Bài thí nghiệm 2:

# Mô phỏng sự tạo ảnh cắt lớp điện toán truyền qua (Transmission Computer Tomography)

## 1. Yêu cầu

Chẩn đoán hình ảnh bằng phương pháp cắt lớp điện toán (Computer Tomography) một trong những kỹ thuật hiện đại được sử dụng phổ biến trong y học và cả trong kỹ thuật công nghệ cao như kính hiển vi cắt lớp khảo sát các vi cấu trúc vật liệu chẳng hạn. Nguyên lý tái tạo ảnh cắt lớp là một ứng dụng hữu hiệu của lý thuyết toán học và thành tựu của công nghệ thông tin trên cơ sở sự hoàn thiện với tốc độ cao về tính năng của máy tính. Chính vì sự quan trọng của nguyên lý tái tạo ảnh cắt lớp ứng dụng trong nhiều kỹ thuật chẩn đoán hình ảnh y học (CT scanner, MRI, siêu âm cắt lớp và optical tomography trong thời gian gần đây), bài thí nghiệm này được xây dựng nhằm giúp sinh viên hiểu rõ hơn về bản chất của vấn đề, các công đoạn cụ thể của quá trình tái tạo ảnh cắt lớp và một số vấn đề phổ dụng trong xử lý ảnh y học. Với mục tiêu chung đó cũng như với thời gian và công cụ cho phép, bài thí nghiệm được thiết kế chỉ cho *phương pháp cắt lớp điện toán truyền qua* nhằm đáp ứng các yêu cầu cụ thể sau:

- Hiểu rõ bản chất vật lý của phương pháp tạo ảnh cắt lớp.
- Tìm hiểu các thuật toán toán học trong phép mô phỏng tín hiệu các góc chiếu (sinogram) – qua phép biến đổi radon – và sự tái tạo ảnh lớp cắt – qua phép biến đổi ngược back projection; cũng như nguyên lý các bộ lọc dùng để xử lý ảnh.
- Thực hành cơ bản cách vận hành các lý thuyết trên trong bài toán mô phỏng sự tạo ảnh cắt lớp truyền qua bằng cách lập trình trong MATLAB với 2 cách: tự lập trình bằng lý thuyết và sử dụng Image Processing Toolbox của MATLAB.

## 2. Điều kiện và trang thiết bị

1) Sinh viên cần có kiến thức tối thiểu về lập trình cơ bản và đồ hoạ trong MATLAB

2) Máy tính có cấu hình đủ mạnh để chạy được chương trình MATLAB 6 trở lên, có cài đặt Image Processing Toolbox.

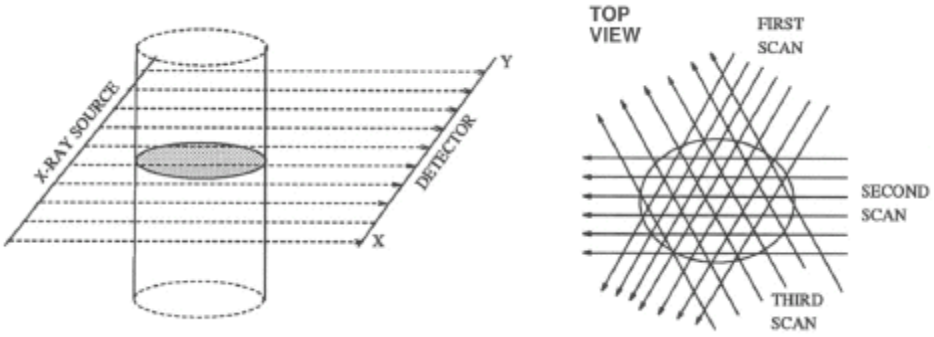
## 3. Tóm tắt lý thuyết

1) Nguyên lý hoạt động của CT scanner - thiết bị sử dụng phương pháp cắt lớp truyền qua [Avinash, Wolbarst]

- Nguyên lý hoạt động của CT là đo sự phân bố không gian của các vật chất trong cơ thể cần được xem xét từ các hướng khác nhau và tính toán các hình ảnh chồng chất từ dữ liệu này. Đây là kỹ thuật chụp ảnh bằng tia X qua đó một mặt phẳng đơn của một bệnh nhân được quét từ các góc khác nhau để cung cấp một hình ảnh mặt cắt của cấu trúc bên trong cơ thể thuộc mặt phẳng đó.

