# EE 415 Term Project: Forward and Backward Projections in X-ray Imaging

Tuba Gök
Electrical and Electronics Engineering
Middle East Technical University(METU)
Ankara, Turkey
tuba.gok@metu.edu.tr

***Abstract*— X-ray imaging holds a pivotal role in the history of medical imaging. This paper delves into the theoretical background of Computed Tomography and the MATLAB-based implementation of forward and backward projection algorithms. The forward projection algorithm simulates the X-ray imaging process by computing line integrals of an object's attenuation coefficients to generate projection data essential for image reconstruction. In contrast, the backward projection algorithm reconstructs images from this projection data, employing various filtering techniques, including non-filtered, filtered, and windowed approaches. Numerical evaluations and visual analyses were conducted to assess reconstruction quality, emphasizing the significance of filtering in enhancing image clarity. Additionally, the effects of windowing techniques were systematically examined.**

**Keywords—X-ray imaging, computed tomography, radon transform, forward projection, backward projection, Ram-lak filter, MATLAB**

## I. Introduction

X-ray imaging was discovered in 1895 by Wilhelm Conrad Roentgen, who observed a new type of ray capable of penetrating various materials and revealing internal structures. This discovery revolutionized science and medicine, with early applications in diagnosing injuries and guiding surgical procedures [1]. Medical X-rays, a form of electromagnetic radiation, are utilized to produce images of internal tissues and structures within the body. When X-rays pass through the body and reach an X-ray detector positioned on the opposite side of the patient, they create an image that represents the "shadows" cast by the internal objects [2]. Over time, advancements like high-vacuum tubes and digital imaging led to the development of Computed Tomography (CT) enabling the reconstruction of cross-sectional images through computational algorithms. CT is a technique that combines X-ray projections at multiple angles with computational algorithms to produce cross-sectional images of the body. CT has become an indispensable tool in modern medicine, offering unparalleled insights into diseases and injuries.

This report investigates the implementation of forward and backward projection algorithms, which form the foundation of CT image reconstruction. Forward projection algorithm simulates the X-ray imaging process by computing the attenuation of rays passing through an object, generating the projection data required for reconstruction and a sonogram for the user to interpret. Backward projection reverses this process, reconstructing images from projection data while addressing challenges such as blurring and low-frequency artifacts.

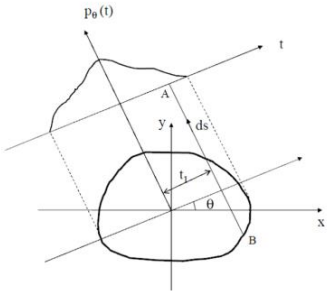


*Figure 1. The geometrical representation of p_θ (t)*

The main objective of this project is to implement these algorithms in MATLAB and evaluate their performance in terms of accuracy, filtering, and windowing techniques. The forward projection algorithm calculates line integrals of an object's attenuation coefficients, producing essential projection data. The backward projection algorithm reconstructs images using unfiltered and filtered techniques, with a focus on the effects of filtering and windowing.

This report is structured as follows: Section II details the mathematical background of forward and backward projections and their implementation in MATLAB. Section III presents the results, including visual analyses and numerical metrics, highlighting the role of filtering and windowing. Section IV discusses the conclusions drawn from the findings and their implications for X-ray imaging. Lastly, a user guide for the GUI, created for the ease of the implementation of both forward and backward projection functions, is added to the Appendix.

## II. Theory and Algorithm

### *A. The Theory of Computed Tomography*

CT is an imaging technique that utilizes X-rays to generate cross-sectional images of the body. It reconstructs these slices by measuring the attenuation coefficients of X-ray beams passing through the object. By leveraging this principle, CT enables the reconstruction of tissue density in a two-dimensional plane perpendicular to the imaging axis [3]. Figure 1 depicts the geometrical principle of obtaining this projection function for the forward problem.

The inverse problem includes determining the reconstructed image using the known projection data acquired in the forward problem. Equations 1-2 give the forward and backward projection line integrals the algorithm implements. In this sense, it can be concluded that the

backward problem is similar to the forward problem algorithm.

$$p_\theta(t) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x,y)\delta(xcos\theta + ysin\theta - t)dxdy \quad (1)$$

$$f_b(x,y) = \int_{0}^{\pi} \int_{-\infty}^{\infty} p_\theta(t)\delta(xcos\theta + ysin\theta - t)dtd\theta \quad (2)$$

This projection data, represented mathematically by a Radon transform in Equation 1, forms the basis for reconstructing the density of tissues in two-dimensional planes perpendicular to the imaging axis. Figure 1 illustrates the geometrical principle of obtaining projection data in the forward problem, where line integrals of the attenuation function $f(x,y)$ are computed along various angles. $p_\theta(t)$ represents the projection data at angle θ, t is the radial distance, and $f(x,y)$is the attenuation coefficient distribution. The inverse problem involves reconstructing the original image, $f_b(x,y)$from the known projection data, $p_\theta(t)$ . This process is represented by the inverse radon transform given in Equation 2.

