Tomographic Imaging: Projection and Backprojection Implementations in MATLAB

Refik Mert Cam
MIDDLE EAST TECHNICAL
UNIVERSITY
Electrical and Electronics Engineering
Ankara, Turkey
refik.cam@metu.edu.tr

***Abstract*—This document contains the implementations of projection and back projections in MATLAB, which are widely used in X-Ray imaging. The project has been conducted as the term project of EE415 Introduction to Medical Imaging course in 2019-2020 Fall semester.**

Keywords—tomographic imaging, projection, back projection, X-Ray imaging, MATLAB, GUI, forward/inverse problems, biomedical imaging

I. INTRODUCTION

In tomographic imaging modality, attenuation of X-ray beams can be used to reconstruct images. As in all imaging, X-ray imaging also includes forward and inverse problems. Forward problem is "the problem of calculating what should be observed for a particular model" **[1].** For X-ray imaging, the forward problem corresponds to the calculation of attenuation of X-rays, while they are passing through a body. As the forward problem, projection has been implemented in MATLAB. Inverse problem is "the problem of determining the nature of a physical feature by examining its effects" **[2].** The inverse problem for X-ray imaging corresponds to inference of the distribution of attenuation coefficients of a body from the measurements. As the inverse problem, back projection algorithm has been implemented in MATLAB. As the final part, a graphical user interface has been designed with MATLAB Appdesigner.

II. BRIEF HISTORY OF X-RAY IMAGING

In 1895, Wilhelm Conrad Roentgen, who was a Professor at Wuerzburg University in Germany, discovered X-rays, after he noticed a fluorescent glow of crystals near cathode-ray tubes. He reached the conclusion that when a high voltage supplied to a tube in which air is evacuated, a new type of beams was emitted. He also realized that these X-ray beams were able to pass through human tissue, but not bones and metals. The first experiment of X-ray imaging was a film of hand of Roentgen's wife, Bertha as shown in Figure 1.

Fig. 1. The film of hand of Roentgen's wife, Bertha **[3]**

Clinically, Dr. Edwin Frost (1866-1935) is credited with making the first diagnostic radiograph in the U.S. at Dartmouth on 3 February 1896 as shown in Figure 2 (left). E. Haschek and O. Lindenthal produced first contrast-enhanced radiograph of veins in hand on January 1896 as shown in Figure 2 (right).

Fig. 2. First American Medical X-ray (left), First Contrast-Enhanced Radipograph of Veins (right) **[4]**

In 1921, a French Scientist Andre Edmond Marie Bocage came up with the idea of tomography. The first tomography idea was moving X-ray source and detector in a synchronized way. This resulted in a blurred version of a iamge of cross-section of a body. However, Bocage did not build the equipment, in 1934 a Dutch Scientist Bernard Zeidses structured the X-ray tomography equipment. The first commercial computerized-tomography scanner was developed by the EMI Company. In 1979, Sir Godfrey Hounsfield won the Nobel Prize with Allan Cormack because of their outstanding achievements for the development of CT scanner **[4].**

III. THEORY OF TOMOGRAPHIC X-RAY IMAGING

While an X-ray beam traverses through a tissue, it is attenuated exponentially as in formula **(1):**

$$I = I_0 \exp\left(-\int \mu(x,y)\, ds\right) \tag{1}$$

$\boldsymbol{\mu}$**(x,y)** corresponds to attenuation coefficient dependent on position.

Therefore, total attenuation can be expressed by a natural logarithm as in formula **(2):**

$$p_\theta(r) = \ln\left(\frac{I}{I_0}\right) = -\int \mu(x,y)\, ds \tag{2}$$

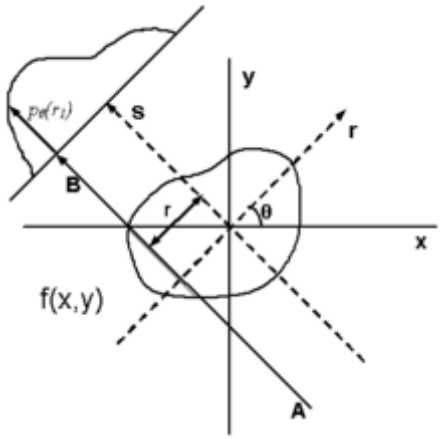

Fig. 3. The Coordinate System that is used for projection function **[5]**

Using the coordinate system in the figure 3, one can express projection function as in formula (3):

$$p_\theta(r) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(x\cos\theta + y\sin\theta - r)\, dx\, dy \tag{3}$$

There are different ways to reconstruct the image from its projection function:

- Backprojection
- Fourier Domain Approach
- Filtered Backprojection
- Iterative Methods
- Algebraic Reconstruction Technique
- Iterative Least Squares
- Simultaneous Iterative Reconstruction Technique
- Deep Learning Reconstruction

In this projection backprojection algorithm has been implemented as the reconstruction technique. Back-projection is very similar with projection, and it can be expressed as in formula (4):

$$f_b(x,y) = \int_0^{\pi}\int_{-\infty}^{\infty} p_\theta(t)\, \delta(xcos\theta + ysin\theta - t)\mathrm{d}t\mathrm{d}\theta \tag{4}$$

When we examine the projection and back-projection formulizations, we can see that the Fourier Transform of the reconstructed image is the multiplication of Fourier Transform of the image and (1/r). Thus, a filtering operation should be conducted in order to obtain better quality reconstructions. In the results and comments section, ramp filter has been used. As for the discussion and further comments section, the effects of Gaussian and Sinusoidal Filters are also discussed.

IV. THE IMPLEMENTATION OF PROJECTION

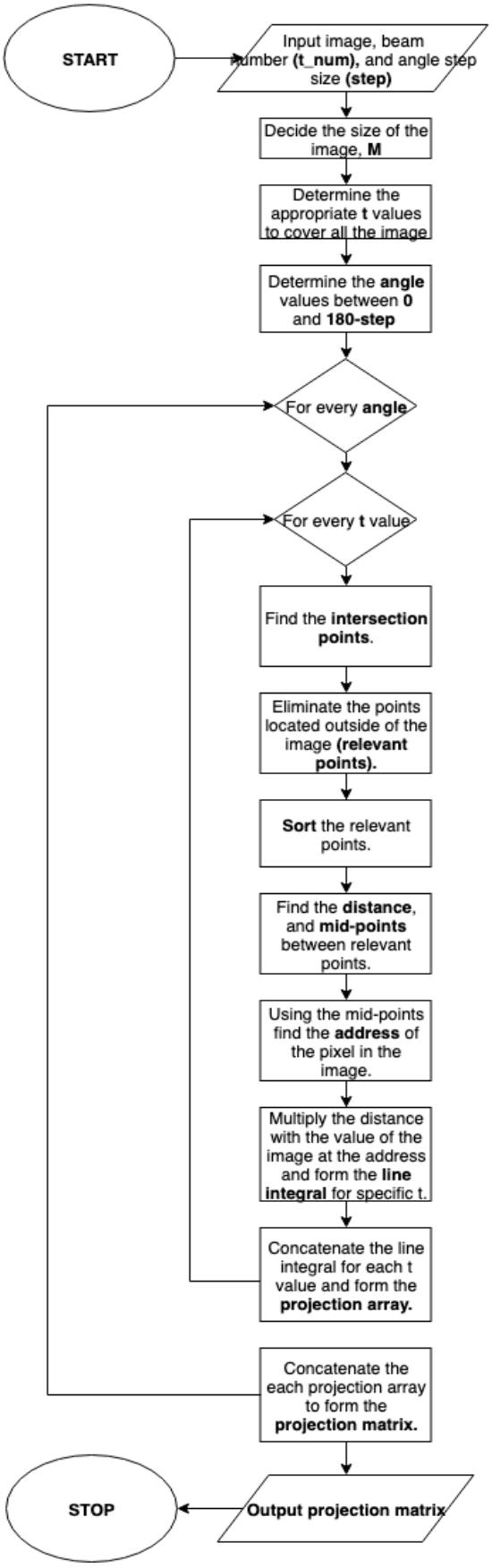

Fig. 4. The Flowchart of the Projection Algorithm

Projection Algorithm explanation:

