Signal and image processing: Assignment 5

Mariuta Nicolae 02-10-2014

Exercise 1.

For solving this exercise and to make the next exercises easier, I have created a function that applies the random transform to any image.

```
function R = fradon(I,step)
N = size(I,1);
M = 180;
R = zeros(round(M/step),N);

for angle = 1 : (180/step)
    rotate = imrotate(I,(angle-1)*step,'nearest','crop');
    for line = 1 : N
        sumLine = 0;
        for column = 1 : N
            sumLine = sumLine + rotate(column,line);
        end
        R(angle,line) = sumLine;
    end
end

end
```

Fig.1 Function that applies the Radon transform over any image with chosen step (number of angles).

The function has the input parameters: *I* (the image) and *step*. The step defines the number of angles between 2 consecutive rotations. The number of lines in the transform matrix(*R*) is equal with *180/step*, where 180 is the angle of the last rotation and the number of columns is equal with width of the input image (which has square shape).

For each line in R, I rotate the image with *(angle-1)*step*. The purpose of *angle-1* is that I want to start from angle 0 and I multiply by *step* to get the desired angle step size between 2 rotations. I used Matlab function *imrotate* with parameters *'nearest'* and '*crop*'. '*nearest*' is the algorithm to choose what is the color of each pixel after rotation(because the pixels no longer get in the grid after they are rotated), which chooses the nearest pixel color. The other options were '*bilinear'* and '*bicubic'*which take the average of surrounding pixels to decide the color of each pixel, but the differences in resulting transform are not very large. The '*crop*' option cuts the image edges that get out of the original matrix size after rotation.

After each rotation, I take each line in the image, sum the pixels and add the result to transformation matrix. The 3 for loops are not a well optimized algorithm and takes some time to calculate the transform but I feel safer to do like this.

I have applied this function for a blank image with a white pixel and the *box.png* image that was given.

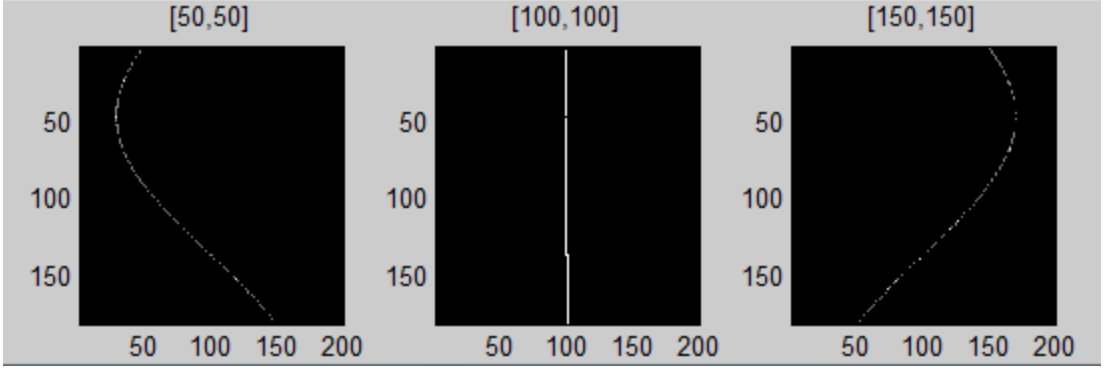

Fig.2 Radon transform applied to black image having a white point in different locations

I have placed the white point in several locations to see the differences. When I placed it in the middle of the image, I get a white line in the transform because during the rotation, the point does not move at all and it will always be on the same line for each angle.

But if the point is not located in the middle of the image, I get a curve with sinusoidal shape which shape and location depends on the location of the point in the original image.

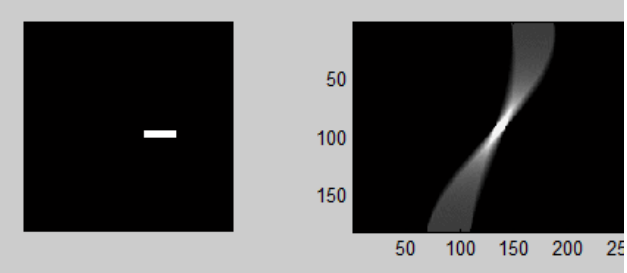

Fig.3 Radon transform applied to *box.png* image

In case of *box.png* I get an image with sinusoidal shape. It seems there are several sinusoids that are very close to each other closer to the middle of transform and further from each other when they are closer to the edges. The area with brighter pixels correspond to the angles close to 90 degrees, when the box is upside down and when calculating the sum of pixels on column, then more pixels are considered and the resulting value is higher.