- Nguyên lý của CT là phải có được các số đo suy giảm qua mặt phẳng của một mặt cắt có độ dày xác định của bệnh nhân. Một nguồn X quang được sử dụng để quét bệnh nhân dọc theo mặt phẳng này, trong khi đó đầu dò ở phía đối diện đo tia X bị suy giảm dọc theo mặt phẳng này và máy tính ghi nhận các số liệu này – do đó phương pháp được gọi là *truyền qua* (transmission). Một khi bệnh nhân đã được quét từ phía bên này đến phía bên kia của mặt phẳng, cả nguồn X quang và đầu dò quay xung quanh bệnh nhân bởi một giá trị đã được định trước và quá trình quét tịnh tiến được lặp lại.

- Các thành phần bên trong của bệnh nhân được phân tích bởi máy vi tính dưới dạng một nhóm các khối thể tích nhỏ. Mỗi khối thể tích nhỏ này có hệ số suy giảm khối riêng của nó và được gọi là voxel. Voxel càng nhỏ thì độ phân giải của hình ảnh càng cao.




- Để tạo ra hình ảnh của một mặt cắt, máy tính phải tính toán hệ số suy giảm khối trung bình μ của mỗi voxel. Các hệ số này có thể được xác định theo phương pháp đại số với một lượng rất lớn phương trình. Tuy nhiên, một phương pháp đơn giản hơn được gọi là phương pháp "Filtered Back Projection (hình chiếu ngược có lọc)" đã được sử dụng phổ biến trong các máy CT ngày nay. Tia X quét ngang được thu thập thành tập hợp hình chiếu được thực hiện xuyên qua bệnh nhân theo một hướng cụ thể nào đó trong mặt phẳng cắt.

- Để tái tạo lại hình ảnh từ các tín hiệu tia X truyền qua, mỗi voxel phải được chiếu từ nhiều hướng khác nhau. Một tập hợp dữ liệu đầy đủ cần đến nhiều hình chiếu ở các góc quay khác nhau xung quanh mặt cắt.

Tóm lại, quá trình quét CT để tạo ra hình ảnh các lớp cắt trong một bệnh nhân bao gồm ba bước chính sau đây :

**THU NHẬN HÌNH CHIẾU → TÁI TẠO LỚP CẮT → HIỂN THỊ HÌNH ẢNH**

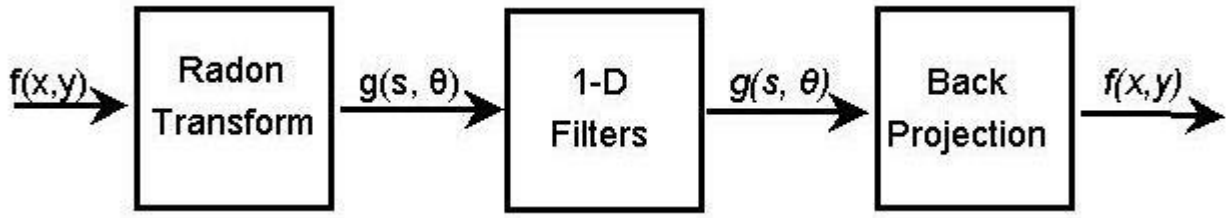
2) Các thuật toán [Avinash, Rosenberg ] :

Trình tự mô phỏng tạo ảnh cắt lớp bao gồm các bước sau :

a) Chiếu một lớp cắt (2-D) xuống một phương xác định theo một góc nào đó tạo ra một hình chiếu 1-D → Thu nhận một tập hợp hình chiếu 1-D

b) Mỗi hình chiếu được xử lý qua bộ lọc

c) Các hình chiếu đã lọc sẽ được chiếu ngược và tổng hợp đồng thời tạo ra ảnh lớp cắt gốc