1. The algorithm takes image, beam number, and angle step as inputs.
2. It decides the size of the image, M.
3. To cover the whole image, it creates an array of t = $[-M/2 \quad M/2]$ with element number equal to beam number.
4. Creates an angle array $[0 \quad 180 - step\ size]$ with the differences between consecutive two elements equal to step size.
5. For every angle
   a. For every t value
      i. Finds the intersection points.
      ii. Eliminates the intersection points which are outside of the image.
      iii. Sorts the relevant points.
      iv. Find the distance and mid-points of relevant points.
      v. Using the mid-points information, find the address of the relevant pixel.
      vi. Multiply the distances and relevant pixel values, then add them together to find the line integrals.
      vii. Concatenate each line integral to form the projection array.
   b. Concatenate each projection array to form the projection matrix.
6. It gives the projection matrix as output.

In short, the algorithm calculates the line integrals for specified image, beam number and angles. This is achieved by adding multiplications of distance that beam traverses with the image value at those pixels. The two for loop concatenates these line integrals first horizontally, then vertically to compose the projection matrix of the image.

While finding relevant points, I check whether the maximum element of the column (x,y pair) is less than or equal to M/2 and the minimum element of the column is greater than or equal to -M/2. If the point is in this interval, then I concatenate this point to the relevant_points array. We find relevant points because if the element is not in that interval, there is no corresponding pixel value in the image of interest.

V. THE IMPLEMENTATION OF BACK-PROJECTION

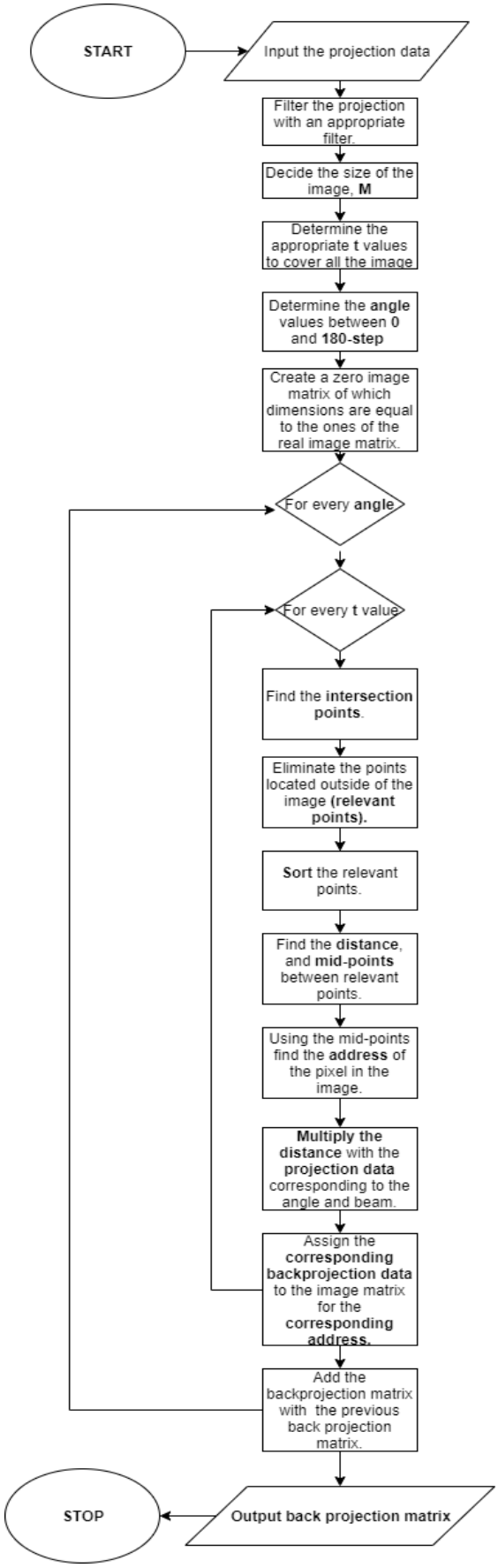

Fig. 5. The Flowchart of the Back-Projection Algorithm

Back-Projection Algorithm explanation:

Firstly, we filter the projection data with a ramp filter in order to eliminate the effect of point-spread function of the system, so we increase the spatial resolution.

Then, we follow the previous projection algorithm:

1. It decides the size of the image, M.
2. To cover the whole projection data, it creates an array of t = $[-M/2 \quad M/2]$ with element number equal to beam number.
3. Creates an angle array $[0 \quad 180 - step\ size]$ with the differences between consecutive two elements equal to step size.
4. For every angle
   a. For every t value
      i. Finds the intersection points.
      ii. Eliminates the intersection points which are outside of the image.
      iii. Sorts the relevant points.
      iv. Find the distance and mid-points of relevant points.
      v. Using the mid-points information, find the address of the relevant pixel.

Here, our back-projection algorithm differs from the previous projection algorithm. This time:

> We multiply each distance with the back projection data (corresponding to the angle and beam number), then it updates the corresponding address of the image matrix by adding this multiplication.

[We created the back-projection image matrix initially as a zero matrix of which dimensions are equal to the dimensions of the original image matrix, and then we update the corresponding address of this matrix by adding the back-projection value in each iteration.]

In the filtering part of our inverse problem, we want to eliminate the effects of the PSF of the system. I implemented the filtering in MATLAB with the following steps:

- First, I created the one-dimensional triangle shaped high pass filter with the length equal to beam number.
- I have used repmat (built-in function to replicate a vector for creating a matrix), and replicated the high-pass filter in the amount of angle number.
- Then, I have taken the FFT of the each projection vector corresponding to each angle with fft() built-in function
- I multiplied the high-pass matrix with the FFT of the projection matrix pointwise.
- As the final step, I have calculated the IFFT of the multiplication in the previous step, and I obtained the filtered projection data.

VI. GUI IMPLEMENTATION

A Graphical User Interface has been designed with MATLAB Appdesigner. In the GUI, there are following windows as it can be seen in Figure 6:

- Select your input image
- Select your filter type
- Projection and back
- Projection & Back Projection buttons
- Beam number slider
- Step size
- Angle Number
- Original Image
- Projection Image (Radon Transform)
- Reconstructed Image
- The Filter Plot
- Projection Plot at a Specific Angle

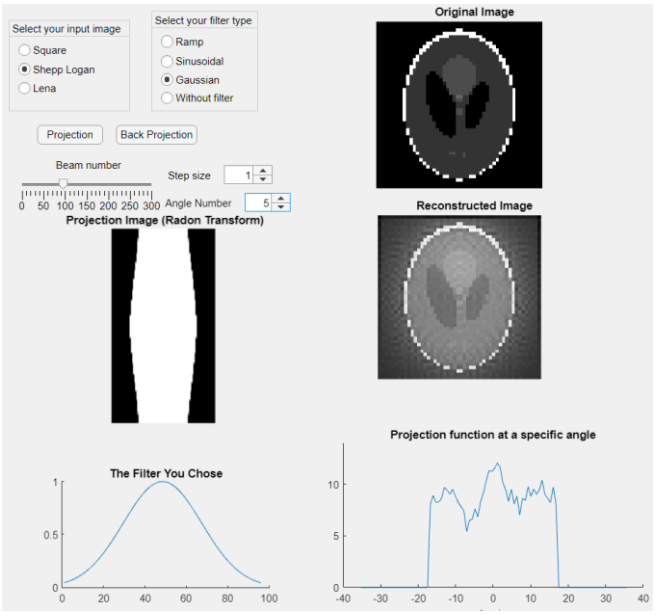

Fig. 6. GUI

While one is using the GUI, s/he should follow these steps:

- Select which image to be used as input
- Decide the beam number and step size for projection
- Push the Projection button
- Decide which filter to be used for back projection
- Push the Back Projection button
- By changing angle number, the projection function at a specific angle can be seen

VII. RESULTS AND COMMENTS

A. Results for Projection Algorithm

In this section, I have provided 4 outputs for each of 3 different images, which are sample image given in the algorithm, Square image, and Lena image. I have selected first image as the sample image because it is easy to calculate projection values by hand. Then, I have increased the complexity of images with Square and Lena, respectively.

