Symulator tomografu komputerowego

1. Skład grupy

Gustaw Kustoń 145215

2. Zastosowany model tomografu

Zastosowano model równoległy.

3. Język programowania i biblioteki

Wybranym językiem programowania jest Python.

Biblioteki wykorzystane do obliczeń to numpy, math, oraz bresenham (zawierająca implementację algorytmu bresenhama). Do tego wykorzystano bibliotekę matplotlib do wyświetlania obrazów, oraz pydicom do obsługi obrazów w formatach DICOM.

4. Opisy funkcji programu

a) **Pozyskiwanie odczytów dla poszczególnych detektorów:**

```
xCenter = r * sin(np.radians(i * alfa))
yCenter = r * cos(np.radians(i * alfa))
row_sinogram = []
row_num = []
row_coords = []
for j in range(n):
    sum_sinogram = 0
    xE = r * sin(np.radians(i * alfa + (j - (n - 1) / 2) * l))
    yE = r * cos(np.radians(i * alfa + (j - (n - 1) / 2) * l))
    xD = round(xE - 2 * xCenter + width / 2)
    yD = round(yE - 2 * yCenter + height / 2)
    xE = round(xE + width / 2)
    yE = round(yE + height / 2)
    row_coords.append([xE, yE, xD, yD])
    for pixel in list(bresenham(xE, yE, xD, yD)):
        if 0 <= pixel[0] < width and 0 <= pixel[1] < height:
            sum_sinogram += image_gray[pixel[1]][pixel[0]]
```

Na początku obliczany jest wektor idący od środka obrazu do pierwszego detektora. Następnie obliczane są pozycje kolejnych detektorów i przy pomocy wcześniejszego wektora (pomnożonego razy dwa) odpowiadających im punktów po drugiej stronie elipsy. Kolejny krok użycie dla tych danych algorytmu bresenhama. Jednocześnie należy sprawdzać, czy dany pixel należy do obrazu, bo jest to prostokąt wpisany w elipsę.

b) Ustalanie jasności poszczególnych punktów obrazu wynikowego, oraz jego przetwarzanie końcowe:

```
for pixel in list(bresenham(xE, yE, xD, yD)):
    if 0 <= pixel[0] < width and 0 <= pixel[1] < height:
        sum_sinogram += image_gray[pixel[1]][pixel[0]]
        pixel_number_in_line[pixel[1]][pixel[0]] += 1
```

Na etapie pobierania danych przez detektory liczona jest też ilość pixeli (należących do obrazu) na, którą natrafił ten detektor.

```
for j in range(n):
    for pixel in list(
        bresenham(
            coords[i][j][0], coords[i][j][1], coords[i][j][2], coords[i][j][3]
        )
    ):
        if 0 <= pixel[0] < width and 0 <= pixel[1] < height:
            new_image[pixel[1]][pixel[0]] += (
                sinogram[i][j] / pixel_number_in_line[pixel[1]][pixel[0]]
            )
```

Wartość ta używana jest przy odtwarzaniu obrazu, gdzie dzięki niej liczymy średnią.

```
def find_min_max(arr):
    max_value = 0
    min_value = 999999999
    for i in range(len(arr)):
        for j in arr[i]:
            if j < min_value:
                min_value = j
            if j > max_value:
                max_value = j
    return (min_value, max_value)
```

```
min_value, max_value = find_min_max(sinogram)
plt.imshow(sinogram, cmap="gray", vmin=min_value, vmax=max_value)
plt.show()
```

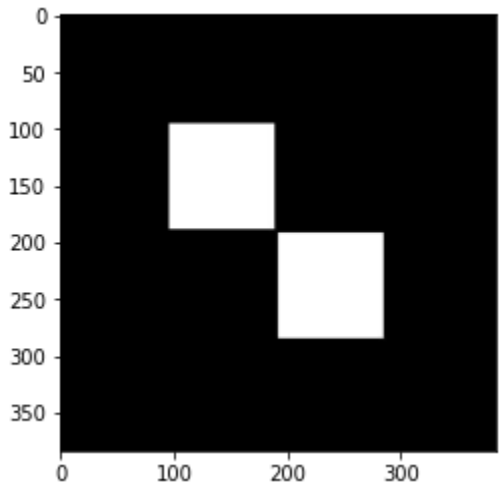
Normalizacja odbywa się automatycznie przez bibliotekę matplotlib i wymaga jedynie znalezienia najmniejszej i największej wartości w obrazie.