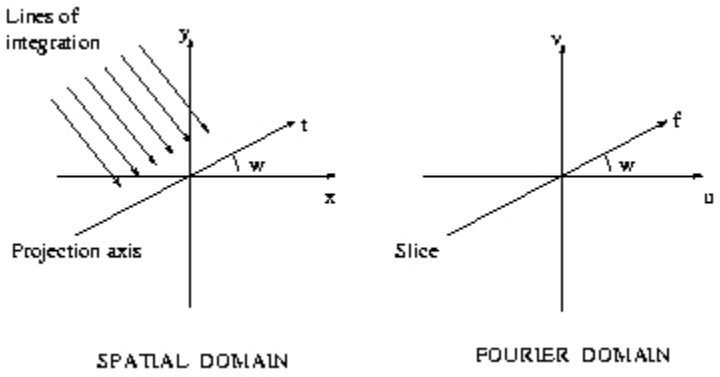

a) Thu nhận tập hợp hình chiếu 1-D: trong thực tế, các hình chiếu này là tín hiệu các đầu dò thu nhận được ứng với mỗi lần chiếu ở một góc quay nào đó. Trong mô phỏng, ta sử dụng phép biến đổi Radon. Phép biến đổi Radon của một hàm hai biến nào đó được định nghĩa như tích phân đường của nó dọc theo một phương tạo một góc ϕ so với trục y và cách gốc toạ độ một khoảng s. Biểu thức của phép biến đổi có dạng sau:

$$g(\phi,s) = \iint f(x,y)\delta(x\sin\phi - y\cos\phi - s)\,dxdy$$

với hàm delta δ(z) được định nghĩa như sau: $\delta(z) = \begin{cases} 1 & , z = 0 \\ 0 & , z \neq 0 \end{cases}$

Qua phép biến đổi Radon ứng với N góc khác nhau, ta thu nhận được tập hợp N hình chiếu ký hiệu *g(ϕ, s)*. Đồ thị tập hợp hình chiếu đó gọi là sinogram.

b) Tái tạo ảnh cắt lớp: sử dụng phương pháp chiếu ngược có lọc (Filtered Back Projection). Phương pháp này được xây dựng trên cơ sở *định lý chiếu lớp Fourier* (Fourier Slice Theorem) được hiểu như sau [Avinash]:




- Gọi *G(ϕ, ω)* là hàm biến đổi Fourier của hình chiếu *g(s, phi)* :

$$G(\phi,\omega) = \int e^{-j\omega s} g(\phi,s)\,ds$$

- Theo định lý *chiếu lớp Fourier* , *G(ϕ, ω)* đồng thời cũng là hàm biến đổi 2-D *F(u,v)* của *f(x,y)* với sự đổi biến *u = ω*sin(ϕ) , v = -ω*cos(ϕ).*

- Sự tái tạo ảnh lớp cắt được thực hiện bằng phép biến đổi Fourier 2-D ngược *có lọc* qua biểu thức sau:

$$f(x,y) = \frac{1}{4\pi^2}\iint G(\phi,\omega)e^{j\omega(x\sin\phi - y\cos\phi)}|\omega|\,d\omega\,d\phi$$

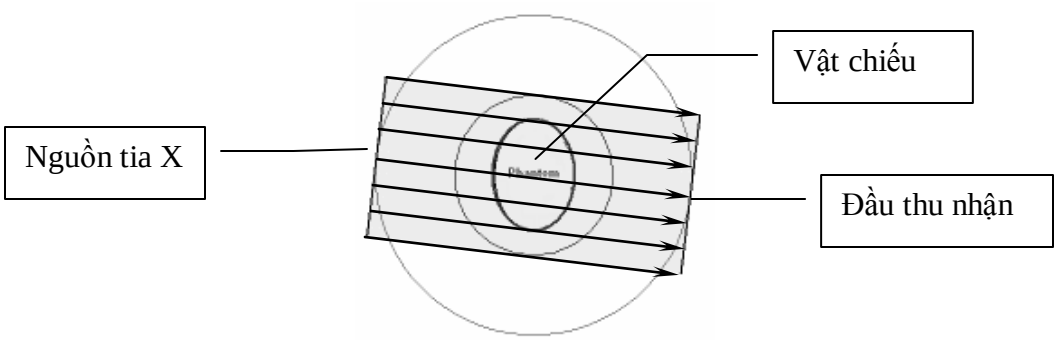
Lưu ý hình chiếu được nhân với hàm |ω| trong miền Fourier, điều đó tương đương với tích chập tương ứng trong miền không gian (x, y) – ý nghĩa bộ lọc. Ý tưởng của bộ lọc xuất phát từ các tần số thu nhận được cũng như tính phức của phép biến đổi Fourier. Ví dụ: Nếu có bộ lọc ưu tiên cho miền tần số cao, hình tái tạo rõ nét những nhiều nhiễu và ngược lại nếu bộ lọc ưu tiên cho miền tần số thấp, nhiễu giảm nhưng hình tái tạo bị mờ.Vấn đề quan trọng là áp dụng bộ lọc dạng nào cho phù hợp. Đó cũng là mục tiêu khảo sát trong bài thí nghiệm.

