Chapter 1: Introduction

1. Tomography: -

Tomography is an imaging technique that generates cross-sectional views of an object and with the help of that cross-sectional view, we try to visualize the internal structures. The term "Tomography" comes from two Greek words 'tomos' which means 'slice or cut' and "graphia" which means 'describing'.

- Different examples of tomographic problems:

  - ♦ Reconstructing a map using two pictures from a different view

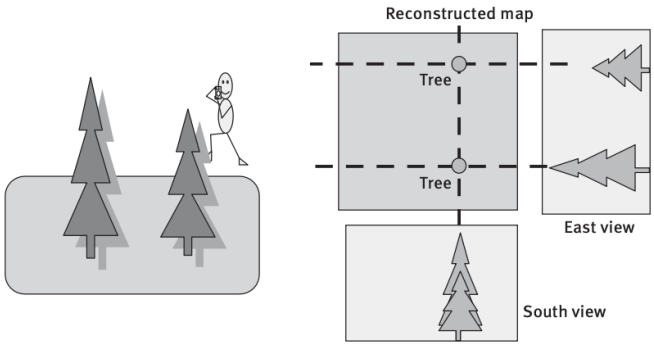
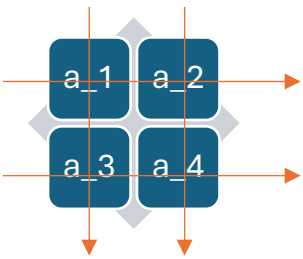


Fig1. Reconstruct an image using data from two separate photographs.

  - ♦ Mathematical Tomography Challenge

    Let's say a 2 by 2 matrix:



$$a_1 + a_2 = 9,$$
$$a_3 + a_4 = 13,$$
$$a_1 + a_3 = 10,$$
$$\text{and } a_2 + a_4 = 12.$$

We get, $a_1=4$, $a_2=5$, $a_3=6$, $a_4=7$.

In the above mathematical tomography challenge, the sum across rows and columns can also be interpreted as the aggregation of values along rays, line integrals, or projections.

2. Projection:

To grasp the idea of Projection (ray sum, line integral, radon transform). Here are some examples:

Example 1: The object is a uniform disk at the center (with linear density x). The Projection (line integral) is represented as:

$$p(s)=x\ t=2x\sqrt{r^2-s^2} \quad \text{if r} > |\text{s}|.$$
$$p(s)=0 \text{ otherwise}$$

where "r" is the disk's radius.

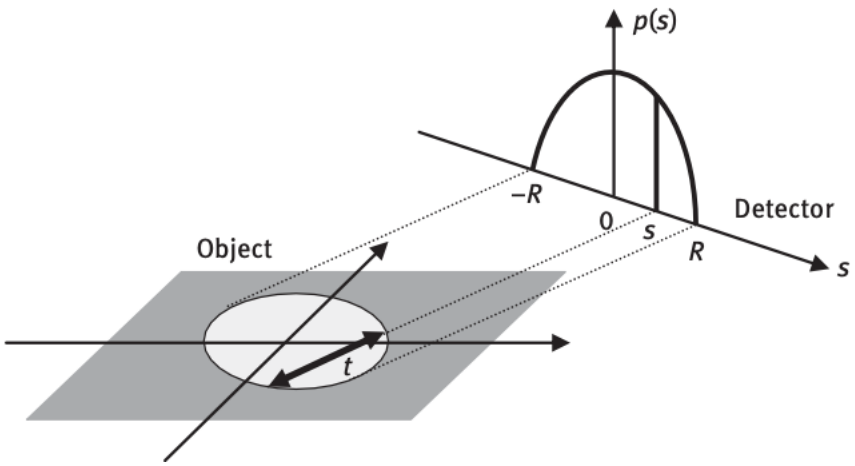


Fig 2. Line integral across the uniform disk

Example 2: When an object is more complicated the projection $p(s,\theta)$ different for different view angles θ.

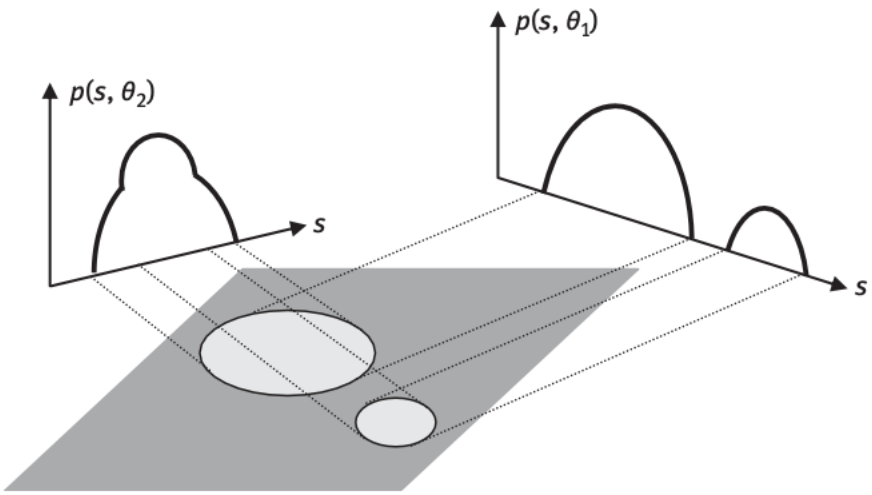


Fig 3. Projections different for different view angle

3. Image Reconstruction:

Image reconstruction is a critical process in medical imaging, allowing for the creation of visual representations of internal structures from projection data. One of the common image reconstruction strategies involves using backprojection and filtering, illustrated here with a simple point source example.

Example: A point source on a 2-D Plane

Firstly, we consider a dot with a value of 1 on an empty 2-D plane with an X-Y coordinate and a rotating detector around the origin which captures projection at different angles θ.

Projection Formation:

- Thus, for a given angle θ, the projection value $p(s,\theta)$ is established.
- s is a value in direction, which is orthogonal to the detector, and represents the coordinate of the point where the line crosses the detector.
- If the line intersects the point source, $p(s,\theta)$ is 1; if the line does not intersect the point source, $p(s,\theta)$ is 0.

Backprojection:

- Projections are taken at different angles, resulting in spikes where the point source is located.
- To reconstruct the image, evenly distribute all activity of each spike along its trajectory.
- Overlay these evenly distributed activities from various angles to pinpoint the location of the point source in the x-y plane. This process creates an initial blurred image.

Filtering:

- To minimize the overall blurring effect, add negative "wings" around each spike in the projections prior to backprojection.
- This filtering step will refine the image and provide a clearer final image.

FBP (Filtered Back Projection) Algorithm:

- The process of combining filtering and backprojection is referred to as the FBP algorithm.
- It is commonly used in medical imaging to reconstruct accurate and clear images from projection data.

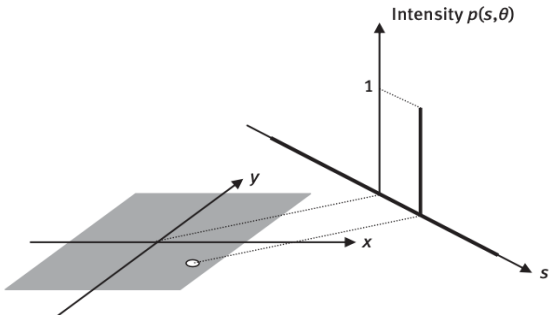


Fig 4: Projection of a single point source object

4. Backprojection:

Backprojection is not the inverse of projection. Backprojection alone is not sufficient to reconstruct an image. One will not get the original image after backprojection. But backprojection is an important step in image reconstruction. Translating the raw projection data into a visual format, this step forms a basic step for many reconstruction algorithms such as FBP.

5. Fourier Transform:

The idea behind the Fourier transform is that the function p(s) can be represented as a weighted sum of sine and cosine terms with various frequencies ω, using the weighting function P(ω). This weighting function P(ω) is the Fourier transform of p(s).