I have compared the projection functions with Radon transform for Square and Lena images, while I have compared the projection of sample image with hand calculations.

First image: Sample image given in the algorithm

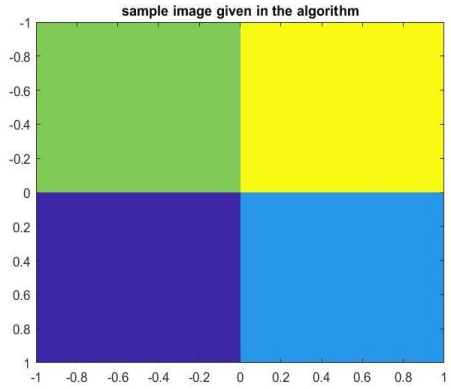

Fig. 7. The Sample Image

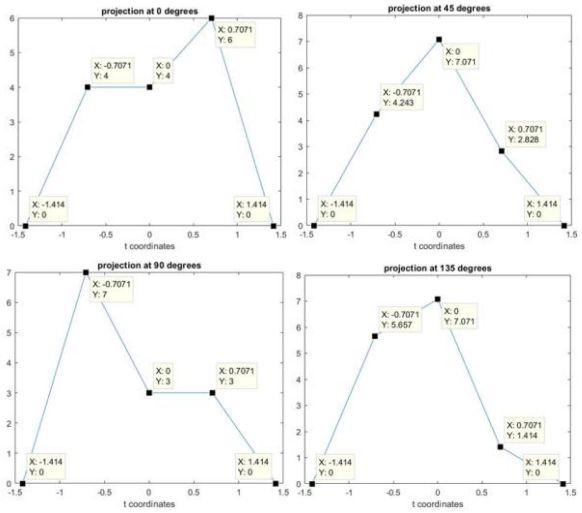

Fig. 8. Projection Functions of the sample image from different angles 0 degree (upper left), 45 degree (upper right), 90 degree (bottom left), 135 degree (bottom right)

Sample image = [1 2; 3 4]
Projection function at 0 degree = [0 4 4 6 0]
Projection function at 45 degree= [0 $3\sqrt{2}$ $5\sqrt{2}$ $2\sqrt{2}$ 0]
Projection function at 90 degree = [0 7 3 3 0]
Projection function at 135 degree = [0 $4\sqrt{2}$ $5\sqrt{2}$ $\sqrt{2}$ 0]

The calculated projection function is the same with hand calculations, but there is just a little difference at t=0. This is because of that there is no pixel value at t=0, and algorithm replicates the projection value for previous t.

Second image: Square image

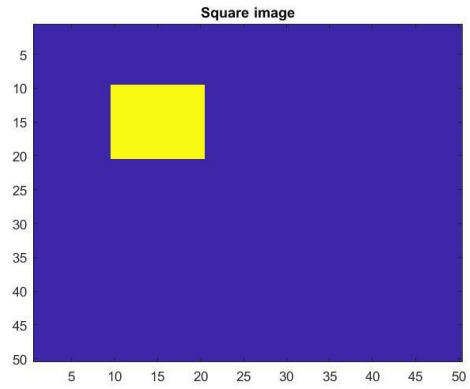

Fig. 8. The Square Image

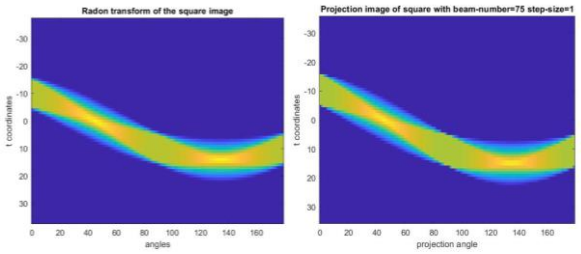

Fig. 9. Radon Transform of the Square Image (left), Projection Image of the Square Image (right)

As one can see from the figure 9, radon transform (built-in) and projection function (calculated by the algorithm) is pretty similar.

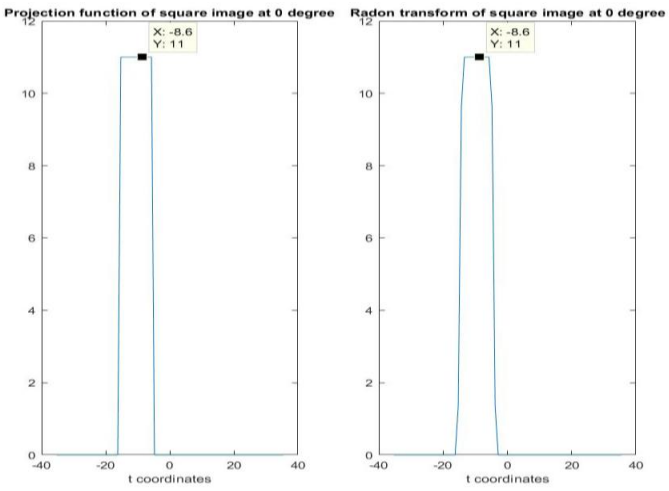

Fig. 10. Radon Transform of the Square Image (right), Projection Image of the Square Image (left) at 0 degree

For projection at 0 degree, it is expected to see a rectangular pulse since the image has zero values except a square region. Furthermore, the Radon transform, and projection function are similar as it can be seen in Figure 10. However, there is a little deviation because Radon function takes the center pixel as floor((size(image)+1)/2).

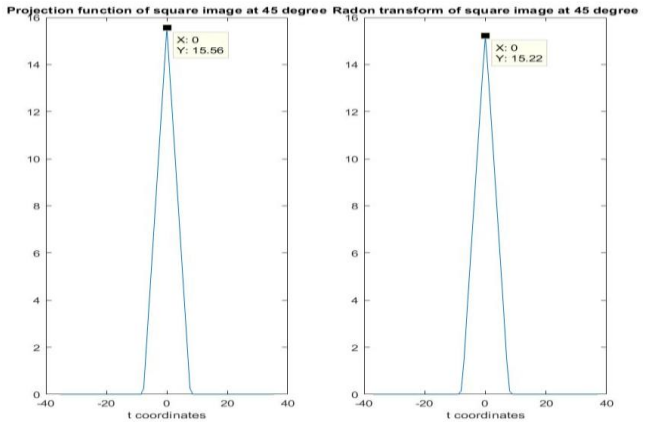

Fig. 11. Radon Transform of the Square Image (right), Projection Image of the Square Image (left) at 45 degree

For projection at 45 degree, a triangular projection function is expected because beams traverse through an equilateral quadrangle. It is also consistent with the Radon transform of the square image as shown in Figure 11.