c) **Odczyt i zapis plików DICOM:**

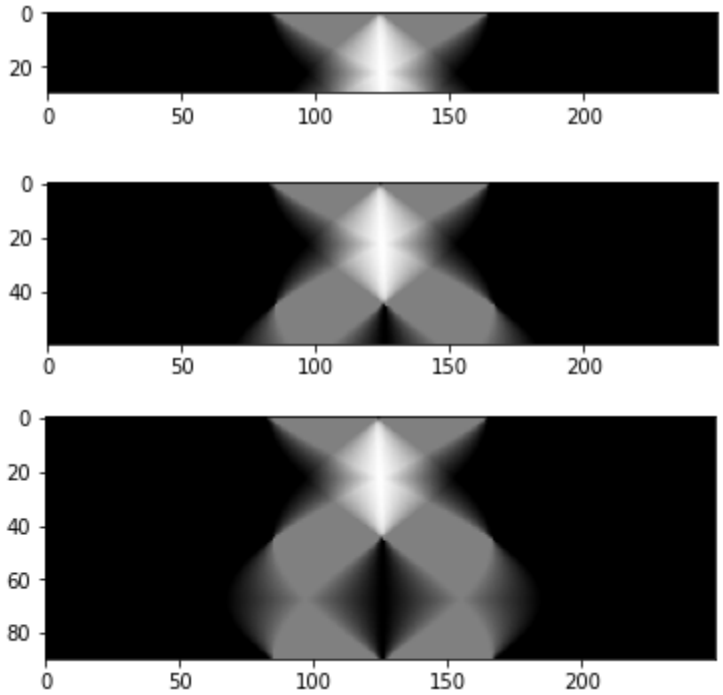
```
ds = dcmread("dicom_examples/Kwadraty2.dcm")

pat_name = ds.PatientName
display_name = pat_name.family_name + ", " + pat_name.given_name
print(f"Patient's Name...: {display_name}")
if hasattr(ds, "PatientID"):
    print(f"Patient ID.......: {ds.PatientID}")
if hasattr(ds, "Modality"):
    print(f"Modality.........: {ds.Modality}")
if hasattr(ds, "StudyDate"):
    print(f"Study Date.......: {ds.StudyDate}")
if hasattr(ds, "Rows") and hasattr(ds, "Columns"):
    print(f"Image size.......: {ds.Rows} x {ds.Columns}")
if hasattr(ds, "PixelSpacing"):
    print(f"Pixel Spacing....: {ds.PixelSpacing}")

image = ds.pixel_array
```

Odczyt plików wymaga jedynie sprawdzania, które atrybuty on posiada, a których nie. Zapis plików zaimplementowany został za pomocą funkcji z eKursów.

