Tomograf – projekt 1

Informatyka w medycynie

1. Skład grupy

Mateusz Kreczmer 151736

Piotr Krzyszowski 151909

2. Zastosowany model tomografu (stożkowy lub równoległy)

W naszym rozwiązaniu zastosowaliśmy równoległy model tomografu.

3. Zastosowany język programowania oraz dodatkowe biblioteki

Nasz projekt zrealizowaliśmy w pythonie, używając przy tym następujących bibliotek:

- os
- streamlit
- PIL
- numpy
- matplotlib.pyplot
- skimage
- pydicom
- math

4. Opis głównych funkcji programu (ilustracja za pomocą fragmentów kodu źródłowego)

a. pozyskiwanie odczytów dla poszczególnych detektorów

Główną funkcją programu jest przeprowadzenie transformacji Radona oraz odwrotnej transformacji Radona w celu rekonstrukcji obrazu na podstawie zebranych danych sinograficznych. Poniżej zawarte są fragmenty kodu odpowiadające za pozyskiwanie odczytów dla poszczególnych detektorów:

```python
def calculate_sinogram_data(self):
    results = np.zeros((self.scan_count, self.devices_count))

    for i, angle in enumerate(self.measurement_angles):
        results[i] = self.radon_transform(angle)

    results = normalize_array(results)

    self.sinogram = results
    self.transposed_sinogram = np.transpose(results)

def radon_transform(self, angle):
    emitters_coords = self.get_emitters_coords(angle)
    detectors_coords = self.get_detectors_coords(angle)
    lines = self.get_lines_between_devices(emitters_coords, detectors_coords)

    result = normalize_array(np.array([np.sum((self.image[tuple(line)])) for line in lines]))

    return result
```

```python
def get_emitters_coords(self, angle):
    return self.get_devices_coords(angle)

def get_detectors_coords(self, angle):
    return self.get_devices_coords(angle + 180)[::-1]

def get_devices_coords(self, angle):
    devices_angles = np.linspace(0, self.range_angle, self.devices_count) + radians(angle)
    center_x, center_y = self.center
    devices_x = (self.radius * np.cos(devices_angles) - center_x).astype(int)
    devices_y = (self.radius * np.sin(devices_angles) - center_y).astype(int)
    devices_coords = list(zip(devices_x, devices_y))

    return devices_coords
```

b. ustalanie jasności poszczególnych punktów obrazu wynikowego oraz jego przetwarzanie końcowe (np. uśrednianie, normalizacja)

Funkcja *inverseRadonTransform* odpowiedzialna jest za przeprowadzenie odwrotnej transformacji Radona, która polega na odtworzeniu obrazu na podstawie sinogramu. W tej funkcji jasność poszczególnych punktów obrazu wynikowego jest ustalana poprzez sumowanie odpowiednich wartości pikseli na podstawie danych z sinogramu.

Natomiast funkcja *calcResultData* przetwarza ostateczny obraz wynikowy po zakończeniu odwrotnej transformacji Radona. W tej funkcji dokonywana jest normalizacja obrazu, czyli dostosowanie wartości pikseli do zakresu ustalonego przez lowerBound i upperBound.

```
def calculate_result_data(self):
    for _, angle in enumerate(self.measurement_angles):
        self.inverse_radon_transform(angle)

    self.normalization_matrix[self.normalization_matrix == 0] = 1
    self.result_image = normalize_array(self.result_image / self.normalization_matrix)

def inverse_radon_transform(self, angle):
    print(angle)
    emitters_coords = self.get_emitters_coords(angle)
    detectors_coords = self.get_detectors_coords(angle)
    lines = self.get_lines_between_devices(emitters_coords, detectors_coords)

    for i, line in enumerate(lines):
        for point in np.transpose(line):
            if int(angle / (180 / self.scan_count)) < self.scan_count:
                self.result_image[point[0], point[1]] += np.transpose(self.transposed_sinogram)[int(angle / (180 / self.
                scan_count)), i]
                self.normalization_matrix[point[0], point[1]] += 1
```

c. odczyt i zapis plików DICOM (wymagania na 4.0).

Klasa *DicomIO* umożliwia odczyt i zapis plików DICOM. Metoda *read* służy do odczytu istniejącego pliku DICOM z dysku, a metoda *write* służy do zapisu danych pacjenta oraz obrazu do nowego pliku DICOM. Dodatkowo, metoda *get_patient_data* umożliwia dostęp do danych pacjenta wczytanych z pliku DICOM.

