Raport laboratoria: Informatyka w Medycynie
Projekt: Symulator Tomografu

Jakub Gardecki 141218 L7
Mateusz Kempa 139952 L7

Zastosowany model tomografu

Model stożkowy.

Zastosowany język programowania oraz dodatkowe biblioteki

Język programowania: Python

Najważniejsze biblioteki: matplotlib, numpy, ipywidgets, math, skimage, bresenham, pydicom.

Pozyskiwanie odczytów dla poszczególnych detektorów.

Pozyskiwanie odczytów zostało zaimplementowane zgodnie z podanymi na laboratoriach wzorami. Na podstawie współczynnika alfa zależnego od liczby kroków obliczana jest pozycja emitera i detektorów.

```
alfa = 360 / krok
# Tworzenie sinogramu
for i in range(0, krok):
    sys.stdout.write(str(i) + " / " + str(krok - 1) + '\r')
    sys.stdout.flush()
    xEmiter = int(round(promien * math.cos(math.radians(alfa * i)) + srodek[1], 1))
    yEmiter = int(round(promien * math.sin(math.radians(alfa * i)) + srodek[0], 1))
    for j in range(0, liczba_detektorow):
        xDetektor = int(round(promien * math.cos(
            math.radians(alfa * i + 180 - rozpietosc / 2 + j * rozpietosc / (liczba_detektorow - 1))), 1) + srodek[1])
        yDetektor = int(round(promien * math.sin(
            math.radians(alfa * i + 180 - rozpietosc / 2 + j * rozpietosc / (liczba_detektorow - 1))), 1) + srodek[0])
        Linia = bresenham(xEmiter, yEmiter, xDetektor, yDetektor)
        for x, y in Linia:
            sinogram[i, j] += zdj[x, y]
```

Filtrowanie sinogramu, zastosowany rozmiar maski

Program pozwala na zastosowanie filtracji sinogramu. Po wygenerowaniu maski wykonany jest splot sinogramu z maską, co zwiększa jakość wygenerowanych obrazów wyjściowych. W programie zastosowaliśmy zmienny rozmiar maski w zależności od liczby detektorów:
Rozmiar maski = liczba detektorów / 2.

Tworzenie maski

```
def tworzenie_maski(liczba_detektorow):
    rozmiar_maski = liczba_detektorow //2
    if(rozmiar_maski % 2 == 0):
        rozmiar_maski += 1
    maska = np.zeros((rozmiar_maski))
    top = -4/(np.pi**2)
    srodek = rozmiar_maski // 2
    for i in range(0, rozmiar_maski):
        maska[i] = (0 if (i-srodek)%2==0 else (top/((i-srodek)**2)) )
    maska[srodek]=1
    return maska
```

Splot

```
def splot(tab,maska,liczbaSkanow):
    wyj_sinogram = np.zeros_like(tab)
    for i in range(liczbaSkanow):
                wyj_sinogram[i] = np.convolve(tab[i], maska, mode = 'same')
    return wyj_sinogram
```

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

Jako przetwarzanie końcowe obrazu została zastosowana normalizacja.

Normalizacja

```
def normalizacja(zdj):
    min_value = np.min(zdj)
    max_value = np.max(zdj)

    for i in range(len(zdj)):
        for j in range(len(zdj[i])):
            zdj[i, j] = (zdj[i, j] - min_value) / (max_value - min_value)

    return zdj
```

Wyznaczanie wartości miary RMSE na podstawie obrazu źródłowego oraz wynikowego

Pierwiastek błędu średnio kwadratowego został wyznaczony w następujący sposób:

Pierwiastek błędu średniokwadratowego

```
def RMSE(tabA, tabB):
    return np.sqrt(np.mean((tabA-tabB)**2))
```

Odczyt i zapis plików DICOM

Program został rozszerzony również w możliwość generowania i odczytu wynikowego obrazu wraz z informacjami pacjenta w formacie DICOM.