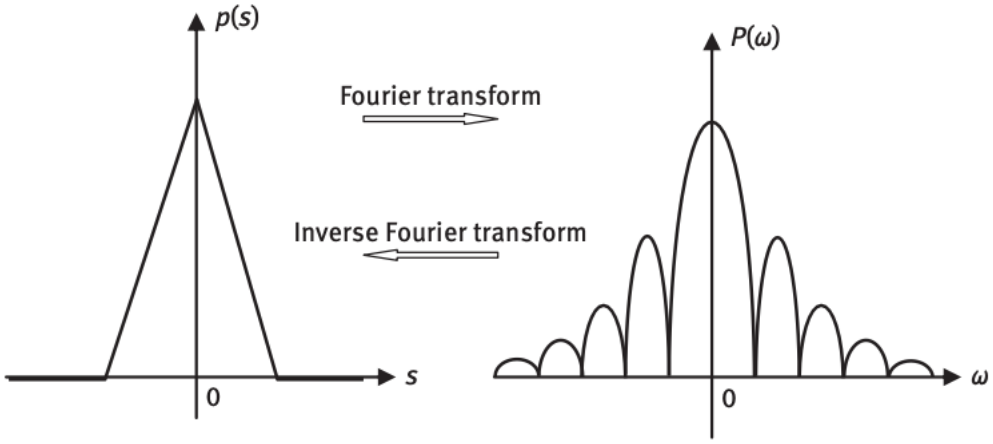


Fig 5: A Fourier transform pair

6. Central Slice Theorem:

Introduction: Fourier Slice Theorem is also referred to as the Central Slice Theorem and is one of the key principles of CT. It defines a crucial connection between the projections of an object and its 2D Fourier transform, easing the reconstruction from projection data.

Fourier Slice Theorem: This theorem states that the one-dimensional Fourier transform of an object's projection at a specific angle corresponds to a slice of the two-dimensional Fourier transform of the object at the same angle.

Fourier Transform of the Projection: Computing the 1D Fourier transform of the projection $p(s, \theta)$ with respect to s results in:

$$\mathcal{F}\big(p(s,\theta)\big) = P(\omega, \theta)$$

Where ω is varying frequency

Chapter 2: Image Reconstruction

1. Projection (Radon Transform): -

Consider a density function $f(x, y)$ across the x-y plane. The projection $p(s, \theta)$ can then be represented as:

$$p(s,\theta) = \iint_{-\infty}^{\infty} f(x,y)\delta(\mathrm{x}\ \cos\theta + \mathrm{y}\ \sin\theta - \mathrm{s})dx\ dy$$

The normal form of a line: $\mathrm{x}\ \cos\theta + \mathrm{y}\ \sin\theta = s$

Assume $\vec{x} = \mathrm{x}\,\hat{\imath} + \mathrm{y}\,\hat{\jmath}$, $\vec{\theta} = \cos\theta\,\hat{\imath} + \sin\theta\ \hat{\jmath}$

Then,

$$p(s,\theta) = \iint_{-\infty}^{\infty} f(x,y)\delta\big(\vec{x}\ \vec{\theta} - \mathrm{s}\big)dx\ dy$$

Now by changing the coordinate system, we get

$$s = x\cos\theta + y\sin\theta$$
$$t = y\cos\theta - x\sin\theta$$

$$ds\ dt = J\ dx\ dy$$

$$J = \begin{bmatrix} \frac{\partial s}{\partial x} & \frac{\partial s}{\partial y} \\ \frac{\partial t}{\partial x} & \frac{\partial t}{\partial y} \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix} = 1$$

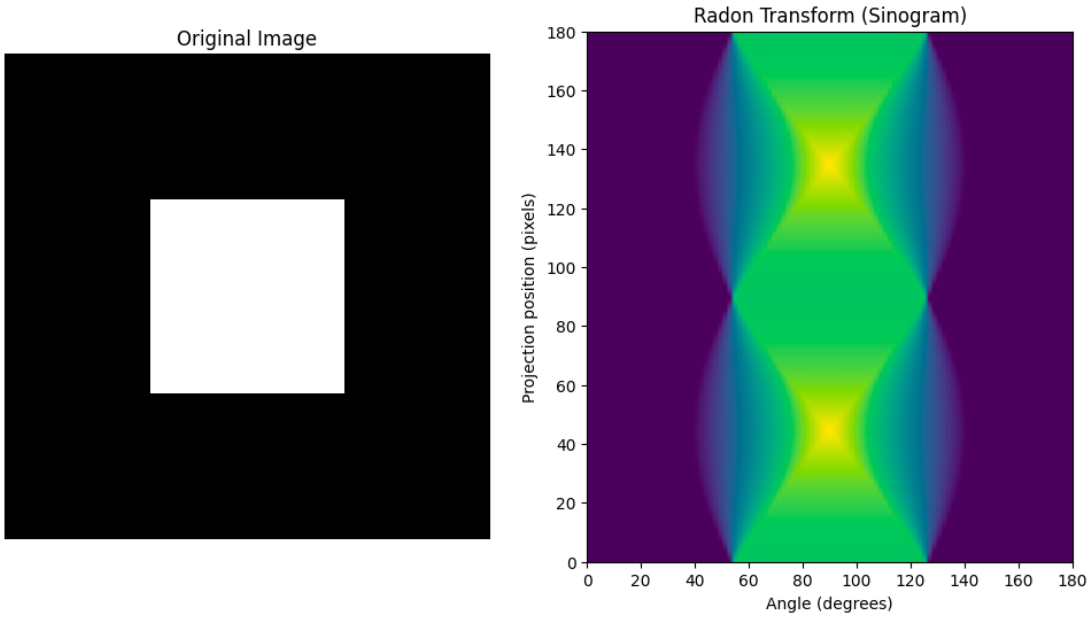
$$p(s,\theta) = \int_{-\infty}^{\infty} f(s\cos\theta - t\sin\theta\,, s\sin\theta + t\cos\theta)dt$$

$$p(s,\theta) = \int_{-\infty}^{\infty} f(s\ \vec{\theta} + t\ \vec{\theta}\ \perp)\ dt$$

CODE FOR RADON TRANSFORM OF ROTATING OBJECT

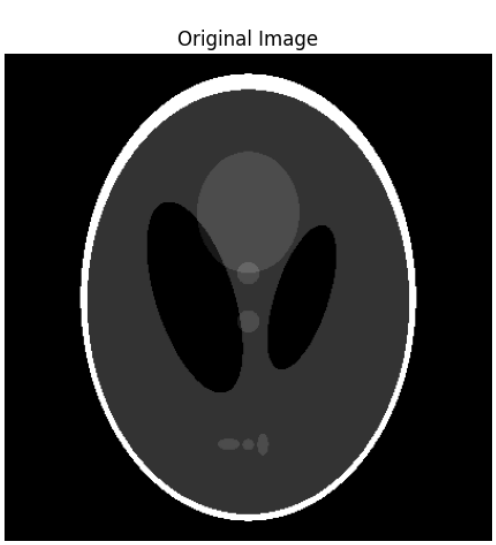
```
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import rotate
def radon(image,theta=None):
    if theta is None:
        theta=np.linspace(0,180,max(image.shape))
        #Initializes a 2D numpy array called sinogram with zeros.
        #The array has dimensions: len(theta): Number of rows equal to the number of
angles.
        #image.shape[0]: Number of columns equal to the height of the input image.
        #dtype=np.float64: Specifies the data type as 64-bit floating point.
    sinogram=np.zeros((len(theta),image.shape[0]),dtype=np.float64)
    for i,angle in enumerate(theta):
        rotated_image=rotate(image,angle,reshape=False)
        sinogram[i]=np.sum(rotated_image,axis=0)

    return sinogram

#Example:
image=np.zeros((100,100))
image[30:70,30:70]=1
theta=np.linspace(0,180,450)
sinogram=radon(image,theta)
# Plot the original image and its Radon transform (sinogram)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

ax1.set_title("Original Image")
ax1.imshow(image, cmap='gray')
ax1.axis('off')

ax2.set_title("Radon Transform (Sinogram)")
ax2.imshow(sinogram, extent=(0, 180, 0, sinogram.shape[0]), aspect='auto')
ax2.set_xlabel("Angle (degrees)")
ax2.set_ylabel("Projection position (pixels)")

plt.show()
```