4. Mô tả chương trình mô phỏng sự tạo ảnh cắt lớp CT

Chương trình mô phỏng sự tạo ảnh cắt lớp với mục đích tái hiện các cơ sở lý thuyết của các thuật toán cắt lớp và xử lý ảnh tái tạo trên cơ sở các sinogram được mô phỏng. Trong phần mô tả này, chương trình thực hiện các bước cơ bản tạo ra một công cụ hiệu quả giúp các sinh viên hiểu rõ nguyên lý thu nhận tái tạo ảnh và thực hành một số kỹ năng xử lý ảnh cắt lớp.

Chương trình được viết bằng MATLAB nhằm giới thiệu sinh viên làm quen sử dụng một công cụ lập trình kỹ thuật đắc lực với nguồn tài nguyên thư viện phong phú trong nhiều lĩnh vực. Phần mã nguồn dưới đây là những phần cơ bản của một chương trình đã được thử nghiệm có kết quả và chỉ mang tính chất minh họa. Sinh viên hòan tòan có thể chỉnh sửa tùy theo thiết kế riêng và mức độ thành thục trong khả năng lập trình.

Chương trình mô tả dưới đây mô phỏng sự tạo ảnh cắt lớp với cấu hình đơn giản nhất: nguồn tia X là nguồn rộng với tia song song bao trùm cả kích thước ngang của mặt cắt.




Chương trình bao gồm 2 chương trình con:

- BP1.m : chương trình mô phỏng tạo ảnh cắt lớp bằng lập trình các thuật toán biến đổi theo lý thuyết.
- BP2.m : chương trình mô phỏng tạo ảnh cắt lớp bằng các lệnh biến đổi có sẵn trong thư viện Image Processing Toolbox của MATLAB.

Cả hai chương trình bao gồm 4 phần:

- Phần 1: Đọc ảnh vào bộ nhớ của MATLAB. Lưu ý câu lệnh có ký tự ”%” phía trước trong MATLAB là câu lệnh trơ, có thể dùng ký tự này để tạm bỏ những câu lệnh chức năng không dùng đến mà không phải xoá chúng.

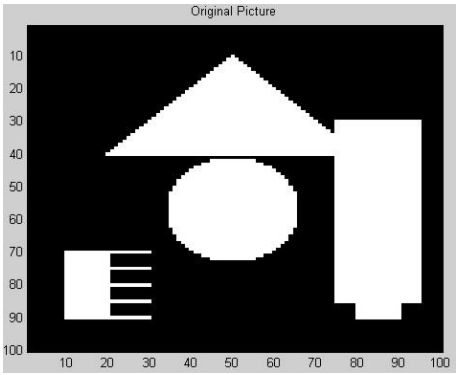
```
% BP1.m
% TOMOGRAPHY SIMULATION BY BP ALGORITHM PROGRAMMING
% Usage : BP1(IMAGE, N)
%
% IMAGE - a picture loaded into some matric in Matlab
% N - the number of projections you want taken between 0 and 180

function BP1 = BP1(image,N)

%  Load the image into Matlab
disp('Loading Image into Matlab...');
 image; %Goi chuong trinh tao anh trong Matlab
%[image,MAP,out]=bmpread('logan.bmp'); %Doc file BMP 16bit
%[image,MAP]=bmpread('CT1.bmp'); %Doc file BMP 8bit
```

Ảnh có thể được tạo bằng lệnh của MATLAB, ví dụ:

```
% picstandard.m
```

```
%
% Usage : picstandard;
%
% Here we make a picture (100X100) with various shapes

function picstandard = picture2()

picstandard = zeros(100,100);

x1 = 1:100;
y1 = 1:100;
[x,y] = meshgrid(x1,y1);

%  Rectangle on left
 picstandard(70:90, 10:20) = ones(21,11);
 picstandard(70:90, 80:90) = ones(21,11);

%  Rectangle on Right
 picstandard(30:85, 75:95) = ones(56,21);

%  Thin Horiz. Line
 picstandard(70,10:30) = ones(1,21);
 picstandard(75,10:30) = ones(1,21);
 picstandard(80,10:30) = ones(1,21);
 picstandard(85,10:30) = ones(1,21);
 picstandard(85,10:30) = ones(1,21);
 picstandard(90,10:30) = ones(1,21);

%  Circle
 p = (x-50).^2 + (y-57).^2;
 size = 250;
 picstandard(find(p<=size)) = ones(1,length(find(p<=size)));

%  Triangle
 i = 0;
 n = 30;
 picstandard(10,50) = ones(1,1);
 for i = 1:n,
     picstandard(i+10,50-i: 50+i) = ones(1,2*i+1);
 end;
```

