_size, N_p, Ra, rnl, ...
geo, Outdisplay)
```

- **Input:**
  - `TrImage`: The test image, options are: `'Peppers'`, `'Matches'`, `'Binary'`, `'Zirconium'`, `'Steel'`, `'D53'`
  - `D`: The given(learned) dictionary, note that the dictionary should be obtained from a training image similar to the test image.
  - `patch_size`: The patch sizes of the blocks in the image, note that the image size is a multiple of the patch size.
  - `N_p`: Number of tomographic projections

  - `Ra`: Range of the angles for the tomographic projections, e.g., full-range $[0°, 180°]$, limited-angle $[0°, 120°]$
  - `rnl`: Gaussian additive noise level in the tomographic data
  - `geo`: Choose a parallel-beam CT geometry or a fan-beam CT geometry, options = `'fanbeam'` and `'parallelbeam'`, default value = `'parallelbeam'`.
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default = 0.

- **Output:**
  - `TrI`: The exact image considered in the tomographic problem. The size of the TrI is predefined.
  - `X_sol`: The tomographic reconstruction solution recovered from α
  - `Alpha`: The sparse representation/solution in the given dictionary obtained by solving our tomographic reconstruction problem
  - `Error_Sol`: Relative reconstruction error
  - `tau`: The sparsity regularization parameter
  - `delta`: The block artifacts regularization parameter
  - `Sparsity`: The sparsity percentage of the representation vector α
  - `Resudial_Fit`: The residual of the tomographic data fitting term
  - `AlphaNorm1`: $\|\alpha\|_1$
  - `AlphaNorm1Avr`: $\|\alpha\|_1/q$
  - `t_end`: Total time used by the TFOCS to find the representation vector (α) for a specific regularization parameter (τ and δ)

- **Example:**

```
TImage = 'Peppers';
load('Dic.mat','D');
patch_size = [5 5];
Ra = 180;
N_p = 25;
rnl = 0.01;
geo = 'fanbeam';
Outdisplay = 1;
[TrI, X_sol, Alph] = ...
RecM_Algorithm( TrImage, D, patch_size, N_p, Ra, rnl, geo, Outdisplay )
```

2.4 The Tensor Dictionary Learning problem

`DL_Tensor_demo.m`	Demo script for the dictionary learning (DL) problem in the tensor form.
`DicLear_T.m`	The function routine to learn a tensor dictionary with the given parameters.
`Lea_Algorithm_T_D2.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the tensor form such that $D \in \mathsf{D}_2$.
`Lea_Algorithm_T_Dinf.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the tensor form such that $D \in \mathsf{D}_\infty$.
`dykstra.m`*	Finds a point in the intersection of two convex sets by iteratively projecting onto each of the convex sets.
`montageplot`*	This function is a tool to illustrate the dictionary elements as images.
`tprod`*	Computes the tensor-tensor product using FFTs [3].
`tran`*	Returns the tensor transpose [3].
`fronorm`*	Computes Frobenius norm of a tensor [3].

2.4.1 `DL_Tensor_demo.m`

- **Description**: Demo script for the dictionary learning (DL) problem in the tensor form.
- **Limit**: The test images are predefined in the codes.
- **Dependencies**: `DicLear_T`, `Lea_Algorithm_T_D2`, `Lea_Algorithm_T_Dinf`, `montageplot`, `dykstra`, `tprod`, `tran`, `fronorm`
- **Usage:** `DL_Tensor_demo`

2.4.2 `DicLear_T.m`

- **Description**: The function loads the test images, extract training patches from those images and computes a dictionary with the given parameters by finding a local optimum to:

$$\min_{\mathcal{D},\mathcal{H}} \ \frac{1}{2}\|\mathcal{Y} - \mathcal{D} * \mathcal{H}\|_{\mathrm{F}}^2 + \lambda\|\mathcal{H}\|_{\mathrm{sum}} + I_{\mathsf{D}}(\mathcal{D}) + I_{\mathbb{R}_+^{s\times t\times Q}}(\mathcal{H}), \tag{3}$$

where the patches of the dictionary are of size $P \times Q$,
$\|\mathcal{H}\|_{\mathrm{sum}} = \sum_{i,j,k} |H_{ijk}|$,

$$\mathsf{D}_2 \equiv \{\mathcal{D} \in \mathbb{R}_+^{P\times s\times Q} \mid \|\mathcal{D}(:,i,:)\|_{\mathrm{F}} \leq \sqrt{pr},\ i = 1,\ldots,s\},$$

and

$$\mathsf{D}_\infty \equiv \{\mathcal{D} \in \mathbb{R}_+^{P\times s\times Q} \mid \|\mathcal{D}(:,i,:)\|_{\mathrm{F}} \leq 1,\ i = 1,\ldots,s\}.$$

The code accepts a range of different values for λ. This function can also run without any input parameter and with the default values defined in the code.