The inverse problem distributes projection data across the reconstruction grid, summing contributions from all angles to reconstruct the original image theoretically. However, practical implementations face challenges such as blurring due to the dominance of low-frequency components in the projection data, which is addressed in Section II-E. The following subheadings present the function guides for the forward and backward projection algorithms, detailing how the algorithms implement the theoretical principles underlying these problems.

### *B. Forward Projection Function*

The function named *forward_projection* computes the forward projection of an object based on its known attenuation coefficient distribution and simulates the x-ray imaging process. The function takes three input variables: image, number of beams, and step size. While the image is expected to be a string variable defining the name of the image to be loaded in the .mat form, the other two variables are numbers defining the length and the number of the output vector of the function, respectively.

The *forward_projection* plots the sonogram of the image to the user. The *π/step_size* number of output vectors that have length as the *number_of_beams* are created and can serve as the inputs for the back projecting the image. The function can implement parallel ray projection and provide accurate projections for non-complex images.

### *C. The Theoretical Basis of the Forward Projection Algorithm*

The image variable introduced to the function is loaded and converted into matrix form. The rows and columns of the image matrix are obtained and defined as 'M' and 'N' variables respectively in the function, to determine the axes of the coordinate system the image is placed in. The t values of the $p_\theta(t)$, projection function, are linearly spaced between the $[-\frac{M}{\sqrt{2}}, \frac{M}{\sqrt{2}}]$ considering the number of rows and the columns are equal for the sample images. The sample image inputs used to test the algorithm have equal number of rows and columns. Therefore in the remaining part of the paper, M will denote both row and column number, the calculations are also simplified in this manner. However, the algorithms of the forward and backward projection are also capable of handling rectangularly shaped images, i.e. when the M and N variables are not equal to each other. The number of parallel ray beams determines the number of linearly spaced t values. Theta values of the projection function , $\theta$ , are linearly spaced between [0,180-step_size] angles, such that the t-axis scans through the angles not including 180° degrees since it yields the same value as 0°. After defining the specifications, the function must compute line integrals for the radon transform, and the yielding result vectors must be created. However, the size of the result vector changes at each iteration, scanning each parallel ray beam at each projection angle resulting in inefficiency. Since the number and the length of the result vectors can be determined beforehand using step_size and number_of_beams variables, a zeros matrix is created.

As Equation 1 shows, the projection function aimed to be computed calculates the projection of an object onto lines defined by radial positions along the t-axis and angles given by the theta values. It sums the attenuation along each projection line, forming a projection matrix or sinogram, which is essential for reconstructing the object in tomographic imaging. To achieve this, two main loops are defined: one iterating over the theta values (angles) and the other over the t values (radial positions). At each angle, for each beam crossing the image, the intersection points of the beam with the object must be determined. These intersection points are used to compute the Radon transform by summing the attenuation along the beam path.

$$xcos(\theta) + ysin(\theta) = t \quad (3)$$

At each iteration, the intersection points are calculated using Equation 3. It is essential to use the round( ) function of MATLAB, to make sure the linearly spaced x and y points have integer values which is useful when finding the related pixel points in the image placed in the coordinate system. Without use of the round() the related pixels in the image might be determined inaccurately resulting discrepancies in the projection at a known angle. However, not all computed points rely on the object, and the ones relying on the outside must be eliminated since they do not contribute to the attenuation along this ray beam. Since the boundaries of the image are determined using abs(M/2) value, the relevant points can be calculated using this value and sorted to obtain an organized structured manner.

$$distance = \sqrt{((x_2 - x_1)^2 + (y_2 - y_1)^2)} \quad (4)$$

$$mid\ points = \frac{x_2+x_1}{2} + \frac{y_2+y_1}{2} \quad (5)$$

Equations 4 & 5 define the computations between two relevant consecutive intersection points along the projection line which are significant for accurate tomographic reconstruction. The distance defines the length of the line segment passing through the object, which determines the extent of attenuation to be summed along that path. The midpoint, on the other hand, provides a reference point along the projection line, helping to correctly position the resulting data in the sinogram.

