Sprawozdanie – Symulacja tomografu komputerowego

1. Informacje ogólne

Skład grupy: **Kacper Garncarek 148114, Kamil Kałużny 148121**, grupa L8

Zastosowany model tomografu: równoległy.

Zastosowany język programowania: python, w tym biblioteki numpy, matplotlib, opencv, ipywidgets, Ipython,, skimage, pydicom

2. Główne funkcje programu

a. Pozyskiwanie odczytów dla poszczególnych detektorów.

```
for i, alpha in enumerate(angles):
    angles_det = np.deg2rad(np.linspace(0, phi, detector_num) + alpha - phi/2)
    angles_emit = np.deg2rad(np.linspace(0, phi, detector_num) + alpha - phi/2 + 180)

    detectors = np.zeros((detector_num, 2), dtype=int)
    emitters = np.zeros((detector_num, 2), dtype=int)

    for j, (angle_det, angle_emit) in enumerate(zip(angles_det, angles_emit)):
        xd = np.round(r * np.cos(angle_det) - x0).astype(int)
        yd = np.round(r * np.sin(angle_det) - y0).astype(int)
        xe = np.round(r * np.cos(angle_emit) - x0).astype(int)
        ye = np.round(r * np.sin(angle_emit) - y0).astype(int)
        detectors[j][0], detectors[j][1] = xd, yd
        emitters[detector_num-j-1][0], emitters[detector_num-j-1][1] = xe, ye

    pixels = np.zeros(detector_num, None)
    for j, (det_coord, emit_coord) in enumerate(zip(detectors, emitters)):
        beam_coord = bresenham(tuple(det_coord), tuple(emit_coord))
        pixels[j] = np.sum([img[coord] for coord in beam_coord])

    sinogram[i,] = pixels
```

Skorzystaliśmy z modelu addytywnego, odczyty dla każdego z detektorów to suma wartości pikseli znajdujących się na linii emiter – detektor.

b. Filtrowanie sinogramu

```
def apply_filter(sinogram, kernel_size):
    kernel = [1 if k == 0 else 0 if k % 2 == 0 else ((-4/(np.pi**2)) / (k**2)) for k in range(-kernel_size//2+1, kernel_size//2+1)]
    filtered_sinogram = np.zeros_like(sinogram)

    for i in range(sinogram.shape[0]):
        filtered_sinogram[i,] = np.convolve(sinogram[i,], kernel, mode='same')

    return filtered_sinogram
```

W wyniku serii eksperymentów, które miały określić optymalny rozmiar maski doszliśmy do wniosków, że w większości przypadków im większy rozmiar tym bardziej wyraźny zrekonstruowany obraz. Z tego względu domyślna długość maski w naszym programie jest równa długości wiersza sinogramu.

c. Ustalenie jasności poszczególnych punktów obrazu wynikowego oraz jego przetwarzanie końcowe

```
def inverse_radon_transform(sinogram, delta, out_scale=5.0, show_iter=True):

    angles = np.arange(0, 180 + delta, delta)
    detectors_num = sinogram.shape[1]
    size = np.round(np.sqrt((detectors_num) ** 2 / out_scale)).astype('int')
    reconstr_img = np.zeros((size, size))
    x0, y0 = np.mgrid[:size, :size] - size // 2
    x = np.arange(detectors_num) - detectors_num // 2
    angles = np.deg2rad(angles)

    for col, angle in zip(sinogram, angles):
        s = -x0 * np.sin(angle) + y0 * np.cos(angle)
        # t = x0 * np.cos(angle) + y0 * np.cos(angle)
        reconstr_img += np.interp(s, x, col)

        if show_iter:
            plt.imshow(reconstr_img, cmap='gray')
            plt.show()
            IPython.display.clear_output(wait=True)

    return reconstr_img
```

Obraz został zrekonstruowany w oparciu o interpolacje wartościami sinogramu. Zastosowano normalizacje MinMax.