- **Dependencies**: `Lea_Algorithm_T_D2`, `Lea_Algorithm_T_Dinf`, `dykstra`, `tprod`, `tran`, `fronorm`
- **Usage**:

  `[D, H, Norm1H, Norm1HSlic, Density, NrNZero, t_end, output] = ...`
  `DicLear_T(TImage, patch_size, s, Lambda, DSet, t, Outdisplay)`
- **Inputs**:
  - `TImage`: The name of the test image, the test image options are: `Peppers`, `Matches`, `Binary`, `D53`, `Zirconium`, `Steel`
  - `patch_size`: The training patch size
  - `s`: Number of dictionary elements
  - `Lambda`: The regularization parameter, the code accepts a range of different values for λ.
  - `DSet`: The name of the compact and convex set which the dictionary belongs to, The options are: `D_inf` and `D_2`
  - `t`: Number of training patches
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default = 0.
- **Ouputs**:
  - `D`: The tensor dictionary of size: `patch_size(1)`$\times s\times$`patch_size(2)`.
  - `H`: The representation tensor of size: $s \times t\times$`patch_size(2)`.
  - `Norm1H`: $\|\mathcal{H}\|_{\text{sum}}$.
  - `Norm1HSlic`: $\|\mathcal{H}\|_{\text{sum}}/t$.
  - `Density`: The density percentage of the tensor $\mathcal{H}$.
  - `NrNZero`: The absolute number of nonzeros in the representation tensor $\mathcal{H}$.
  - `t_end`: Total time used by the ADMM algorithm to find a solution for a specific λ.
  - `output`: Structured output array of the following fields:
    - `Objective`: The objective function value for each iteration of the ADMM algorithm for each λ: $\frac{1}{2}\|\mathcal{Y} - \mathcal{D} * \mathcal{H}\|_{\text{F}}^2 + \lambda\|\mathcal{H}\|_{\text{sum}}$.
    - `Residual`: The residual value for each iteration of the ADMM algorithm: $\|\mathcal{Y} - \mathcal{D} * \mathcal{H}\|_{\text{F}}$.
    - `StopCr1`, `StopCr2`, `StopCr3`, `StopCr4`, `StopCr5`: The stopping criteria values at each iteration of the ADMM algorithm for each λ.
- **Example**:

```
TImage = 'Peppers';
patch_size = [5 5];
t = 10000;
s = 50;
```

```
Lambda = 1;
DSet = 'D_2';
Outdisplay = 1;
[D,H]=DicLear_T(TImage,patch_size„s,Lambda,DSet,t,Outdisplay);
```

2.4.3 `Lea_Algorithm_T_D2.m`

- **Description**: This function performs the ADMM algorithm in k iterations for the dictionary learning problem in the tensor form, for one value of λ, where the dictionary D belongs to the set D_2, `DSet = 'D_2'`.
- **Usage:**

  ```
  [W, H, Rs1, Rs2, Rs3, Rs4, Rs5, fobjec, ResF] = ...
  Lea_Algorithm_T_D2( X, tol, maxiter, s, Cof_Lambda, U, V, ...
  Cof_rho, Outdisplay );
  ```

- **Input:**
  - `X`: Training tensor of size: `patch_size(1)`$\times t\times$`patch_size(2)`.
  - `tol`: The tolerance for the ADMM convergence.
  - `maxiter`: total number of ADMM iterations
  - `s`: The number of dictionary elements.
  - `Cof_Lambda`: The regularization parameter λ in the dictionary learning problem formulation.
