Taller - Imágenes Médicas
Conceptos de tomografía

William Gómez Roa
wa.gomez@javeriana.edu.co
Bioingeniería y Ciencia de Datos

Carolina Santos Baquero
carolinasantosb@javeriana.edu.co
Bioingeniería

Ejercicio 1: Creación de imagen sintética

Vamos a crear una imágen sintética como la que se muestra a continuación en la Figura 1. Esta imágen sintética constará de 5 circulos que indicaremos por medio de 4 parametros: coordenada en x, coordenada en y, el radio del circulo y el valor de intensidad de sus pixeles.

El algoritmo se intentará explicar a continuación: Observe que la cordenada (0,0) de la imagen se encuentra en el centro de esta. Dicha matriz con el origen en el centro posee 4 cuadrantes que disminuyen hacia la izquierda y hacia arriba, como se ve en los ejes de la imágen.

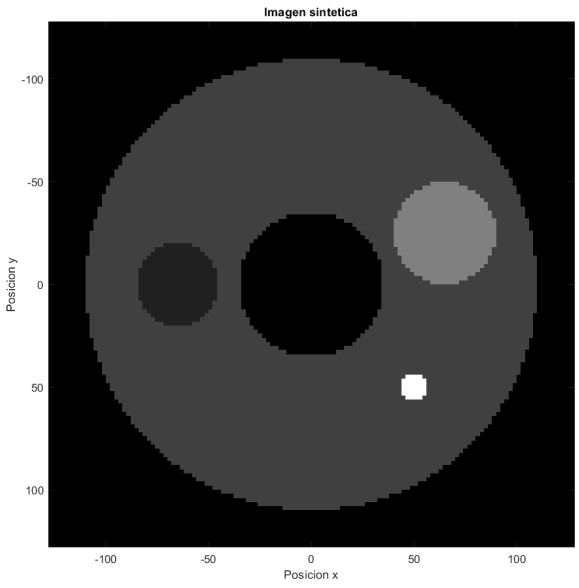

Figure 1: *Phantom Image*

```
% columna 1: centro en x
```

```
%columna 2: centro en y
%columna 3: radio
%columna 4: coef. atenuacion (intensidad)
circ = [ 0 0 110 2;
 -65 0 20 1;
 0 0 35 0;
 65 -25 25 4;
 50 50 7 8];

 % parametros de la imagen: numero de pixeles, tamano, etc.
nx = 128;
ny = 128;
dx = 2; % resolucion: 2 mm / pixel
x = dx * ([1:nx]'-(nx+1)/2);
y = -dx * ([1:ny]'-(ny+1)/2);
xx = x(:,ones(1,nx));
yy = y(:,ones(1,ny))';
% generar los datos para la imagen sintetica
phantom = zeros(nx,ny);
for ii=1:size(circ,1)
 cx = circ(ii,1);
 cy = circ(ii,2);
 rad = circ(ii,3);
 amp = circ(ii,4);
 t = find( ((xx-cx)/rad).^2 + ((yy-cy)/rad).^2 <= 1 );
 phantom(t) = amp * ones(size(t));
end
```

El algoritmo utiliza los parametros del circulo de la matriz **circ** e inicializa 2 matrices (**xx** y **yy**) de una forma especifica, tal que ambas tienen el mismo tamaño de la imagen que se quiere sintetizar y una es reflejo horizontal de la otra. Para el tamaño y resolución especificada, se crean 2 vectores **x** y **y** y sus valores se distribuyen simetricamente respecto al centro de dicho vector con un valor de 0 y aumentan o disminuyen en alguna direccion según la resolución indicada. Poseteriormente estos vectores columna se extienden horizontalmente para formar 2 matrices cuadradas (**xx** y **yy**, las cuales se utilizarán junto con los parametros del circulo para realizar una operación específica que permita obtener la ubicación de cada circulo.

El algoritmo utiliza la ecuación del circulo para ubicar los pixeles que corresponden al circulo indicado en los parametros de la matriz **circ**:

$$\left(\frac{xx - cx}{\text{rad}}\right)^2 + \left(\frac{yy - cy}{\text{rad}}\right)^2 \leq 1$$