Because using the '*crop*' option of *imrotate* function, there is some loss when calculating the transform. In these examples, the loss is not very large because the point and box do not get out of the image very much when it is rotated. Maybe a good option is to place the image into a larger matrix so the image does not get out of the matrix when it is rotated. This and the fact that there is used an approximation method when determining the values of new pixels after rotation, no matter if it is used '*nearest*'','*bilinear'* or '*bicubic'*algorithm, there will be some loss so the inverse transform will return a noisy image.

Exercise 2.

I have, once again, created a function to apply the backprojection to make it easier to use the code for the next exercises and make more experiments.

```
function Ir = fbackpropagation(R,step)
[M N] = size(R);
Ir = zeros(N,N);

for a = 1 : M
   Ia = zeros(N,N);
   for row = 1 : N
       Ia(row,:) = R(a,:);
   end
    Ir = Ir + imrotate(Ia,-(a-1)*step,'nearest','crop');
end

end
```

Fig.4 Function for applying the backpropagation to calculate the inverse Radon transform

I have used the algorithm proposed by John at the course because it is a lot easier and also fast. I have also tried to apply the equation written in the documentation but it had 3 for loops (long computation time) and It was also not working (probably because mistake at calculating the angles and coordinates).

For this function I take the Radon transform matrix that is resulting from applying the function shown at Exercise 1 and also the step size because I need it to reconstruct the original image. For each line in the transform matrix, I create a matrix with size of the original image containing all lines equal to the line that is the considered and then rotate it back corresponding to the angle at which the line was calculated. I sum all the matrixes calculated like this and obtain the original image.

I have applied this function for the transform of the image with white point, for the *box.png* transform and for the *sinogram.png* image that was given.