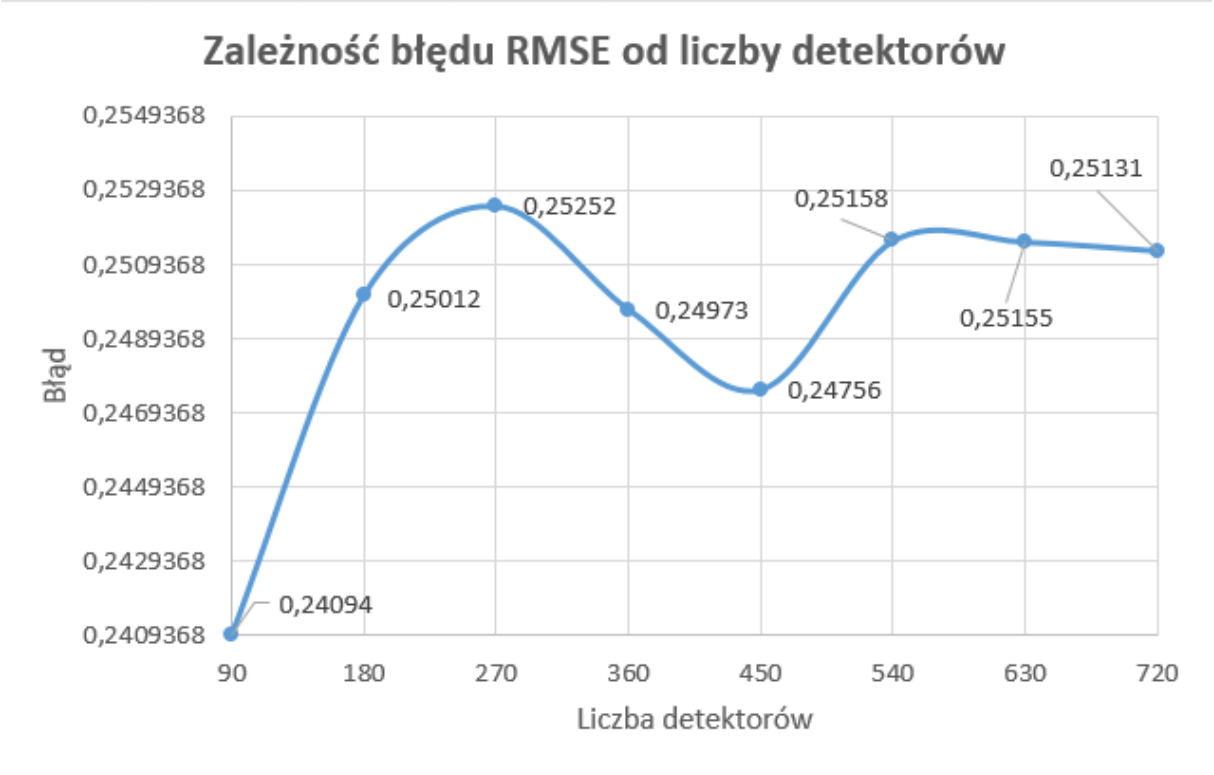
Zapis:

