Matlab implementation to illustrate central slice theorem and back projection using inverse radon transform

Nasser M. Abbasi

Nov 10,2008 Compiled on July 9, 2025 at 2:53am

Figure 1: Matlab implementation to illustrate central slice theorem

1 Introduction

Central slice theorem says that if we make a projection of a 2D image on a projection line, and take the 1D Fourier transform (say A) of the projection itself, and then take a slice (say B) from the 2D Fourier transform of the image itself, then A=B. When taking the slice from the 2D Fourier transform it has to be done on a slice through the center of the 2D spectrum, and along a line parallel to the projection line mentioned above.

This is a simple Matlab implementation to illustrate the above. It allows the user to select a number of images, and the angle of the projection and then it find the projection (using radon transform) and shows the 1D FT of this projection, and then it finds the 2D FT of the image itself.

The 2D FT is shown in a 3D view to allow the user to rotate it.

This program also illustrate back projection by allowing the user to select the number of degrees to project the image at, and then back project all the resulting projecting to

reconstruct the original image (as is done in CT scans). We see that the more angles used, the closer the resulting image will be to the original image.

The 2D FT of the reconstructed image is also shown as number of degrees is increased allowing one to see the slices being added and the final 2D FT convergence to the original image 2D FT.

Streaks in the image are noticed as it is being constructed. The streaks become less noticeable as more angles added. The inverse radon transform is used for back projection.

2 Animation

Here is a screen shot of the Matlab program.

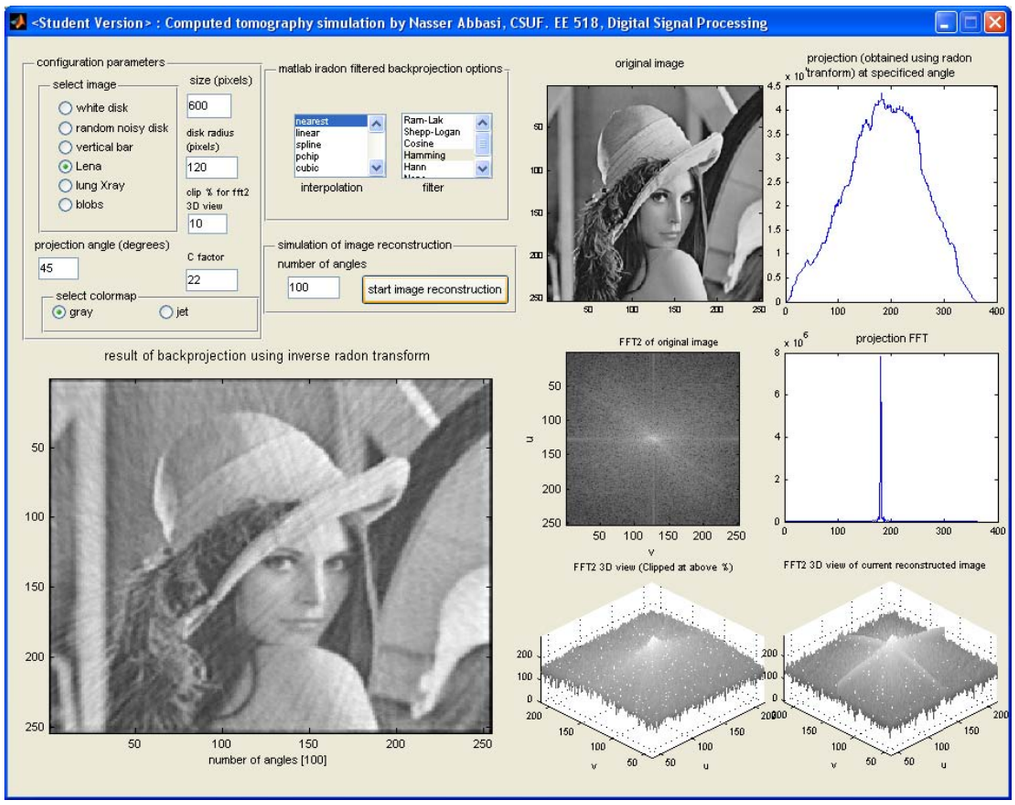

Figure 2: screen shot

3 source code

Current version is here zip file. This is a zip file which contains all the files needed. Download the zip file and extract it. It will creat a new folder. Simply add this folder to your MATLAB PATH, then from the Matlab console type the command

```
nma_projection
```

This will now bring up the GUI to use.

