Symulator tomografu komputerowego

1. Skład grupy

- Tsimafei Kotski 158740

2. Zastosowany model tomografu

W projekcie wykorzystano równoległy model układu emiter–detektor, w którym promienie przechodzą przez obiekt równolegle.

3. Zastosowany język programowania oraz biblioteki

- Język programowania: Python
- Środowisko: Jupyter Notebook
- Biblioteki:
    - numpy – operacje na tablicach i macierzach
    - matplotlib – wizualizacja obrazów i wykresów
    - pydicom – obsługa plików DICOM
    - skimage – przetwarzanie obrazów
    - imageio – wczytywanie obrazów
    - math – obliczenia matematyczne
    - tkinter – tworzenie interfejsów użytkownika
    - PIL – praca z obrazami

4. Opis głównych funkcji programu

4.1 **Pozyskiwanie odczytów dla poszczególnych detektorów**

Odczyty detektorów uzyskiwane są za pomocą algorytmu transformacji Radona, który na podstawie obrazu przetwarza informacje o intensywności pikseli wzdłuż promieni padających na detektory. Detektory rozmieszczone są wzdłuż kątów projekcji, a dla każdego detektora obliczane są średnie intensywności na podstawie trajektorii promieni wyznaczonych przez algorytm Bresenhama

```python
def transformata_radona(obraz, krok_a, zakres_pi, liczba_detektorow, krok_x):
    alfa = math.pi / 2
    liczba_katow = math.ceil(180 / krok_x) + 1
    detektory = [[0, 0] for _ in range(liczba_detektorow)]
    emitery = [[0, 0] for _ in range(liczba_detektorow)]
    dlugosc = len(obraz)
    szerokosc = len(obraz[0])
    r = math.floor(math.sqrt(dlugosc * dlugosc + szerokosc * szerokosc) / 2) # promień obrazu, używany do wyznaczenia rozmiaru nowego obrazu

    obraz1 = [[0 for _ in range(r * 2)] for __ in range(r * 2)] # dwa razy większy rozmiar niż oryginalny obraz
    l, w = obraz.shape
    y_offset = round((r * 2 - l) / 2) # przesunięcie w osi Y
    x_offset = round((r * 2 - w) / 2) # przesunięcie w osi X
    obraz1 = np.array(obraz1.copy())
    obraz1[y_offset:y_offset + l, x_offset:x_offset + w] = obraz

    sinogram = []

    # Obliczanie sinogramu - przejście przez wszystkie kąty
    for i in range(liczba_katow):
        strzał = [0 for _ in range(liczba_detektorow)]  # projekcja 1D obrazu dla danego kąta, zawierająca średnie intensywności promieni dla każdego detektora

        for j in range(liczba_detektorow):
            # Obliczanie pozycji detektora
            x = r * math.cos(alfa + math.pi - (1 / 2) * zakres_pi + j * (zakres_pi / (liczba_detektorow - 1)) + (i * krok_a))
            y = r * math.sin(alfa + math.pi - (1 / 2) * zakres_pi + j * (zakres_pi / (liczba_detektorow - 1)) + (i * krok_a))
            detektory[j] = [round(x), round(y)]
```

```python
    # Obliczanie sinogramu - przejście przez wszystkie kąty
    for i in range(liczba_katow):
        strzał = [0 for _ in range(liczba_detektorow)]  # projekcja 1D obrazu dla danego kąta, zawierająca średnie intensywności promieni dla każdego detektora

        for j in range(liczba_detektorow):
            # Obliczanie pozycji detektora
            x = r * math.cos(alfa + math.pi - (1 / 2) * zakres_pi + j * (zakres_pi / (liczba_detektorow - 1)) + (i * krok_a))
            y = r * math.sin(alfa + math.pi - (1 / 2) * zakres_pi + j * (zakres_pi / (liczba_detektorow - 1)) + (i * krok_a))
            detektory[j] = [round(x), round(y)]

            # Obliczanie pozycji emitera naprzeciw detektora
            x_e = r * math.cos(alfa + (1 / 2) * zakres_pi - j * (zakres_pi / (liczba_detektorow - 1)) + (i * krok_a))
            y_e = r * math.sin(alfa + (1 / 2) * zakres_pi - j * (zakres_pi / (liczba_detektorow - 1)) + (i * krok_a))
            emitery[j] = [round(x_e), round(y_e)]

            # Wyznaczanie pikseli promienia (emiter -> detektor)
            punkty = algorytm_bresenhama(emitery[j][0], emitery[j][1], detektory[j][0], detektory[j][1])

            k = 1  # Licznik próbek na promieniu
            for x_p, y_p in punkty:
                # Sprawdzamy, czy punkt mieści się w granicach obrazu
                if x_offset < x_p + r < x_offset + w and y_offset < y_p + r < y_offset + l:
                    # Dodajemy wartość intensywności pikseli do projekcji
                    strzał[j] += obraz1[x_p + r - 1][y_p + r - 1]
                    k += 1
```

```python
            if k > 1:
                k -= 1
            strzał[j] /= k  # Średnia intensywność promienia

        sinogram.append(strzał)  # Dodajemy wynik dla danego kąta

    sinogram = np.rot90(sinogram, 1, axes=(0, 1))  # Obracamy wynik, by uzyskać właściwy układ sinogramu
    return sinogram
```