d. Wyznaczanie wartości RMSE na podstawie obrazu źródłowego oraz wynikowego

```
def calculate_rmse(original_img, reconstructed):
    return np.sqrt(np.mean((reconstructed - original_img)**2))
```

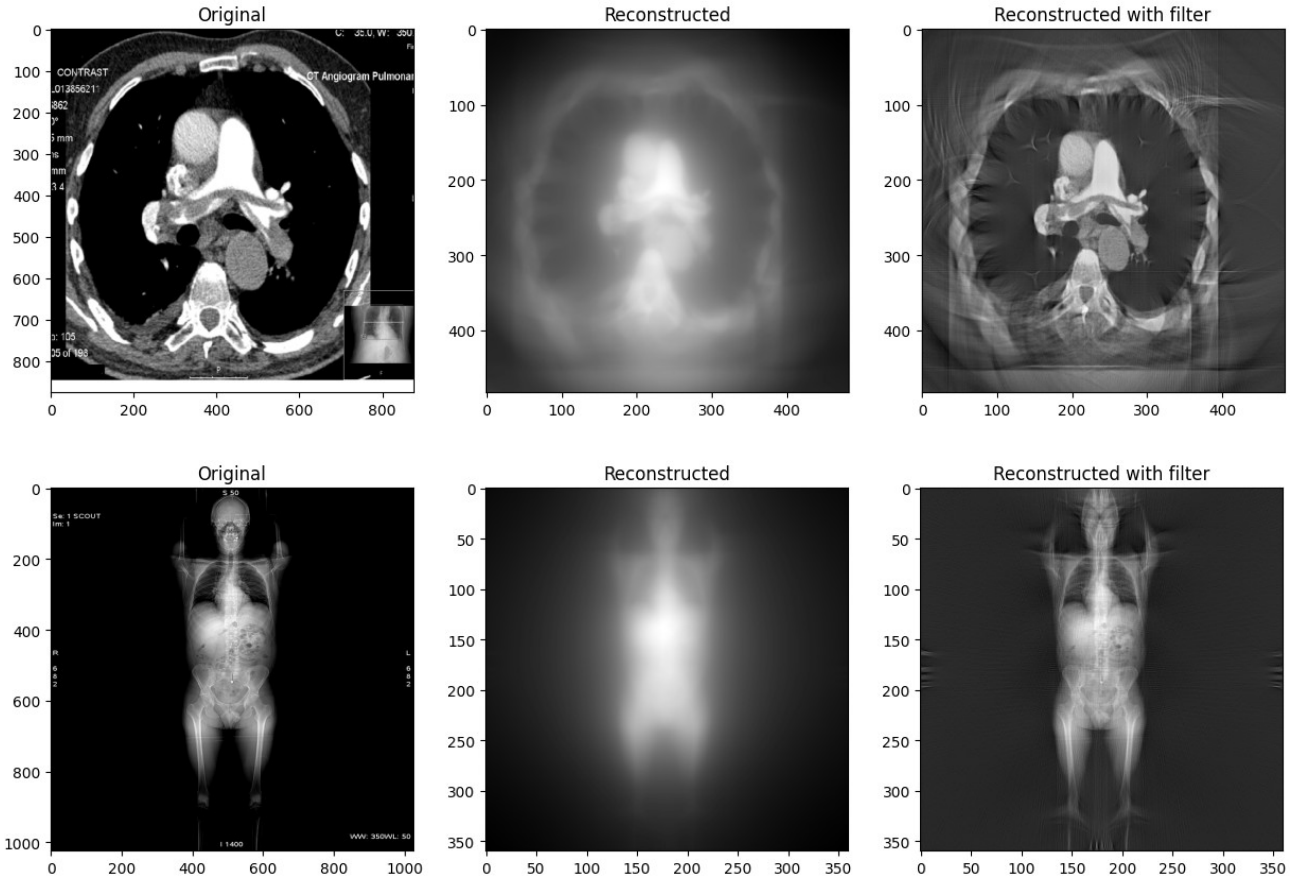
e. odczyt i zapis plików DICOM

```python
def convert_image_to_ubyte(img):
    return img_as_ubyte(rescale_intensity(img, out_range=(0.0, 1.0)))

def save_as_dicom(file_name, img, patient_data):
    img_converted = convert_image_to_ubyte(img)

    # Populate required values for file meta information
    meta = Dataset()
    meta.MediaStorageSOPClassUID = pydicom._storage_sopclass_uids.CTImageStorage
    meta.MediaStorageSOPInstanceUID = pydicom.uid.generate_uid()
    meta.TransferSyntaxUID = pydicom.uid.ExplicitVRLittleEndian

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

    ds.is_little_endian = True
    ds.is_implicit_VR = False

    ds.SOPClassUID = pydicom._storage_sopclass_uids.CTImageStorage
    ds.SOPInstanceUID = meta.MediaStorageSOPInstanceUID

    ds.PatientName = patient_data["PatientName"]
    ds.PatientID = patient_data["PatientID"]
    ds.ImageComments = patient_data["ImageComments"]
    ds.StudyDate = patient_data["StudyDate"]

    ds.Modality = "CT"
    ds.SeriesInstanceUID = pydicom.uid.generate_uid()
    ds.StudyInstanceUID = pydicom.uid.generate_uid()
    ds.FrameOfReferenceUID = pydicom.uid.generate_uid()

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

    ds.ImagesInAcquisition = 1
    ds.InstanceNumber = 1

    ds.Rows, ds.Columns = img_converted.shape

    ds.ImageType = r"ORIGINAL\PRIMARY\AXIAL"

    ds.PhotometricInterpretation = "MONOCHROME2"
    ds.PixelRepresentation = 0

    pydicom.dataset.validate_file_meta(ds.file_meta, enforce_standard=True)

    ds.PixelData = img_converted.tobytes()

    ds.save_as(file_name, write_like_original=False)

    print('Image saved successfully')

def read_dicom(filename):
    ds = pydicom.dcmread(filename)
    return ds.pixel_array
```