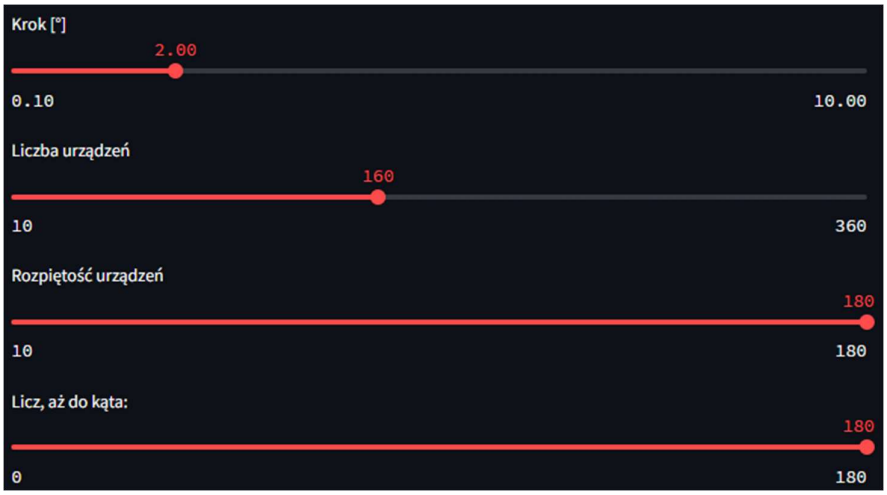
```
class DicomIO:
    def __init__(self):
        self.filename = None
        self.patient_data = None

    def read(self, filename):
        if os.path.exists(filename):
            self.filename = filename
            self.patient_data = pydicom.dcmread(filename)
        else:
            raise FileNotFoundError(f"File not found: {filename}!")

    def get_patient_data(self):
        if self.patient_data:
            return self.patient_data
        else:
            raise ValueError("No data. File not loaded!")
```

```python
def write(self, patient_data, img):
    img_converted = img_as_ubyte(rescale_intensity(img, out_range=(0.0, 1.0)))

    meta = Dataset()
    meta.MediaStorageSOPClassUID = CTImageStorage
    meta.MediaStorageSOPInstanceUID = generate_uid()
    meta.TransferSyntaxUID = ExplicitVRLittleEndian

    ds = FileDataset(None, {}, preamble=b"\0" * 128)
    ds.file_meta = meta

    ds.SOPInstanceUID = meta.MediaStorageSOPInstanceUID
    ds.PatientName = patient_data["PatientName"]
    ds.PatientID = patient_data["PatientID"]
    ds.ImageComments = patient_data.get("ImageComments", "")

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

    ds.Rows, ds.Columns = img_converted.shape

    ds.Modality = "CT"
    ds.BitsAllocated = 8
    ds.BitsStored = 8
    ds.HighBit = 7
    ds.PixelRepresentation = 0
    ds.SamplesPerPixel = 1
    ds.PhotometricInterpretation = "MONOCHROME2"
    ds.ImageType = r"ORIGINAL\PRIMARY\AXIAL"

    ds.PixelData = img_converted.tobytes()

    ds.save_as(self.filename, write_like_original=False)
```

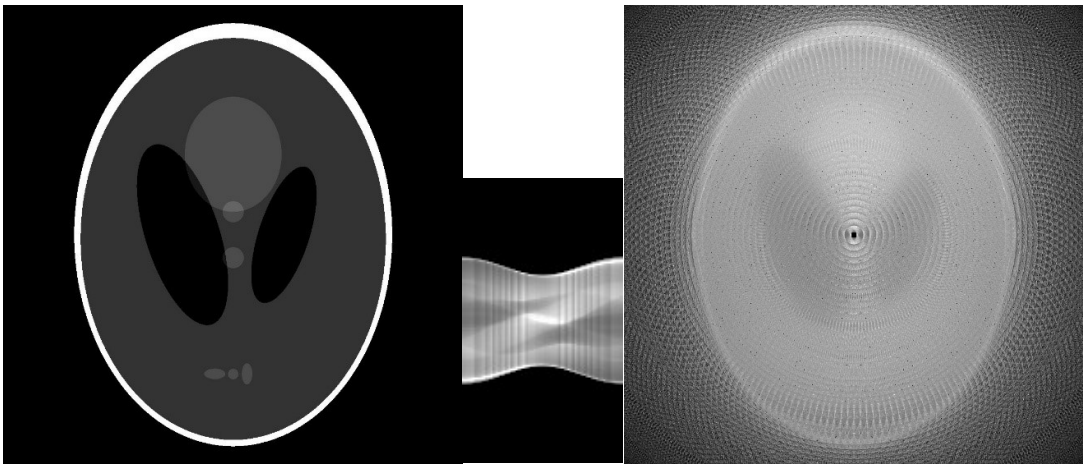
5. Przykład działania programu dla dwóch obrazków wejściowych (przedstawić zarówno obraz oryginalny jak i zrekonstruowany). Jeżeli zaimplementowano filtrowanie, to pokazać także efekt jego zastosowania na osobnym obrazku.

Wybrane zdjęcie: Shepp_logan.jpg

Ustawienia:

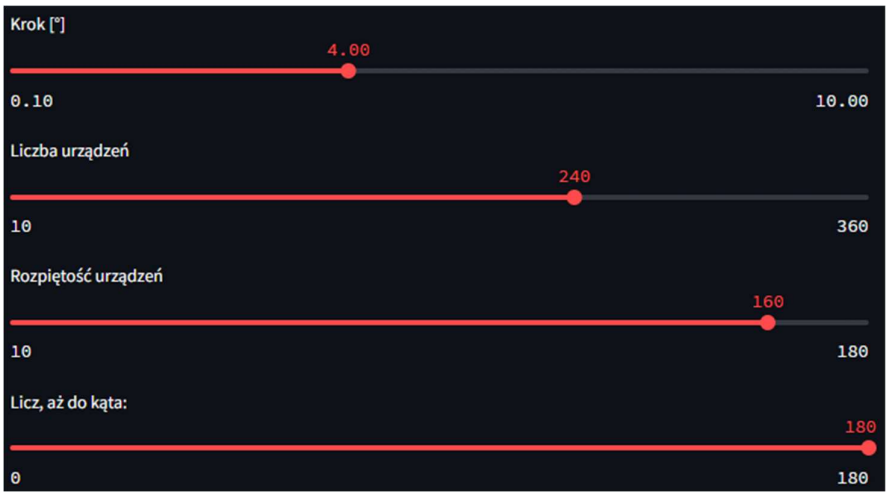


Obrazy (wejściowy/sinogram/wynikowy):



Wybrane zdjęcie: CT_ScoutView-large.jpg

Ustawienia:



Obrazy (wejściowy/sinogram/wynikowy):