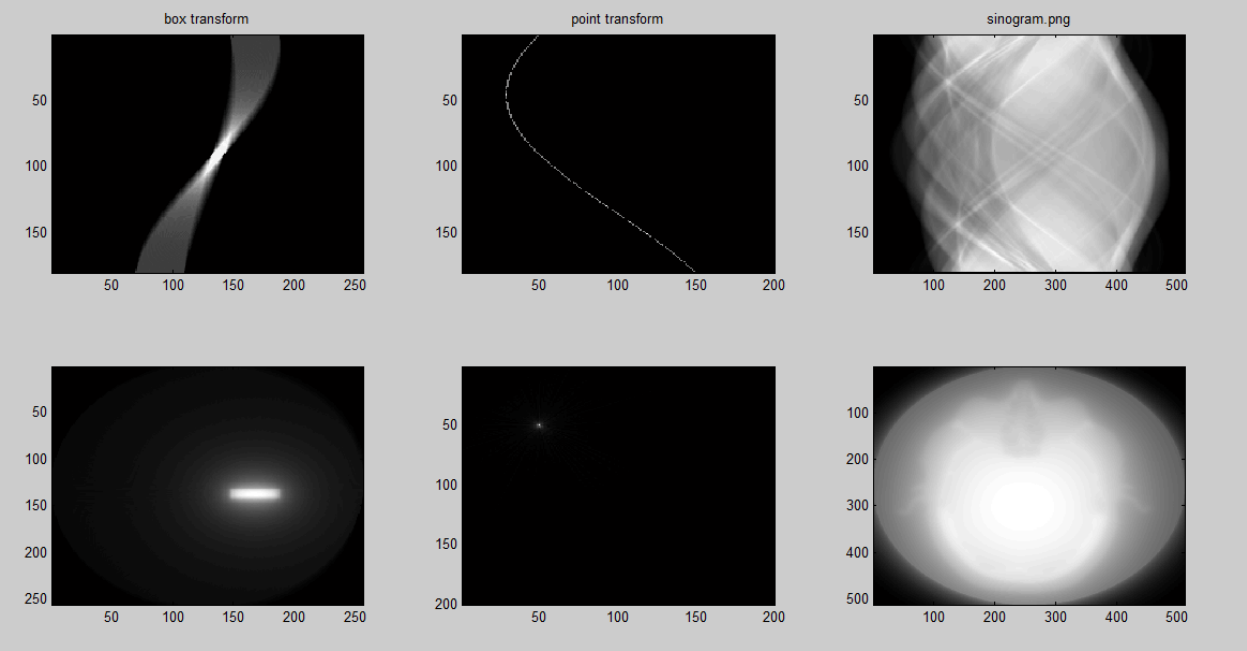

Fig.5backprojectionapplied for 3 sinograms: the box.png transform, the image with point and sinogram.png

About the point image and box transform, the backpropagation has rebuilt the original shapes but the edges are hard to distinguish.

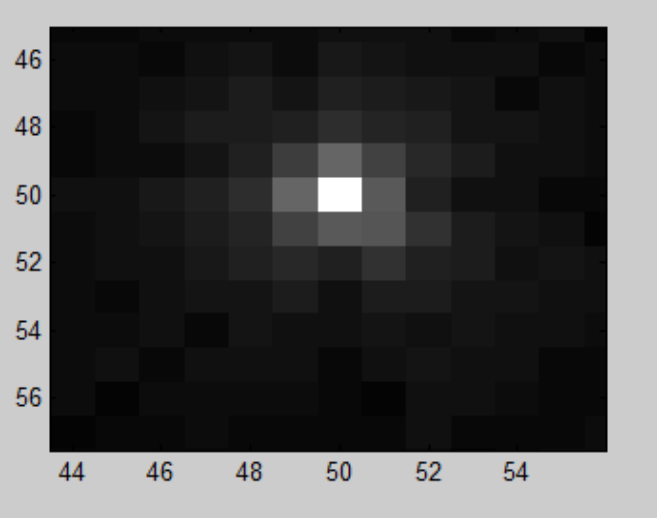

Fig.6 Zoomed result for applying backprojectionfor the sonogram of image containing a white point

About the backprojectionapplied to the sinogram obtained for the image having a white point we get several pixels having grey color (which can be seen after zooming the result). And even if not zoomed image, the point appears to be blurred and hard to distinguish the shape.

The backprojectionof the sinogram of box image gives a rectangle with edges that are hard to distinguish and surrounded by several circles, I think they are corresponding to each rotation of the image and they are result for the use of '*crop*' option of the *imrotate* function when the Radon transform was applied to the image.

For applying backprojection to '*sinogram.png*' image I get image of, probably part of human body but it is very blurred and hard to say what organ is in the image.

Exercise 3.

According to (10) from XP documentation, the filtered backprojection can be done by applying the ramp filter before doing the backprojection. I have created a function to apply the filtering easier.

```
function P= rfiltering( R )
p = double(R);
P = zeros(size(p));

ramp = fftshift(-size(P,2)/2:size(P,2)/2-1);

for r = 1 : size(R,1)
    P(r,:) = fft(p(r,:));
    P(r,:) = real(ifft(abs(ramp).*P(r,:)));
end

end
```

Fig.7 Function for applying the ramp filter to the sonogram

The function takes the sinogram of and image, creates a 1D ramp matrix and then calculate 1D FFT for each line in the sonogram, multiply it with the ramp and the apply the inverse FFT for that line.

I have also tried to apply FFT for whole image which applied 1D FFT for each line and created a 2D ramp filter having all lines equal with *ramp* and then multiply the 2D matrices that result but for reasons that I did not understand, that did not work.

After creating this function, I have applied it to the sinograms obtained before and after applying this function I have also applied the backprojection to reconstruct the original images.