Đoạn mã trên tạo ảnh 100x100 pixels như sau:




hoặc bằng hàm `bmpread` dùng để đọc file ảnh .BMP 100x100 bất kỳ.

```
function [X,map,out3]=bmpread(filename);
%BMPREAD Read a BMP (Microsoft Windows Bitmap) file from disk.
%   [X,MAP]=BMPREAD('filename') reads the 8-bit BMP file and
%   returns the indexed image X and associated colormap
%   MAP. If no extension is given for the filename, the
%   extension '.bmp' is assumed.
%   [X,MAP,OUT]=BMPREAD('filename') reads the 24-bit BMP file
%   from the file 'filename'.
%   BPP=BMPREAD('filename') returns the number of bits per
%   pixel in the BMP file
```

Ví dụ:

```
Hình trái:     [image,MAP,out]=bmpread('logan.bmp'); %Doc file BMP 16bit
Hình phải:     [image,MAP]=bmpread('CT1.bmp'); %Doc file BMP 8bit
```

- Phần 2: Hiển thị ảnh gốc ở góc phần tư thứ nhất của trang kết quả.

```
%  Plot the original image on subplot #1
 figure(1);
 colormap(gray(256));
 subplot(221);
 imagesc(image);
 title('Original Picture');
 disp('Plotting original image...');
```

- Phần 3: Dùng phép biến đổi radon để tạo sinogram các hình chiếu và hiển thị ở góc phần tư thứ hai của trang kết quả.

** Trong BP1:*

```
%  Take the Projections of the image where 'count' is the number
%  of intervals for theta between 1 and 180
 interval = linspace(1, 180, N);
 Proj = proj(image, N);
 disp('Calculating projections of the image...');
 subplot(222);
 imagesc(Proj);
 title('Sinogram - Plot of unchanged projections');
```

```
%  Add Noise to projections
```

- Thêm nhiễu ngẫu nhiên cho ảnh (nếu muốn).

```
disp('Adding noise to each projection...');
X = randn(144);
for i = 1:N
     Proj(:,i)= Proj(:,i) +(2.* X(:,i));
end
```

Đoạn mã trên gọi hàm hình chiếu `proj(image, N)` thông qua thuật toán tính tích phân đường theo một góc nào đó.

```
% This MATLAB function takes an image matrix and vector of angles and then
% finds the 1D projection (Radon transform) at each of the angles.  It
% returns a matrix whose columns are the projections at each angle.

function Projection = proj(IMG, N)

%  Create a vector of values for theta
 interval = linspace(1, 180, N);

%  Take images and pad with zero to eliminate error
 [iLength, iWidth] = size(IMG);
 iDiag = sqrt(iLength^2 + iWidth^2);
 LengthPad = ceil(iDiag - iLength) + 2;
 WidthPad = ceil(iDiag - iWidth) + 2;
 padIMG = zeros(iLength+LengthPad, iWidth+WidthPad);

 padIMG(ceil(LengthPad/2):(ceil(LengthPad/2)+iLength-1), ...
       ceil(WidthPad/2):(ceil(WidthPad/2)+iWidth-1)) = IMG;

%  Here we rotate the image N different angles of theta, and then sum the
columns
%    of each of these projected images in order to build N different
projections
 n = length(interval);
 Projection = zeros(size(padIMG,2), n);
 for i = 1:n
    temp = imrotate(padIMG, 90-interval(i), 'bilinear', 'crop');
    Projection(:,i) = (sum(temp))';
 end
```

* *Trong BP2:* sử dụng thẳng hàm `radon()` của MATLAB

```
%  Take the Projections of the image where 'count' is the number
%  of intervals for theta between 1 and 180
nn = 180/N;
theta1 = 0:nn:nn*N; [proj,xp] = radon(image,theta1);
disp('Calculating projections of the image...');
subplot(222);
imagesc(theta1,xp,proj);
title('Sinogram - Plot of unchanged projections');
```

- Phần 4: Dùng phép biến đổi chiếu ngược để tái tạo ảnh mặt cắt và hiển thị ở góc phần tư thứ 3 và thứ 4 của trang kết quả.