After determining the middle points, the exact row and column addresses are found using Equation 6 & 7 to append the related attenuation coefficient to the relevant pixel. MATLAB has discrete grid indices for the image and the related row and column data may not be found integer so appropriate rounding must be applied. For this purpose floor( ) and ceil( ) are used to map the continuous midpoint values to the nearest discrete grid indices. Floor( ) rounds down the Y-coordinate to determine the row, while ceil( ) rounds up the X-coordinate to determine the column. This ensures the projection data is correctly placed on the discrete image grid.

$$row\_data = \frac{M}{2} - floor(midpoints\_y) \quad (6)$$

$$column\_data = \frac{N}{2} - floor(midpoints\_x) \quad (7)$$

The projection function, $p_\theta(t)$,is modeled by calculating the weighted sum of pixel values and corresponding distances, which are then appended to the result vector. This algorithm is applied along each t-axis at a specific theta angle relative to the x-axis. The use of the round() function, and small numerical precision errors due to the nature of MATLAB are the main shortcomings of this algorithm.

### *D. Backward Projection Function*

The function named *backward_projection* computes the backward projection of an object based on its known projection data given, to reconstruct the previously projected image. The function takes three input variables: input data, reconstructed image size, and the specification of filter use. The input data represents the projection data that can either be extracted from the forward projection or can directly be given as a text file. The input data represents the projection data, which can either be obtained from the forward projection process or provided as a .txt or .mat file. If the input data is not in .mat format and is given as a .txt file in a specific structure, the function processes the file to create a projection data matrix, where rows correspond to the number of projections and columns represent the number of samples in each projection. Each projection is mapped to its appropriate position in the matrix.

The second input, the reconstructed image size, defines the dimensions of the reconstructed image since the function does not inherently know the shape or size of the original image. The size must be provided as a 1×2 matrix, e.g., [rows, cols], allowing for the reconstruction of rectangular, square, or other custom image shapes. Finally, the filter specification addresses the inherent blurring of back-projection. Users can choose from three options: 1 for no filtering, 2 for filtering without a window, and 3 for filtering with a window, such as the Hamming window. The *backward_projection* plots the "Reconstructed Image" in the x-y plane.

### *E. The Theoretical Basis of the Backward Projection Algorithm*

The function begins by preparing and assigning necessary variables based on the user's input. The first step involves checking the format of the input_data, which can be provided either as a direct matrix or as a file in .mat or .txt format. If the input is a .mat file, the function loads the projection data directly, assuming it is structured in the file as a matrix where rows correspond to projection angles and columns to projection samples. For .txt files, the function parses the data line by line to construct the projection data matrix, ensuring each projection is accurately assigned to its corresponding row in the matrix. This preprocessing step ensures the function can handle multiple input formats seamlessly.

The function then determines the number of projections,i.e. num_angles, and the number of beams,i.e. num_beams, from the dimensions of the projection data matrix. These values are critical for further computations, as they define the scope of the main back projection algorithm.

Next, the size of the reconstructed image is initialized based on the user-provided reconstructed_image_size parameter. The dimensions of the reconstructed image are set as rows and cols, and these values are used to define a coordinate grid for the reconstruction process. The function calculates the range of t-values, representing the beam positions, and theta-values, corresponding to the projection angles, using the linspace() function. These ranges are essential for defining the geometry of the back projection process. It is significant to realize the rows and the columns do not necessarily need to match since the original image is unknown. Hence, M and N variables are created separately to represent rows and columns.

To facilitate reconstruction, coordinate grids for x and y are also generated, spanning the range from the negative to the positive half of the image dimensions. This coordinate system forms the basis for mapping projection data onto the reconstructed image. Finally, an empty matrix named reconstructed_image is initialized with zeros to store the cumulative back-projected values during the reconstruction process.

Filtering plays a critical role in the back projection process due to the inherent nature of imaging systems. When examining the impulse responses of projection and back projection systems in the frequency domain, the reconstructed image is modulated by a factor of $\frac{1}{\rho}$, where ρ represents the frequency. This modulation introduces an undesirable low-frequency component, resulting in a blurred reconstruction. To correct this, a high-pass filter is applied to the projection data to amplify higher frequencies, ensuring a sharper and more accurate reconstruction.
In the provided function, filtering is implemented based on the user's choice through the third input parameter. The options allow for no filtering (1), filtering without a window (2), and filtering with a window (3).