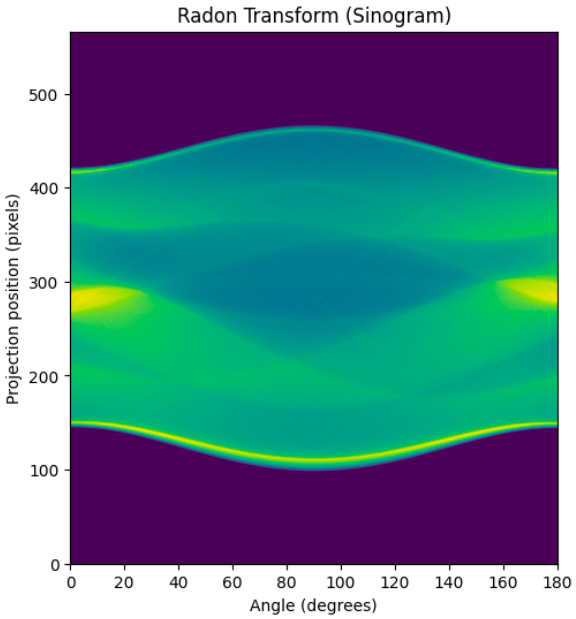
OUTPUT


Original Image
Radon Transform (Sinogram)
Projection position (pixels)
Angle (degrees)
0
20
40
60
80
100
120
140
160
180

```
import numpy as np
import matplotlib.pyplot as plt
from scipy.ndimage import rotate
from skimage.data import shepp_logan_phantom
def radon_transform(image,theta=None):
    if theta is None:
        theta=np.linspace(0,180,max(image.shape))
        #Initializes a 2D numpy array called sinogram with zeros.
        #The array has dimensions: len(theta): Number of rows equal to the number of
angles.
        #image.shape[0]: Number of columns equal to the height of the input image.
        #dtype=np.float64: Specifies the data type as 64-bit floating point.
    sinogram=np.zeros((len(theta),image.shape[0]),dtype=np.float64)
    for i,angle in enumerate(theta):
        rotated_image=rotate(image,angle,reshape=False)
        sinogram[i]=np.sum(rotated_image,axis=0)

    return sinogram

#Example:
image=shepp_logan_phantom()
theta=np.linspace(0,180,450)
sinogram=radon_transform(image,theta)
# Plot the original image and its Radon transform (sinogram)
fig, (ax1, ax2) = plt.subplots(1, 2, figsize=(12, 6))

ax1.set_title("Original Image")
ax1.imshow(image, cmap='gray')
ax1.axis('off')

ax2.set_title("Radon Transform (Sinogram)")
ax2.imshow(sinogram.T, extent=(0, 180, 0, sinogram.shape[0]), aspect='auto')
ax2.set_xlabel("Angle (degrees)")
ax2.set_ylabel("Projection position (pixels)")

plt.show()
```

OUTPUT





2. FBP (Filtered Back Projection):

Introduction: Filtered back projection (FBP) is one of the most common algorithms to reconstruct images from projection data in the CT. It integrates the principle of the Fourier Slice Theorem with the filtering technique to provide high-resolution and accurate images. The FBP method comprises two primary steps: initially filtering the projection data, followed by back projecting the filtered data to generate the image.

Mathematical steps of FBP:

- Collect projections $p(s,\theta)$ at various angles θ, where s represents the position along the detector.
- Filtering (Convolution in frequency domain):
  - ❖ The projection is first filtered to enhance the high-frequency components.
  - ❖ Filtering is done by applying a filter $H(f)$ in the frequency domain. Some examples of filters as Ramp Filter, Shepp-Logan filter etc.
  - ❖ We compute 1D Fourier transform of the projection $p(s,\theta)$:

$$P(f,\theta) = \mathcal{F}\big(p(s,\theta)\big) = \int_{-\infty}^{\infty} p(s,\theta)e^{-i2\pi sf}\,ds$$

  - ❖ Apply the filter $H(f)$ in the frequency domain:

$$\tilde{P}(f,\theta) = P(f,\theta).H(f)$$

  Common filter $H(f)$ is Ramp Filter:

$$H(f) = |f|$$

  - ❖ Now we apply inverse Fourier Transform to get filtered projection $\tilde{p}(s,\theta)$:

$$\tilde{p}(s,\theta) = \mathcal{F}^{-1}\left(\tilde{P}(f,\theta)\right) = \int_{-\infty}^{\infty} \tilde{P}(f,\theta)\, e^{i2\pi sf}\,df$$

- Back Projection:
  - ❖ The filtered projection $\tilde{p}(s,\theta)$ are then backprojected to get a better reconstructed image.
  - ❖ Mathematically, backprojection steps are:

$$f(x,y) = \int_{0}^{\pi} \tilde{p}(x\cos\theta + y\sin\theta\,,\theta)d\theta$$

Conclusion: The FBP algorithm is an effective method for image reconstruction from CT projection data.

PYTHON CODE FOR FBP

```
# Original function from scikit-image project
# Source: https://github.com/scikit-image/scikit-
image/blob/v0.23.2/skimage/transform/radon_transform.py
# License: BSD-3-Clause


import numpy as np
from scipy.fftpack import fft, ifft
from skimage._shared.utils import convert_to_float
from skimage.transform import radon

def inverse_radon(
    sinogram,
    x,
    size_of_output=None,
    name_of_filter="ramp",
    interpolation="linear",
    circle=True,
):
    #Find the dimension of projection image
    if sinogram.ndim != 2:
        raise ValueError('the image should in 2 dimensional')

    # different set of projections
    if x is None:
        x = np.linspace(0, 180, sinogram.shape[1], endpoint=False)

    no_of_angles = len(x)

    #checking whether no of angle is equal to column of sinogram array
    if no_of_angles != sinogram.shape[1]:
        raise ValueError("error")

    type_of_interpolation = ('linear','nearest','cubic')

    if interpolation not in type_of_interpolation:
        raise ValueError("Unknown interpolation")

    filter = ('ramp', 'cosine', 'hamming','shepp-logan', 'hann', None)
```

```
#checking type of filter

if name_of_filter not in filter:
    raise ValueError("Unknown filter")

sinogram = convert_to_float(sinogram)
data_type = sinogram.dtype

# storing the row of sinogram array
shape_of_image = sinogram.shape[0]

# estimating output size from sinogram if output size is not specified
if size_of_output is None:
    if circle:
        size_of_output = shape_of_image
    else:
        size_of_output = int(np.floor(np.sqrt((shape_of_image) ** 2 / 2.0)))

if circle:
    sinogram = _sinogram_circle_to_square(sinogram)
    shape_of_image = sinogram.shape[0]

# Resizing image
# Fourier analysis for less artifacts
padded_proj_size = max(64, int(2 ** np.ceil(np.log2(2 * shape_of_image))))
padded_width = ((0, padded_proj_size - shape_of_image), (0, 0))
image = np.pad(sinogram, padded_width, mode='constant', constant_values=0)

#filter in Fourier domain
fourier_filt = _get_fourier_filter(padded_proj_size, name_of_filter)
proj = fft(image, axis=0) * fourier_filt
radon_filt = np.real(ifft(proj, axis=0)[:shape_of_image, :])

# image reconstruction by interpolation
reconstructed = np.zeros((size_of_output, size_of_output), dtype=data_type)
radius = size_of_output // 2
x_p_r, y_p_r = np.mgrid[:size_of_output, :size_of_output] - radius
x = np.arange(shape_of_image) - shape_of_image // 2
```

```
for col, angle in zip(radon_filt.T, np.deg2rad(x)):
    t = y_p_r * np.cos(angle) - x_p_r * np.sin(angle)
    if interpolation == 'linear':
        interpolant = partial(np.interp, xp=x, fp=col, left=0, right=0)
    else:
        interpolant = interp1d(
            x, col, kind=interpolation, bounds_error=False, fill_value=0
        )
    reconstructed += interpolant(t)

if circle:
    out_reconstruction_circle = (x_p_r**2 + y_p_r**2) > radius**2
    reconstructed[out_reconstruction_circle] = 0.0

return reconstructed * np.pi / (2 * no_of_angles)
```

3. ART (Algebraic Reconstruction Technique):

Solving a system of linear equations: Instead of utilizing an analytical algorithm for image reconstruction, a system of linear equations can be solved to achieve the reconstructed image. This process involves discretizing the image into pixels or voxels.