$$(xx - cx)^2 + (yy + cy)^2 \leq \text{rad}^2$$

Ejercicio 2: Cálculo del sinograma

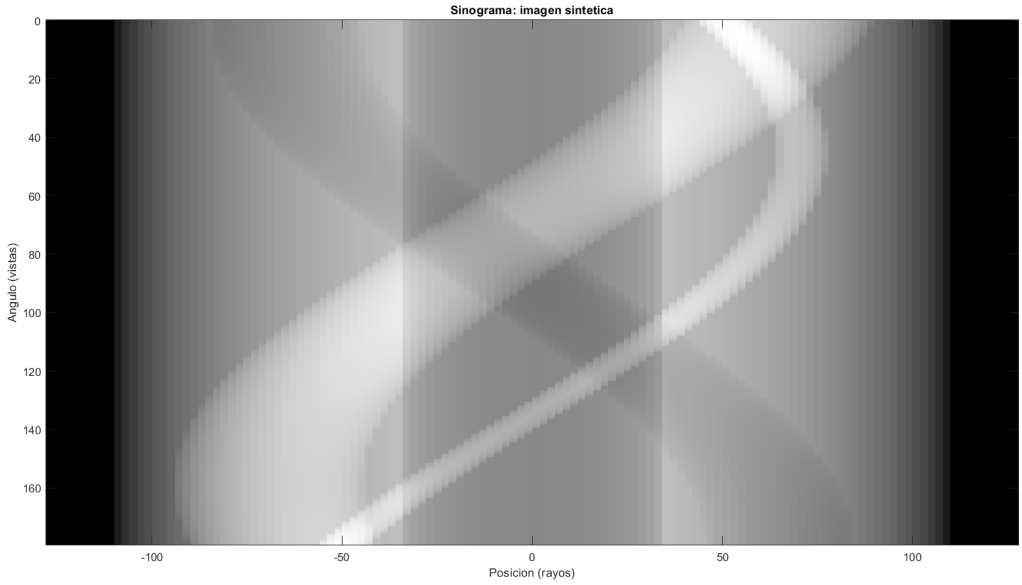

Figure 2: *Sinograma de la imágen sintética*

```
% parametros de la geometria
nr = 128; % numero de muestras radiales
dr = 2; % espaciado de los rayos
na = nr*2; % numero de muestras angulares
r = dr * ([1:nr]'-(nr+1)/2); % posiciones de las muestras radiales
angle = [0:(na-1)]'/na * pi; % posiciones de las muestras angulares
% calcular el sinograma para la imagen sintetica
rr = r(:,ones(1,na));
sg1 = zeros(nr, na);
for ii=1:size(circ,1)
 cx = circ(ii,1);
 cy = circ(ii,2);
 rad = circ(ii,3);
 amp = circ(ii,4);
 tau = cx * cos(angle) + cy * sin(angle);
 tau = tau(:,ones(1,nr))';
 t = find( (rr-tau).^2 <= rad.^2 );
 if ii > 1, amp = amp - circ(1,4); end % discos pequenos embebidos
 sg1(t) = sg1(t)+amp*2*sqrt(rad^2-(rr(t)-tau(t)).^2);
end
% visualizar el sinograma
figure(2)
```

```
 imagesc(r, angle/pi*180, sg1') % se usa la transpuesta (') y el angulo
 colormap('gray') % se muestra en grados
 title('Sinograma: imagen sintetica')
 xlabel('Posicion (rayos)')
 ylabel('Angulo (vistas)')
% definir una variable general para el sinograma, para posterior ejecucion
sinogram = sg1; % sinograma de la imagen sintetica
% visualizar informacion del sinograma
disp(sprintf('numero de rayos = %g', nr))
disp(sprintf('numero de vistas = %g', na))
```

El algoritmo utiliza las cordenadas del centro de cada circulo **cx** y **cy**, para calcular la magnitud de un vector para todos los angulos entre 0 y π y los almacena en la variable **tau**. Estos valores corresponden a la magnitud de la intensidad de la proyección de linea.

$$tau = cx * cos(angle) + cy * sin(angle)$$

Posterior a esto, se utiliza una matriz llamada **rr** la cual tiene valores en sus filas entre $-(nr-1)$ y $nr-1$ con pasos indicados por **dr** y el valor de cada fila es el mismo para todas las columnas. Dicha matriz es restada con **tau** y comparada con el valor del radio para determinar los angulos y pixeles que corresponden al circulo correspondiente:

$$(rr - tau)^2 \leq \text{rad}^2$$

Por último se almacenan dicha información en la matriz **sg1**.

Ejercicio 3: Aplicación de retroproyección simple (laminograma)

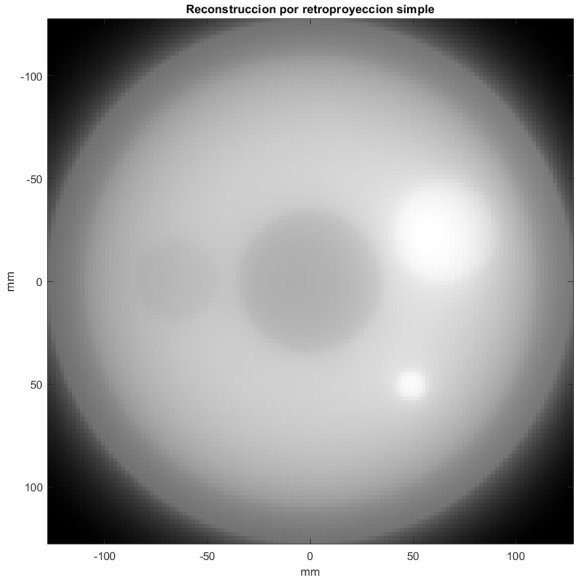

Figure 3: *Retroproyeccion simple*

```
% retroproyeccion simple
lamin = zeros(nx,ny);
for ia = 1:na
 disp(sprintf('angulo %g de %g', ia, na));
 % extraer angulo actual del sinograma
 projection_ia=sinogram(:,ia);
 % difuminar angulo actual en 128*128
 projection_smear=repmat (projection_ia,1,128);
 % 256 proyecciones corresponden a 180 grados
 % entonces se usa ia*180/256 para el angulo de proyeccion actual
 rot= imrotate(projection_smear', ia*180/256, 'bicubic','crop');
 % lamin necesita ser de 128*128
 % por eso el primer argumento en imrotate debe ser de la misma dimension
 lamin=lamin+rot;
end
```

Primero se toma el sinograma del punto anteriorior columna por columna dentro del ciclo *for*, luego para cada columna se utiliza la función *repmat(columna, 1,128)* para extender este vector columna en una matriz con 128 columnas y el mismo valor a lo largo de las filas.

Por último se genera una "rotación" de la matriz generada, rotandola en factores de 180 grados la información y almacenandola en una matriz **lamin**, cuyo tamaño es de 128*128.