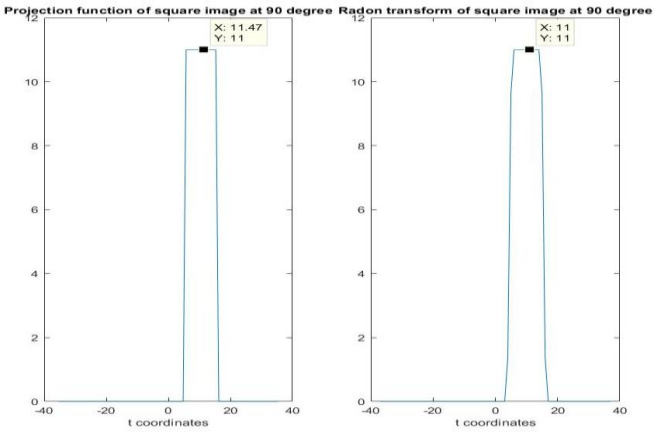

Fig. 12. Radon Transform of the Square Image (right), Projection Image of the Square Image (left) at 90 degree

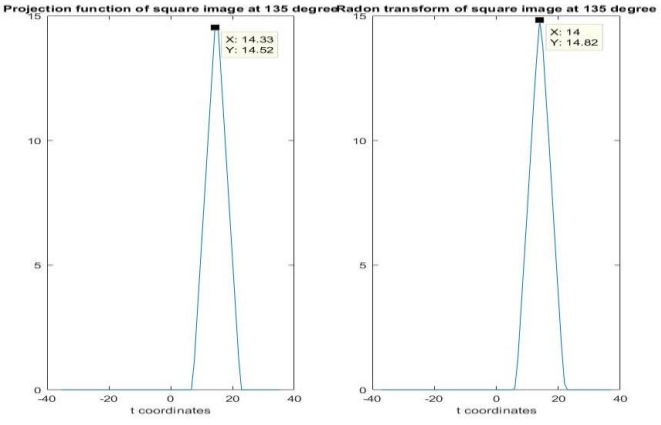

Fig. 13. Radon Transform of the Square Image (right), Projection Image of the Square Image (left) at 135 degree

The projection functions at 90 and 135 degrees are the symmetric versions of the projections at 0 and 45 degrees. This is expected because the Square Image show rotational symmetricity.

Third image: Lena image

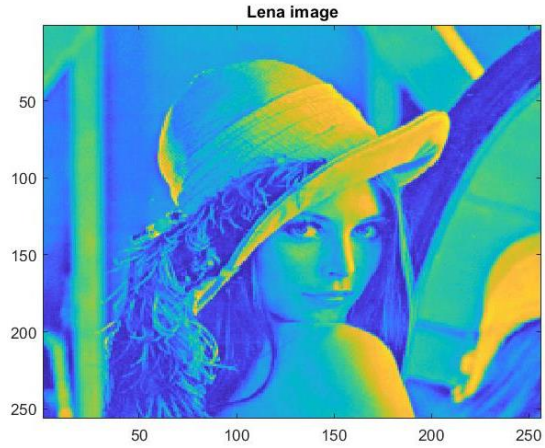

Fig. 14. The Lena Image

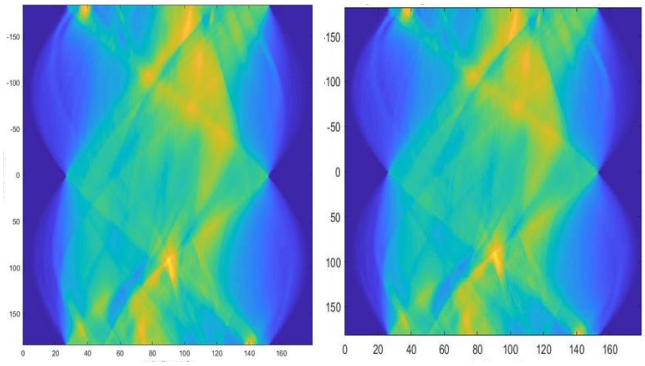

Fig. 15. Radon Transform of the Lena Image (left), Projection Image of the Square Image (right)

From Figure 15, we can infer that our algorithm works properly for complex images like Lena, since projection image is similar with the image of Radon transform.

Since Lena is a complex image, which makes intuitive inference hard, I compared projection functions and Radon transforms for 4 different angles in addition to the general comparison shown at Figure 15. One can see that projection results and Radon transforms are very similar.

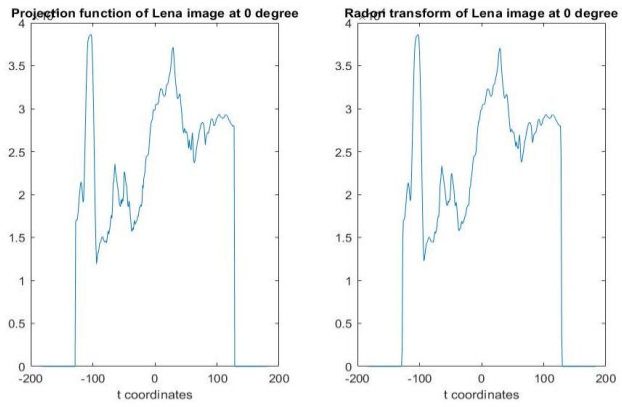

Fig. 16. Radon Transform of the Lena Image (right), Projection Image of the Lena Image (left) at 0 degree

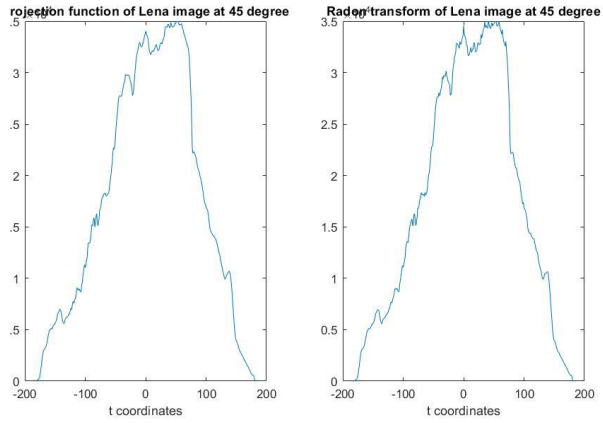

Fig. 17. Radon Transform of the Lena Image (right), Projection Image of the Lena Image (left) at 45 degree

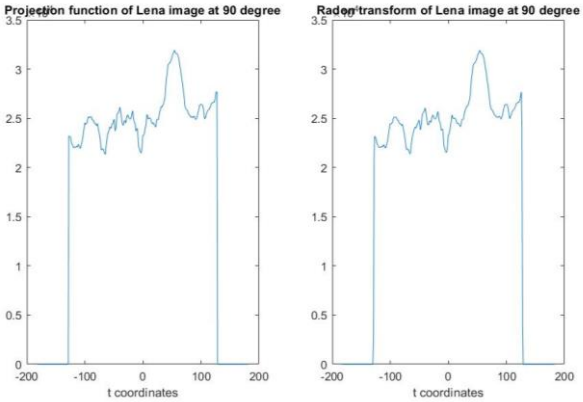

Fig. 18. Radon Transform of the Lena Image (right), Projection Image of the Lena Image (left) at 90 degree

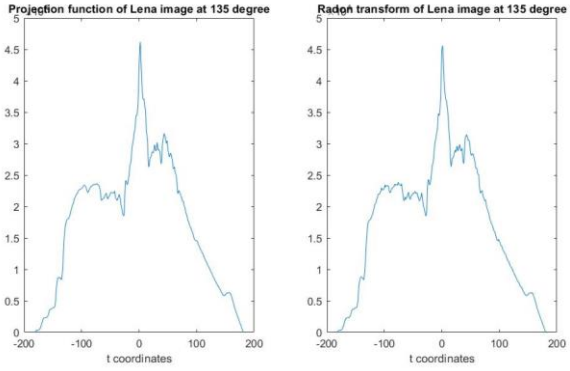

Fig. 19. Radon Transform of the Lena Image (right), Projection Image of the Lena Image (left) at 135 degree

B. Results for Back-Projection Algorithm

In the comments for each of the three different outputs, I have utilized SSIM, PSNR, and MSE indicators to make a quantitative analysis. The indicators and their interpretations are explained below. As an additional comment, SNR (signal to noise ratio) may also be used for quantitative analysis, however, I haven't used the SNR because first we need to corrupt the image with a noise (Gaussian, Poisson or any familiar random noise function) in order to utilize the SNR.