Example: Here, the image pixels x_j (where $j=1,2,....$) are sequentially labelled in a one-dimensional fashion, similar to all projections p_i (where, $i=1,2,..$) relationship between image pixels and projections can be established through a system of linear equations outlined as follows:

$$
\begin{gathered}
x_1 + x_2 + x_3 = p_1 \\
x_4 + x_5 + x_6 = p_2 \\
x_7 + x_8 + x_9 = p_3 \\
x_3 + x_6 + x_9 = p_4 \\
x_2 + x_5 + x_8 = p_5 \\
x_1 + x_4 + x_7 = p_6 \\
2(\sqrt{2}-1)x_4 + (\sqrt{2}-1)x_7 + 2(\sqrt{2}-1)x_8 = p_7 \\
\sqrt{2}(x_1 + x_5 + x_9) = p_8 \\
2(\sqrt{2}-1)x_2 + (\sqrt{2}-1)x_3 + 2(\sqrt{2}-1)x_6 = p_9
\end{gathered}
$$

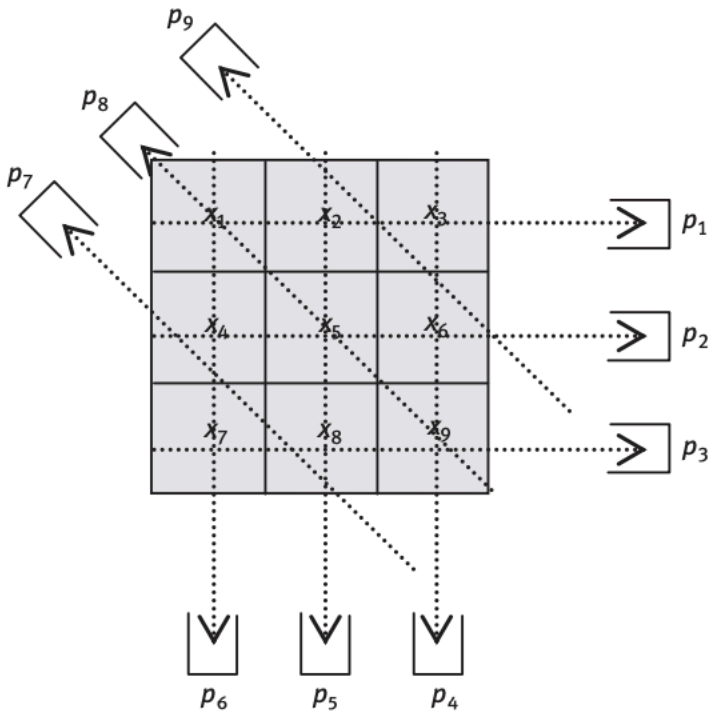


Fig 6: Example of nine unknown and nine measurements

Expressing this relationship in matrix form, we have $AX=P$ where A represents the coefficient matrix of the system, $X=[x_1,x_2,\ldots,x_9]^T$ and $P=[p_1,p_2,\ldots,p_9]^T$. If the inverse of matrix A exists, the reconstructed image is given by $X=A^{-1}P$. However, in practical imaging scenarios, A may not always be a square matrix. In such cases, a generalized inverse of the matrix is utilized. For an overdetermined system, the least squares solution is $X=(A^TA)^{-1}A^TP$ while for an under-determined system, it is $X=A^T(AA^T)^{-1}P$.

ART:

The ART algorithm is an iterative approach referred to as the Kaczmarz method that is applied to image reconstruction. The basic concept of the ART algorithm is to iteratively move the estimated image in a way that satisfies one equation at a time. This concept is depicted in figure 7 as shown below, where three lines L1, L2, and L3 correspond to three different equations. The solution to the system of equations is the point of intersection of these lines. In this case, the image in question contains only two pixels.

Process of ART:

- Initial Guess: First, choose an approximate value for the solution which we will denote as $\overrightarrow{x_0}$.
- Iterative Projection:
  - Step 1: Project $\overrightarrow{x_0}$ perpendicularly to the first line L1 and obtain a new point $\overrightarrow{x_1}$.
  - Step 2: Project the point $\overrightarrow{x_1}$ onto the second line L2 by drawing a line perpendicular to the second line L2, and label the intersection point as $\overrightarrow{x_2}$.

- Step 3: This process can be continued by mapping each of the next points onto the next line (equation) in succession.
- Iteration: Each equation is processed in a loop in one iteration, and the estimate is updated at every step.

❖ The algorithm then moves the estimate in the direction of each line (equation) iteratively. In an ideal world, all the lines will overlap at a certain point, which is the solution to the system of equations.
❖ If the equations exhibit consistency, the algorithm will converge to the solution of the system of equations.
❖ If the equations lack consistency, the algorithm will not converge to a solution for the system of equations and will continually fluctuate without reaching a stable outcome.
❖ Mathematically ART algorithm can be written as:

$$\overrightarrow{x_{next}} = \overrightarrow{x_{current}} - Backproject_{ray}\left(\frac{Project_{ray}(\overrightarrow{x_{current}}) - Measurement_{ray}}{Normalization\ factor}\right)$$

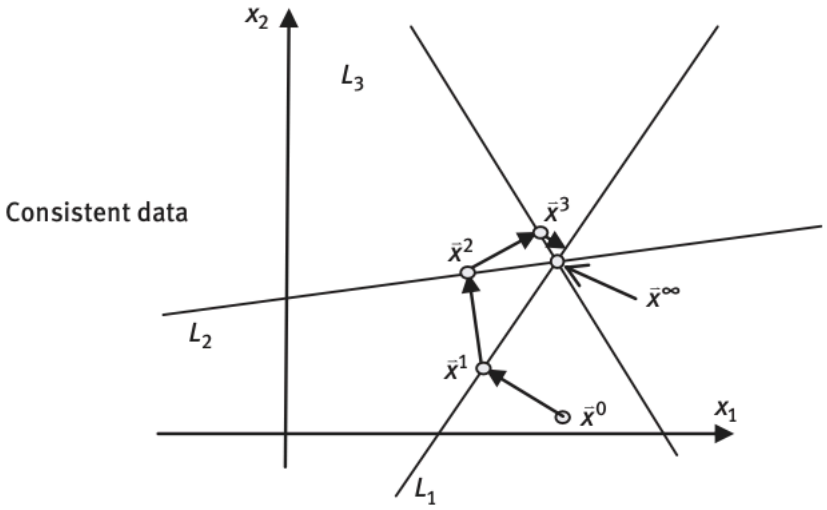


Fig 7: The ART algorithm aims to meet each equation after each update, ensuring consistency in the data.

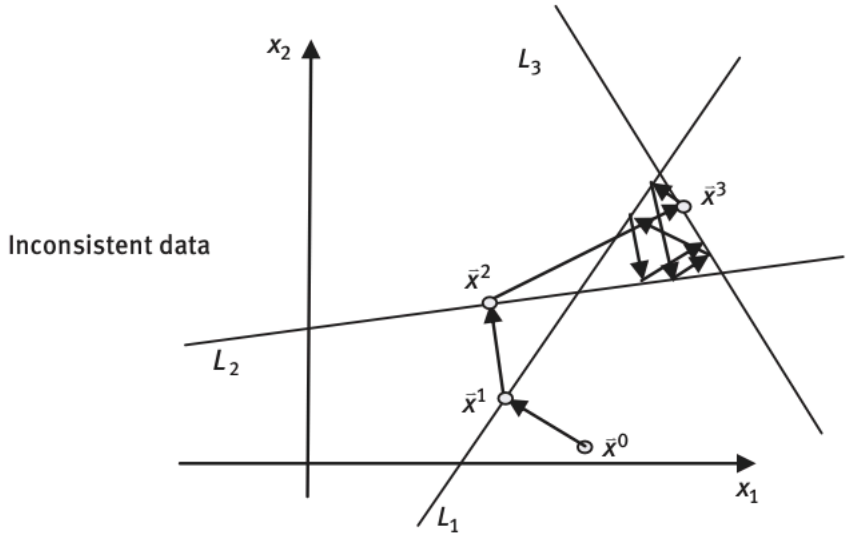


Fig 8: The ART algorithm endeavours to fulfil each equation following every update, even in cases of inconsistent data.