Ejercicio 4: Filtrado de las proyecciones

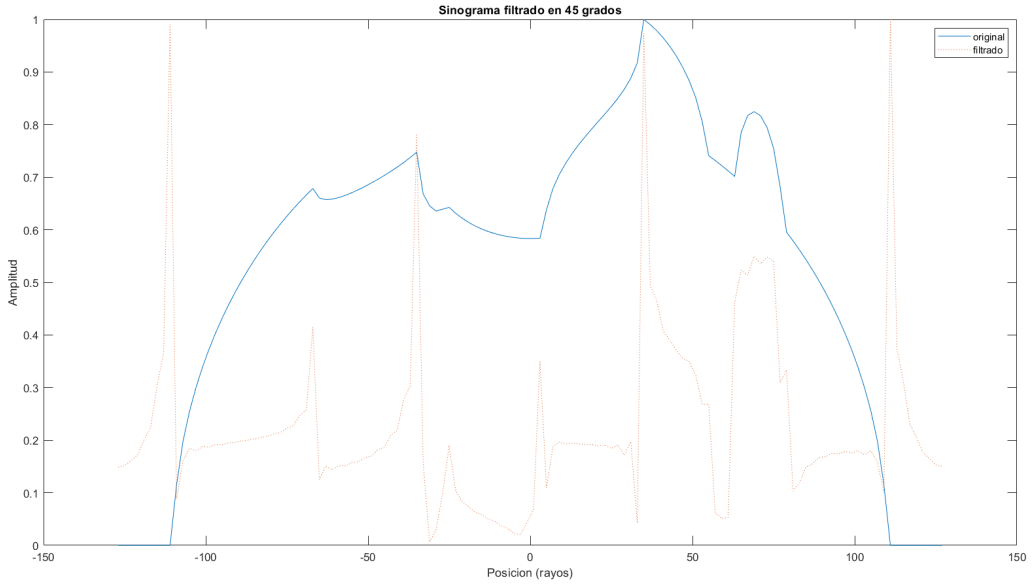

Figure 4: *Phantom Image*

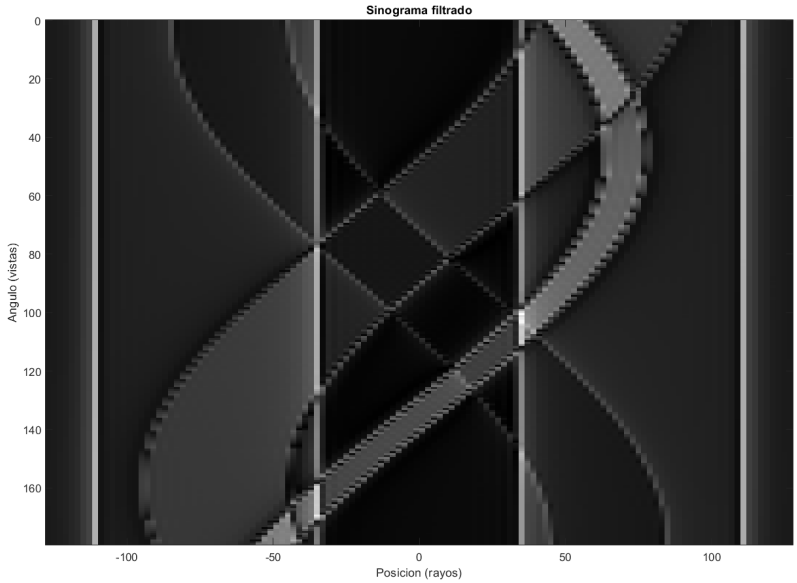

Figure 5: *Phantom Image*

```
% transformar en Fourier el sinograma,
% luego trasladar para centrar las bajas frecuencias
sinogramfiltered=fftshift(fft(sinogram));
% filtrar el sinograma usando un filtro Ram-Lak
a = length(sinogram);
% pendiente -1 hasta 128, luego +1 hasta 256
freqs=linspace(-1, 1, a/2).';
myFilter = abs(freqs);
% asegurarse que el sinograma y el filtro son de las mismas dimensiones
myFilter = repmat(myFilter,1,256);
% multiplicar en frecuencia, luego trasladar, luego transformada inversa
sinogramfilt=abs(ifft(ifftshift(sinogramfiltered.*myFilter)));
```

Inicialmente, se calcula la transformada de Fourier del sinograma. La función fftshift reorganiza las frecuencias para que las bajas frecuencias estén en el centro. Luego de esto se inicializa un filtro Ram-Lak para atenuar las altas frecuencias.

Finalmente multiplica la transformada de Fourier del sinograma con el filtro Ram-Lak. Luego, aplica la transformada inversa de Fourier, para devolver el sinograma al dominio espacial.

En la visualización se muestra la señal filtrada del sinograma para un angulo específico (45°) y se muestra además el sinograma filtrado en dichas frecuencias mencionadas.

Ejercicio 5: Aplicación de retroproyección filtrada

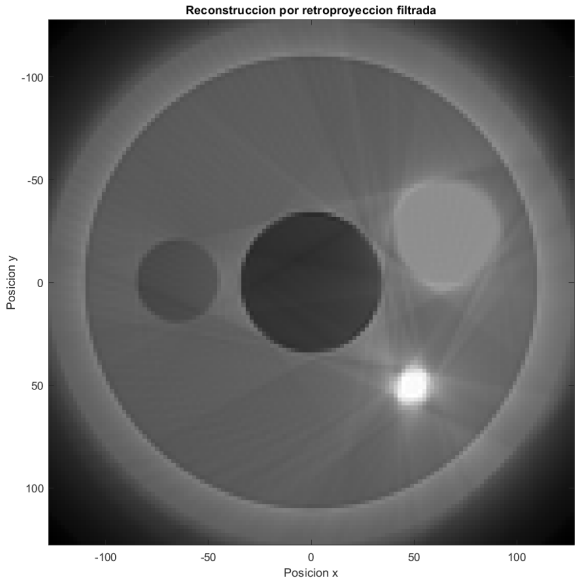

Figure 6: *Retroproyección del sinograma filtrado*

```
% retroproyeccion filtrada
bpf_recon = zeros(nx,ny);
for ia = 1:na
 disp(sprintf('angulo %g de %g', ia, na));
 bpf_ia=sinogramfilt(:,ia);
 bpf_smear=repmat(bpf_ia,1,128);
 % rotando la proyeccion
 rot1= imrotate(bpf_smear', ia*180/256, 'bicubic','crop');
 bpf_recon=bpf_recon+rot1;
end
```

Se inicia recorreindo en un bucle todos los ángulos de proyección desde 1 hasta 256 que hay en el sinograma filtrado.

Se toma cada proyección filtrada y se realiza el mismo procedimiento del Ejercicio 3, es decir se aplica retroporyección simple para el sinograma filtrado.