Dane pacjenta przekazywane pod postacią słownika patient_data.

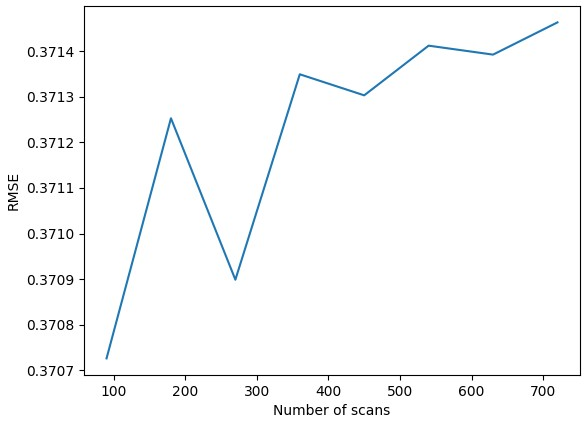
```python
filename = input("Enter file name: ")
patient_data = {}
patient_data["PatientName"] = input("Enter patient name: ")
patient_data["PatientID"] = input("Enter patient id: ") # str(np.random.randint(0,10000))
patient_data["ImageComments"] = input("Enter comments: ")
patient_data["StudyDate"] = datetime.now().date().strftime('%Y%m%d')

dicom_img = convert_image_to_ubyte(reconstructed_filtered)
save_as_dicom(dicom_dir+filename+'.dcm', dicom_img, patient_data)
```