Chapter 3: Analyzing Backprojection of DICOM Image

1. Introduction to DICOM Imaging and Quality Assessment:

- DICOM stands for Digital Imaging and Communications in Medicine, serving as a globally recognized standard ensuring uniform communication among medical imaging devices and systems.

- DICOM refers not only to the file format of medical images but also to the data that describes the images and associated patient data, acquisition parameters, etc. This guarantees that images will always be stored with all the relevant information that would make them valuable in clinical practice.

- PSNR: PSNR, or Peak Signal-to-Noise Ratio, serves as a mathematical metric for evaluating the fidelity of reconstruction in lossy compression codecs, such as image compression. It quantifies the discrepancy between the original and reconstructed data through comparative analysis. Mathematically, PSNR is expressed as:

$$PSNR = 10\,log_{10}(\frac{(MAX)^2}{MSE})$$

Where MAX denotes the highest achievable pixel value within the image, and MSE represents the Mean Squared Error.

- SSIM: SSIM, or Structural Similarity Index, is a technique utilized to gauge the resemblance between two images. It relies on pixel-wise distinctions to quantify the similarity.

2. Implementation:

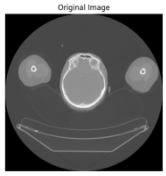
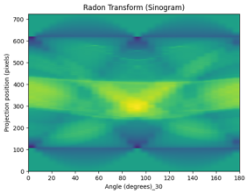
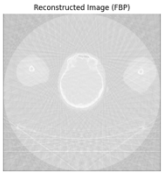
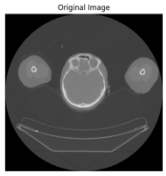
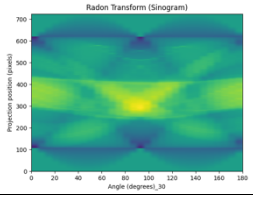
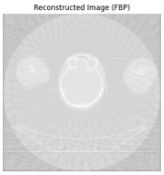
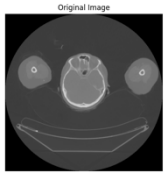
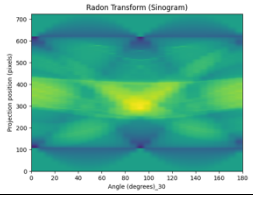
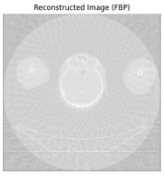
PYTHON CODE FOR ABOVE

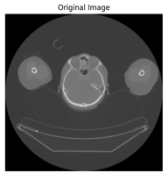
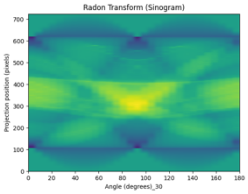
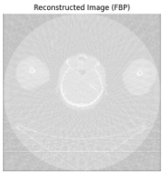
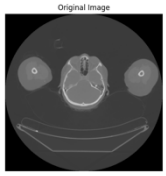
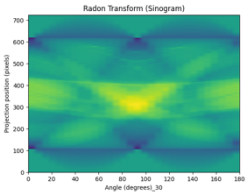
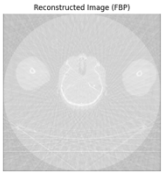
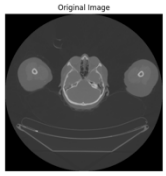
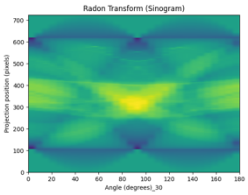
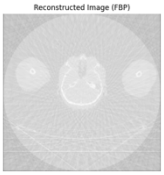
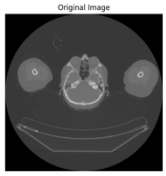
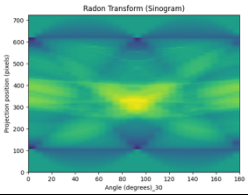
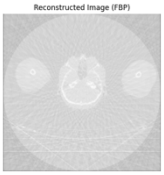
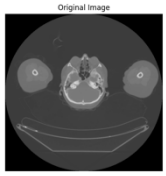
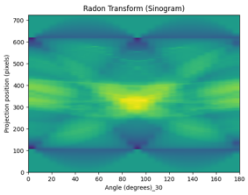
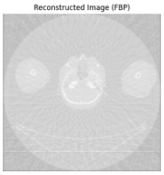
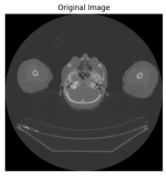
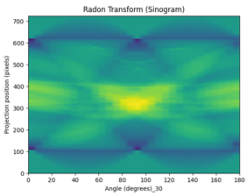
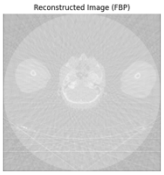
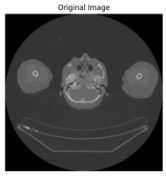
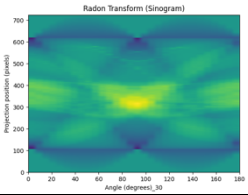
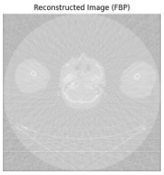
```
import pydicom
import numpy as np
import matplotlib.pyplot as plt
from skimage.transform import radon,iradon
import os
import glob
from skimage.metrics import peak_signal_noise_ratio,
structural_similarity


def get_dicom_files(directory):
    # Use glob to find all .dcm files in the specified
directory
    dicom_files = glob.glob(os.path.join(directory,
"*.dcm"))
    return dicom_files

# Specify the directory containing DICOM files
directory = '/Users/amanraj/Desktop/object'  # Replace with
the path to your folder

# Get the list of all .dcm files in the directory
dicom_files = get_dicom_files(directory)

# Define the different projection angles to test
projections = [30,60,90,120,150,180]

# Initialize matrices to store PSNR and SSIM values
psnr_matrix = np.zeros((len(projections),
len(dicom_files)))
ssim_matrix = np.zeros((len(projections),
len(dicom_files)))

for j, num_projections in enumerate(projections):
```

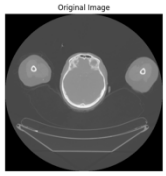
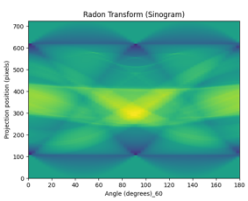
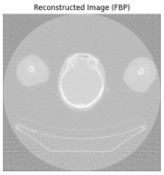
```
    theta = np.linspace(0, 180, num_projections,
endpoint=False)
    for i, dicom_path in enumerate(dicom_files):
        # Step 1: Load the DICOM image
        dicom_image = pydicom.dcmread(dicom_path)

        # Extract image data as a NumPy array
        image = dicom_image.pixel_array

        # Compute the Radon transform
        radon_image = radon(image,theta, circle=False)

        # Reconstruct the image using FBP
        reconstruction_fbp = iradon(radon_image,
theta=theta, circle= False)

         # Calculate PSNR
        psnr_value = peak_signal_noise_ratio(image,
reconstruction_fbp)
        psnr_matrix[j, i] = psnr_value

        # Calculate SSIM
        ssim_value, _ = structural_similarity(image,
reconstruction_fbp,  data_range=image.max() - image.min(),
full=True)
        ssim_matrix[j, i] = ssim_value

        # Plot the original image, its Radon transform
(sinogram), and the reconstructed image
        fig, (ax1, ax2, ax3) = plt.subplots(1, 3,
figsize=(18, 6))

        ax1.set_title("Original Image")
        ax1.imshow(image, cmap='gray')
        ax1.axis('off')

        ax2.set_title("Radon Transform (Sinogram)")
        ax2.imshow(radon_image, extent=(0, 180, 0,
radon_image.shape[0]), aspect='auto')
        ax2.set_xlabel(f"Angle(degree)_{num_projections}")
```