Por últmio se visualiza la impagen reconstgruida de la Figura 6

Ejercicio 6: Transformada de Radón y reconstrucción de la imagen

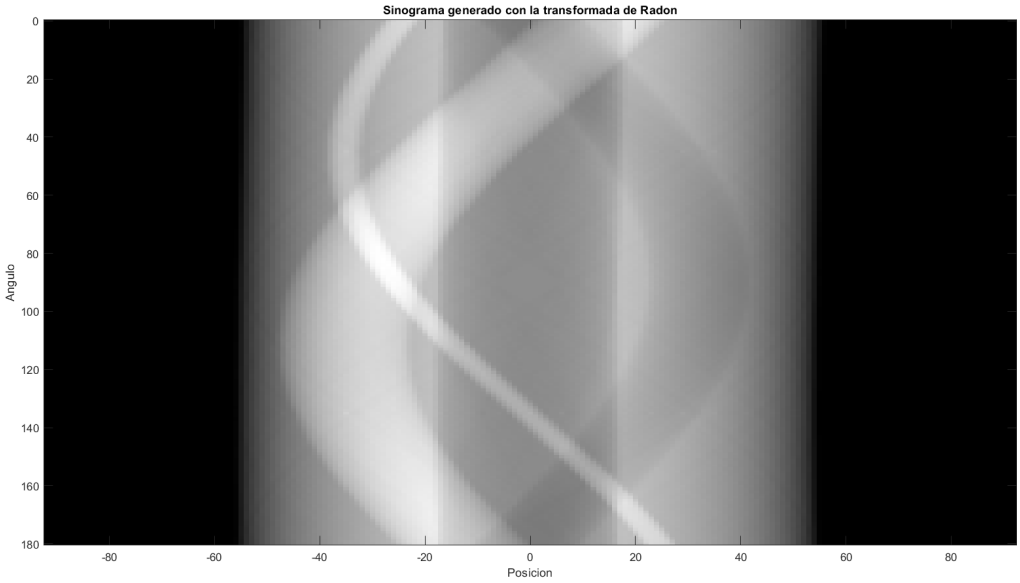

Figure 7: *Phantom Image*

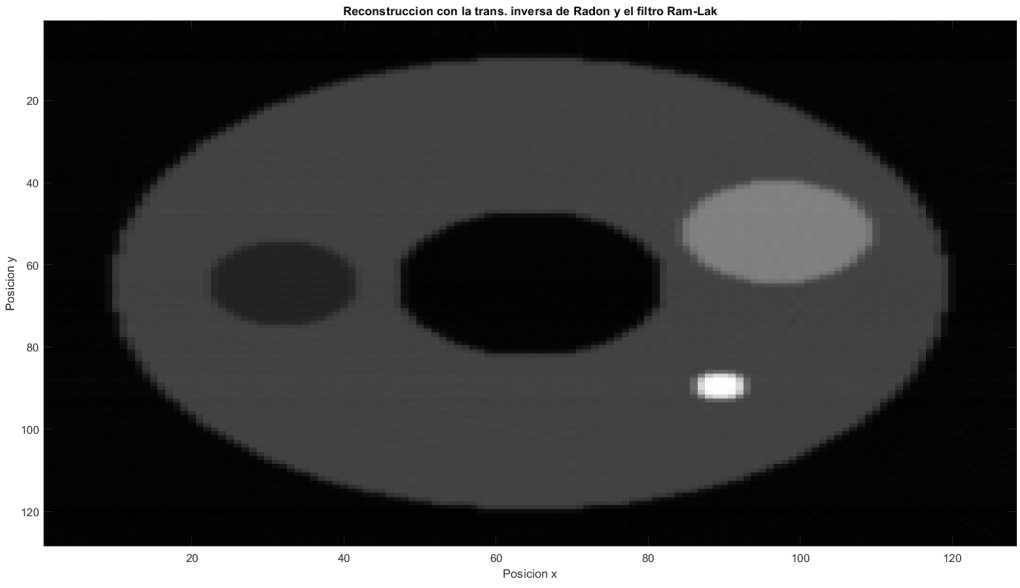

Figure 8: *Transformada de Radón de la imágen sintetizada.*

```
% angulos para la proyeccion
theta=0:180;
% obtener el sinograma aplicando la transformada de Radon
```

```
[R,rad_angles]=radon(phantom,theta);
% visualizar el sinograma obtenido
figure(7)
 imagesc(rad_angles,theta,R');
 colormap('gray');
 title('Sinograma generado con la transformada de Radon')
 xlabel('Posicion')
 ylabel('Angulo')
% reconstruir la imagen con la transformada inversa, aplicando un filtro
RamLak_filtered=iradon(R, theta, 'linear','Ram-Lak', 1.0, size(phantom,1))
   ;
RamLak_filtered=imrotate(RamLak_filtered,90);
% visualizar la imagen reconstruida
figure(8)
 imagesc(RamLak_filtered);
 colormap('gray');
 title('Reconstruccion con la trans. inversa de Radon y el filtro Ram-Lak'
    )
 xlabel('Posicion x')
 ylabel('Posicion y')
```

Inicialmente se aplica la ***Transformada de Radon***, directamente sobre la imagen sintetizada en el Ejercicio 1 para los ángulos entre 0 y 180. El sinograma resultante se visualiza en la Figura 7.

Por último, se reconstruye la imagen con la transformada inversa de Radon, se utiliza un filtro RamLak entre los parametros y dicha imágen recosntruida se observa en la Figura 8.

Ejercicio 7

Genere diferentes reconstrucciones por retroproyección filtrada para la imagen sintética original usando la transformada de Radón en MATLAB, esta vez variando el filtro utilizado al momento de la reconstrucción. Las opciones de filtro a probar (en vez de 'Ram-Lak') son: 'Shepp-Logan', 'Hamming' y 'Hann'. Analice en su reporte las posibles diferencias que puedan identificarse del uso de estos filtros.

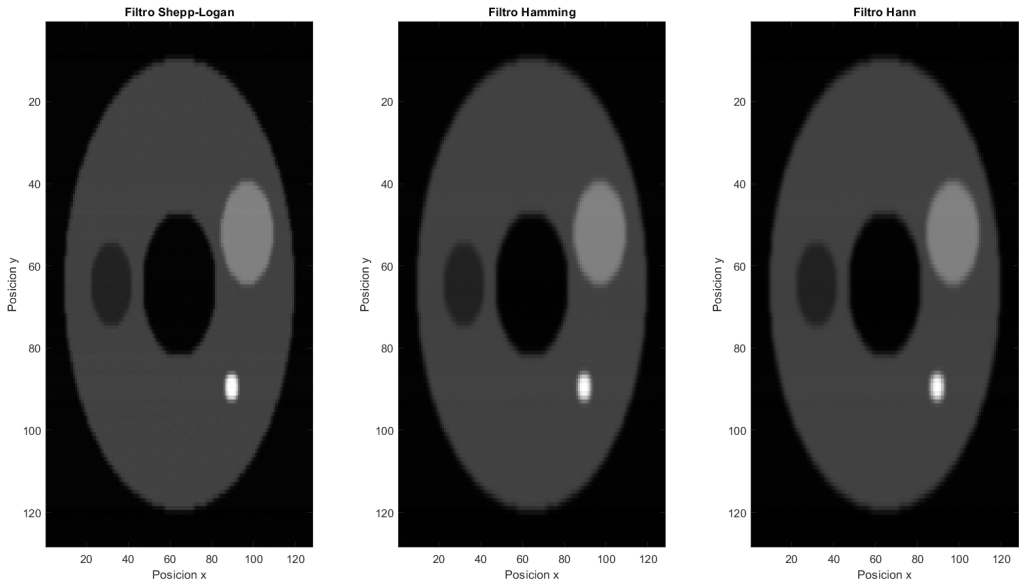

Figure 9: *Reconstrucciones de Sinograma con Transformada Inversa de Radon utilizando distintos filtros*

```
% obtener el sinograma aplicando la transformada de Radon
theta=0:180;
[R,rad_angles]=radon(phantom,theta);

% reconstruir la imagen con la transformada inversa, aplicando un filtro
SheppLogan_filtered=iradon(R, theta, 'linear','Shepp-Logan', 1.0, size(
    phantom,1));
SheppLogan_filtered=imrotate(SheppLogan_filtered,90);
Hamming_filtered=iradon(R, theta, 'linear','Hamming', 1.0, size(phantom,1)
    );
Hamming_filtered=imrotate(Hamming_filtered,90);
Hann_filtered=iradon(R, theta, 'linear','Hann', 1.0, size(phantom,1));
Hann_filtered=imrotate(Hann_filtered,90);

% visualizar las imagenes reconstruidas
figure()

subplot(1,3,1)
 imagesc(SheppLogan_filtered);
 colormap('gray');
 title('Filtro Shepp-Logan')
 xlabel('Posicion x')
 ylabel('Posicion y')
```

```
subplot(1,3,2)
imagesc(Hamming_filtered);
colormap('gray');
title('Filtro Hamming')
xlabel('Posicion x')
ylabel('Posicion y')

subplot(1,3,3)
imagesc(Hann_filtered);
colormap('gray');
title('Filtro Hann')
xlabel('Posicion x')
ylabel('Posicion y')
```