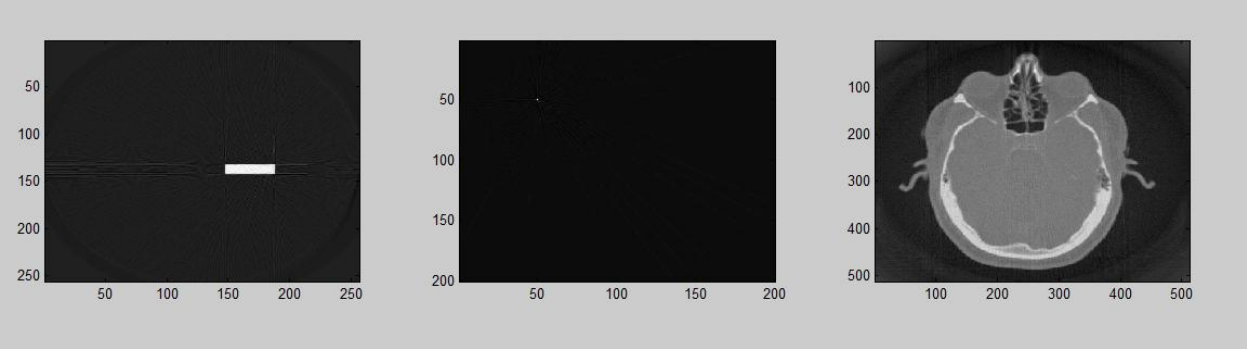

Fig.8 Results for applying the filtered backprojection for the 3 sinograms

By using the filtered backprojection, the results are a lot better, the reconstructed images having a lot better contrast than what I got at Exercise 2. The box shape is a lot easier to distinguish and it is no longer being surrounded by so many circles, the point is easier to distinguish because it is no longer surrounded by so many grey pixels. The image of the human organ is also having a lot better contrast , the shape and parts of the organs can be easily distinguished (I think it is an eye) and now can actually be used by a medic to diagnose it.

What I notice, maybe because of the filter, the edges of the resulting image are no longer very black and the larger the object is, the brighter are the pixels surrounding it(the area surrounding the small white point is almost completely black). In the Exercise 2, the edges of the reconstructed images are completely black.

Exercise 4.

Now, that I have implemented the functions considering the step size where it was necessary, this exercise will be easy to solve.

For lower step sizes the result is the same as for step =1. But for larger values I notice that some grey lines appear, and their number is equal to number of angles considered for calculating the Radon transform. For 90 degrees where only 2 angles are considered (0 and 90 degrees) only 2 lines appear. I have seen that even for step size 1, in case of zooming the image there are some rays that are converging at the location of point. The terminations of these lines form an oval but this is also because of the '*crop*' that is done at rotating the image.

Another thing to notice is that in case of filtered backprojection, the point also becomes a little blurred and it becomes almost invisible for only 2 projections(only 2 perpendicular lines appear). This means that a large step makes a degraded result in the reconstruction of the image by applying the backprojection.

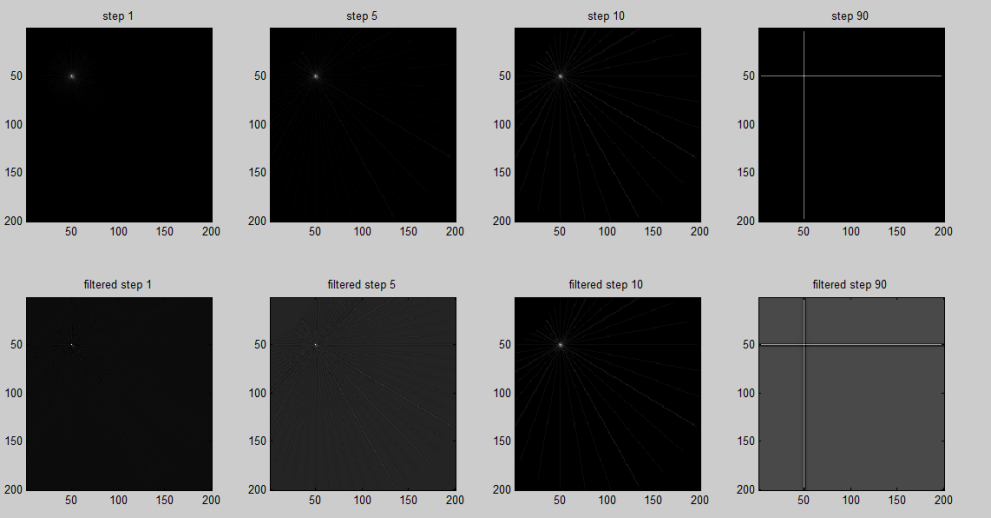

Fig.9 Different step sizes considered for simple and filtered backprojection of image with white point.