* *Trong BP1:*

```
%   Using the projections from above we can back-project
%   the image using our backprojection algorithm
 disp('Reconstructing the image from back-projection...');
 BP = backuf(Proj, interval); % Hàm chiếu ngược không lọc
 subplot(223);
 BP = BP(22:121,22:121);
```

```
 imagesc(BP);
 title('Unfiltered Back-Projected Image');

%   Using the projections from above we can now filter and back-project
%   the image using our backprojection algorithm
 disp('Reconstructing the image from back-projection...');
 BP = back(Proj, interval);  % Hàm chiếu ngược có lọc
 subplot(224);
 BP = BP(22:121,22:121);
 imagesc(BP);
 title('Filtered Back-Projected Image');
```

Trong đoạn mã trên có sử dụng hàm chiếu ngược `back` hoặc `backuf` lập trình theo thuật toán:

```
function BackI = back(PR, THETA)

% Determine the size of the projected, PR, image
n = size(PR,1);
sideSize = n;

% * * * FILTER THE PROJECTIONS * * *

%  No Filter applied
% filtPR = PR;

%  Ramp Filter with Cutoff
filtPR = projfilter(PR);

%  Convert THETA to radians and subtract pi so the reconstructed image
%  has the same orientation
 th = pi - ((pi/180)*THETA);

%  Prepare image
 m = length(THETA);
 BackI = zeros(sideSize,sideSize);

%  Find the middle index of the projections
 midindex = (n+1)/2;

%  Create x and y matrices
 x = 1:sideSize;
 y = 1:sideSize;
 [X,Y] = meshgrid(x,y);
 xpr = X - (sideSize+1)/2;
 ypr = Y - (sideSize+1)/2;

 for i = 1:m
    % Use the backprojection algorithm to determine which areas
    % on the projected images add up
    filtIndex = round(midindex + xpr*sin(th(i)) - ypr*cos(th(i)));

    % While "in bounds" then add the point
    BackIa = zeros(sideSize,sideSize);
    spota = find((filtIndex > 0) & (filtIndex <= n));
    newfiltIndex = filtIndex(spota);
    BackIa(spota) = filtPR(newfiltIndex(:),i);
```

```
    BackI = BackI + BackIa;
 end

BackI = BackI./m;
```

Và hàm lọc cho hình chiếu `projfilter` như sau:

```
% The code below takes a matrix of projections and returns
% a matrix with filtered projections whose Fourier Transform
% is |w|.  By uncommenting the vairous filters, various aspects
% of the reconstructed picture will be emphasized or lost.

function FIL = projfilter(Image)

[L,C]=size(Image);

% - - - - - - Filters - - - - - - - -

%  Uncomment the different filters below to change which one is
%  applied to the projections we calculate in proj.m

w = [-pi : (2*pi)/L : pi-(2*pi)/L];

% Ramp filter (Linear High-Pass filter)
 Filt = fftshift(abs(w));

% The opposite of the Ramp Function (Linear Low-Pass Filter)
%Filt = (abs(w));

% IIR Filter
%a=[1,-.9];
%b=[1];

% FIR-Box car-length 32
%a=1;
%b=[1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1];

% Butterworth Filter (Low-Pass Filter)
%fcutoff = 1000;
%Filt=3*10^5*((1+.414.*(fcutoff./(abs(w))).^2).^-1);

% Sinusoidal Filter (peak at 25)
% Filt = abs(sin(w));

% Stretched Sinusoidal Filter (peak at 42)
%L = 170;
%w = [-pi : (2*pi)/L : pi-(2*pi)/L];
%Filt = abs(sin(w));
%Filt = Filt(1:144);
%Filt = Filt .* 3.5;

% - - - - - - - - - - - - - - - - -

% Below we use the Fourier Slice Theorem to filter the image
for i = 1:C,
    IMG = fft(Image(:,i));
```

```
    FiltIMG = IMG.*Filt';
    %FiltIMG = filter (b, a, IMG);
    FIL(:,i) = ifft(FiltIMG);
end

% Remove any remaining imaginary parts
FIL = real(FIL);
```

* *Trong BP2:* sử dụng thẳng hàm ngược `iradon()` của MATLAB

```
%   Using the projections from above we can back-project
%   the image using our backprojection algorithm
disp('Reconstructing the image from back-projection...');
BP = iradon(proj,theta1);
subplot(223);
imagesc(BP);
title('BP Image: Ram-Lak filter, linear interp.');



%  Add Noise to projections
% disp('Adding noise to each projection...');
% X = randn(145);
% for i = 1:N
%     proj(:,i)= proj(:,i) +(2.* X(:,i));
% end

%   Using the projections from above we can now filter and back-project
%   the image using our backprojection algorithm
 disp('Reconstructing the image from back-projection...');
 BP = iradon(proj,theta1,'spline','Hamming');
 subplot(224);
 imagesc(BP);
 title('BP Image: Hamming filter, spline interp.');
```