Las diferencias que puedan identificarse del uso de estos filtros, es que al aplicar la ventana de Shepp-Logan a la imagen se mejora la calidad de la imagen reconstruida y ayuda a reducir artefactos, especialmente en áreas con alta variación de densidad. Por otra parte, la ventana de Hamming, suaviza las transiciones y bordes, lo que resulta una imagen con menos detalles finos, pero con menos ruido en el dominio de la frecuencia. Además, la ventana de Hann, suaviza las transiciones y reduciendo las fugas espectrales, a su vez, con lóbulos laterales más anchos en el espectro de frecuencia en comparación con Hamming. Con lo dicho, las diferencias en las imágenes procesadas con estas ventanas radican en cómo afectan la nitidez de los bordes, la atenuación del ruido y la representación de detalles finos en la imagen.

Ejercicio 8

Replique el proceso completo (sinograma – ejercicio 2, retroproyección simple – ejercicio 3, sinograma filtrado – ejercicio 4, retroproyección filtrada – ejercicio 5, reconstrucción con la transformada de Radón – ejercicio 6) para una nueva imagen sintética con al menos 7 círculos de diferentes tamaños, posiciones e intensidades (ejercicio 1). Identifique claramente el filtro utilizado al momento de aplicar la transformada inversa de Radón.

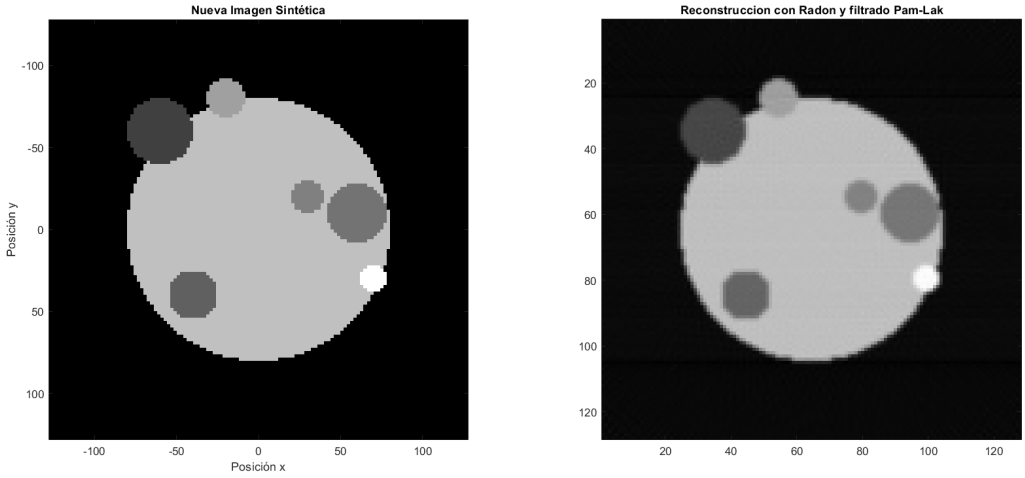

Figure 10: *Nueva imágen sintética y imágen reconstruida con la transformada inversa de Radon*

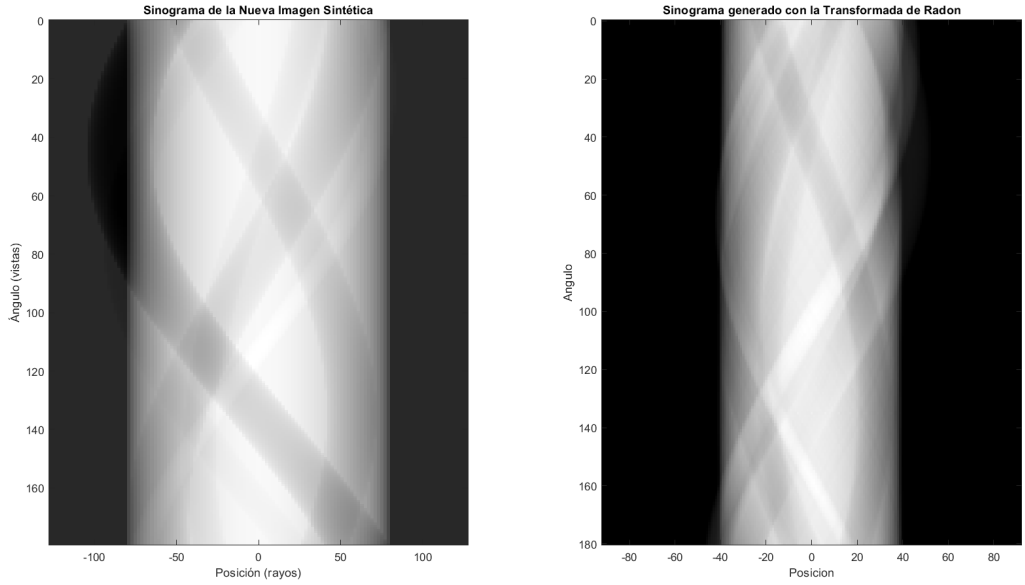

Figure 11: *Sinograma de la nueva imagen sintética y el generado con la transformada de Radon*

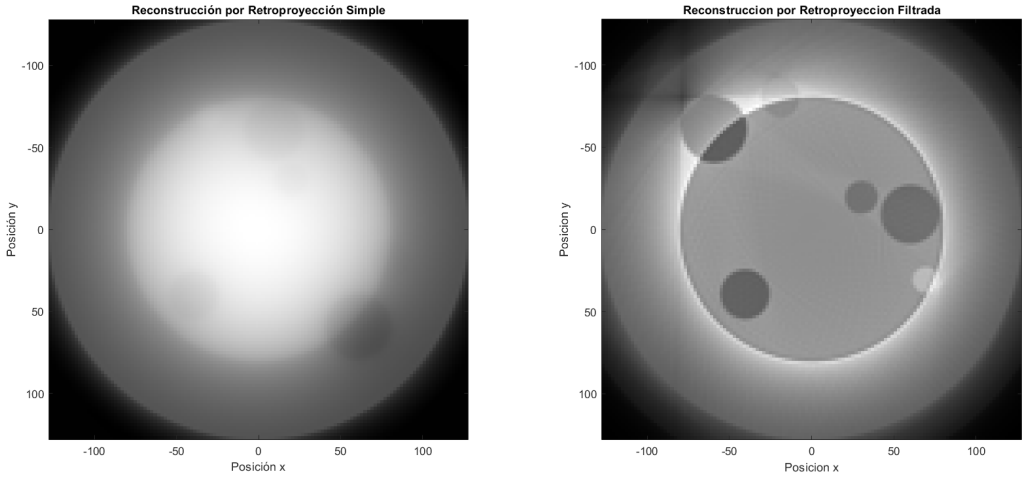

Figure 12: *Reconstrucción por retroproyección simple y filtrada con el filtro Ram-Lak*