4.2 **P**rzetwarzanie końcowe obrazu

Rekonstrukcja obrazu polega na zastosowaniu odwrotnej transformacji Radona, która przywraca pierwotny obraz na podstawie wcześniej obliczonego sinogramu. Po obróbce sinogramu, intensywności pikseli są przypisane do odpowiednich punktów w siatce obrazowej.

```python
def odwrotna_transformata_radona(sinogram, krok_x):
    x = np.arange(sinogram.shape[0]) - sinogram.shape[0] / 2 # współrzędne X wyśrodkowane względem obrazu
    y = x.copy()
    ox, oy = np.meshgrid(x, y) #siatka współrzędnych (ox, oy)
    num = sinogram.shape[0]
    rekonstrukcja = np.array([[0 for _ in range(num)] for __ in range(num)])  # pusta macierz dla rekonstrukcji obrazu
    katy = np.arange(0.0, 180, krok_x) * (-1) * math.pi / 180  # tablica kątów w radianach (od 0 do 180 stopni)


    for j in range(len(katy)):
        _projekcja = np.array([[0 for _ in range(num)] for __ in range(num)])
        rotacja = ox * math.sin(katy[j]) - oy * math.cos(katy[j])  # Obliczanie nowych współrzędnych po obrocie
        punkty = np.round(rotacja + num / 2)  # Przesunięcie i zaokrąglenie
        punkty = punkty.astype('int')  # Konwersja na liczby całkowite


        p1, p2 = np.where((punkty >= 0) & (punkty < num)) # indeksy pikseli, które mieszczą się w obrazie
        linia = sinogram[:, j] # linia danych dla aktualnego kąta z sinogramu
        _projekcja[p1, p2] = linia[punkty[p1, p2]] # Przypisujemy wartości intensywności do odpowiednich pikseli w _projekcja
        rekonstrukcja += _projekcja

    rekonstrukcja = rekonstrukcja // len(katy) # Uśredniamy wynik dla wszystkich kątów
    return rekonstrukcja
```