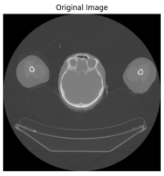
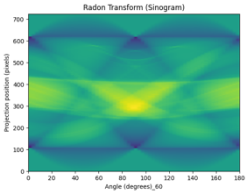
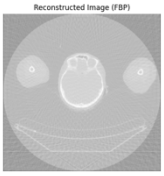
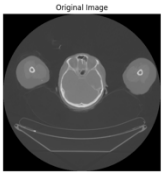
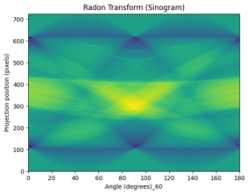
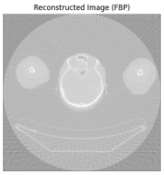
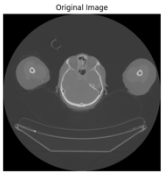
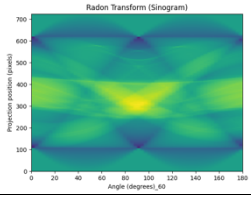
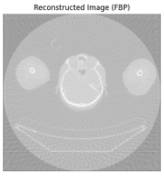
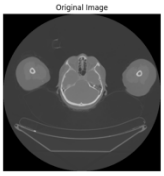
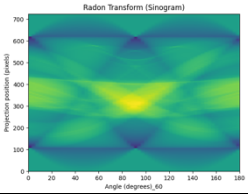
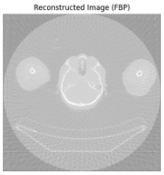
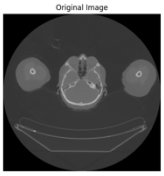
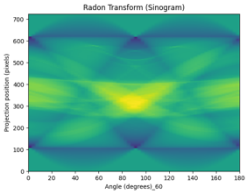
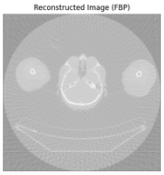
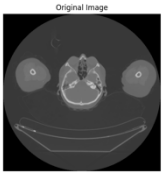
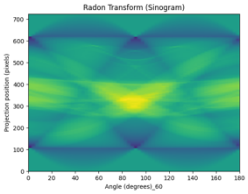
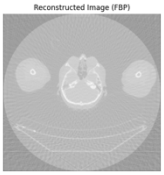
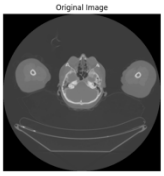
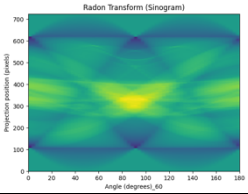
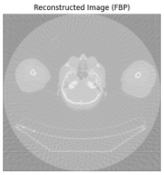
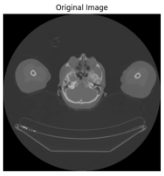
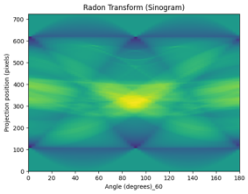
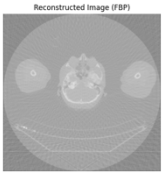
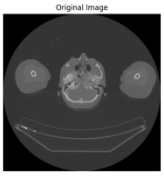
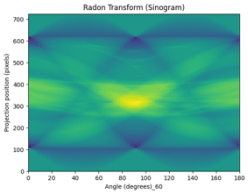
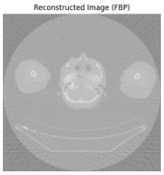
```
        ax2.set_ylabel("Projection position (pixels)")

        ax3.set_title("Reconstructed Image (FBP)")
        ax3.imshow(reconstruction_fbp, cmap='gray')
        ax3.axis('off')

        plt.show()

# Display PSNR and SSIM matrices
print("PSNR Matrix:")
print(psnr_matrix)

print("\nSSIM Matrix:")
print(ssim_matrix)

print(dicom_image.pixel_array)
print(reconstruction_fbp)
```

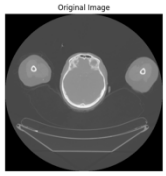
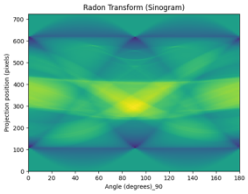
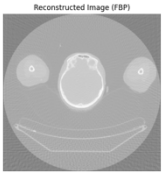
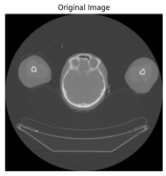
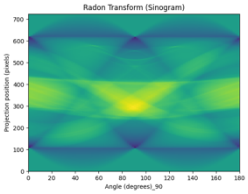
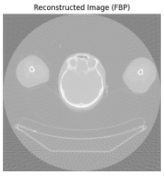
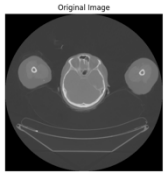
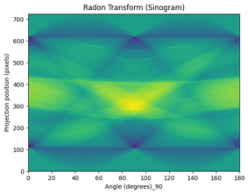
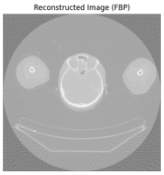
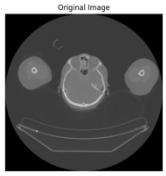
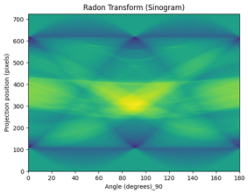
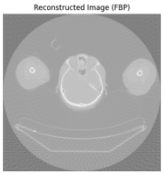
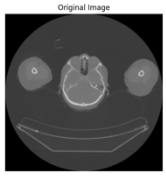
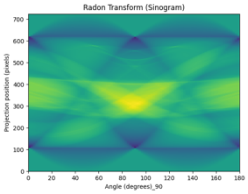
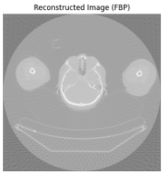
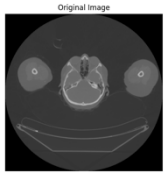
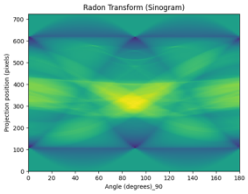
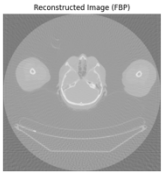
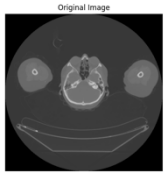
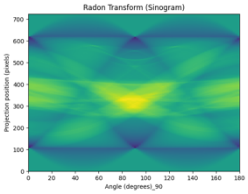
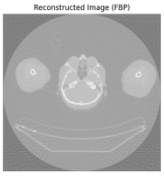
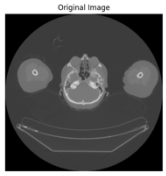
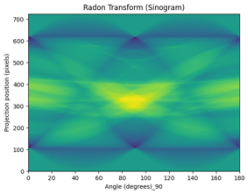
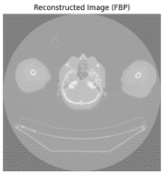
OUTPUT

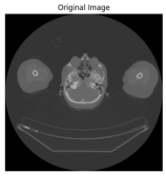
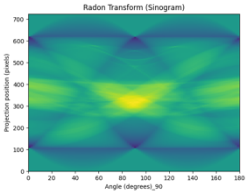
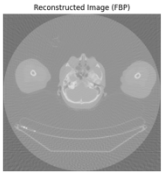
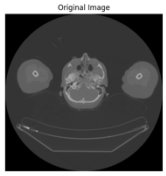
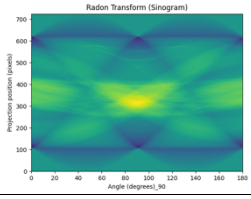
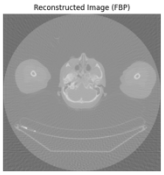
Sr.no.	Ground Truth	Projections_30	FBP
1			
2			
3			