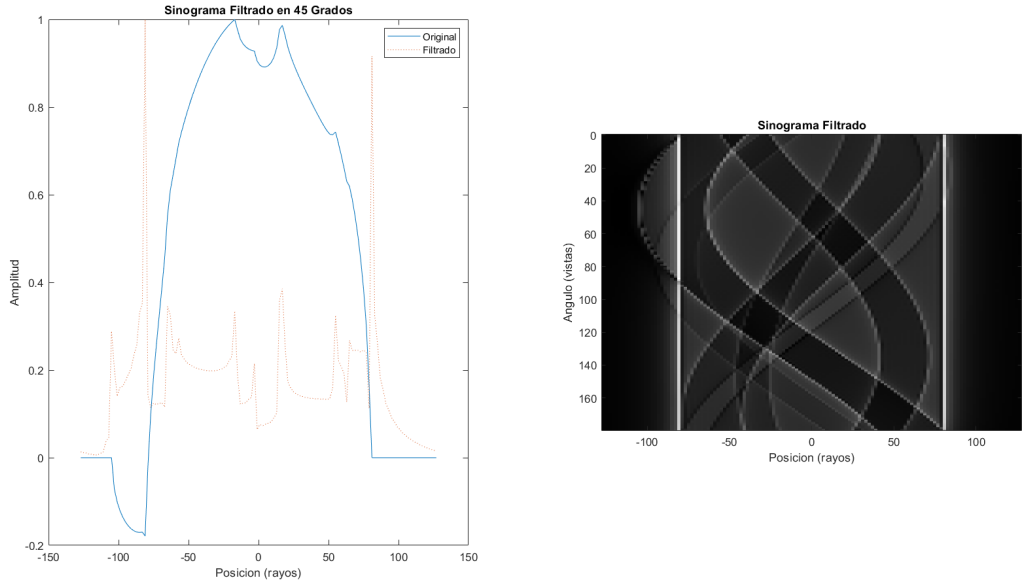

Figure 13: *Sinograma filtrado y la señal en 45 grados*

```
clc, clear, close all;

% PARAMETROS CIRCULOS
circles = [0 0 80 3;
           -40 40 15 1.5;
```

```
            30 -20 10 2;
            -60 -60 20 1;
            70 30 8 4;
            -20 -80 12 2.5;
            60 -10 18 1.8];

% PARAMETROS IMAGEN SINTETICA NUEVA
nx = 128;
ny = 128;
dx = 2; % Resoluci n: 2 mm / pixel
x = dx * ([1:nx]' - (nx+1)/2);
y = -dx * ([1:ny]' - (ny+1)/2);
xx = x(:, ones(1, nx));
yy = y(:, ones(1, ny))';

% IMAGEN SINTETICA NUEVA
new_phantom = zeros(nx, ny);
for ii = 1:size(circles, 1)
    cx = circles(ii, 1);
    cy = circles(ii, 2);
    rad = circles(ii, 3);
    amp = circles(ii, 4);
    t = find(((xx - cx) / rad).^2 + ((yy - cy) / rad).^2 <= 1);
    new_phantom(t) = amp * ones(size(t));
end

% SINOGRAMA
nr = 128; % N mero de muestras radiales
dr = 2; % Espaciado de los rayos
na = nr * 2; % N mero de muestras angulares
r = dr * ([1:nr]' - (nr+1)/2); % Posiciones de las muestras radiales
angle = [0:(na-1)]' / na * pi; % Posiciones de las muestras angulares

rr = r(:, ones(1, na));
sinogram = zeros(nr, na);
for ii = 1:size(circles, 1)
    cx = circles(ii, 1);
    cy = circles(ii, 2);
    rad = circles(ii, 3);
    amp = circles(ii, 4);
    tau = cx * cos(angle) + cy * sin(angle);
    tau = tau(:, ones(1, nr))';
    t = find((rr - tau).^2 <= rad.^2);
```

```
    if ii > 1
        amp = amp - circles(1, 4);
    end
    sinogram(t) = sinogram(t) + amp * 2 * sqrt(rad^2 - (rr(t) - tau(t))
       .^2);
end

% RETROPROYECCION SIMPLE
lamin = zeros(nx, ny);
for ia = 1:na
    projection_ia = sinogram(:, ia);
    projection_smear = repmat(projection_ia, 1, 128);
    rot = imrotate(projection_smear', ia*180/256, 'bicubic', 'crop');
    lamin = lamin + rot;
end

% SINOGRAMA FILTRADO
sinogramfiltered = fftshift(fft(sinogram));
a = length(sinogram);
freqs = linspace(-1, 1, a/2).';
myFilter = abs(freqs);
myFilter = repmat(myFilter, 1, 256);
sinogramfilt = abs(ifft(ifftshift(sinogramfiltered .* myFilter)));

% RETROPROYECCION FILTRADA
bpf_recon = zeros(nx, ny);
for ia = 1:na
    bpf_ia = sinogramfilt(:, ia);
    bpf_smear = repmat(bpf_ia, 1, 128);
    rot1 = imrotate(bpf_smear', ia*180/256, 'bicubic', 'crop');
    bpf_recon = bpf_recon + rot1;
end

% RECONSTRUCCION CON TRANSFORMADA DE RADON
theta = 0:180;
[R, rad_angles] = radon(new_phantom, theta);

% RECONSTRUCCION CON TRANSFORMADA INVERSA DE RADON Y FILTRO RAM-LAK
RamLak_filtered = iradon(R, theta, 'linear', 'Ram-Lak', 1.0, size(
   new_phantom, 1));
RamLak_filtered = imrotate(RamLak_filtered, 90);
```

```
% Visualizar la nueva imagen sint tica
figure(1)

subplot(1,2,1)
imagesc(x, y, new_phantom')
colormap('gray')
axis('square')
title('Nueva Imagen Sint tica')
xlabel('Posici n x')
ylabel('Posici n y')

subplot(1,2,2)
% Visualizar la imagen reconstruida con filtro Ram-Lak
imagesc(RamLak_filtered)
colormap('gray')
axis('square')
title('Reconstruccion con Radon y filtrado Pam-Lak')


figure(2)
subplot(1,2,1)
% Visualizar el sinograma de la nueva imagen sint tica
imagesc(r, angle/pi*180, sinogram')
colormap('gray')
title('Sinograma de la Nueva Imagen Sint tica')
xlabel('Posici n (rayos)')
ylabel(' ngulo  (vistas)')

subplot(1,2,2)
% Visualizar el sinograma obtenido
imagesc(rad_angles, theta, R')
colormap('gray')
title('Sinograma generado con la Transformada de Radon')
xlabel('Posicion')
ylabel('Angulo')

figure(3)
subplot(1,2,1)
% Visualizar la imagen reconstruida mediante retroproyecci n simple
imagesc(x, y, lamin')
colormap('gray')
axis('square')
title('Reconstrucci n por Retroproyecci n Simple')
```

```
xlabel('Posici n x')
ylabel('Posici n y')

subplot(1,2,2)
% Visualizar la imagen reconstruida mediante retroproyeccion filtrada
imagesc(x, y, max(bpf_recon, 0));
colormap('gray')
axis('square')
title('Reconstruccion por Retroproyeccion Filtrada')
xlabel('Posicion x')
ylabel('Posicion y')


figure(4)
subplot(1,2,1)
% Visualizar el sinograma filtrado en Theta = 45 grados
plot(r, sinogram(:, 64) ./ max(sinogram(:, 64)), '-',...
     r, sinogramfilt(:, 64) ./ max(sinogramfilt(:, 64)), ':');
title('Sinograma Filtrado en 45 Grados')
legend('Original', 'Filtrado')
xlabel('Posicion (rayos)')
ylabel('Amplitud')

subplot(1,2,2)
% Visualizar el sinograma completo filtrado
imagesc(r, angle/pi*180, sinogramfilt')
colormap('gray')
axis('image')
title('Sinograma Filtrado')
xlabel('Posicion (rayos)')
ylabel('Angulo (vistas)')
```