4.3 **Odczyt i zapis plików DICOM**

Funkcja zapisu plików DICOM została zaimplementowana przy użyciu biblioteki pydicom, co pozwala na tworzenie plików DICOM z danymi pacjenta oraz szczegółami dotyczącymi badania. Funkcja odczytu umożliwia odczyt zapisanych plików DICOM i wyświetlenie obrazu pacjenta. Proces obejmuje wprowadzenie danych pacjenta przez interfejs, który generuje i zapisuje plik DICOM w formacie zgodnym ze standardem.

```python
import numpy as np
import cv2
import pydicom as pd
from pydicom.dataset import FileDataset, FileMetaDataset, validate_file_meta
from pydicom.uid import UID, generate_uid
import ipywidgets as widgets
from IPython.display import display

# Funkcja do zapisania pliku DICOM
def dicom_write(file, data):

    image = cv2.imread(data['image'], cv2.IMREAD_GRAYSCALE)

    file_meta = FileMetaDataset()
    file_meta.MediaStorageSOPClassUID = UID('1.2.840.10008.5.1.4.1.1.2')
    file_meta.MediaStorageSOPInstanceUID = generate_uid()
    file_meta.ImplementationClassUID = generate_uid()
    file_meta.TransferSyntaxUID = UID('1.2.840.10008.1.2.1')

    ds = FileDataset(file, {}, file_meta=file_meta, preamble=b'\0'*128)
    ds.PatientName = data['name']
    ds.PatientID = data['id']
    ds.is_little_endian = True
    ds.is_implicit_VR = False
    ds.StudyDate = data['date']

    ds.SeriesInstanceUID = generate_uid()
    ds.StudyInstanceUID = generate_uid()
    ds.FrameOfReferenceUID = generate_uid()
```

```python
    ds.BitsStored = 8
    ds.BitsAllocated = 8
    ds.SamplesPerPixel = 1
    ds.HighBit = 7

    ds.Rows = image.shape[0]
    ds.Columns = image.shape[1]

    ds.ImagesInAcquisition = '1'
    ds.InstanceNumber = 1
    ds.ImageType = r'ORIGINAL\PRIMARY\AXIAL'

    ds.RescaleIntercept = '0'
    ds.RescaleSlope = '1'
    ds.PixelSpacing = r'1\1'
    ds.PhotometricInterpretation = 'MONOCHROME2'
    ds.PixelRepresentation = 0

    ds.ImageComments = data['comment']
    ds.PixelData = image.tobytes()

    validate_file_meta(ds.file_meta, enforce_standard=True)

    ds.save_as(file, write_like_original=False)
```

```python
# Funkcja do wprowadzania danych pacjenta
def input_patient_data():
    name_widget = widgets.Text(value='', description='Imię pacjenta:')
    id_widget = widgets.Text(value='', description='ID pacjenta:')
    date_widget = widgets.DatePicker(description='Data badania:')
    image_widget = widgets.Text(value='', description='Ścieżka obrazu:')
    comment_widget = widgets.Text(value='', description='Komentarze:')

    def on_button_click(b):
        # Zapisanie danych
        patient_data = {
            'name': name_widget.value,
            'id': id_widget.value,
            'date': date_widget.value.strftime('%Y%m%d') if date_widget.value else '',
            'image': image_widget.value,
            'comment': comment_widget.value,
        }
        # Wywołanie funkcji zapisu DICOM
        dicom_write('file.dcm', patient_data)

    button = widgets.Button(description="Utwórz DICOM")
    button.on_click(on_button_click)

    display(name_widget, id_widget, date_widget, image_widget, comment_widget, button)

# Uruchomienie interfejsu do wprowadzania danych
input_patient_data()
```