  - `U,V`: The initial value for the auxiliary tensor variables in the ADMM algorithm.
  - `Cof_rho`: The augmented Lagrangian parameter.
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default = 0.
- **Output:**
  - `W`: The dictionary tensor for the corresponding λ.
  - `H`: The representation tensor for the corresponding λ.
  - `fobjec`: The objective function value for each iteration of the ADMM algorithm: $\frac{1}{2}\|\mathcal{Y}-\mathcal{D}*\mathcal{H}\|_{\mathrm{F}}^2+\lambda\|\mathcal{H}\|_{\mathrm{sum}}$.
  - `ResF`: The residual value for each iteration of the ADMM algorithm: $\|\mathcal{Y}-\mathcal{D}*\mathcal{H}\|_{\mathrm{F}}$.
  - `Rs1, Rs2, Rs3, Rs4, Rs5`: The stopping criteria values at each iteration of the ADMM algorithm.

2.4.4 `Lea_Algorithm_T_Dinf.m`

- **Description**: This function performs the ADMM algorithm in k iterations for the dictionary learning problem in the tensor form, for one value of λ, where the dictionary D belongs to the set D_∞, `DSet = 'D_inf'`.

- **Usage:**

```
[W, H, Rs1, Rs2, Rs3, Rs4, Rs5, fobjec, ResF] = ...
Lea_Algorithm_T_Dinf( X, tol, maxiter, s, Cof_Lambda, U, V, ...
Cof_rho, Outdisplay );
```

- **Input, Output**: **See** `Lea_Algorithm_T_D2`

2.5 The Approximation Error by the Tensor Dictionary

`MAET_demo.m`	Demo script file illustrates how to compute the approximation error of a test problem in the cone defined by the given tensor dictionary.
`MAE_T.m`	Computes the approximation error for the tensor formulation.
`Linear_TenLSQ_BestRec`*	The linear operator for the approximation error problem that is used by TFOCS.
`tprod`*	Computes the tensor-tensor product using FFTs [3].
`tran`*	Returns the tensor transpose [3].

2.5.1 `MAET_demo.m`

- **Description**: This demo script file illustrates how to compute the approximation error of a test problem in the cone defined by the dictionary for the tensor formulation. This code handle various dictionaries for a range of values for λ.

- **Dependencies**: TFOCS Toolbox, `MAE_T.m`, `Linear_TenLSQ_BestRec`, `tprod`, `tran`

- **Limit**: This codes uses the TFOCS optimization solver version 1.3.1 (`http://cvxr.com/tfocs/`). Make sure TFOCS is properly installed on the computer.

- **Usage:** `MAET_demo`

2.5.2 `MAE_T.m`

- **Description**: This function computes the approximation error for the tensor formulation i.e., how well we can represent the exact image in the cone defined by the dictionary. To evaluate the approximation error, i.e., the distance of the exact image x^{exact} to its projection on the cone

$$\mathsf{G} = \{\mathcal{D} * \vec{\mathcal{Z}} \mid \vec{\mathcal{Z}} \in \mathbb{R}_+^{s\times 1\times Q}\} \subseteq \mathbb{R}_+^{p\times 1\times Q},$$

  we compute the solutions $\alpha_j^\star$ to the q approximation problems for all blocks $j = 1, 2, \ldots, q$ in x^{exact}. This code handle various dictionaries for a range of values for λ.

- **Dependencies**: TFOCS Toolbox, `Linear_TenLSQ_BestRec`, `tprod`, `tran`

- **Limit:** The test images are predefined, the user should modify the test images in the code for other test problem or image size. This code uses the TFOCS optimization solver (`http://cvxr.com/tfocs/`). Make sure TFOCS is properly installed on the computer.
- **Usage:**

  ```
  [Appr_Error_Mean, Appr_Error, Alph, Norm1Alph, SparsityAlph] =
  MAE_T(TrImage, D, Lambda, patch_size, Outdisplay);
  ```

- **Input:**
  - `TrImage`: The test image, options are: `'Peppers'`, `'Matches'`, `'Binary'`, `'Zirconium'`, `'Steel'`, `'D53'`.