1. **No Filtering**: When the third input is set to 1, the function skips any modifications to the projection data, resulting in a reconstructed image that retains the blurring caused by the low-frequency components.
2. **Filtering without a Window**: When the third input is set to 2, the function implements a high-pass filter known as the ramp filter or Ram-Lak filter. This

filter is constructed in the frequency domain by creating a linear ramp function using the absolute values of frequencies. Each row of the projection data matrix is then transformed into the frequency domain using the fft() command. The ramp filter is applied multiplicatively to enhance the high-frequency components, and the modified projection data is transformed back to the spatial domain using the ifft().

3. **Filtering with a Window**: When the third input is set to 3, an additional smoothing window, such as the Hamming window, is applied to the ramp filter together forming a bandpass filter. This window reduces spectral leakage by tapering the edges of the filter, providing a trade-off between sharpness and noise suppression [4]. The combined effect of the ramp filter and windowing ensures both high-frequency amplification and smoother transitions in the frequency domain.

This filtering process prepares the projection data for back projection by addressing the low-frequency dominance inherent in the imaging system's frequency response. Figure 2 depicts the typical high-pass filter designed to account for low-frequency dominance.

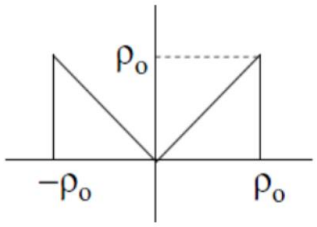


*Figure 2. The typical High Pass filter*

### *F. Defining the Main Loops of the Function*

The main loop of the backward_projection function implements the core mathematical framework of the back-projection process stated in Equation 2. This equation strongly resembles the forward projection algorithm where f(x,y) is integrated instead of the projection data. Therefore, the same main loop in forward projection is used by just changing the last loop where the distances are multiplied by the projection data, instead of the image data as in the case of forward projection.

In the function, the outer loop iterates over all angles, spanning the range from 0 to π. Each angle corresponds to a specific projection of the object, and this loop ensures that every projection contributes to the reconstructed image. For each angle, the inner loop processes all beam positions along the detector. The beam position t determines the locations in the image that are influenced by the corresponding projection value.

Using Equation 3 the function calculates the intersection points of the projection line with the grid of the reconstructed image. After computing these intersection points, the function identifies valid pixel coordinates within the image boundaries. The distances between consecutive intersection points along the projection line are calculated to quantify the contributions of the projection data. Midpoints of these segments are used to pinpoint the specific pixels in the reconstructed image that need updating. The corresponding projection data value, weighted by the calculated distances, is added to the identified pixels. This step simulates the integral in the mathematical equation, summing contributions from all projections to reconstruct the image in the spatial domain.

The main loop of the backward projection function has the same shortcomings as the projection function, since it nearly follows the same logic as explained in the Section II-A. However, in the filtering part of the function, it is important to take the real part of the inverse fourier transform, due to the precision errors of the MATLAB, since in this project the taken projection data is assumed to be real and noise-free. The inverse fft() of MATLAB introduces very small imaginary components which have to be deleted according to our assumptions.

## III. Results & Discussions

### *A. Projection*

Figure 3 is obtained for 101 number of beams with 1 step size for the square.mat sample image when theta equals 0 degree. In this figure, p-values exactly correspond to 11 and beam index corresponds to 28.

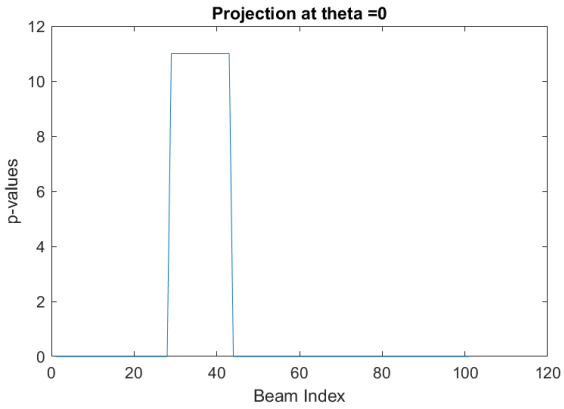


*Figure 3. Projection vs beam plot at 0 degree for simple image*

It is known that the square.mat sample image has 50*50 matrix size with the square image lying in the 10 to 20 for both the row and the column values. The image is placed into an x-y grid meaning that it is placed in between [-25,25] both for x and y axes in the coordinate system. According to our coordinate system, image lies between -15 and -5 points in the x-axis, and between 5 and 15 points in the y-axis. Since our selected projection is in angle 0 using Equation 1, Equation 8 can be found using the new denoted x and y axes along with the theta value, where boundaries of the integrals corresponding to the replacement of the square in the x-y axis.

$$p_0(t) = \int_{5}^{15} \int_{-15}^{-5} 1 * \delta(x - t) dx dy \qquad (8)$$

According to the forward problem's algorithm, summing 1's from 5 to 15 since these numbers are also included gives 11 as expected. It is important that, the algorithm follows

discrete sampling in which it sums over the given range instead of integrating. Also, 101 beams are passed through the image, which results in the 28th beam passing through from the -15 point on the x-axis and the 44th beam passing through the -5 point on the x-axis. Hence, the forward projection results are compatible with the hand-calculated results.