Imię pacje... Tsimafei Kotski

ID pacjenta: 158740

Data bada... 04/04/2005

Ścieżka o... Shepp_logan.jpg

Komentarze: Projekt - tomograf

Utwórz DICOM

Odczyt DICOM

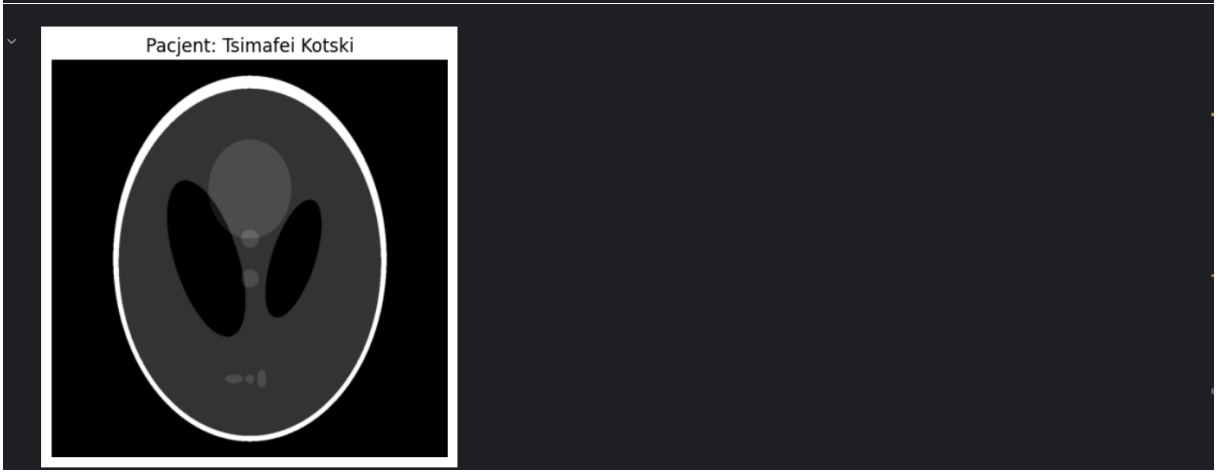
```python
# Funkcja do odczytu pliku DICOM
def dicom_read(file):
    dicom = pd.dcmread(file)
    print(dicom)
    plt.imshow(np.array(dicom.pixel_array).astype(np.uint8), cmap='gray')
    plt.title(f"Pacjent: {dicom.PatientName}")
    plt.axis('off')
    plt.show()

dicom_read('file.dcm')
```