  - `D`: The given (learned) tensor dictionary, note that the dictionary should be obtained from a training image similar to the test image.
  - `Lambda`: The regularization parameter in the dictionary learning problem formulation, note that it should be consistence with the dictionary D.
  - `patch_size`: The patch sizes of the dictionary elements, does not necessarily need to be provided, default = `[size(D,1),size(D,3)]`.
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 prints progress of the algorithm, default = 0.
- **Outputs:**
  - `Appr_Error_Mean`: Mean approximation error for each λ.
  - `Appr_Error`: The vector of approximation errors for each block in the image for each λ.
  - `Alph`: The best representation/approximation of the jth block in the cone ($\alpha_j^\star$) for each λ.
  - `Norm1Alph`: $\|\alpha_j^\star\|_1$ for each block j.
  - `SparsityAlph`: The sparsity percentage of the representation vector $\alpha_j^\star$ for each block.
- **Example:**

  ```
  TImage = 'Peppers';
  load('DicTensor.mat','D');
  Lambda = 1;
  patch_size = [5 5];
  Outdisplay = 1;
  [Appr_Error_Mean, Appr_Error, Alph] = ...
  MAE_T(TrImage, D, Lambda, patch_size, Outdisplay);
  ```

2.6 Tomographic Reconstruction with Tensor Dictionary

`RecT_demo`	Demo script for the tomographic reconstruction problem in the tensor form.
`RecT_Algorithm`	Solves the tomographic reconstruction problem in the tensor form for the asked test image by TFOCS.
`Perm_Vec_Tensor`*	Returns the permutation vector which gives a permutation to the solution.
`L_Matrix`*	Returns the matrix L, used to penalize the block artifacts in the reconstruction formulation.
`Linear_Opr_Tensor`*	Defines a linear operator for the tomographic reconstruction tensor formulation used by TFOCS.
`prox_nuclearposl1`*	Computes the prox operator for nuclear norm + l_1 norm regularization iteratively using a Dykstra-like proximal algorithm.
`tprod`*	Computes the tensor-tensor product using FFTs [3].
`tran`*	Returns the tensor transpose [3].

2.6.1 `RecT_demo.m`

- **Description**: Demo script for the tomographic reconstruction problem in the tensor form. This script illustrates the use of the dictionary learning approach in the discrete tomographic reconstruction problem using tensor definitions in [3]. Note that this script solves a large scale sparse approximation problem and many iterations are needed to converge to the solution and this is not an bug/error of the code.

- **Dependencies**: TFOCS Toolbox, `RecT_Algorithm`, `Perm_Vec_Tensor`, `L_Matrix`,
`Linear_Opr_Tensor`, `prox_nuclearposl1`, `tprod`, `tran`

- **Limit:** The test images are predefined, the user should modify the test images in the code for other test problem or image size. This code handles one dictionary for a specific λ and s. This codes uses the TFOCS optimization solver version 1.3.1 (`http://cvxr.com/tfocs/`)

- **Usage:** `RecT_demo`

2.6.2 `RecM_Algorithm.m`

- **Description**: This function solves the tomographic reconstruction problem in the tensor formulation for the given test problem by TFOCS. The problem is given by:

$$\begin{array}{ll} \min_{\mathcal{C}\in\mathbb{R}^{s\times q\times Q}} & \frac{1}{2m}\|A\Pi\mathtt{vec}(\mathcal{D}*\mathcal{C})-b\|_2^2+\mu\varphi_\nu\big(\mathcal{C}\big)+\delta^2\psi\big(\Pi\mathtt{vec}(\mathcal{D}*\mathcal{C})\big) \\ \text{s.t.} & \mathcal{C}\geq 0, \end{array} \tag{4}$$

with regularization parameters $\mu, \delta > 0$, where

$$\psi(z) = \frac{1}{M(M/P-1) + N(N/Q-1)} \frac{1}{2} \|Lz\|_2^2.$$

x is an $M \times N$ image, the patches of the dictionary are of size $P \times Q$ and q is the total number of non-overlapping patches in the image x. We consider two different ways to impose a sparsity prior on $\mathcal{C}$ in the form $\Phi_{\text{sp}}(\mathcal{C}) = \mu\, \varphi_\nu(\mathcal{C})$, $\nu = 1, 2$, where μ is a regularization parameter and

$$\varphi_1(\mathcal{C}) = \frac{1}{q}\|\mathcal{C}\|_{\text{sum}}, \qquad \varphi_2(\mathcal{C}) = \frac{1}{q}\big(\|\mathcal{C}\|_{\text{sum}} + \|C\|_*\big), \tag{5}$$

Let $\tau = \mu/q$. The code can handle a range of values for both τ and δ.