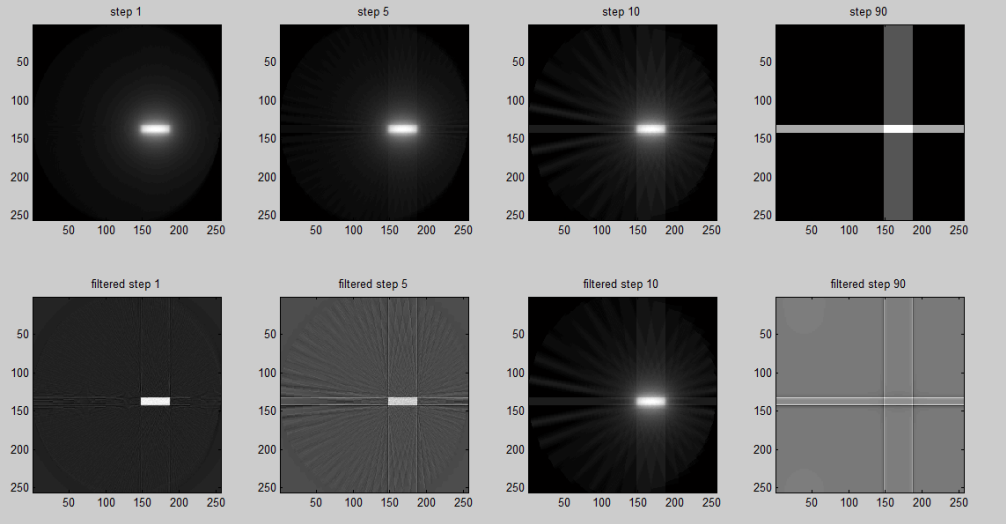

Fig.10 Different step sizes considered for simple and filtered backprojection of image with box.

Almost the same thing happens for the box image. The lines become wider beams but there also appears degradation in the result of backprojection, until the object can no longer be distinguished.

I have also done this to the image obtained after applying filtered backprojection (calculated the radon transform and backprojection again for it) and also for lena image (even if this method is mostly for medical purpose).