Executed at 2025.04.15 15:11:43 in 102ms

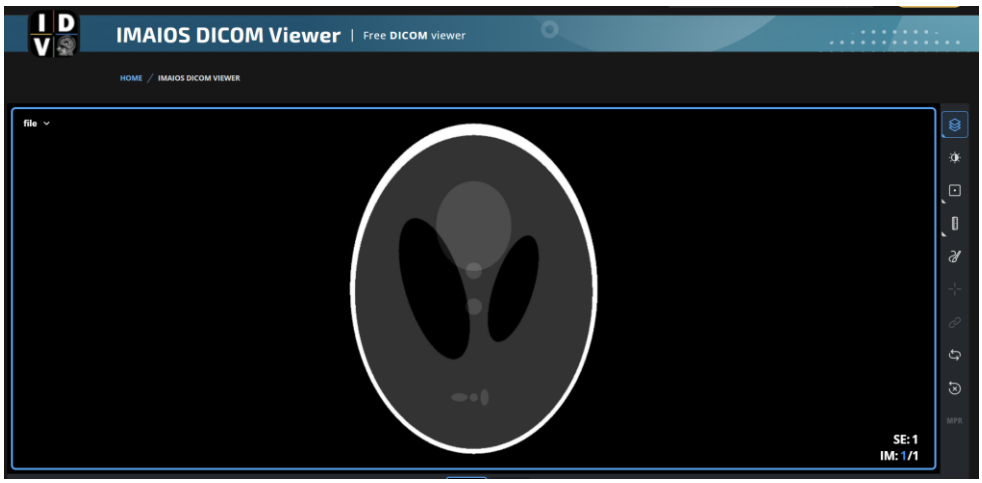
```
Dataset.file_meta -------------------------------
(0002, 0000) File Meta Information Group Length  UL: 242
(0002, 0001) File Meta Information Version       OB: b'\x00\x01'
(0002, 0002) Media Storage SOP Class UID         UI: CT Image Storage
(0002, 0003) Media Storage SOP Instance UID      UI: 1.2.826.0.1.3680043.8.498.726858502229543186322851694479466296666
(0002, 0010) Transfer Syntax UID                 UI: Explicit VR Little Endian
(0002, 0012) Implementation Class UID            UI: 1.2.826.0.1.3680043.8.498.31413375906044188102807508242074151294
(0002, 0013) Implementation Version Name         SH: 'PYDICOM 2.4.4'
-------------------------------------------------
```

```
(0008, 0008) Image Type                      CS: ['ORIGINAL', 'PRIMARY', 'AXIAL']
(0008, 0020) Study Date                      DA: '20050404'
(0010, 0010) Patient's Name                  PN: 'Tsimafei Kotski'
(0010, 0020) Patient ID                      LO: '158740'
(0020, 000d) Study Instance UID              UI: 1.2.826.0.1.3680043.8.498.90739843317053869108915243359917625210
(0020, 000e) Series Instance UID             UI: 1.2.826.0.1.3680043.8.498.88077989850877823433968805490602146615
(0020, 0013) Instance Number                 IS: '1'
(0020, 0052) Frame of Reference UID          UI: 1.2.826.0.1.3680043.8.498.11214148971559985494849586164883529192
(0020, 1002) Images in Acquisition           IS: '1'
(0020, 4000) Image Comments                  LT: Array of 18 elements
(0028, 0002) Samples per Pixel               US: 1
(0028, 0004) Photometric Interpretation      CS: 'MONOCHROME2'
(0028, 0010) Rows                            US: 1024
(0028, 0011) Columns                         US: 1024
(0028, 0030) Pixel Spacing                   DS: [1, 1]
(0028, 0100) Bits Allocated                  US: 8
(0028, 0101) Bits Stored                     US: 8
(0028, 0102) High Bit                        US: 7
(0028, 0103) Pixel Representation            US: 0
(0028, 1052) Rescale Intercept               DS: '0.0'
(0028, 1053) Rescale Slope                   DS: '1.0'
(7fe0, 0010) Pixel Data                      OB: Array of 1048576 elements
```

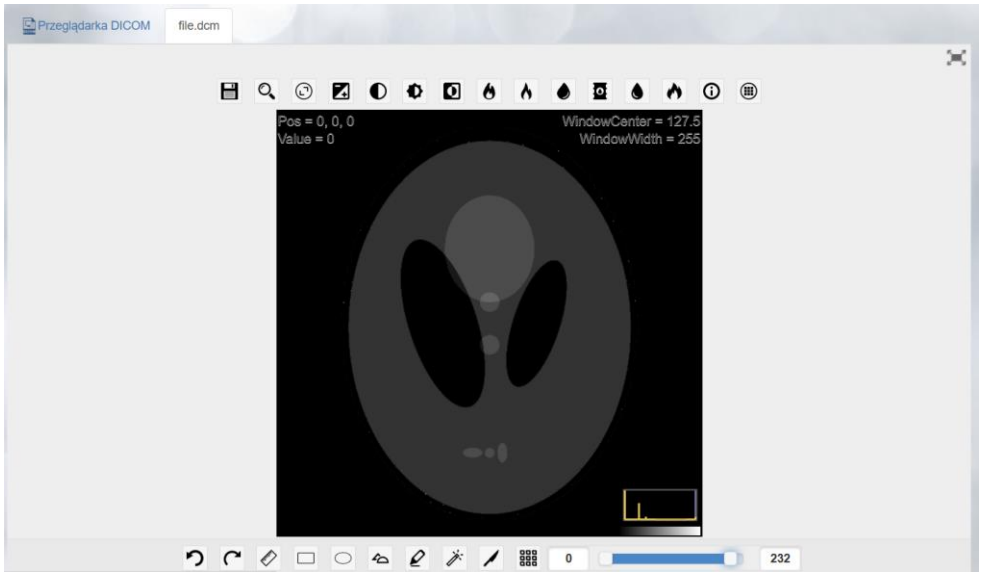


Poprawność zapisanego pliku została zweryfikowana na stronach:

https://www.imaios.com/en/Imaios-Dicom-Viewer

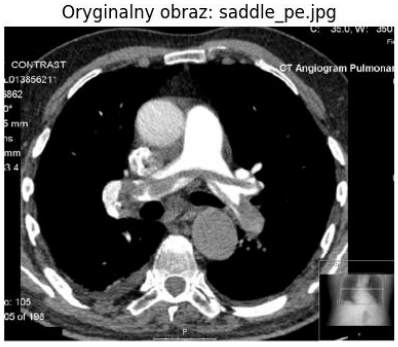


https://www.fviewer.com/pl/view-dicom

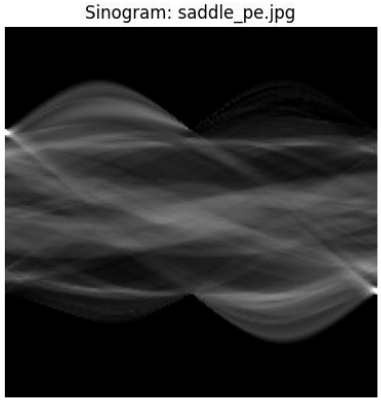


5. Przykład działania programu

Oryginalny obraz: saddle_pe.jpg



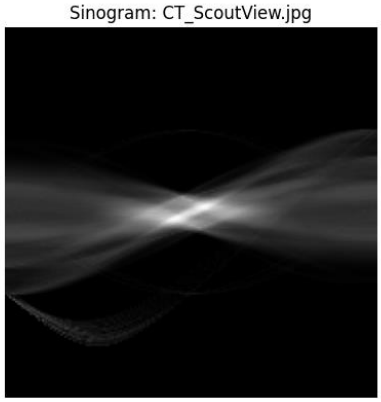
Sinogram: saddle_pe.jpg



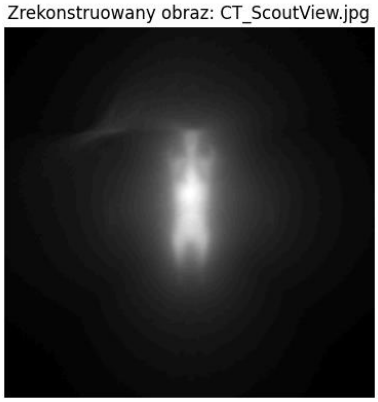
Zrekonstruowany obraz: saddle_pe.jpg

Oryginalny obraz: CT_ScoutView.jpg