El filtro utilizado al momento de aplicar la transformada inversa de Radón es el filtro Ram-Lak, puesto que este filtro de reconstrucción es comúnmente utilizado en la transformada inversa de Radón en la tomografía computarizada. En este caso, se especifica como 'Ram-Lak' en la función iradon.

El proceso que se está realizando en el código presentado es, primero generar una imagen sintética que contiene múltiples círculos de diferentes tamaños, posiciones e intensidades, basado en el Ejercicio 1. La imagen se crea utilizando coordenadas cartesianas y se visualiza como una representación gráfica de los círculos. Luego, se simula el proceso de adquisición de datos. El sinograma se crea en el Ejercicio 2, mediante la proyección de rayos a través de la imagen sintética desde diferentes ángulos. Este sinograma representa las mediciones de absorción de rayos X a lo largo de las trayectorias de los rayos a través de la imagen. Posteriormente, en el Ejercicio 3, se realiza una retroproyección simple de las proyecciones angulares para reconstruir la imagen. Esto implica sumar las contribuciones de las proyecciones a través de ángulos diferentes para obtener una imagen reconstruida inicial. En el Ejercicio 4, se aplica un procesamiento de filtrado al

sinograma para mejorar la calidad de la reconstrucción. Se utiliza un filtro Ram-Lak en el dominio de la frecuencia para realzar las características de la imagen y reducir artefactos. El filtro Ram-Lak es esencial para reducir artefactos y mejorar la precisión en la reconstrucción final. En cada paso del proceso, se visualizan los resultados para evaluar la calidad de la imagen a medida que se aplican los diferentes métodos y técnicas de procesamiento. En conjunto, este código simula y demuestra un flujo de trabajo típico de tomografía computarizada para generar imágenes médicas a partir de datos de proyección.

A continuación, en el Ejercicio 5, se lleva a cabo una retroproyección filtrada utilizando el sinograma filtrado. Este proceso refina aún más la reconstrucción utilizando la información mejorada del sinograma. Finalmente, en el "Ejercicio 6", se utiliza la transformada de Radon para generar un nuevo sinograma y luego se aplica la transformada inversa de Radon para obtener la reconstrucción final. En este caso, se utiliza un filtro Ram-Lak nuevamente en la transformada inversa de Radon para mejorar la calidad de la imagen reconstruida.

Ejercicio 9

Replique el proceso completo (sinograma – ejercicio 6, retroproyección simple – ejercicio 3, sinograma filtrado – ejercicio 4, retroproyección filtrada – ejercicio 5) para una imagen de su elección, garantizando que la información del sinograma se construye utilizando la transformada de Radón (ya no manualmente como en el ejercicio 2), y que las variables de geometría para el proceso de retroproyección se ajustan de acuerdo a las características de la imagen seleccionada. Se sugiere trabajar con imágenes de tamaño cuadrado e intensidades en escala de grises (1 solo canal) para facilidad del ejercicio.

```
% Cargar la imagen y definir la geometria
A=imread("eco_gray.png");

A=A(:,1:276);

%Obtener las dimensiones de la imagen
[nx ,ny] = size(A)


dx = 2; % resolucion: 2 mm / pixel
x = dx * ([1:nx]'-(nx+1)/2);
y = -dx * ([1:ny]'-(ny+1)/2);

na=395;

nr=128;
r = dr * ([1:nr]'-(nr+1)/2); % posiciones de las muestras radiales


% obtener el sinograma aplicando la transformada de Radon
theta=0:180;
```

```
[R,rad_angles]=radon(A,theta);

R=R';

figure(1)
 imagesc(rad_angles,theta,R);
 colormap('gray');
 title('Sinograma generado con la transformada de Radon')
 xlabel('Posicion')
 ylabel('Angulo')


% RETROPROYECCION SIMPLE
lamin = zeros(181, 181);
for ia = 1:na
    projection_ia = R(:, ia);
    projection_smear = repmat(projection_ia, 1, 181);
    rot = imrotate(projection_smear', ia*180/256, 'bicubic', 'crop');
    lamin = lamin + rot;
end

figure(2)
 imagesc(x, y, lamin);
 colormap('gray');
 axis('image')
 title('Reconstruccion por retroproyeccion simple')
 xlabel('mm') %x y y en imagesc nos da limites escalares en mm
 ylabel('mm')


% SINOGRAMA FILTRADO
sinogramfiltered = fftshift(fft(R));
a = length(R);
freqs = linspace(-1, 1,181).';
myFilter = abs(freqs);
myFilter = repmat(myFilter, 1, 395);
sinogramfilt = abs(ifft(ifftshift(sinogramfiltered .* myFilter)));

figure(3)
 imagesc(r, angle/pi*180, sinogramfilt');
 colormap('gray');
 axis('image');
 title('Sinograma filtrado')
```

```
 xlabel('Posicion (rayos)')
 ylabel('Angulo (vistas)')

% RETROPROYECCION FILTRADA
bpf_recon = zeros(181, 181);
for ia = 1:na
    bpf_ia = sinogramfilt(:, ia);
    bpf_smear = repmat(bpf_ia, 1, 181);
    rot1 = imrotate(bpf_smear', ia*180/256, 'bicubic', 'crop');
    bpf_recon = bpf_recon + rot1;
end

figure(4)
 imagesc(x, y, max(bpf_recon,0));
 colormap('gray');
 axis('image')
 title('Reconstruccion por retroproyeccion filtrada')
 xlabel('Posicion x')
 ylabel('Posicion y')
```