Figure 4 depicts the projection data versus beam number plot taken at a 45-degree angle to repeat the hand calculations and verify the forward algorithm.

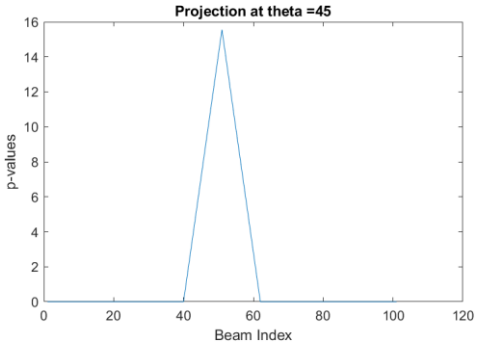


*Figure 4. Projection vs beam plot at 45 degrees for simple image*

In Figure 4, the peak of the p-values approximately corresponds to 15.56 and the projection data is nonzero for beam numbers in between 40 to 62. For this projection Equation 9 is obtained using the boundaries of the square image and the 45 degrees theta value. Since at this theta value the diagonal of the square image is obtained and equals to $11\sqrt{2}$, which is approximately 15.56, the computations align with the hand-calculations.

$$p_{45}(t) = \int_{5}^{15} \int_{-15}^{-5} 1 * \delta\left(\frac{x}{\sqrt{2}} + \frac{y}{\sqrt{2}} - t\right) dxdy \qquad (8)$$

Figures 5 and 6 are the results of the complex image corresponding to *forward_projection('lena.mat',400,1)* command which are the sonogram, and the projection data plotted for $0^{th}$ degree, respectively.

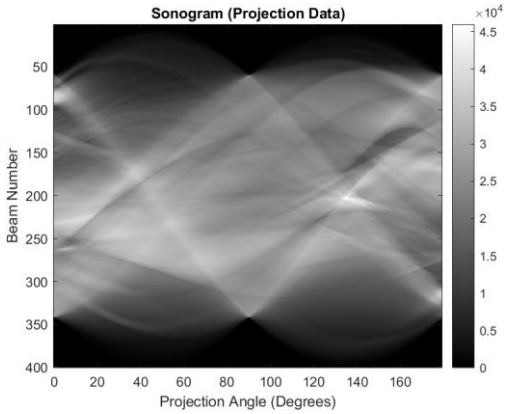


*Figure 5. Sonogram of the complex image*

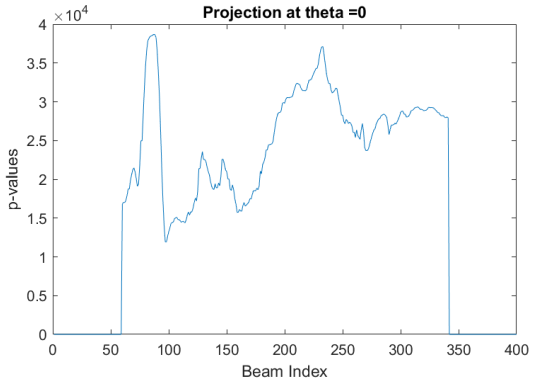


*Figure 6. Projection data plot at 0 degrees for complex image*

This plot aligns with expectations based on the original image of *lena.mat* depicted in Figure 7. The first peak, occurring around the 85th beam, corresponds to the red rectangular region in Figure 7, while the second prominent peak in the projection data roughly aligns with the line passing through the nose tip of the Lena image, represented by the green rectangular area.

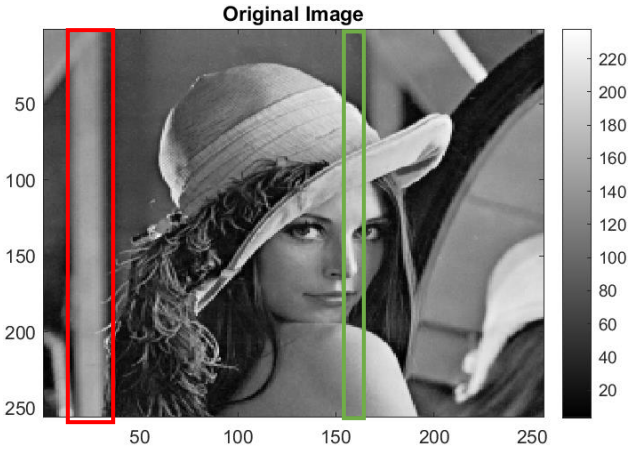


*Figure 7. Original of the complex image*

### B. *Backprojection: Affect of Filtering*

Figures 8 and 9 depict the nonfiltered and the filtered reconstructions of the square sample image.