Note that this function solves a large scale sparse approximation problem and many iterations are needed to converge to the solution and this is not an bug/error of the code.

- **Dependencies**: TFOCS, `Perm_Vec_Tensor`, `L_Matrix`, `Linear_Opr_Tensor`, `prox_nuclearposl1`, `tprod`, `tran`
- **Limit:** The test images are predefined, the user should modify the test images in the code for other test problem or image size. This code handles one dictionary for a specific λ and s. The regularization parameters $\tau, \delta > 0$ are predefined in the code. The user can easily modify the code for other values of regularization parameter.
- **Usage:**

```
[TrI, X_sol, C, Error_Sol, tau, delta, Sparsity, Resudial_Fit, ...
AlphaNorm1,AlphaNorm1Avr, t_end] = ...
RecT_Algorithm(TrImage, D, N_p, Ra, rnl, geo, patch_size, ...
Reg, Outdisplay)
```

- **Input:**
  - `TrImage`: The test image, options are: `'Peppers'`, `'Matches'`, `'Binary'`, `'Zirconium'`, `'Steel'`, `'D53'`.
  - `D`: The given (learned) tensor dictionary, note that the dictionary should be obtained from a training image similar to the test image.
  - `N_p`: Number of tomographic projections.
  - `Ra`: Range of the angles for the tomographic projections, e.g., full-range $[0°, 180°]$, limited-angle $[0°, 120°]$.
  - `rnl`: Gaussian additive noise level in the tomographic data.
  - `geo`: Choose a parallel-beam CT geometry or a fan-beam CT geometry, options = `'fanbeam'` and `'parallelbeam'`, default value = `'parallelbeam'`.
  - `patch_size`: The patch sizes of the blocks in the image, note that the image size is a multiple of the patch size.
    Default = `[size(D,1),size(D,3)]`.

  - `Reg`: Regularization on the representation in the dictionary. Options are sparsity prior (l_1 norm regularization) and sparsity+low-rank (l_1+nuclear norm regularization), options are 'l1pos' and 'nuclearl1pos', default = 'l1pos'.
  - `Outdisplay`: The display option on the screen, when set to 0 no display on the screen, when set to 1 printes progress of the algorithm default = 0.

- **Output:**
  - `TrI`: The exact image considered in the tomographic problem. The size of the TrI is predefined.
  - `X_sol`: The tomographic reconstruction solution recovered from optimal $\mathcal{C}$.
  - `C`: The sparse representation/solution in the given dictionary obtained by solving our tomographic reconstruction problem.
  - `Error_Sol`: Relative reconstruction error.
  - `tau`: The sparsity regularization parameter.
  - `delta`: The block artifacts regularization parameter.
  - `Sparsity`: The sparsity percentage of the representation vector $\mathcal{C}$.
  - `Resudial_Fit`: The residual of the tomographic data fitting term.
  - `AlphaNorm1`: $\|\mathcal{C}\|_1$
  - `AlphaNorm1Avr`: $\|\mathcal{C}\|_1/q$
  - `t_end`: Total time used by the TFOCS to find the representation tensor ($\mathcal{C}$) for a specific regularization parameter (τ and δ).

- **Example:**

```
TImage = 'Peppers';
load('Dic.mat','D');
patch_size = [5 5];
Ra = 180;
N_p = 25;
rnl = 0.01;
geo = 'fanbeam';
Reg = 'l1pos';
Outdisplay = 1;
[TrI, X_sol, C, Error_Sol] = ...
RecT_Algorithm( TrImage, D, N_p, Ra, rnl, geo, patch_size, ...
Reg, Outdisplay );
```

3 Alphabetic List of Package Functions

`DicLear_M.m`	The function routine to learn a matrix dictionary with the given parameters.