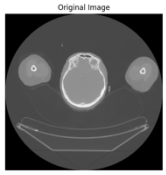
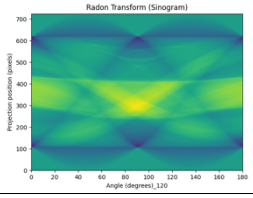
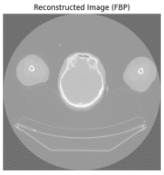
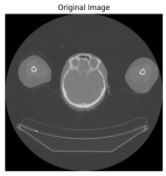
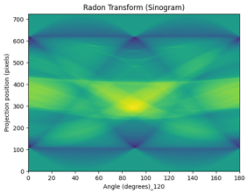
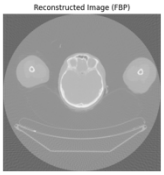
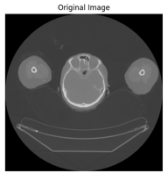
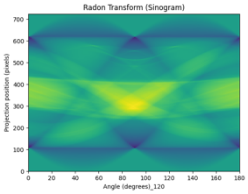
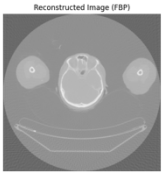
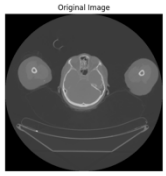
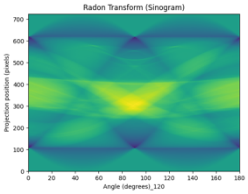
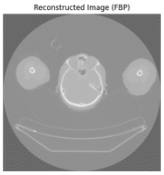
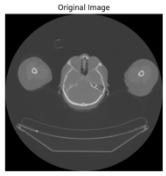
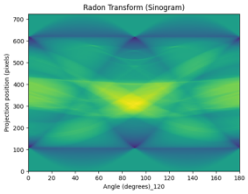
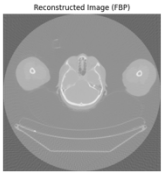
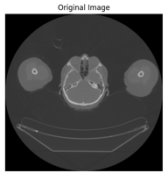
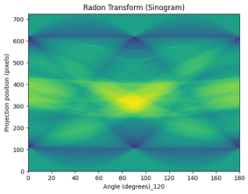
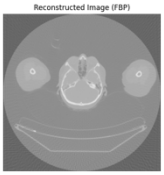
4			
5			
6			
7			
8			
9			
10			

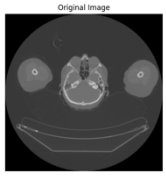
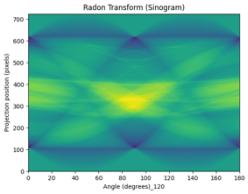
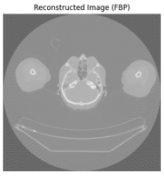
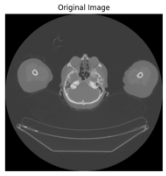
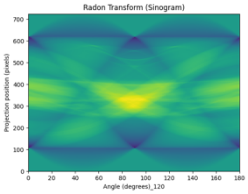
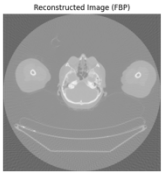
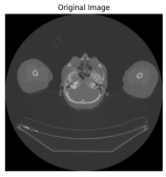
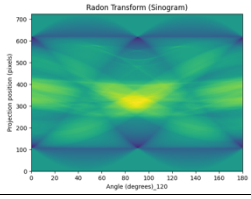
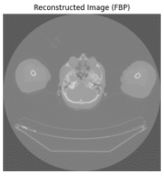
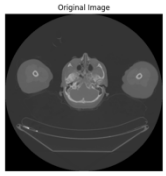
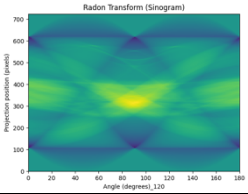
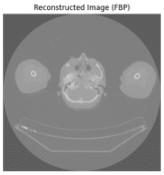
Sr.no.	Ground Truth	Projections_60	FBP
1			

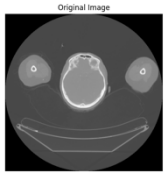
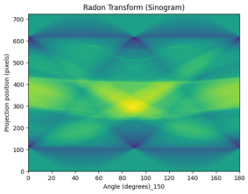
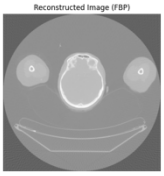
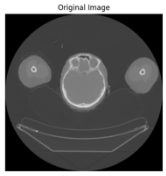
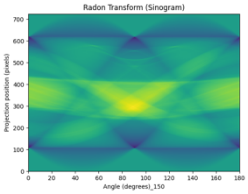
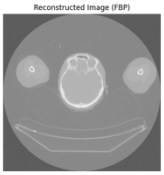
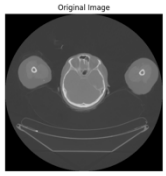
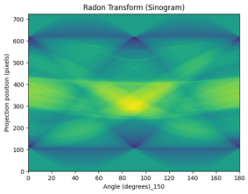
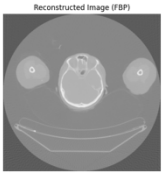
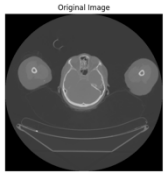
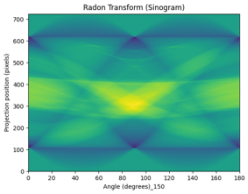
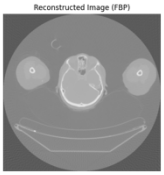
2	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
3	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
4	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
5	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
6	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
7	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
8	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
9	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)
10	 Original Image	 Radon Transform (Sinogram) Projection position (pixels) Angle (degrees)_60	 Reconstructed Image (FBP)

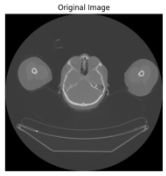
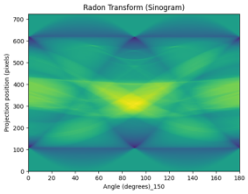
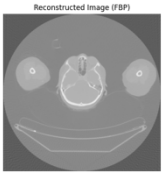
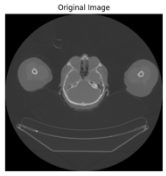
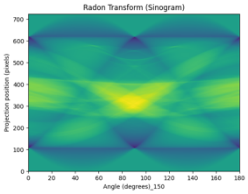
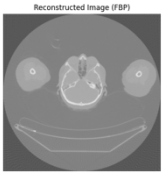
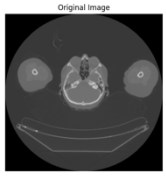
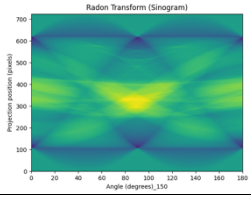
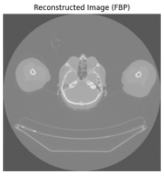
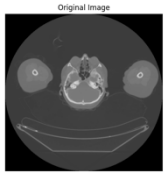
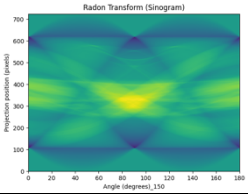
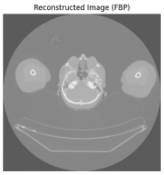
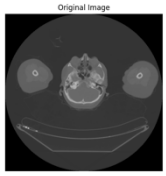
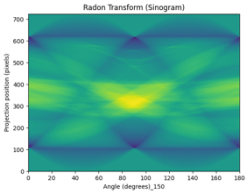
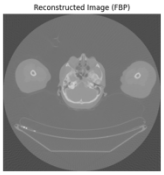
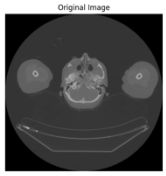
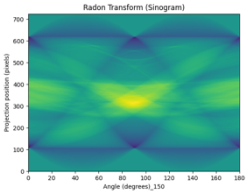
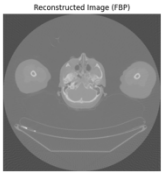
Sr.no.	Ground Truth	Projections_90	FBP
1			
2			
3			
4			
5			
6			
7			
8			