4 Soruce code listing

4.1 nma_projection.m

```
function varargout = nma_projection(varargin)
%
%  illustrate central slice theorem and reconstruction
%  of 2D images from backprojections using inverse radon transform
%
%  work related to Math 597 project, summer 2008
%  by Nasser Abbasi 6/2/08
%

% NMA_PROJECTION M-file for nma_projection.fig
%      NMA_PROJECTION, by itself, creates a new NMA_PROJECTION or raises the existing
%      singleton*.
%
%      H = NMA_PROJECTION returns the handle to a new NMA_PROJECTION or the handle to
%      the existing singleton*.
%
%      NMA_PROJECTION('CALLBACK',hObject,eventData,handles,...) calls the local
%      function named CALLBACK in NMA_PROJECTION.M with the given input arguments.
%
%      NMA_PROJECTION('Property','Value',...) creates a new NMA_PROJECTION or raises the
%      existing singleton*.  Starting from the left, property value pairs are
%      applied to the GUI before nma_projection_OpeningFunction gets called.  An
%      unrecognized property name or invalid value makes property application
%      stop.  All inputs are passed to nma_projection_OpeningFcn via varargin.
%
%      *See GUI Options on GUIDE's Tools menu.  Choose "GUI allows only one
%      instance to run (singleton)".
%
% See also: GUIDE, GUIDATA, GUIHANDLES

% Edit the above text to modify the response to help nma_projection

% Last Modified by GUIDE v2.5 14-Nov-2008 14:04:41

% Begin initialization code - DO NOT EDIT
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @nma_projection_OpeningFcn, ...
                   'gui_OutputFcn',  @nma_projection_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
end
```

```
% --- Executes just before nma_projection is made visible.
function nma_projection_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
% varargin   command line arguments to nma_projection (see VARARGIN)

% Choose default command line output for nma_projection
handles.output = hObject;

set(handles.groupBtnTag,'SelectionChangeFcn',...
    @groupBtnTag_SelectionChangeFcn);
set(handles.colormapTag,'SelectionChangeFcn',...
    @colormapTag_SelectionChangeFcn);

% Update handles structure
set(handles.figure1,'Name',...
    'Computed tomography simulation by Nasser Abbasi, CSUF. EE 518, Digital Signal Processing')
guidata(hObject, handles);

process(hObject,0);

end

% --- Outputs from this function are returned to the command line.
function varargout = nma_projection_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Get default command line output from handles structure
varargout{1} = handles.output;
end


%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%
function groupBtnTag_SelectionChangeFcn(hObject, eventdata)
process(hObject,0);
end

%%%%%%%%%%%%%%%%%%
%
%
%%%%%%%%%%%%%%%%%%%
function colormapTag_SelectionChangeFcn(hObject, eventdata)
handles   = guidata(hObject);
h         = get(handles.colormapTag);
hColorMap = h.SelectedObject;
switch get(hColorMap,'Tag')
    case 'jetTag'
        colormap(jet);
    case 'grayTag'
        colormap(gray);
```

```
end

end

%%%%%%%%%%
%
%%%%%%%%%%
function process(hObject,doSimulation)

%retrieve GUI data, i.e. the handles structure
handles = guidata(hObject);

nPixels  = 600;
radius   = 120;
c        = 22
clip     = 10;

h         = get(handles.groupBtnTag);
hselected = h.SelectedObject;

%Now obtain colormap
h          = get(handles.colormapTag);
hColorMap  = h.SelectedObject;
switch get(hColorMap,'Tag')
    case 'jetTag'
        colormap(jet);
    case 'grayTag'
        colormap(gray);
end

%Now obtain the angle of projection and display projection
h     = get(handles.angleTag);
angle = str2double(h.String);

switch get(hselected,'Tag')   % Get Tag of selected object
    case 'blackDiskTag'
        A = makeDisk(0,nPixels,radius);
        p = radon(A,angle);

   case 'verticalBarTag'
        [A,p] = makeVerticalBar(nPixels);

    case 'randomDiskTag'
        A = makeDisk(1,nPixels,radius);
        p = radon(A,angle);

    case 'lenaTag'
        fileName = 'lena.jpg';
        A        = imread(fileName,'jpg');
        p        = radon(A,angle);

    case 'lungTag'
        fileName = 'lung.jpg';
        A        = imread(fileName,'jpg');
        [nRow,nCol] = size(A);
        d = min(nRow,nCol);
        A = imresize(A, [d d]);
        p = radon(A,angle);
```

```
     case 'blobsTag'
        fileName     = 'blobs.gif';
        A            = imread(fileName,'gif');
        [nRow,nCol]  = size(A);
        d            = min(nRow,nCol);
        A            = imresize(A, [d d]);
        p            = radon(A,angle);

    otherwise
        % Code for when there is no match.
end

axes(handles.originalImageAxes);
imagesc(A);
axis(handles.originalImageAxes,'image');

%now display projection.
axes(handles.projectionAxes);
stairs(p);