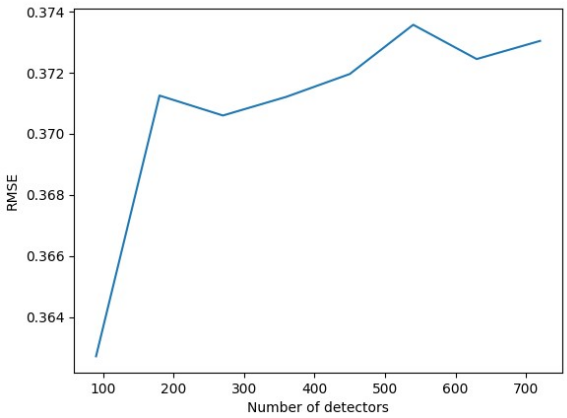
3. Przykład działania



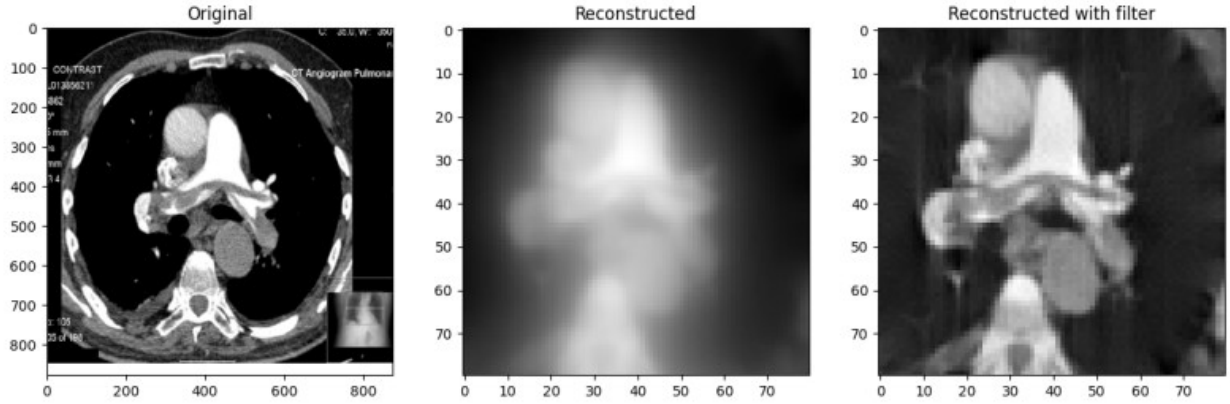
4. Wyniki eksperymentu

Wykresy zmian RMSE w zależności od zmian parametrów.

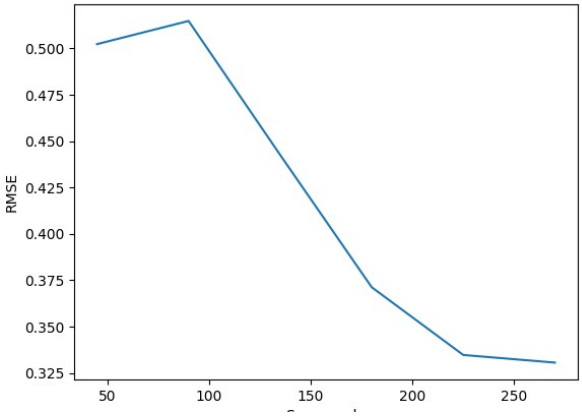




Niestety analiza wykresów zależności RMSE od liczby skanów oraz detektorów nie ma sensu ponieważ nie udało nam się przed deadlinem zmodyfikować błędu, który powodował że rekonstrukcja obrazów nie jest w takiej samej skali jak obraz wejściowy. Dla tego przykładu:

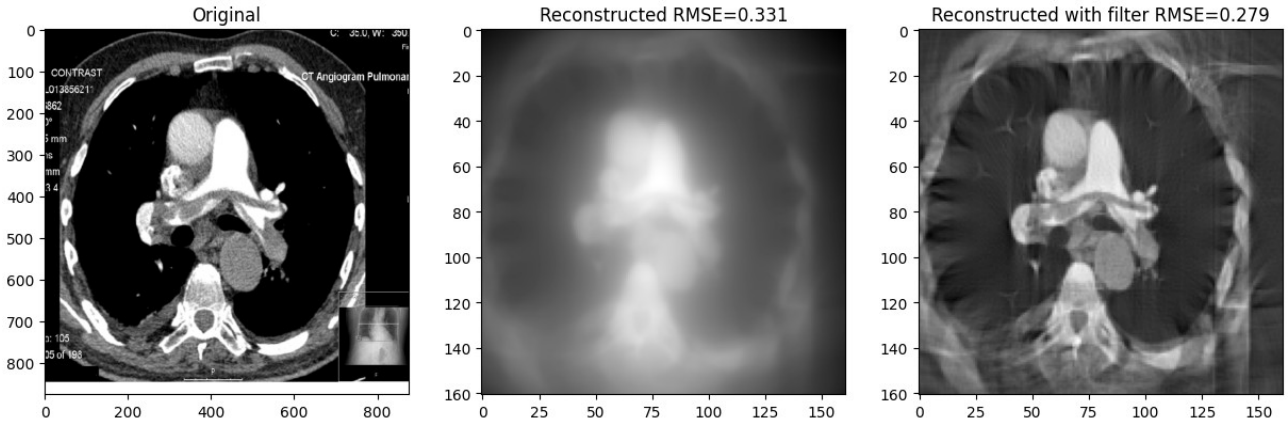


Dla kąta rozpiętości wygląda to nieco lepiej, jednak ma to również związek z niepoprawną skalą zrekonstruowanego obrazu:

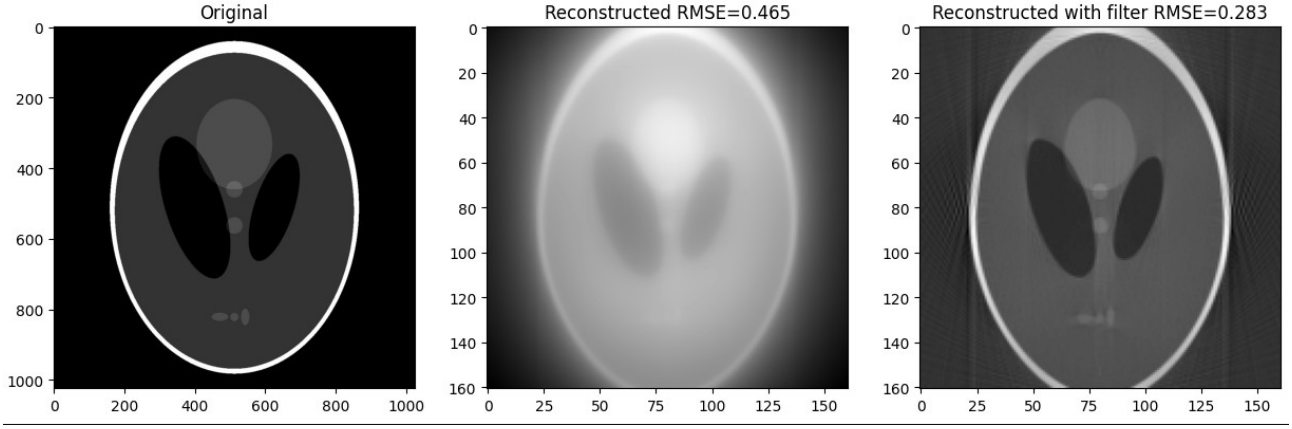


Wyniki dla dwóch wybranych obrazów dla następujących parametrów: liczba detektorów = 360, liczba skanów = 360, rozpiętość wachlarza = 270 stopni:

Obraz 1



Obraz 2



W obu przypadkach zaobserowano znacząca poprawę jakości obrazu po użyciu filtrowania.