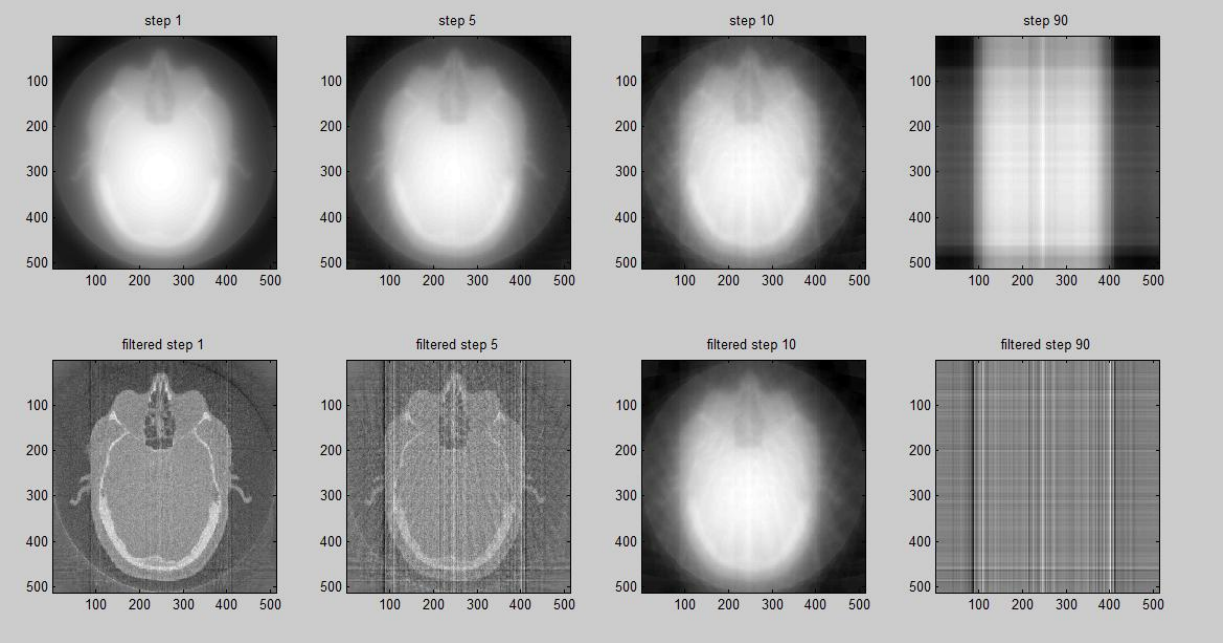

Fig.11 Different step sizes considered for simple and filtered backprojection of image that was reconstructed from *sinogram.png*

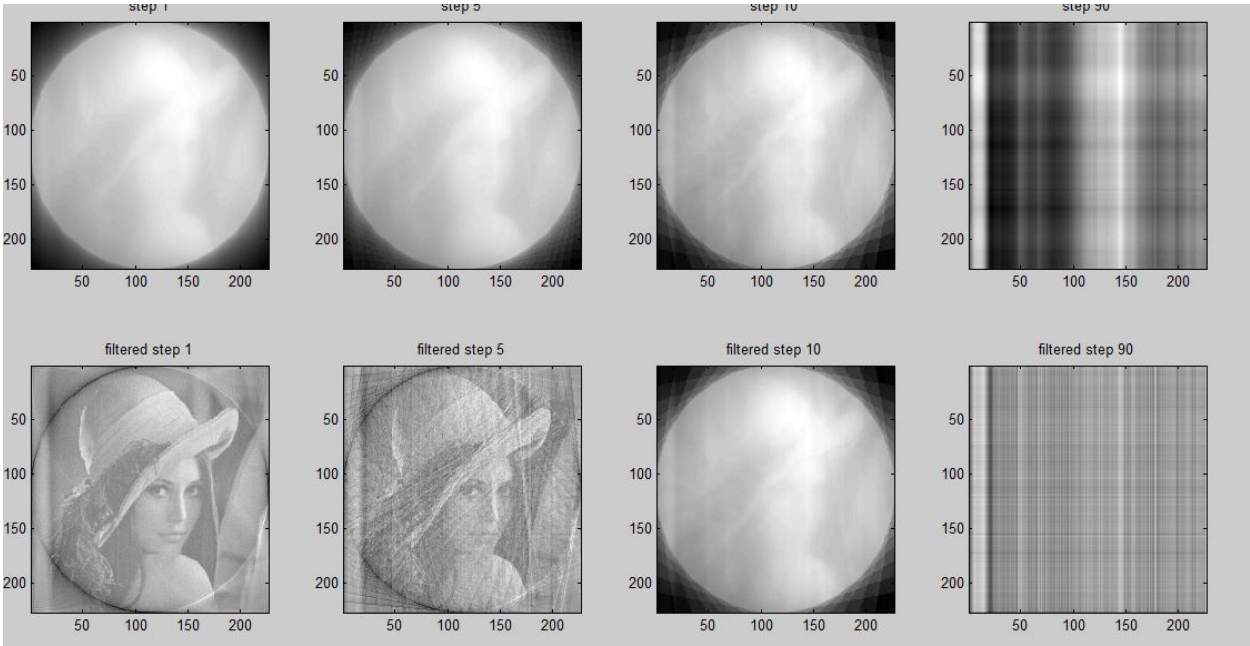

Fig.12 Different step sizes considered for simple and filtered backprojection of *lena.tiff*

For low step size, the objects can be distinguished but for large step size the original images can no longer be distinguished, until they only become some lines.

As a conclusion, I think it happens the same thing as in case of sampling. If the sample rate is enough high then we can see the original image but in case of too low sampling rate the result

becomes too noisy. Because, in fact, then we take the image at different angles and calculate the line from Radon transform, it is much like doing a sampling. In case of taking too low number of angles, the backprojection will no longer obtain the original image, especially at the filtered backprojection where FFT is also done and may appear the aliasing effect (because of that the Nyquist about having the sampling rate at least double maximum frequency of the source is no longer satisfied).