%now make fft of projection
axes(handles.normalProjectionTransformAxes);
YProjection = fft(p);
YshiftedProjection = fftshift(YProjection);
plot(abs(YshiftedProjection));

%now make 2D fft of original image
Y2D = fft2(double(A));
Y2Dshifted = abs(fftshift(Y2D));
axes(handles.twoDTransformAxes);
imagesc(c*log(1+Y2Dshifted));

xlabel('v'); ylabel('u');
axis(handles.twoDTransformAxes,'image');

%
%display the 2D spectrum on 3D
make3dspectrum(handles.FFT2on3DAxes,abs(Y2Dshifted),clip,c);

if doSimulation
    backProjection(hObject,A,clip,c);
end

end
%%%%%%%%%%
%
%%%%%%%%%%
function disk=makeDisk(isRandom,nPixels,r)
BLACK =0;
WHITE =255;

disk    = zeros(nPixels);
xCenter = ceil(nPixels/2);
yCenter = ceil(nPixels/2);

for i=1:nPixels
    for j = 1:nPixels
```

```
        xReal = i-xCenter;
        yReal = j-yCenter;
        distance = sqrt(xReal^2+yReal^2);
        if distance>r
           disk(i,j) = BLACK;
        else
           if isRandom
              disk(i,j) = WHITE*rand;
           else
              disk(i,j) = WHITE;
           end
        end
    end
end
end

%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%%
function angleTag_Callback(hObject, eventdata, handles)
% hObject    handle to angleTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of angleTag as text
%        str2double(get(hObject,'String')) returns contents of angleTag as a double

process(hObject,0);
end

%%%%%%%%%%%%%%%%%%
%
%%%%%%%%%%%%%%%%%%

% --- Executes during object creation, after setting all properties.
function angleTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to angleTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

%%%%%%%%%%
%
%%%%%%%%%%
function [A,p]=makeVerticalBar(nPixels)
A=[0 0 1 0 0;
   0 0 1 0 0;
   0 0 1 0 0;
   0 0 1 0 0;
   0 0 1 0 0];

p=[1;
```

```
    1;
    1;
    1;
    1];
end

%%%%%%%%%%%%
%
%%%%%%%%%%%%%
function nAnglesTag_Callback(hObject, eventdata, handles)
% hObject    handle to nAnglesTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: get(hObject,'String') returns contents of nAnglesTag as text
%        str2double(get(hObject,'String')) returns contents of nAnglesTag as a double

end

% --- Executes during object creation, after setting all properties.
function nAnglesTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to nAnglesTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: edit controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on button press in BtnTag.
function BtnTag_Callback(hObject, eventdata, handles)
% hObject    handle to BtnTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

%Now obtain how many angles to use for backprojection
process(hObject,1);
end
%%%%%%%%%%
%
%%%%%%%%%%%
function backProjection(hObject,A,clip,c)

%retrieve GUI data, i.e. the handles structure
handles = guidata(hObject);

%Now obtain how many angles to use for backprojection
h       = get(handles.nAnglesTag);
nAngles = str2double(h.String);

angles = zeros(nAngles,1);
delta=180/(nAngles+1)
for i=1:nAngles
    angles(i)=delta*i;
end
```

```
R      = radon(A,angles);

[the_intepolation,the_filter] = get_iradon_options(handles);

for i = 1:nAngles
    I = iradon(R(:,1:i),angles(1:i),the_intepolation,the_filter);
    axes(handles.radonSimAxes);
    imagesc(I);

    xlabel(sprintf('number of angles [%d]',i));

    make3dspectrum(handles.FFT2on3DcurrentAxes,abs(fftshift(fft2(I))),clip,c)
end

end


% --- Executes on selection change in iradonInterpTag.
function iradonInterpTag_Callback(hObject, eventdata, handles)
% hObject    handle to iradonInterpTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns iradonInterpTag contents as cell array
%        contents{get(hObject,'Value')} returns selected item from iradonInterpTag
end

% --- Executes during object creation, after setting all properties.
function iradonInterpTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to iradonInterpTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end

% --- Executes on selection change in iradonFilterTag.
function iradonFilterTag_Callback(hObject, eventdata, handles)
% hObject    handle to iradonFilterTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)

% Hints: contents = get(hObject,'String') returns iradonFilterTag contents as cell array
%        contents{get(hObject,'Value')} returns selected item from iradonFilterTag
end

% --- Executes during object creation, after setting all properties.
function iradonFilterTag_CreateFcn(hObject, eventdata, handles)
% hObject    handle to iradonFilterTag (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    empty - handles not created until after all CreateFcns called

% Hint: listbox controls usually have a white background on Windows.
```

```
%       See ISPC and COMPUTER.
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
```