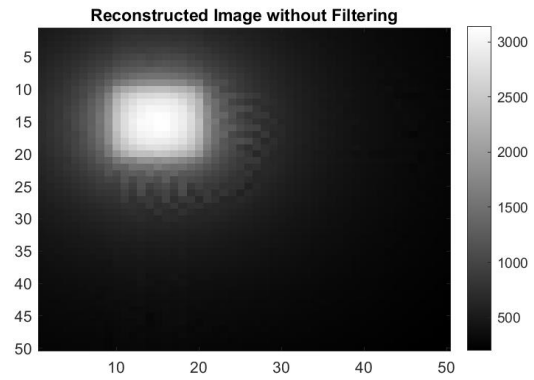


*Figure 8. Reconstructed square image without filtering with [50,50] image size*

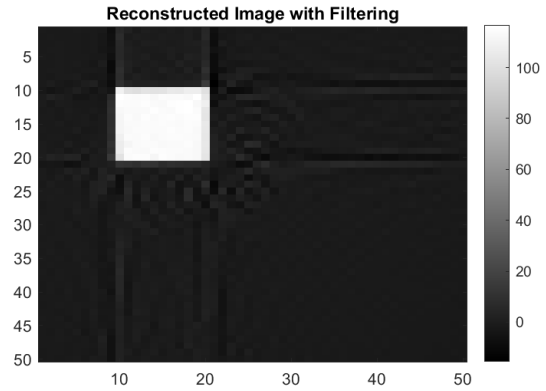


*Figure 9. Reconstructed square image with filtering with [50,50] image size*

As observed from Figure 8, the image lacks sharpness and appears blurred especially in the edges, which aligns with the theoretical expectation of unfiltered back projection. This behavior arises due to the absence of a high-pass filter so that the low-frequency components dominate reconstruction, leading to a loss of detail and clarity. The unfiltered reconstruction directly accumulates the projection data across all angles without compensating for the low-frequency effects. The main loops in the function sum up contributions for each projection angle and beam position, and strongly resemble the forward projection step. However, without filtering, these contributions are not adjusted for the $\frac{1}{\rho}$, modulation, resulting in the observed smoothing and reduced contrast. While the overall structure of the original image is recognizable, fine details and edges are not well-defined, and small sinusoidal distortions are added to the image. The small sinusoidal distortions are more observable when the image size is increased as Figure 10 depicts the unfiltered reconstruction of the projection data of square image with [256,256] image size.

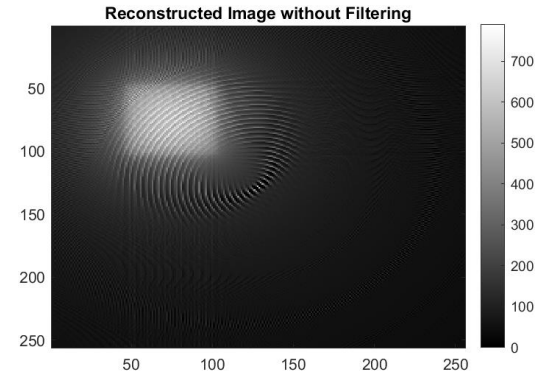


*Figure 10. Reconstructed square image unfiltered with [256,256] image size*

After applying the high-pass Ram-Lak filter, the edges of the square become well-defined as the filter mitigates the dominance of low-frequency components by emphasizing high-frequency details. This enhancement results in a sharper reconstructed image that closely resembles the original. The improvement is further validated quantitatively by comparing the mean squared error (MSE) of the unfiltered and filtered reconstructed images relative to the original. The MSE for the unfiltered image is 0.039, whereas it significantly decreases to $9.17 \times 10^{-4}$ after filtering, demonstrating the expected reduction in error following the application of the filter.

Figures 11 and 12 show the unfiltered and filtered reconstructed images for the complex image *lena.mat*, generated using the projection data obtained through the *forward_projection('lena.mat', 400, 1)* command, as previously described under Section III-A.