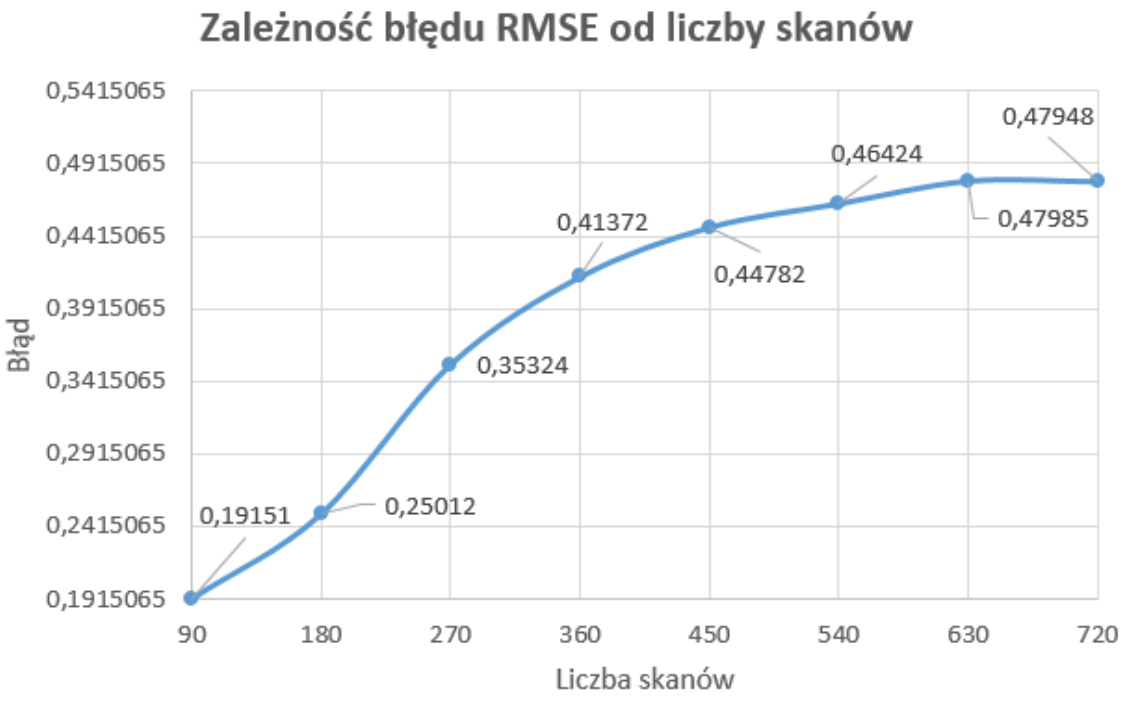
```
def save_dicom_file(pic, patient_data, filename="dicom/test.dcm"):
        filename_little_endian = tempfile.NamedTemporaryFile(suffix=".dcm").name
        file_meta = Dataset()
        file_meta.MediaStorageSOPClassUID = '1.2.840.10008.5.1.4.1.1.2'
        file_meta.MediaStorageSOPInstanceUID = "1.2.3"
        file_meta.ImplementationClassUID = "1.2.3.4"

        ds = FileDataset(filename_little_endian, {},
                         file_meta=file_meta, preamble=b"\0" * 128)
        ds.PatientName = patient_data["PatientName"]
        ds.PatientWeight = patient_data["PatientWeight"]
        ds.ImageComments = patient_data["ImageComments"]
        ds.StudyDate = patient_data["StudyDate"]
        ds.PatientBirthDate = patient_data["BirthDate"]


        if pic.dtype != np.uint8:
            img_max = np.max(pic)
            img_min = np.min(pic)
            pic = ((pic - img_min) / (img_max - img_min) * 255)
            pic = pic.astype(np.uint8)

        ds.SpecificCharacterSet = 'utf-8'
        ds.PixelData = pic.tobytes()
        ds.Rows, ds.Columns = pic.shape[0:2]
        ds.BitsStored = 8
        ds.BitsAllocated = 8
        ds.HighBit = 7
        ds.SamplesPerPixel = 1
        ds.PhotometricInterpretation = "MONOCHROME2"
        ds.PixelRepresentation = 0

        ds.save_as(filename, write_like_original=False)
        print("Plik pomyślnie zapisany.")

def on_create_dicom_clicked(x):
    global OdtworzoneZdj
    patient_data = {}
    patient_data["PatientName"] = imie_pacjenta_input.value
    patient_data["PatientWeight"] = waga_pacjenta_input.value
    patient_data["ImageComments"] = zdj_komentarz.value
    patient_data["StudyDate"] = data_badania_pacjenta.value
    patient_data["BirthDate"] = data_urodzenia_input.value
    print("Dane Pacjenta: ")
    print(patient_data)

    matplotlib.image.imsave("zdj/pom.jpg", OdtZdj)
    dicom_pic = imread("zdj/pom.jpg")
    dicom_pic = rgb2gray(dicom_pic)
    save_dicom_file(dicom_pic,patient_data)

```

Odczyt:

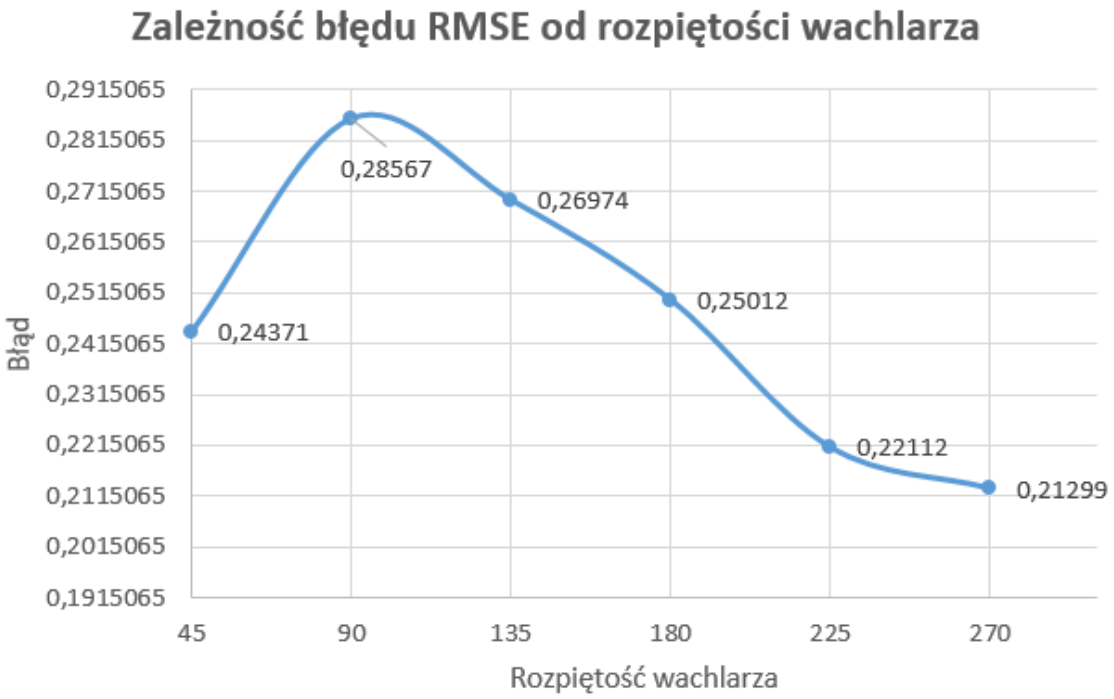
```python
def load_dicom_file(filename="dicom/test.dcm"):
        ds = pydicom.dcmread(filename)
        image = ds.pixel_array
        # Convert the image data type to uint8
        if image.dtype != np.uint8:
            img_max = np.max(image)
            img_min = np.min(image)
            image = 255 * (image - img_min) / (img_max - img_min)

        patient_data = {}

        if "PatientName" in ds:
            patient_data["PatientName"]=ds.PatientName
        if "ImageComments" in ds:
            patient_data["ImageComments"]=ds.ImageComments
        if "StudyDate" in ds:
            patient_data["StudyDate"]=ds.StudyDate
        if "PatientBirthDate" in ds:
            patient_data["BirthDate"]=ds.PatientBirthDate
        if "PatientWeight" in ds:
            patient_data["PatientWeight"]=ds.PatientWeight

        plt.imshow(image.astype(np.uint8), cmap=plt.cm.bone)
        return   image.astype(np.uint8), patient_data


def on_load_dicom_clicked(x):
    image, data = load_dicom_file()
    print(data)
```

Wyniki eksperymentu





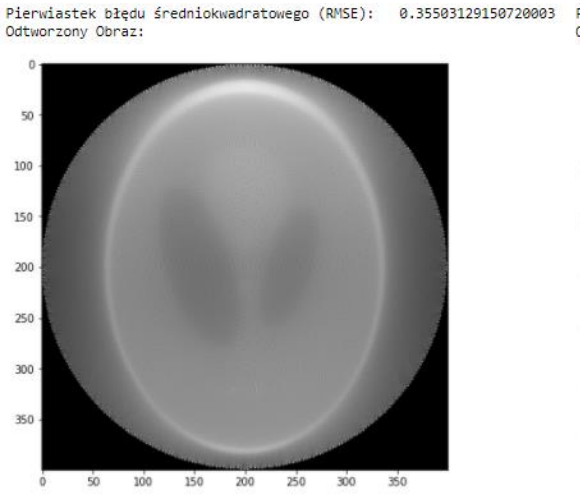
Pomimo tego, że wraz ze zwiększeniem liczby detektorów czy to liczby skanów zwiększa się widoczność szczegółów na zdjęciach to wraz wzrostem tych wartości rośnie rozmazanie obrazu wynikowego i rośnie jasna poświata (nierównomierny rozkład próbek w dziedzinie czasu) co skutkuje wzrostem błędu RMSE. Rozwiązaniem tego problemu byłoby zastosowanie filtracji jednak w tym badaniu nie została ona zastosowana.