4.2 nma_build.m

```
function nma_build_HW5()

list = dir('*.m');

if isempty(list)
    fprintf('no matlab files found\n');
    return
end

for i=1:length(list)
    name=list(i).name;
    fprintf('processing %s\n',name)
    p0 = fdep(list(i).name,'-q');
    [pathstr, name_of_matlab_function, ext] = fileparts(name);

    %make a zip file of the m file and any of its dependency
    p1=dir([name_of_matlab_function '.fig']);
    if length(p1)==1
        files_to_zip =[p1(1).name;p0.fun];
    else
        files_to_zip =p0.fun;
    end

    zip([name_of_matlab_function '.zip'],files_to_zip)

end

end
```

4.3 make3dspectrum.m

```
%*****************************************************
%       Copyright (C) 2008 Nasser Abbasi
%  Free to use and modify for academic and research only
%*****************************************************

function make3dspectrum(axesHandle,F,clip,c)

axes(axesHandle);
L      = size(F,1);
[i,j]  = find(F>(clip/100)*(max(max(F))));
d      = F;
d(sub2ind(size(F),i,j)) = 0;
r      = round(.15*L);
extent = r:L-r;
mesh(extent,extent,c*log(1+d(extent,extent)));
axis('tight');
```

```
shading interp;
xlabel('u');
ylabel('v');
view(-45,50);
end
```

4.4 get_iradon_options.m

```
%*******************************************************
%       Copyright (C) 2008 Nasser Abbasi
%  Free to use and modify for academic and research only
%*******************************************************
function [the_intepolation,the_filter] = get_iradon_options(client_handles)

k = get(client_handles.iradonFilterTag,'Value');
switch k
    case 1
        the_filter = 'Ram-Lak';
    case 2
        the_filter = 'Shepp-Logan';
    case 3
        the_filter =  'Cosine' ;
    case 4
        the_filter = 'Hamming' ;
    case 5
        the_filter = 'Hann' ;
    case 6
        the_filter = 'None';
end

k = get(client_handles.iradonInterpTag,'Value');
switch k
    case 1
        the_intepolation = 'nearest';
    case 2
        the_intepolation = 'linear';
    case 3
        the_intepolation =  'spline' ;
    case 4
        the_intepolation= 'pchip' ;
    case 5
        the_intepolation = 'cubic' ;
end
```