- Ghi chú:

Chương trình BP2 thường cho kết quả tái tạo tốt hơn chương trình BP1 vì lệnh phép biến đổi ngược `iradon` đã được lập trình một cách tối ưu thông qua sự chọn bộ lọc (mặc định Ram-Lak) và phép nội suy (mặc định là linear). Tuy nhiên sinh viên khó lòng hiểu được phần chương trình cụ thể của 2 lệnh `radon` và `iradon` mà chỉ có thể áp dụng một cách máy móc. Cho nên chương trình BP1 tạo cơ hợi cho sinh viên có thể tự lập trình thuật toán và thử nghiệm nhiều dạng bộ lọc khác mà lệnh `iradon` không cho phép.

Một số kết quả minh hoạ:

* *BP1:*

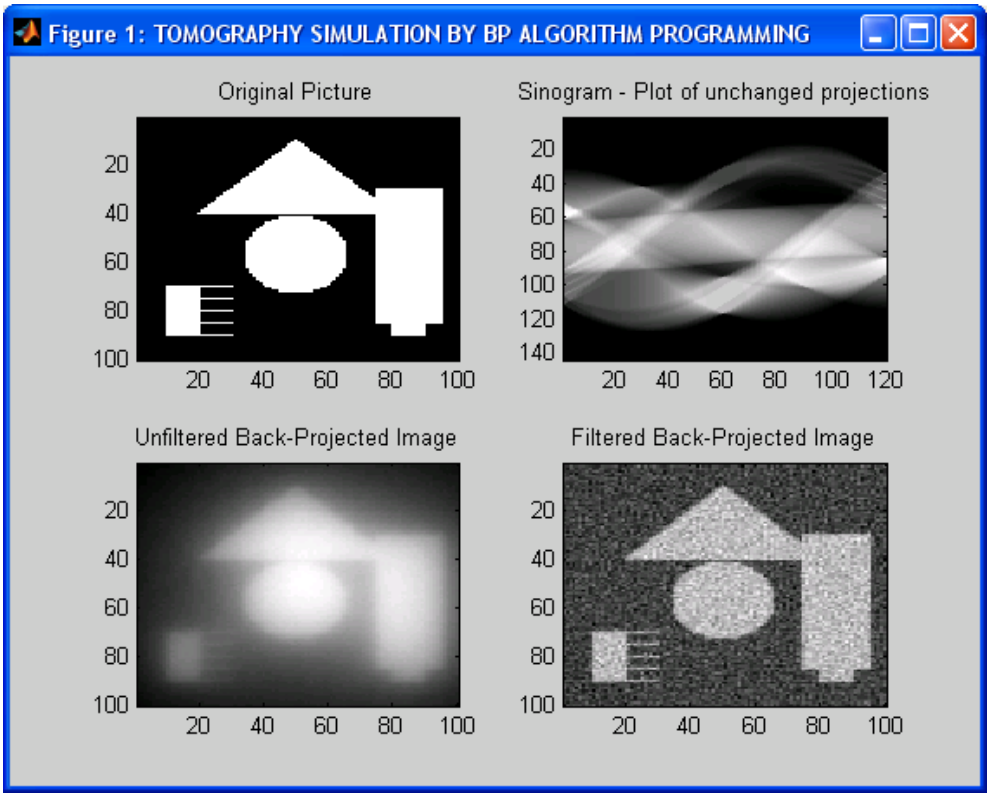


Figure 1: TOMOGRAPHY SIMULATION BY BP ALGORITHM PROGRAMMING
Original Picture
Sinogram - Plot of unchanged projections
Unfiltered Back-Projected Image
Filtered Back-Projected Image