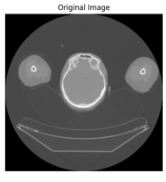
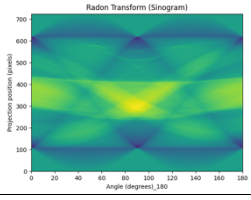
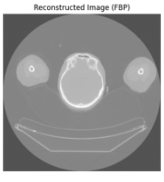
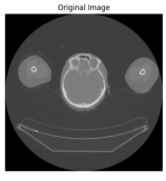
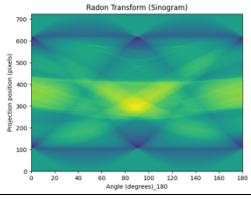
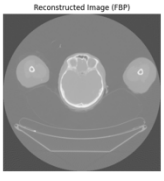
9			
10			

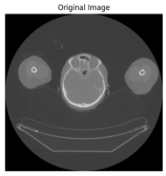
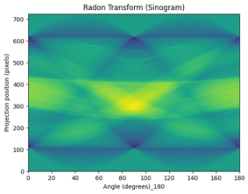
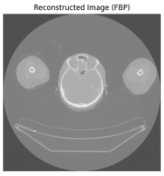
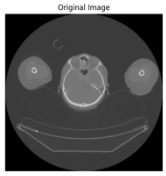
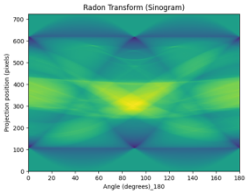
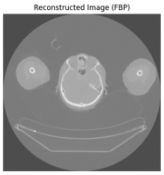
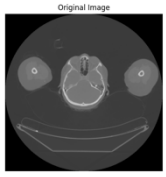
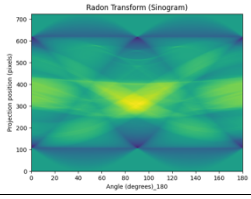
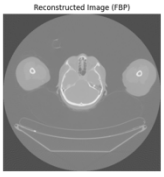
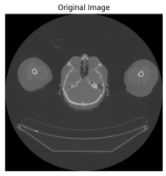
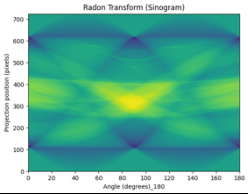
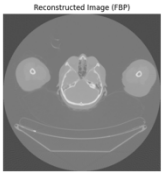
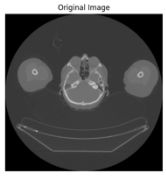
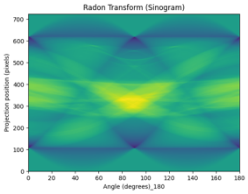
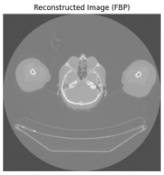
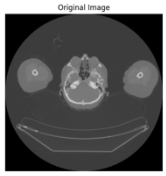
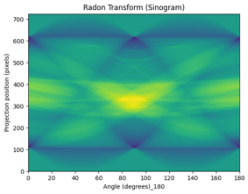
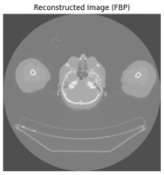
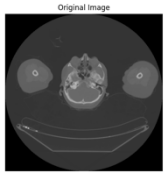
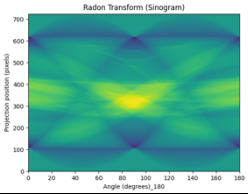
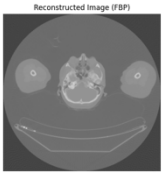
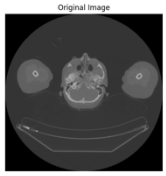
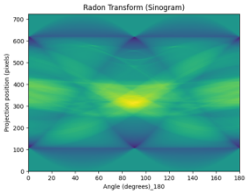
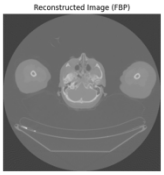
Sr.no.	Ground Truth	Projections_120	FBP
1			
2			
3			
4			
5			
6			

7			
8			
9			
10			

Sr.no.	Ground Truth	Projections_150	FBP
1			
2			
3			
4			

5			
6			
7			
8			
9			
10			

Sr.no.	Ground Truth	Projections_180	FBP
1			
2			

3			
4			
5			
6			
7			
8			
9			
10			

- PSNR Table:

Projection number	OBJ 1	OBJ 2	OBJ 3	OBJ 4	OBJ 5	OBJ 6	OBJ 7	OBJ 8	OBJ 9	OBJ 10
30	39.85957049	39.808657	39.85899031	39.82316372	39.75296015	39.68863157	39.658518 4	39.79195655	39.76338106	39.77826008
60	39.85954822	39.808634 9	39.85896815	39.82314153	39.75293832	39.68861004	39.65849702	39.79193468	39.76335919	39.77823795
90	39.85954177	39.80862855	39.85896167	39.82313511	39.75293206	39.68860387	39.65849084	39.79192828	39.76335284	39.77823159
120	39.85953886	39.80862568	39.85895878	39.82313224	39.75292926	39.68860111	39.65848812	39.79192544	39.76335003	39.77822876
150	39.85953727	39.80862409	39.8589572	39.82313067	39.75292769	39.68859957	39.65848659	39.79192386	39.76334847	39.77822722
180	39.85953633	39.80862314	39.85895622	39.8231297	39.75292675	39.68859863	39.65848567	39.79192291	39.76334753	39.77822625

- SSIM Table:

Projection number	OBJ 1	OBJ 2	OBJ 3	OBJ 4	OBJ 5	OBJ 6	OBJ 7	OBJ 8	OBJ 9	OBJ 10
30	.28567009	0.2886650 2	0.3063612 6	0.2951599 3	0.2798540 4	0.2759669 6	0.2479583 4	0.3110879 5	0.3099805 7	0.2801562 2
60	.28566919	0.2886642 4	.30636043	0.2951591 6	0.27985533	0.2759662 6	0.2479574 6	0.3110871 9	0.3099798 3	0.2801554 3
90	0.2856690 2	0.2886640 6	.30636025	0.2951590 1	0.2798531 1	0.2759660 6	0.2479572 2	0.3110870 1	0.3099796 5	.28015525
120	0.2856689 8	0.2886640 2	0.3063602 1	0.2951589 7	0.2798530 7	0.2759660 1	0.2479571 6	0.3110869 7	0.3099796 2	0.2801552 2

150	0.2856689	0.2886640 1	0.3063602 1	0.2951589 7	0.2798530 6	0.275966	0.2479571 6	.31108697	.30997961	0.2801552 2
180	0.2856689 7	.28866401	0.3063602	0.2951589 6	0.2798530 6	0.275966	.24795716	0.3110869 6	0.3099796 1	0.2801552 1

Chapter 4: Result and Conclusions

The reconstructive methods using a transform-based approach were able to produce good-quality images from fewer projections, as evident from the PSNR and SSIM values achieved. This approach appears to have a very high possibility of decreasing radiation exposure in medical imaging and not compromising the accuracy of diagnosis at the same time.

References

1. *Gengsheng Lawrence Zeng. "Medical Image Reconstruction", Springer Science and Business Media LLC, 2010*
2. *B.R. Ramesh, N. Srinivasa, K. Rajgopal, "An Algorithm for Computing the Discrete Radon Transform With Some Applications", Proceedings of the Fourth IEEE Region 10 International Conference, TENCON '89, 1989.*
3. *AC Kak, M Slaney, "Principles of Computerized Tomographic Imaging", IEEE Press 1988.*
4. *Van der Walt, S., Schönberger, J. L., Nunez-Iglesias, J., Boulogne, F., Warner, J. D., Yager, N., ... and Yu, T. "scikit-image: Image Processing in Python." Accessed [10 June 2024]. https://scikit-image.org.*
5. *scikit-image. (2024). radon_transform.py. In scikit-image/scikit-image. Retrieved from https://github.com/scikit-image/scikit-image/blob/v0.23.2/skimage/transform/radon_transform.py#L187-L323*