In MATLAB, the indicators are implemented with SSIM, PSNR, and MSE built-in functions. Their inputs are the reconstructed images and reference images.

Structural Similarity Index (SSIM): It is used for measuring the similarity between two images. It is calculated on image windows for both reference and the reconstructed image. It is formulated between two windows of common size *NxN* as in formula (5):

$$\text{SSIM}(x,y) = \frac{(2\mu_x\mu_y+c_1)(2\sigma_{xy}+c_1)}{(\mu_x^2+\mu_y^2+c_1)(\sigma_x^2+\sigma_y^2+c_2)} \quad (5)$$

μ_x: average of x

μ_y: average of y

σ_x^2: variance of x

σ_y^2: variance of y

σ_{xy}: covariance of x and y

$c_1 = (k_1 L)^2$, $c_1 = (k_1 L)^2$ variables to stabilize the division in case of small denominator

L: dynamic range of the pixel-values

k_1=0.01 and k_2=0.03 as default

When we decide an image is reconstructed well or not, SSIM may guide us with other quantitative comparison measures. If SSIM is high (close to 1), the reconstructed image resembles the reference image enough and then we can say that the image has been reconstructed well. If SSIM is low (close to 0), the image doesn't look like reference image, and is not reconstructed well **[6].**

Mean Square Error (MSE): It measures the average of the squared error pixel by pixel for the reconstructed and the reference images. Its formulation is as follows in formula (6):

$$\text{MSE}(x,y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - y_i)^2 \quad (6)$$

If the MSE is low, we can say that the reconstructed image values are close to the ones of the reference image, so we can say that the image is reconstructed well for low MSE values. When the MSE is high, it is vice versa **[7].**

Peak Signal to Noise Ratio (PSNR): PSNR is the proportion of the square of maximum pixel value to the mean square error in dB magnitude **[8].** It is formulated as follows in formula (7):

$$\text{PSNR} = 10\log_{10}\left(\frac{MAX^2}{MSE}\right) \quad (7)$$

MAX: maximum pixel value in the original image

MSE: mean square error between the reconstructed and reference images

As in the SSIM case, high PSNR value is an indicator of better reconstruction.

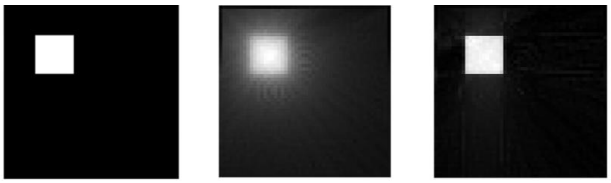

Fig. 20. (left) The square image (middle) Reconstructed image without filtering (right) Reconstructed image with filtering when the number of beams=100 and step size=1°. Ramp type filter is used.

Table 1. Quantitative Analysis of the Reconstruction for the Square Image

	PSNR	SSIM	MSE
Back projection	5.4210	0.0950	0.2870
Filtered-back projection	20.0188	0.5110	0.0100

From the Figure 20, we can see that when we reconstructed the image with filtering, the outcome becomes better. However, there is still differences between reconstructed and the reference images, this is because we use only a limited number of beams while we do projection and back projection. If we increase the beam number and decrease the step size, it is clear that we obtain better reconstructions, however, this increases the computational complexity and therefore the durations for the projection and back projection processes. Furthermore, we can see that the reconstructed image without filtering is diffused due to the point-spread function, therefore, the spatial resolution is low. When we filter the projection, we actually eliminate the effect of the PSF, so we obtain better reconstructions.
Our quantitative analysis in Table 1. also verifies our previous comments. PSNR and SSIM values are higher and MSE is lower for filtered back projection with respect to the ones of back projection without filtering.

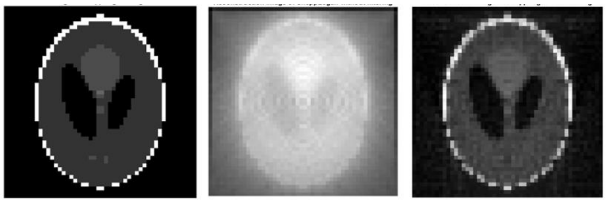

Figure 21 (left) The Shepp Logan image (middle) Reconstructed image without filtering (right) Reconstructed image with filtering when the number of beams=100 and step size=1°. Ramp type filter is used.

Table 2. Quantitative Analysis of the Reconstruction for the Shepp Logan Image

	PSNR	SSIM	MSE
Back projection	10.0232	0.1883	0.0995
Filtered-back projection	19.1922	0.4515	0.0120

By looking at the Figure 21, we can make the similar comments for output 2 as we did for output 1. Additionally, we should say that as the Shepp Logan image is more complex than the square image, its reconstruction without filtering is blurrier than the one for the square image. This is because more PSF distributions are overlapped in the reconstruction of Shepp Logan.

Our quantitative analysis in Table 2. also verifies the previous comments. PSNR and SSIM values are higher and the MSE value is lower for the filtered back projection with respect to the ones of back projection without filtering.

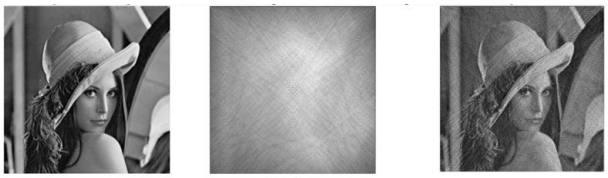

Figure 22 (left) The Lena image (middle) Reconstructed image without filtering (right) Reconstructed image with filtering when the number of beams=200 and step size=1°. Ramp type filter is used.

Table 3. Quantitative Analysis of the Reconstruction for the Lena Image

	PSNR	SSIM	MSE
Back projection	14.3443	0.0470	0.0368
Filtered-back projection	25.4343	0.1938	0.0029

Again, from the Figure 22, we observe that the result for filtered back projection is better than the reconstruction with only back projection. This is because we eliminate the effect of PSF with filtering. Furthermore, we observe that in the filtered back projection reconstruction, there exists linear artifacts. The reason for that is the limited number of beams. And we also see that the reconstructed image is faded than the original image, this qualitatively shows us the contrast resolution phenomenon. While making projection and back projection, I have increased the beam number to 200 from 100 because the Lena image is more complex than the previous ones. The drawback of the increasing beam numbers was longer durations.

Again, quantitative analysis in Table 3. verifies our qualitative analysis. PSNR and SSIM values are higher and MSE is lower for filtered back projection with respect to the ones of back projection without filtering.

VIII. DISCUSSION AND FURTHER COMMENTS

As the first further comment, when we increase the beam number and decrease the step size, the reconstruction quality increases whereas due to the increased computational complexity the duration also increases. This can be seen in Figure 23.

Figure 23 (left) Reconstructed Lena image with ramp filtering when the number of beams=100 and step size=1°. (right) Reconstructed Lena image with ramp filtering when the number of beams=200 and step size=1°.