** BP2:*

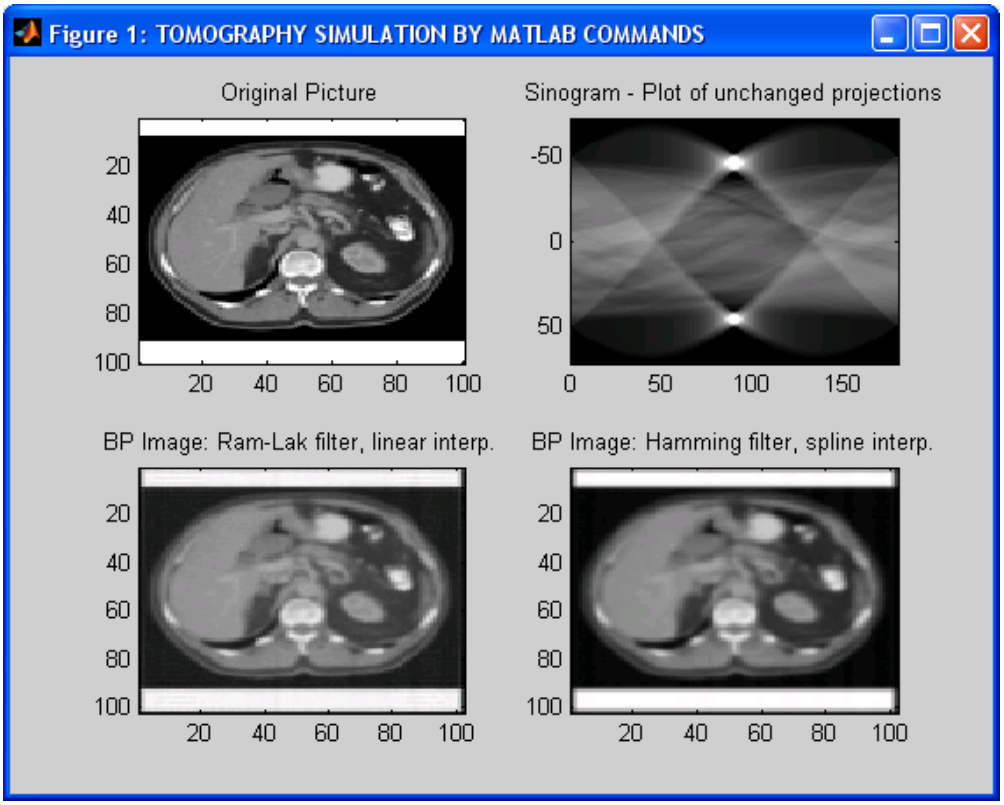


Figure 1: TOMOGRAPHY SIMULATION BY MATLAB COMMANDS
Original Picture
Sinogram - Plot of unchanged projections
BP Image: Ram-Lak filter, linear interp.
BP Image: Hamming filter, spline interp.

5. Nhiệm vụ và câu hỏi

1) Đọc kỹ phần tóm tắt lý thuyết và phần giải thích chương trình mã nguồn để hiểu rõ các ý nghĩa và nội dung lập trình.

2) Vận hành MATLAB và tìm hiểu các câu lệnh cơ bản phục vụ cho bài thí nghiệm. Chú ý tìm hiểu kỹ 2 lệnh `radon` và `iradon` (*viết mô tả hai hàm trên trong báo cáo*).

3) Dùng tài liệu và phần help của Matlab khảo sát các dạng hàm lọc trong phép biến đổi back projection (*viết tổng quan báo cáo*)

4) Thực hiện lập trình phần chương trình BP1 và BP2 và chạy thử nghiệm:

- Chép các phần mã nguồn trong phần lý thuyết vào trong giao diện M-file của MATLAB, liên kết và chạy thử. Chú ý sử dụng debug để xác định chỉnh sữa lỗi.
- Tạo các file ảnh để thử nghiệm: chú ý phải là file BMP 100x100 pixels (có thể sử dụng các ảnh khác với các ảnh có trong thư mục chương trình đã cung cấp).
- *Lưu lại mã nguồn và kết quả để làm báo cáo.*

5) Thử thay đổi các dạng bộ lọc khác nhau trong BP1 và BP2 và nhận xét.

6) *Nhiệm vụ mở rộng (không bắt buộc):* Tìm hiểu lệnh `fanbeam` và `ifanbeam` và xây dựng chương trình tạo ảnh cắt lớp trên cơ sở các hàm đó (*Lưu lại mã nguồn và kết quả để làm báo cáo)*.

6. Tài liệu tham khảo:

1. Avinash C. Kak, Malcolm Slaney: "Principles of Computerized Tomographic Imaging" – IEEE Press - 1988
2. Anthony Brinton Wolbarst: "Physics of Radiology", Prentice Hall International, Inc.- 1993
3. Kevin M. Rosenberg: "Ctsim – The Open-Source Computed Tomography Simulator"
4. http://bigwww.epfl.ch