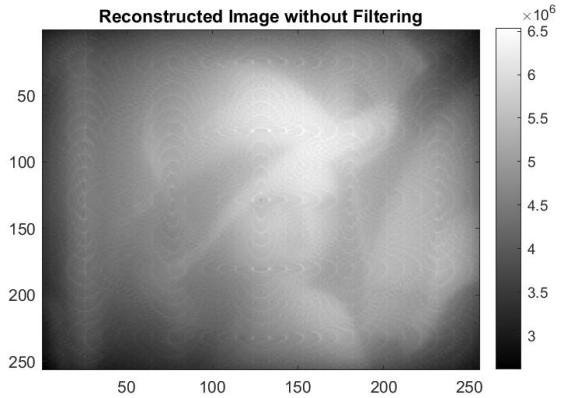


*Figure 11. Reconstructed complex image without filtering in [256,256] image size*

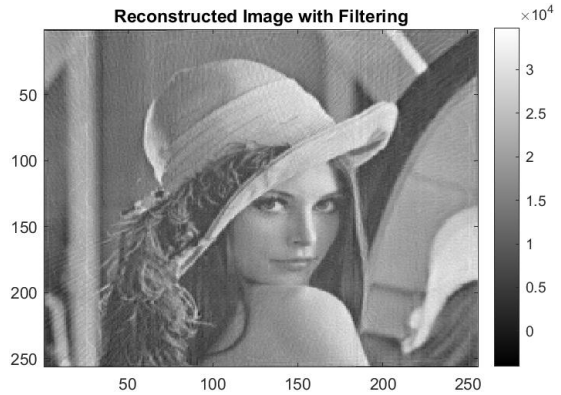


*Figure 12. Reconstructed complex image with filtering in [256,256] image size*

As observed in the case of the simple image, the use of a high-pass filter enhances boundary definition and adds sharpness to the reconstructed image. Figure 11 exhibits blurriness and distortions caused by possible under sampling in the forward projection process, likely due to an insufficient number of beams or an inadequate step size. The blurriness arises from the inherent nature of the Fourier transform in back projection, where the $\frac{1}{\rho}$ component dominates, amplifying low frequencies and causing a loss of detail. To address this, the Fourier transform is multiplied by ρ to compensate for the low-frequency dominance, followed by an inverse transform. This compensation restores detail, sharpens the image, reduces the MSE, and, as shown in Figure 12, results in a reconstructed image that more closely resembles the original.

### *C. Filtered Backprojection: The Effect of Windowing*

It is evident that the edges and fine details of the reconstructed complex image, such as the contours of the

face, hat, and feathers, are better preserved with the use of the Ram-Lak filter compared to the unfiltered reconstruction. However, the absence of windowing introduces some visual artifacts. These artifacts result from the sudden truncation of high frequencies at the boundaries of the projection data, which is inherent to the Ram-Lak filter when used without smoothing. When windows, which act as low-pass filters, are combined with the Ram-Lak filter, the result is a bandpass filter that smooths the reconstructed image while preserving details. This approach improves the overall similarity between the original and reconstructed images. However, the smoothing effect slightly increases the MSE compared to the filtered case alone, though it remains lower than the unfiltered case, with an MSE value of 0.023.

There exist different types of windowing techniques such as Hamming, Hanning, and Blackman windows. These windows, combined with the Ram-Lak filter result in variation of the blurriness. Applying window to the image results in a trade-off between sharpness and the noise reduction, but according to our algorithm our projection data is noise-free. So, there will be differences between the edge sharpness and the overall frequency resolutions. Figures 13 and 14 depict the reconstructed complex image using Hamming and Blackman windows respectively.


*Figure 13. Reconstructed filtered complex image with Hamming window*


*Figure 14. Reconstructed filtered complex image with Blackman window*

For the main code, the Hamming window is chosen as default but can be changed according to the need. The Hamming window is chosen for its ability to suppress spectral leakage effectively while maintaining sufficient frequency resolution. Compared to other windowing methods, such as the Hanning or Blackman windows, the Hamming window provides a better trade-off between noise suppression and edge sharpness [4]. The Hanning window provides similar results but slightly less sharpness, while the Blackman window offers greater artifact suppression at the cost of more blurred details [4]. When Figures 13 & 14 are compared it is seen that Blackman window introduces more blurring, as expected. Apart from the visual facts, this difference can also be proven quantitively, since the MSE calculated for Figure 13 is 0.0139 and for Figure 14 it is 0.0151. Increase in the MSE also shows how hamming window is the optimal windowing technique for this algorithm. As explained for the sample image, other ringing distortions are mainly caused due to under sampling for both filtered and filtered with window reconstructed image. Figures 11-14 corresponds to reconstructed image for 400 number of beams with 1 step size. When the number of beams is increased, these distortions are also eliminated as Figure 15 shows filtered reconstructed image with hamming window for 700 number of beams and 1 step size projection data.