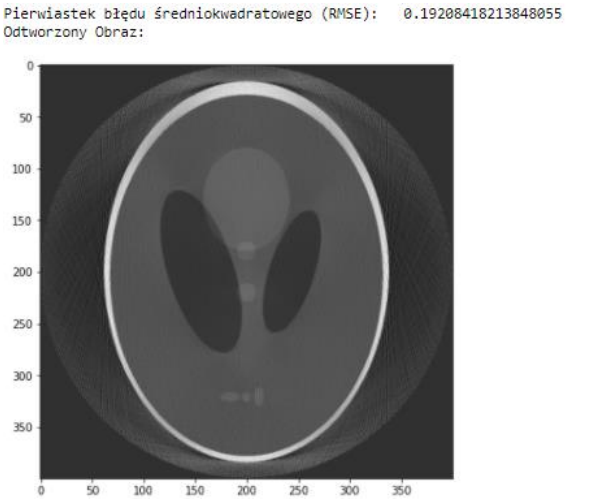
W tym przypadku od rozpiętości wachlarza przewyższającej 90 stopni zauważamy spadek miary RMSE. Jest to uzasadnione tym że wraz z zwiększaniem się rozpiętości wachlarza obraz wynikowy jest lepszej jakości. Pozostaje odpowiedź na pytanie dlaczego wartości dla 45 i 90 stopni są takie nietypowe. Jest to anomalia związana z tym faktem iż przy tak małej rozpiętości wachlarza obraz który nie znajduje się tylko w centrum zdjęcia jest praktycznie nieodtworzony (w eksperymencie został użyty obraz Shepp_logan). Przy takim małym kącie zostaje prześwietlony punkt centralny zdjęcia a jego reszta jest praktycznie nierozpoznawalna.

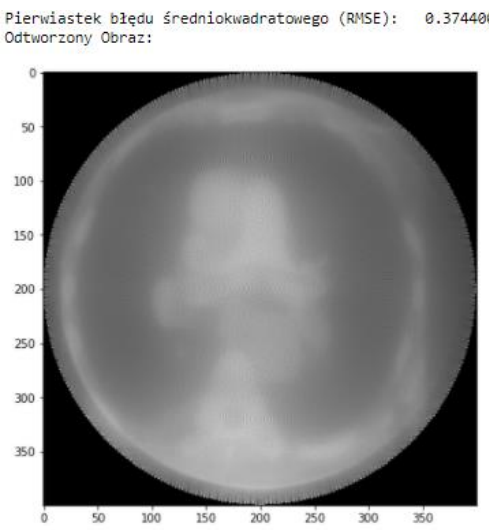
Dla dwóch wybranych obrazów oraz następujących parametrów: liczba detektorów = 360, liczba skanów = 360, rozpiętość wachlarza = 270 stopni, wykonać dwa warianty obliczeń -- z włączonym i wyłączonym filtrowaniem sinogramu.

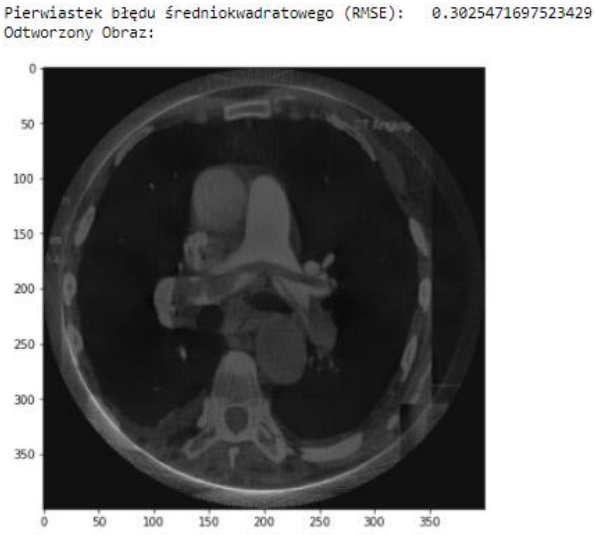
Obraz przed filtracją



Obraz po filtracji







Na piewszy rzut oka widać jak duże efekty przynosi filtracja sinogramu. Po filtracji obraz wynikowy jest bardzo zbliżony do obrazu wejściowego czego nie można powiedzieć o obrazie przed filtracją. Porównując miarę RMSE dla obu obrazów również widać bardzo duże zmniejszenie błędu po zastosowaniu filtra. Są to poprawy wyników jakich nie jesteśmy w stanie uzyskać nawet w najwyższej konfiguracji rozpiętości wachlarza, liczby skanów czy też liczby detektorów.