`DicLear_T.m`	The function routine to learn a tensor dictionary with the given parameters.
`DL_demo.m`	Demo and example script for the dictionary learning (DL) problem in the matrix form.
`DL_Tensor_demo.m`	Demo script for the dictionary learning (DL) problem in the tensor form.
`dykstra.m`	Finds a point in the intersection of two convex sets by iteratively projecting onto each of the convex sets.
`fronorm`	Computes Frobenius norm of a tensor [3].
`L_Matrix`	Returns the matrix L, used to penalize the block artifacts in the reconstruction formulation.
`Lea_Algorithm_D2.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the Matrix form such that $D \in Ð_2$.
`Lea_Algorithm_Dinf.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the Matrix form such that $D \in Ð_\infty$.
`Lea_Algorithm_T_D2.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the tensor form such that $D \in \mathsf{D}_2$.
`Lea_Algorithm_T_Dinf.m`	Performs the ADMM algorithm in k iterations for the dictionary learning problem in the tensor form such $D \in \mathsf{D}_\infty$.
`Linear_Opr`	Defines a linear operator for the tomographic reconstruction matrix formulation used by TFOCS.
`Linear_Opr_Tensor`	Defines a linear operator for the tomographic reconstruction tensor formulation used by TFOCS.
`Linear_TenLSQ_BestRec`	The linear operator for the approximation error problem that is used by TFOCS.
`MAE_M.m`	Computes the approximation error for the matrix formulation.
`MAE_T.m`	Computes the approximation error for the tensor formulation.
`MAEM_demo.m`	Demo script file that illustrates how to compute the approximation error of a test problem in the cone defined by the given dictionary.
`MAET_demo.m`	Demo script file that illustrates how to compute the approximation error of a test problem in the cone defined by the given tensor dictionary.
`montageplot.m`	This function is a tool to illustrate the dictionary elements as images.

`Perm_Vec`	Returns the permutation vector which gives a permutation to the solution.
`Perm_Vec_Tensor`	Returns the permutation vector which gives a permutation to the solution.
`prox_nuclearposl1`	Computes the prox operator for nuclear norm + l_1 norm regularization iteratively using a Dykstra-like proximal algorithm.
`RecM_Algorithm`	Solves the tomographic reconstruction problem in the matrix form for the asked test image by TFOCS.
`RecM_demo`	Demo script for the tomographic reconstruction problem in the matrix form.
`RecT_Algorithm`	Solves the tomographic reconstruction problem in the tensor form for the asked test image by TFOCS.
`RecT_demo`	Demo script for the tomographic reconstruction problem in the tensor form.
`tprod`	Computes the tensor-tensor product using FFTs [3].
`tran`	Returns the tensor transpose [3].

References

[1] S. Becker, E. J. Candèes and M. Grant, *Templates for convex cone problems with applications to sparse signal recovery*, Mathematical Programming Computation, **3**, 165–218, 2011.

[2] S. Boyd, N. Parikh, E. Chu, B. Peleato, and J. Eckstein. *Distributed optimization and statistical learning via the alternating direction method of multipliers*, Foundations and Trends in Machine Learning, **3**(1), 1–122, 2010.

[3] M. E. Kilmer and C. D. Martin, *Factorization strategies for third-order tensors*, Linear Algebra and its Applications **435**, 641–658, 2011.

[4] P. C. Hansen and M. Saxild-Hansen, *AIR Tools - a Matlab package of algebraic iterative reconstruction methods*, Journal of Computational and Applied Mathematics, **236**, 2167–2178, 2012.

[5] S. Soltani, M. S. Andersen, and P. C. Hansen *Tomographic image reconstruction using Training Images*, `http://arxiv.org/abs/1503.01993`, submitted, 2015.

[6] S. Soltani, M. E. Kilmer, and P. C. Hansen, *A Tensor-Based Dictionary Learning Approach to Tomographic Image Reconstruction*, `http://arxiv.org/abs/1506.04954`, submitted, 2015.

[7] `http://multibandtexture.recherche.usherbrooke.ca/normalized_brodatz.html`

[8] `http://www.one-eighty-degrees.com/service/microstructural-investigations/`