*Figure 15. Reconstructed image for forward_projection('lena.mat', 700, 1) forward projection result*

## IV. Conclusion

This study implemented and evaluated forward and backward projection algorithms as essential components of CT imaging, emphasizing their theoretical foundations and algorithmic implementation. The forward projection function simulated the X-ray imaging process by calculating line integrals of attenuation coefficients, generating projection data essential for image reconstruction. This process was validated through theoretical calculations, as well as visual and numerical analyses using MATLAB, which demonstrated the algorithm's accuracy in producing projection data, including sonograms.

The backward projection algorithm reconstructed images from projection data, addressing inherent challenges such as blurring caused by low-frequency dominance. The application of the high-pass Ram-Lak filter significantly

enhanced the sharpness and clarity of reconstructed images by compensating for low-frequency artifacts. Moreover, windowing techniques, including Hamming and Blackman windows, demonstrated the effect of windows on to the sharpness.

Further analysis explored the impact of sampling parameters, including the number of beams and step size, on reconstruction quality. Results indicated that insufficient sampling introduced artifacts, and increasing the number of beams and reducing step size were shown to decrease these effects, leading to more accurate and visually faithful reconstructions.

Overall, this project highlights the critical role of filtering in enhancing the quality of the reconstructed image by emphasizing high-frequency components, while the application of a window creates a smoother image, though with a slight reduction in sharpness.

## REFERENCES


[1] NDE Education Resource Center, "Interaction of Radiation with Matter," [Online]. Available: https://www.nde-ed.org/NDETechniques/Radiography/interactionradmat.xhtml.

[2] National Institute of Biomedical Imaging and Bioengineering, "X-rays," [Online]. Available: https://www.nibib.nih.gov/science-education/science-topics/x-rays.

[3] Radiopaedia, "Computed Tomography," [Online]. Available: https://radiopaedia.org/articles/computed-tomography.

[4] A. V. Oppenheim and R. W. Schafer, *Discrete-Time Signal Processing*, 3rd ed. Upper Saddle River, NJ, USA: Pearson, 2010, ch. 7, pp. 562–567.

## APPENDIX

### USER'S GUIDE FOR FORWARD & BACKPROJECTION GUI

The Forward & Backprojection GUI is designed to provide a simple interface for simulating X-ray imaging and reconstructing images using forward and backward projection functions detailly explained under Section II. The *Forward Projection* section of GUI allows users to compute projection data for a given image. To use this feature, users need to input the name of the image file in .mat format, specify the number of beams, define the step size for angular increments, and optionally provide a specific angle for detailed analysis. Upon clicking the *Run Forward* button, the projection data for the specified angle is plotted, displaying a graph of p-values against the beam index for visual analysis.

The *Backprojection* section facilitates image reconstruction using either the projection data generated from the forward projection or an external .txt file. Users must input the desired reconstructed image size and select a filter option. Filtering options include no filtering, Ram-Lak filtering, and Ram-Lak filtering combined with a Hamming window to enhance reconstruction quality and corresponding to inputs 1,2 and 3 respectively. The reconstructed image is displayed alongside the original image for comparison, and the applied filter type is indicated in the title of the reconstructed image. Figure 16 depicts an example use of the Forward & Backprojection GUI with appropriate inputs.

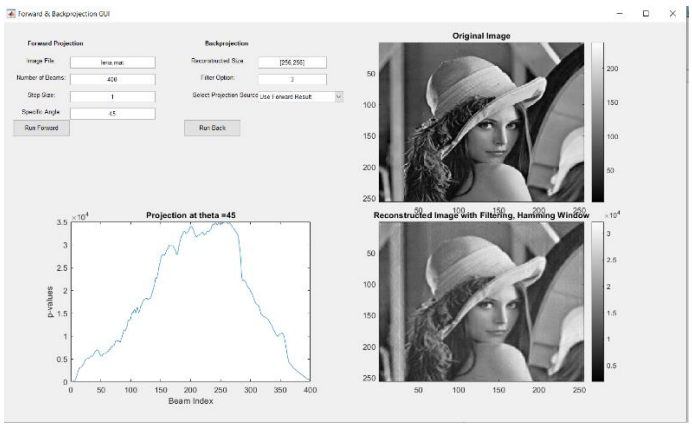


*Figure 16. Example use of GUI*