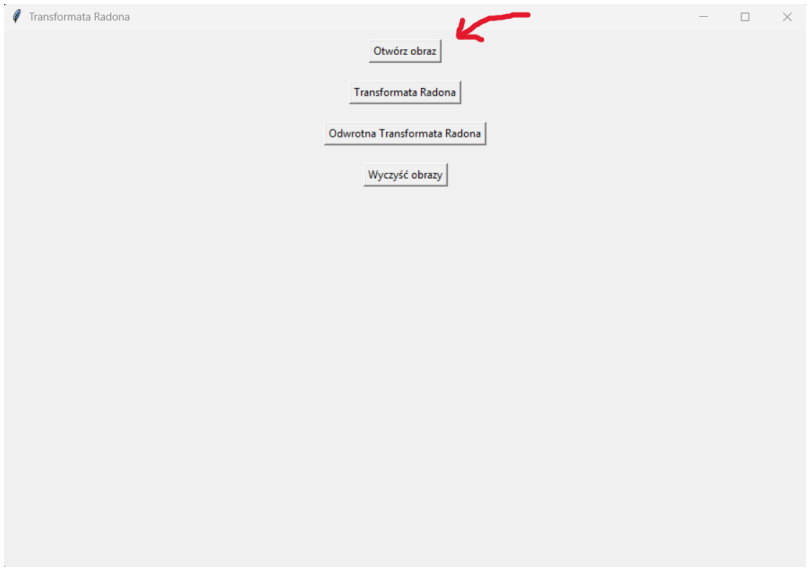
Sinogram: CT_ScoutView.jpg

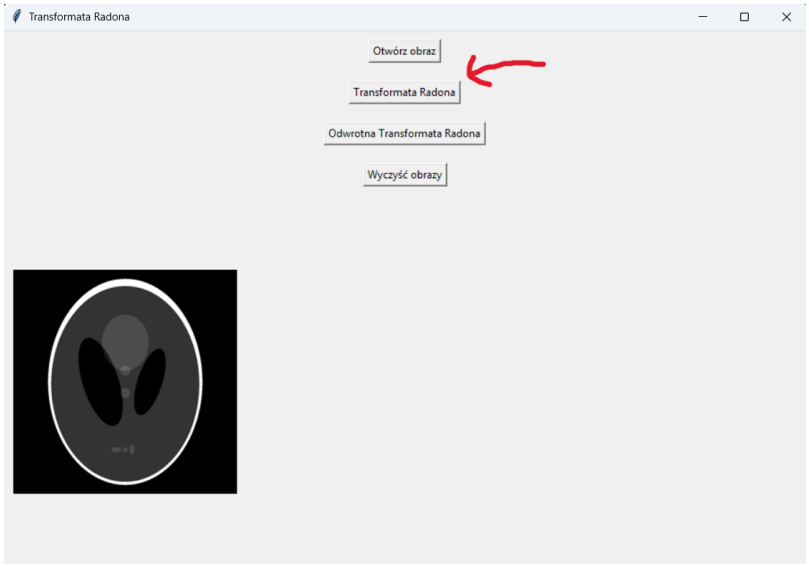


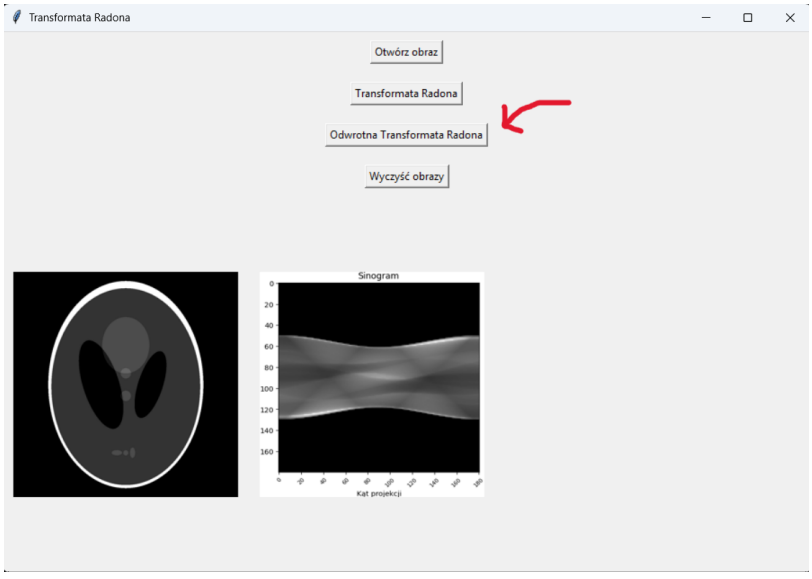
Zrekonstruowany obraz: CT_ScoutView.jpg

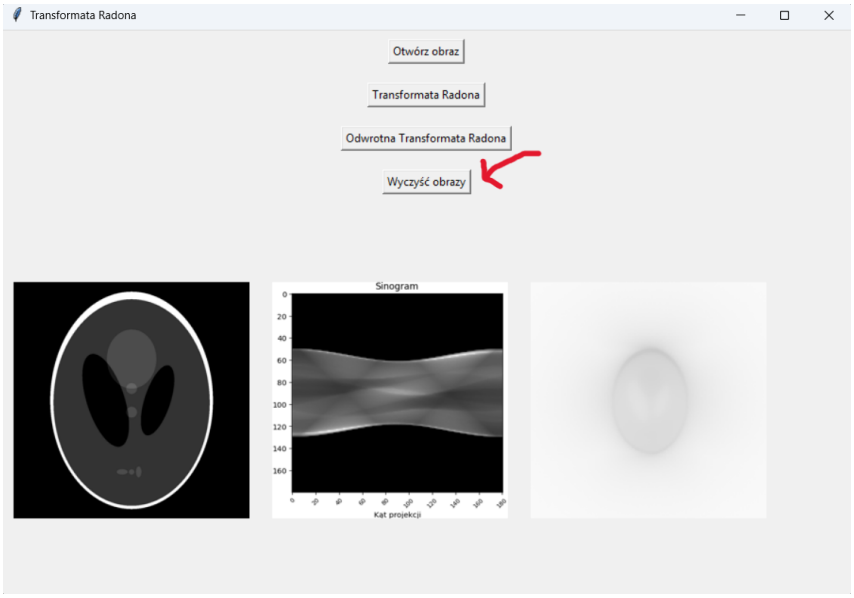


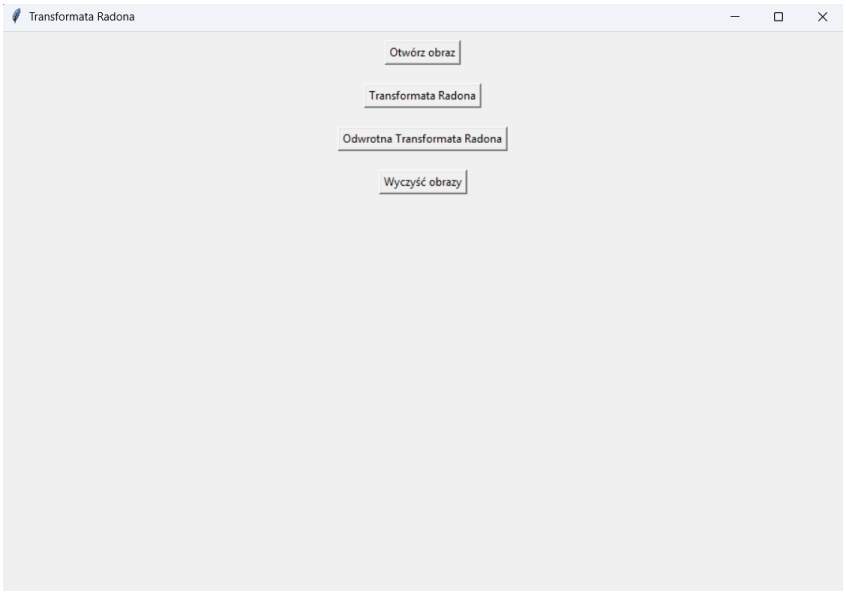
6. User Interface


Transformata Radona
Otwórz obraz
Transformata Radona
Odwrotna Transformata Radona
Wyczyść obrazy


Transformata Radona
Otwórz obraz
Transformata Radona
Odwrotna Transformata Radona
Wyczyść obrazy


Transformata Radona
Otwórz obraz
Transformata Radona
Odwrotna Transformata Radona
Wyczyść obrazy
Sinogram
Kąt projekcji


Transformata Radona
Otwórz obraz
Transformata Radona
Odwrotna Transformata Radona
Wyczyść obrazy
Sinogram
0
20
40
60
80
100
120
140
160
Kąt projekcji


Transformata Radona
Otwórz obraz
Transformata Radona
Odwrotna Transformata Radona
Wyczyść obrazy