As the second further comment, sinusoidal, gaussian, and ramp filters have been used for filtering operations. We can see from the Figure 24 that the contrast of images changes with different filters. This is because as one can see in Figure 25 the bandwidths of filters are different and this causes different frequency components are treated differently especially near high frequencies. Furthermore, the Gaussian filter is not 0 ‘zero’ for DC components, so it doesn’t eliminate totally the DC component, which corresponds to the smooth sections of the image.

Figure 24 (left) Reconstructed Lena image with Ramp filter (middle) Reconstructed Lena image with Sinusoidal filter (right) Reconstructed Lena image with Gaussian filter when the number of beams=200 and step size=1°.

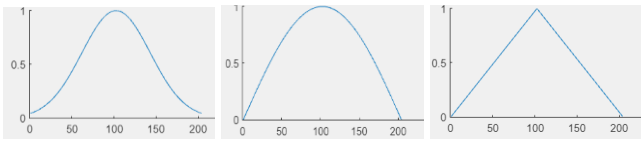

Figure 25 (left) Gaussian filter (middle) Sinusoidal filter (right) Ramp filter

As a further discussion, there is no noise in our projections and back projection algorithms, it may be included a noise to the original image and the outcomes can be examined. A noise is a high frequency component, and our filters show also high-pass characteristics. Thus, the filters may increase the noise component and corrupt our reconstructed images.

REFERENCES

[1]"forward problem | Encyclopedia.com", *Encyclopedia.com*, 2019. [Online]. Available:
[2]"inverse problem | Encyclopedia.com", *Encyclopedia.com*, 2019. [Online]. Available:
[3]"History", *Nde-ed.org*, 2019. [Online]. Available:
[4]*Inis.iaea.org*, 2019. [Online]. Available:
[5]"Tomographic reconstruction - Wikipedia", *Wikizeroo.org*, 2019. [Online]. Available:
[6]"Structural similarity - Wikipedia", *Wikizeroo.org*, 2019. [Online]. Available:.
[7]"Mean squared error - Wikipedia", *Wikizeroo.org*, 2019. [Online]. Available:
[8]"Peak signal-to-noise ratio - Wikipedia", *Wikizeroo.org*, 2019. [Online]. Available:

APPENDIX

APPENDIX.A: Here the MATLAB code is given as the appendix for interested readers.

```
classdef ee415_term_project < matlab.apps.AppBase
    % Properties that correspond to app components
    properties (Access = public)
        UIFigure                    matlab.ui.Figure
        SelectyourinputimageButtonGroup  matlab.ui.container.ButtonGroup
        SquareButton                matlab.ui.control.RadioButton
        SheppLoganButton            matlab.ui.control.RadioButton
        LenaButton                  matlab.ui.control.RadioButton
        SelectyourfiltertypeButtonGroup  matlab.ui.container.ButtonGroup
        RampButton                  matlab.ui.control.RadioButton
        SinusoidalButton            matlab.ui.control.RadioButton
        GaussianButton              matlab.ui.control.RadioButton
        WithoutfilterButton         matlab.ui.control.RadioButton
        ProjectionButton            matlab.ui.control.Button
        BeamnumberSliderLabel       matlab.ui.control.Label
        BeamnumberSlider            matlab.ui.control.Slider
        StepsizeSpinnerLabel        matlab.ui.control.Label
        StepsizeSpinner             matlab.ui.control.Spinner
        UIAxes                      matlab.ui.control.UIAxes
        UIAxes_2                    matlab.ui.control.UIAxes
        UIAxes2                     matlab.ui.control.UIAxes
        BackProjectionButton        matlab.ui.control.Button
        UIAxes3                     matlab.ui.control.UIAxes
        UIAxes4                     matlab.ui.control.UIAxes
        AngleNumberSpinnerLabel     matlab.ui.control.Label
        AngleNumberSpinner          matlab.ui.control.Spinner
    end

    properties (Access = public)
        projection_public
        projection_filtered_public
        t_public
        theta_public
        M_public
    end



    methods (Access = private)
        % Selection changed function: SelectyourinputimageButtonGroup
        function SelectyourinputimageButtonGroupSelectionChanged(app, event)
            selectedButton = app.SelectyourinputimageButtonGroup.SelectedObject;

            if app.SquareButton.Value
                matdata=load('square.mat');
                image=matdata.square;
                [M M]=size(image);
                app.M_public=M;
                gray_original=mat2gray(image);
                imshow(gray_original,'Parent',app.UIAxes);
            end
```

```
    if app.SheppLoganButton.Value
        matdata=load('SheppLogan.mat');
        image=matdata.SheppLogan;
        [M M]=size(image);
        app.M_public=M;
        gray_original=mat2gray(image);
        imshow(gray_original,'Parent',app.UIAxes);
    end

    if app.LenaButton.Value
        matdata=load('lena.mat');
        image=matdata.lena;
        [M M]=size(image);
        app.M_public=M;
        gray_original=mat2gray(image);
        imshow(gray_original,'Parent',app.UIAxes);
    end

end
% Selection changed function: SelectyourfiltertypeButtonGroup
function SelectyourfiltertypeButtonGroupSelectionChanged(app, event)

    selectedButton = app.SelectyourfiltertypeButtonGroup.SelectedObject;

    value = app.BeamnumberSlider.Value;
    M=app.M_public;
    t_num=value;
    t=linspace(-M*sqrt(2)/2,M*sqrt(2)/2,t_num);
    app.t_public=t;
    step=app.StepsizeSpinner.Value;
    theta=0:step:(180-step);
    app.theta_public=theta;


    projection=app.projection_public;
    t=app.t_public;
    theta=app.theta_public;

    if app.RampButton.Value
        high_pass=triang(length(t))';
        plot(high_pass,'Parent',app.UIAxes3);
    end
    if app.GaussianButton.Value
        ind=linspace(0,10,length(t));
        high_pass=gaussmf(ind,[2 5])';
        plot(high_pass,'Parent',app.UIAxes3);
    end
    if app.SinusoidalButton.Value
        ind=linspace(0,pi,length(t));
        high_pass=sin(ind);
        plot(high_pass,'Parent',app.UIAxes3);
    end
    if app.WithoutfilterButton.Value
        plot(ones(length(t)),'color',[0 0 1],'Parent',app.UIAxes3);
```

```
            end


        end
        % Button pushed function: ProjectionButton
        function ProjectionButtonPushed(app, event)
            if app.SquareButton.Value
                matdata=load('square.mat');
                image=matdata.square;
                [M M]=size(image);
                app.M_public=M;
                x_val=-M/2:M/2;
                y_val=-M/2:M/2;
            end
            if app.SheppLoganButton.Value
                matdata=load('SheppLogan.mat');
                image=matdata.SheppLogan;
                [M M]=size(image);
                x_val=-M/2:M/2;
                y_val=-M/2:M/2
            end

            if app.LenaButton.Value
                matdata=load('lena.mat');
                image=matdata.lena;
                [M M]=size(image);
                x_val=-M/2:M/2;
                y_val=-M/2:M/2;
            end
            value = app.BeamnumberSlider.Value;
            t_num=value;
            t=linspace(-M*sqrt(2)/2,M*sqrt(2)/2,t_num);
            app.t_public=t;
            step=app.StepsizeSpinner.Value;
            theta=0:step:(180-step);
            app.theta_public=theta;
            projection=[];
            for i=1:length(theta)
                projection_i_j=[];
                intersection_points_y=[];
                intersection_points_x=[];
                for j=1:length(t)
                    intersection_points_y=(t(j)-
x_val*cos(theta(i)/180*pi))/sin(theta(i)/180*pi);
                    points_y=[intersection_points_y; y_val];
                    points_y=wthresh(points_y,'h',1e-10);
                    intersection_points_x=(t(j)-
y_val*sin(theta(i)/180*pi))/cos(theta(i)/180*pi);
                    points_x=[x_val; intersection_points_x];
                    points_x=wthresh(points_x,'h',1e-10);
                    points=[points_x points_y];
                    relevant_points=[];
                    for k=1:size(points,2)
                        if(round(max(points(:,k)),8)<= round(M/2,8) &&
round(min(points(:,k)),8)>=round(-M/2,8))
```

```
                        relevant_points=[relevant_points points(:,k)];
                    end
                end
                relevant_points=rmmissing(relevant_points,2);

                %sorting relevant points
                relevant_points=unique(round(relevant_points',8),'rows');
                dist=[];
                mid_points=[];
                for m=1:(size(relevant_points,1)-1)
                    dist=[dist; sqrt(sum((relevant_points(m,:)-
relevant_points(m+1,:)).*(relevant_points(m,:)-relevant_points(m+1,:))))];
                    mid_points=[mid_points;
relevant_points(m,:)/2+relevant_points(m+1,:)/2];
                end

                if ~isempty(mid_points)
                    rowdata=M/2-floor(mid_points(:,1));
                    columndata=M/2+ceil(mid_points(:,2));
                    address=[rowdata columndata];
                end

                if isempty(mid_points)
                    projection_i_j=[projection_i_j 0];
                else
                    r=1;
                    add=0;
                    for p=1:length(dist)
                        add=add+image(address(r,1),address(r,2))*dist(p);
                        r=r+1;
                    end
                    projection_i_j=[projection_i_j add];
                end
            end
            projection=[projection; projection_i_j];
        end
        imshow(projection,'Parent',app.UIAxes_2);
        app.projection_public=projection;


    end
    % Button pushed function: BackProjectionButton
    function BackProjectionButtonPushed(app, event)
        M=app.M_public;
        projection=app.projection_public;
        value = app.BeamnumberSlider.Value;
        M=app.M_public;
        t_num=value;
        t=linspace(-M*sqrt(2)/2,M*sqrt(2)/2,t_num);
        step=app.StepsizeSpinner.Value;
        theta=0:step:(180-step);

        if app.RampButton.Value
            high_pass=triang(length(t))';
            plot(high_pass,'Parent',app.UIAxes3);
```

```
            high_pass=repmat(high_pass,length(theta),1);
            %%passing to frequency domain, multiplication, and then back to spatial
            %%domain
            fourier=fft(projection');
            fourier=fourier';
            projection_freqdom=fourier.*high_pass;
            projection_filtered=ifft(projection_freqdom');
            projection_filtered=projection_filtered';
            app.projection_filtered_public=projection_filtered;
        end
        if app.GaussianButton.Value
            ind=linspace(0,10,length(t));
            high_pass=gaussmf(ind,[2 5]);
            plot(high_pass,'Parent',app.UIAxes3);
            high_pass=repmat(high_pass,length(theta),1);
            %%passing to frequency domain, multiplication, and then back to spatial
            %%domain
            fourier=fft(projection');
            fourier=fourier';
            projection_freqdom=fourier.*high_pass;
            projection_filtered=ifft(projection_freqdom');
            projection_filtered=projection_filtered';
            app.projection_filtered_public=projection_filtered;
        end
        if app.SinusoidalButton.Value
            ind=linspace(0,pi,length(t));
            high_pass=sin(ind);
            plot(high_pass,'Parent',app.UIAxes3);
            high_pass=repmat(high_pass,length(theta),1);
            %%passing to frequency domain, multiplication, and then back to spatial
            %%domain
            fourier=fft(projection');
            fourier=fourier';
            projection_freqdom=fourier.*high_pass;
            projection_filtered=ifft(projection_freqdom');
            projection_filtered=projection_filtered';
            app.projection_filtered_public=projection_filtered;
        end
        if app.WithoutfilterButton.Value
            projection_filtered=projection;
            filter=ones(length(t));
            plot(filter,'Parent',app.UIAxes3);
            app.projection_filtered_public=projection_filtered;
        end
        x_val=-M/2:M/2;
        y_val=-M/2:M/2;
        back_projection=zeros(M,M);
        for i=1:length(theta)
            intersection_points_y=[];
            intersection_points_x=[];
            for j=1:length(t)
                intersection_points_y=(t(j)-
x_val*cos(theta(i)/180*pi))/sin(theta(i)/180*pi);
                points_y=[intersection_points_y; y_val];
                points_y=wthresh(points_y,'h',1e-10);
```

```
                    intersection_points_x=(t(j)-
