The Matlab Curvelet Toolbox
Beta version

Laurent Demanet

August 31, 2004

1 Setup

Copy the archive file CurveletToolbox.tar in the directory of your choice, and expand it with the linux command

```
tar xvf CurveletToolbox.tar
```

This will create a new directory CurveletToolbox containing the toolbox matlab files. Don't forget to add the directory to your matlab path.

2 Specs

The new curvelet transform as implemented in the toolbox enjoys the following properties:

- Geometrically faithful to the continuous transform as explained in Candes and Donoho (2004).
- Numerically tight up to machine precision: as a result the pseudo-inverse is simply the adjoint.
- The algorithmic complexity is $O(N \log N)$.
- Comes as a complex-valued or real-valued transform.
- The coarsest scale subband consists of Meyer wavelets, all the other subbands consist of angularly selective curvelets.
- The redundancy is about 5 to 10.
- The input image can be a rectangle with arbitrary height and length.

- I also implemented a very redundant transform where each scale-angle subband has the same size as the original image.

The implementation is based on the idea of (1) decomposing the frequency plane into wedges by multiplying the Fourier transform of the data by appropriate windows, and (2) and applying a local slanted inverse 2-D Fourier transform on parallelograms containing these wedges. Parallelograms can be turned into rectangles by an adequate ‘wrapping’ technique. In particular, the cartesian grid is used throughout the computation and no interpolation is necessary in the frequency plane.

(All this will be expanded elsewhere)

3 How to use it

```
function C = FCT(x, is_real, nbscales, nbangles_coarse)

% FCT.m - Fast Curvelet Transform
%
% Inputs
%   x           M-by-N matrix
%
% Optional Inputs
%   is_real     forces a real-valued or complex-valued transform.
%                   0: complex-valued curvelets
%                   1: real-valued curvelets
%                   2: real-valued Wilson curvelets
%               [default set to 1 if x is real-valued, 0 otherwise]
%   nbscales    number of scales including the coarsest wavelet level
%               [default set to ceil(log2(min(M,N))/2)]
%   nbangles_coarse
%               number of angles at the 2nd coarsest level, minimum 8,
%               must be a multiple of 4. [default set to 16]
%
% Outputs
%   C           Cell array of curvelet coefficients: C{j}{l}(k1,k2) is
%               the coefficient at
%                   - scale j: integer, from finest to coarsest scale,
%                   - angle l: integer, starts at the top-left corner and
%                   increases clockwise,
%                   - position k1,k2: both integers, size varies with j
%                   and l.
%               C{end} is a vector containing additional information:
%                   size(x,1), size(x,2), is_real
```

```
function x = FICT(C)

% FICT.m - Fast Inverse Curvelet Transform
% This is in fact the adjoint, also the pseudo-inverse
%
% Inputs
%   C           Cell array containing curvelet coefficients (see
%               description in FCT.m)
%
% Outputs
%   x           M-by-N matrix



function [Angles, Xlocations, Ylocations] = GetParam(C)

% GetParam.m - Obtain the angle and location relative to each curvelet
% coefficient
%
% Inputs
%   C           Cell array C{j}{l}(k1,k2) containing curvelet
%               coefficients (see description in FCT.m)
%
% Outputs
%   Angles      Cell array Angles{j}(l) giving the angle, in degrees,
%               between the x-axis and the codirection of each curvelet
%               indexed by j and l (angle increases counterclockwise from 0
%               to 180, or from 0 to 360).
%   XLocations  Cell array XLocations{j}{l}(k1,k2) giving the x coordinate,
%               on the original M-by-N image grid, of the curvelet indexed
%               by j, l, k1 and k2 (x increases from left to right, from 1
%               to N).
%   YLocations  Same for the y coordinate (y increases from top to bottom,
%               from 1 to M).
%
% See also FCT.m, FICT.m



function C = RCT(x, is_real, nbscales, nbangles_coarse)

% RCT.m - Redundant Curvelet Transform
%
% Can be seen as a numerically invertible discretization of the Continuous
% Curvelet Transform on the rectangular grid of the original image.
```

```
%
% Inputs
%   x           M-by-N matrix
%
% Optional Inputs
%   is_real     forces a real-valued or complex-valued transform.
%                   0: complex-valued curvelets
%                   1: real-valued Wilson curvelets
%               [default set to 1 if x is real-valued, 0 otherwise]
%   nbscales    number of scales including the coarsest wavelet level
%               [default set to ceil(log2(min(M,N))/2)]
%   nbangles_coarse
%               number of angles at the 2nd coarsest level, minimum 8,
%               must be a multiple of 4. [default set to 16]
%
% Outputs
%   C           Cell array of curvelet coefficients: C{j}{l}(t,k1,k2) is
%               the coefficient at
%                   - scale j: integer, from finest to coarsest scale,
%                   - angle l: integer, starts at the top-left corner and
%                   increases clockwise,
%                   - type t: 1 for 'cosine', 2 for 'sine' curvelets,
%                   - position k1, k2: both integers, k1 takes on N values
%                   and k2 M values, independently of j and l.
%               C{end} is a vector containing additional information:
%                   [size(x,1), size(x,2), is_real]
%
% See also IRCT.m
```

```
function x = IRCT(C)

% IRCT.m - Inverse Redundant Curvelet Transform
%
% This is in fact the adjoint, also the pseudo-inverse
%
% Inputs
%   C           Cell array containing curvelet coefficients (see
%               description in RCT.m)
%
% Outputs
%   x           M-by-N matrix
%
% See also RCT.m
```

```
function D = SoftThresh(C,thresh,power)

% SoftThresh.m - Curvelet soft-thresholding
%
% Inputs
%   C           Cell array containing curvelet coefficients coming from
%               RCT.m, using the option is_real = 1.
%   thresh      Amount to which the curvelet shrinkage will be
%               proportional. The actual value of the threshold also
%               depends on the size of the original image (through Donoho's
%               sqrt(2*log(N))) and is proportional to the L^2 norm of each
%               tight frame element.
%   power       Scalar: in Fourier, Log(amplitude noise)/Log(radius). For white
%               noise, power = 0 [default]. For Radon noise, power = 1/2.
%
% Outputs
%   D           Cell array containing thresholded curvelet coefficients
%
% See also RCT.m, IRCT.m
```