Figure 14: *Imagen original utilizada*

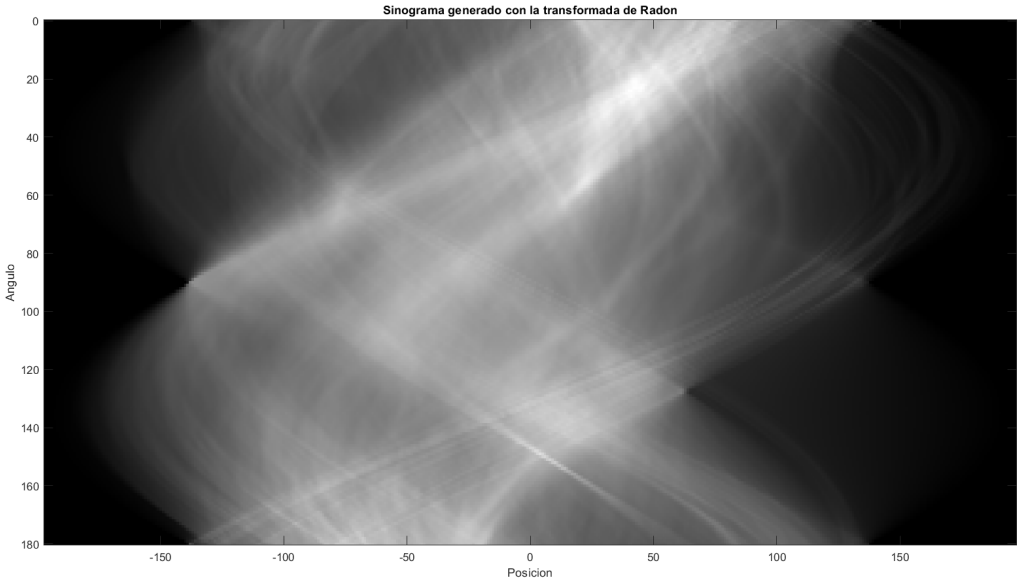

Figure 15: *Sinograma obtenido con la transforamda de Radon*

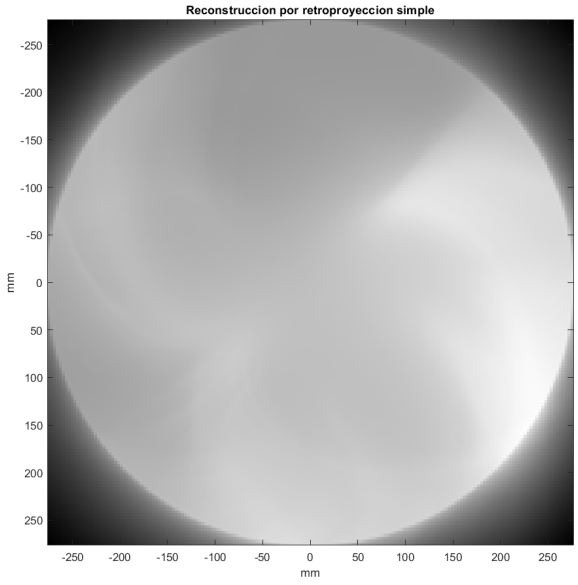

Figure 16: *Retroproyección simple*

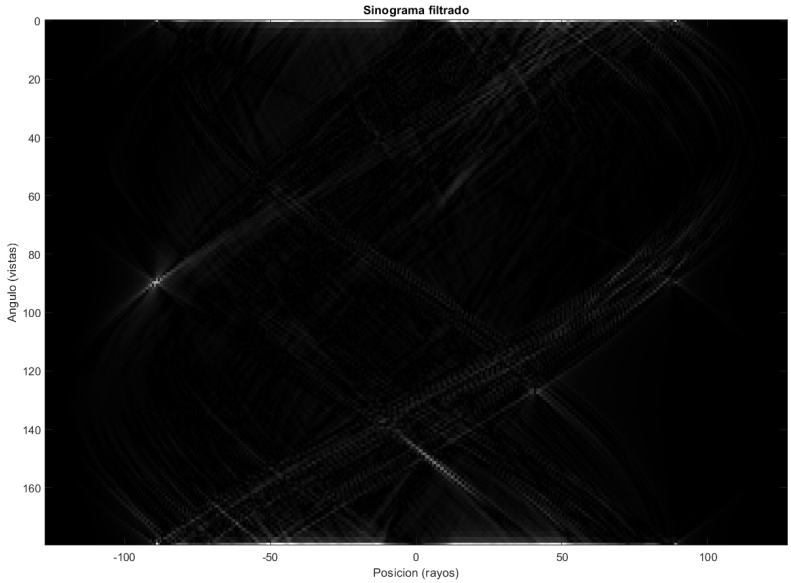

Figure 17: *Filtrado del sinograma*

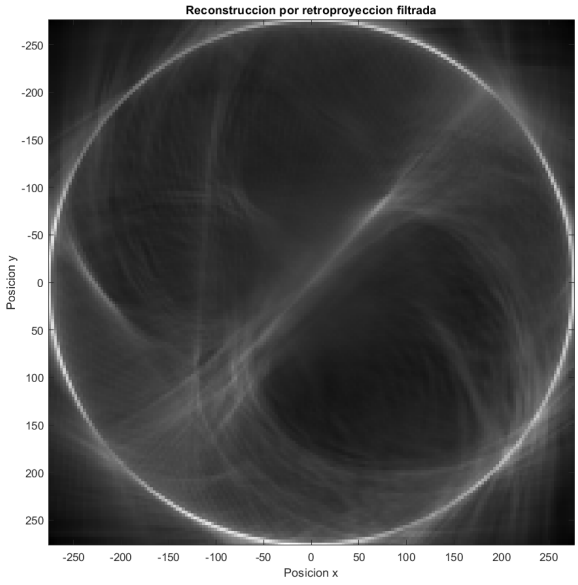

Figure 18: *Retroproyección filtrada*