y_val*sin(theta(i)/180*pi))/cos(theta(i)/180*pi);
                    points_x=[x_val; intersection_points_x];
                    points_x=wthresh(points_x,'h',1e-10);
                    points=[points_x points_y];
                    relevant_points=[];
                    for k=1:size(points,2)
                        if(round(max(points(:,k)),8)<= round(M/2,8) &&
round(min(points(:,k)),8)>=round(-M/2,8))
                            relevant_points=[relevant_points points(:,k)];
                        end
                    end
                    relevant_points=rmmissing(relevant_points,2);
                    %sorting relevant points
                    relevant_points=unique(round(relevant_points',8),'rows');
                    dist=[];
                    mid_points=[];
                    for m=1:(size(relevant_points,1)-1)
                        dist=[dist; sqrt(sum((relevant_points(m,:)-
relevant_points(m+1,:)).*(relevant_points(m,:)-relevant_points(m+1,:))))];
                        mid_points=[mid_points;
relevant_points(m,:)/2+relevant_points(m+1,:)/2];
                    end

                    if ~isempty(mid_points)
                        rowdata=M/2-floor(mid_points(:,1));
                        columndata=M/2+ceil(mid_points(:,2));
                        address=[rowdata columndata];
                    end

                    if ~isempty(mid_points)
                        u=size(mid_points,1);
                        for m=1:(u-1)

back_projection(rowdata(m),columndata(m))=projection_filtered(i,j)*dist(m,1)+back_proje
ction(rowdata(m),columndata(m));
                        end
                    end
                end
            end
            grayrecon=mat2gray(abs(back_projection));
            imshow(grayrecon,'Parent',app.UIAxes2);
        end
        % Value changed function: AngleNumberSpinner
        function AngleNumberSpinnerValueChanged(app, event)
            value = app.AngleNumberSpinner.Value;
            value1 = app.BeamnumberSlider.Value;
            M=app.M_public;
            t_num=value1;
            t=linspace(-M*sqrt(2)/2,M*sqrt(2)/2,t_num);
            projection=app.projection_public;
            plot(t,projection(value,:),'Parent',app.UIAxes4);

        end
    end
```

```
    % App initialization and construction
    methods (Access = private)
        % Create UIFigure and components
        function createComponents(app)
            % Create UIFigure
            app.UIFigure = uifigure;
            app.UIFigure.AutoResizeChildren = 'off';
            app.UIFigure.Position = [100 100 786 725];
            app.UIFigure.Name = 'UI Figure';
            app.UIFigure.Resize = 'off';
            % Create SelectyourinputimageButtonGroup
            app.SelectyourinputimageButtonGroup = uibuttongroup(app.UIFigure);
            app.SelectyourinputimageButtonGroup.AutoResizeChildren = 'off';
            app.SelectyourinputimageButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @SelectyourinputimageButtonGroupSelectionChanged, true);
            app.SelectyourinputimageButtonGroup.Title = 'Select your input image';
            app.SelectyourinputimageButtonGroup.Position = [24 592 140 106];
            % Create SquareButton
            app.SquareButton = uiradiobutton(app.SelectyourinputimageButtonGroup);
            app.SquareButton.Text = 'Square';
            app.SquareButton.Position = [11 60 61 22];
            app.SquareButton.Value = true;
            % Create SheppLoganButton
            app.SheppLoganButton = uiradiobutton(app.SelectyourinputimageButtonGroup);
            app.SheppLoganButton.Text = 'Shepp Logan';
            app.SheppLoganButton.Position = [11 38 94 22];
            % Create LenaButton
            app.LenaButton = uiradiobutton(app.SelectyourinputimageButtonGroup);
            app.LenaButton.Text = 'Lena';
            app.LenaButton.Position = [11 16 65 22];
            % Create SelectyourfiltertypeButtonGroup
            app.SelectyourfiltertypeButtonGroup = uibuttongroup(app.UIFigure);
            app.SelectyourfiltertypeButtonGroup.AutoResizeChildren = 'off';
            app.SelectyourfiltertypeButtonGroup.SelectionChangedFcn =
createCallbackFcn(app, @SelectyourfiltertypeButtonGroupSelectionChanged, true);
            app.SelectyourfiltertypeButtonGroup.Title = 'Select your filter type';
            app.SelectyourfiltertypeButtonGroup.Position = [189 592 123 116];
            % Create RampButton
            app.RampButton = uiradiobutton(app.SelectyourfiltertypeButtonGroup);
            app.RampButton.Text = 'Ramp';
            app.RampButton.Position = [11 70 58 22];
            app.RampButton.Value = true;
            % Create SinusoidalButton
            app.SinusoidalButton = uiradiobutton(app.SelectyourfiltertypeButtonGroup);
            app.SinusoidalButton.Text = 'Sinusoidal';
            app.SinusoidalButton.Position = [11 47 77 22];
            % Create GaussianButton
            app.GaussianButton = uiradiobutton(app.SelectyourfiltertypeButtonGroup);
            app.GaussianButton.Text = 'Gaussian';
            app.GaussianButton.Position = [11 26 73 22];
            % Create WithoutfilterButton
            app.WithoutfilterButton =
uiradiobutton(app.SelectyourfiltertypeButtonGroup);
            app.WithoutfilterButton.Text = 'Without filter';
            app.WithoutfilterButton.Position = [11 5 89 22];
```

```
            % Create ProjectionButton
            app.ProjectionButton = uibutton(app.UIFigure, 'push');
            app.ProjectionButton.ButtonPushedFcn = createCallbackFcn(app,
@ProjectionButtonPushed, true);
            app.ProjectionButton.Position = [57 554 76 22];
            app.ProjectionButton.Text = 'Projection';
            % Create BeamnumberSliderLabel
            app.BeamnumberSliderLabel = uilabel(app.UIFigure);
            app.BeamnumberSliderLabel.HorizontalAlignment = 'right';
            app.BeamnumberSliderLabel.Position = [73 520 81 22];
            app.BeamnumberSliderLabel.Text = 'Beam number';
            % Create BeamnumberSlider
            app.BeamnumberSlider = uislider(app.UIFigure);
            app.BeamnumberSlider.Limits = [0 300];
            app.BeamnumberSlider.Position = [39 506 150 3];
            app.BeamnumberSlider.Value = 1;
            % Create StepsizeSpinnerLabel
            app.StepsizeSpinnerLabel = uilabel(app.UIFigure);
            app.StepsizeSpinnerLabel.HorizontalAlignment = 'right';
            app.StepsizeSpinnerLabel.Position = [203 508 55 22];
            app.StepsizeSpinnerLabel.Text = 'Step size';
            % Create StepsizeSpinner
            app.StepsizeSpinner = uispinner(app.UIFigure);
            app.StepsizeSpinner.Position = [273 508 56 22];
            % Create UIAxes
            app.UIAxes = uiaxes(app.UIFigure);
            title(app.UIAxes, 'Original Image')
            app.UIAxes.PlotBoxAspectRatio = [1 0.618951612903226 0.618951612903226];
            app.UIAxes.Box = 'on';
            app.UIAxes.XColor = [0.9412 0.9412 0.9412];
            app.UIAxes.YColor = [0.9412 0.9412 0.9412];
            app.UIAxes.ZColor = [0.9412 0.9412 0.9412];
            app.UIAxes.Color = [0.9412 0.9412 0.9412];
            app.UIAxes.Position = [360 444 404 274];
            % Create UIAxes_2
            app.UIAxes_2 = uiaxes(app.UIFigure);
            title(app.UIAxes_2, 'Projection Image (Radon Transform)')
            app.UIAxes_2.XColor = [0.9412 0.9412 0.9412];
            app.UIAxes_2.YColor = [0.9412 0.9412 0.9412];
            app.UIAxes_2.ZColor = [0.9412 0.9412 0.9412];
            app.UIAxes_2.Color = [0.9412 0.9412 0.9412];
            app.UIAxes_2.Position = [1 184 407 293];
            % Create UIAxes2
            app.UIAxes2 = uiaxes(app.UIFigure);
            title(app.UIAxes2, 'Reconstructed Image')
            xlabel(app.UIAxes2, 'X')
            ylabel(app.UIAxes2, 'Y')
            app.UIAxes2.PlotBoxAspectRatio = [1 0.881028938906752 0.881028938906752];
            app.UIAxes2.GridColor = [0.9412 0.9412 0.9412];
            app.UIAxes2.MinorGridColor = [0.9412 0.9412 0.9412];
            app.UIAxes2.XColor = [0.9412 0.9412 0.9412];
            app.UIAxes2.YColor = [0.9412 0.9412 0.9412];
            app.UIAxes2.ZColor = [0.9412 0.9412 0.9412];
            app.UIAxes2.Color = [0.9412 0.9412 0.9412];
            app.UIAxes2.Position = [360 207 404 287];
```

```
            % Create BackProjectionButton
            app.BackProjectionButton = uibutton(app.UIFigure, 'push');
            app.BackProjectionButton.ButtonPushedFcn = createCallbackFcn(app,
@BackProjectionButtonPushed, true);
            app.BackProjectionButton.Position = [148 554 94 22];
            app.BackProjectionButton.Text = 'Back Projection';
            % Create UIAxes3
            app.UIAxes3 = uiaxes(app.UIFigure);
            title(app.UIAxes3, 'The Filter You Chose')
            app.UIAxes3.PlotBoxAspectRatio = [1 0.511254019292605 0.511254019292605];
            app.UIAxes3.Color = [0.9412 0.9412 0.9412];
            app.UIAxes3.Position = [27 22 356 163];
            % Create UIAxes4
            app.UIAxes4 = uiaxes(app.UIFigure);
            title(app.UIAxes4, 'Projection function at a specific angle')
            xlabel(app.UIAxes4, 't axis')
            app.UIAxes4.Color = [0.9412 0.9412 0.9412];
            app.UIAxes4.Position = [391 4 373 225];
            % Create AngleNumberSpinnerLabel
            app.AngleNumberSpinnerLabel = uilabel(app.UIFigure);
            app.AngleNumberSpinnerLabel.HorizontalAlignment = 'right';
            app.AngleNumberSpinnerLabel.Position = [200 476 82 22];
            app.AngleNumberSpinnerLabel.Text = 'Angle Number';
            % Create AngleNumberSpinner
            app.AngleNumberSpinner = uispinner(app.UIFigure);
            app.AngleNumberSpinner.ValueChangedFcn = createCallbackFcn(app,
@AngleNumberSpinnerValueChanged, true);
            app.AngleNumberSpinner.Position = [297 476 53 22];
        end
    end
    methods (Access = public)
        % Construct app
        function app = ee415_term_project
            % Create and configure components
            createComponents(app)
            % Register the app with App Designer
            registerApp(app, app.UIFigure)
            if nargout == 0
                clear app
            end
        end
        % Code that executes before app deletion
        function delete(app)
            % Delete UIFigure when app is deleted
            delete(app.UIFigure)
        end
    end
end
```