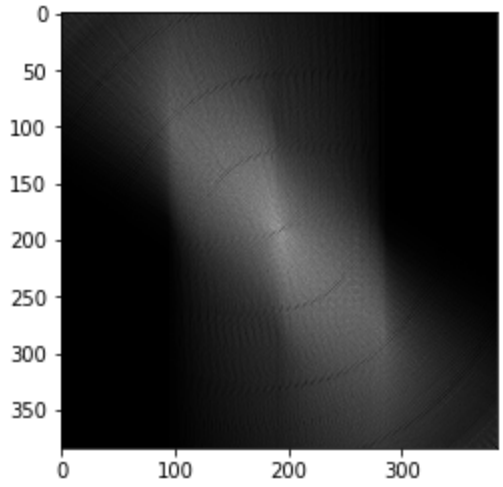
5. Przykładowe wyniki

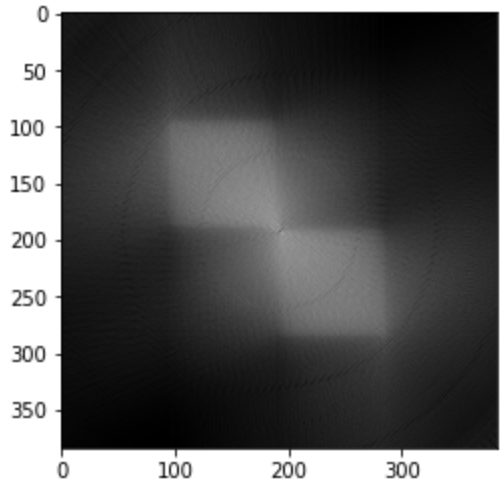


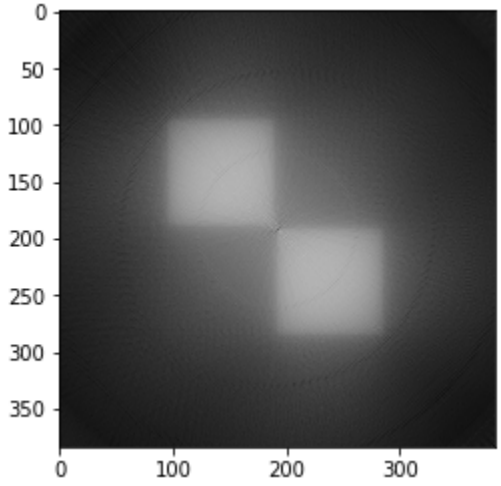
Oryginalny obraz.



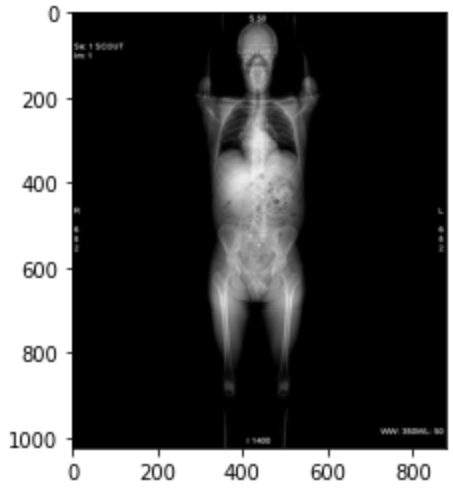
Sinogram - wyniki pośrednie i ostateczny.



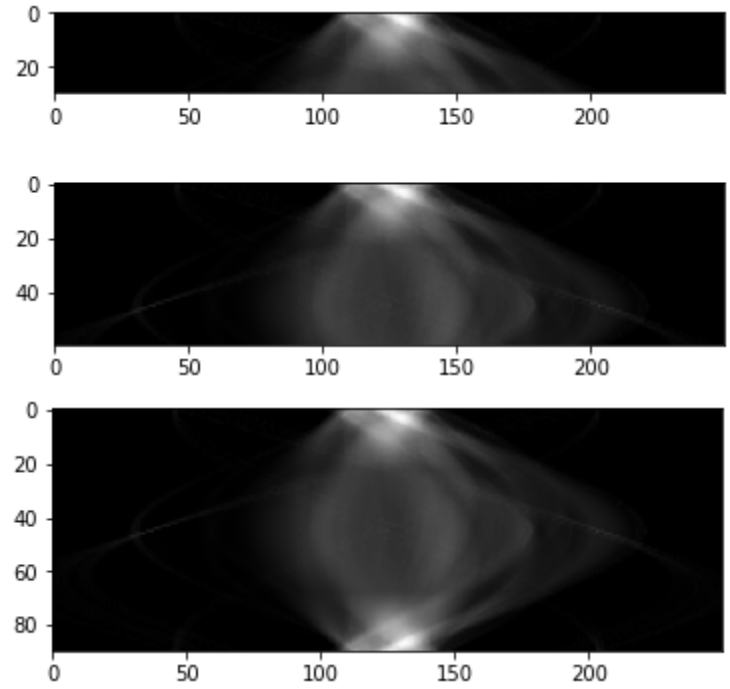




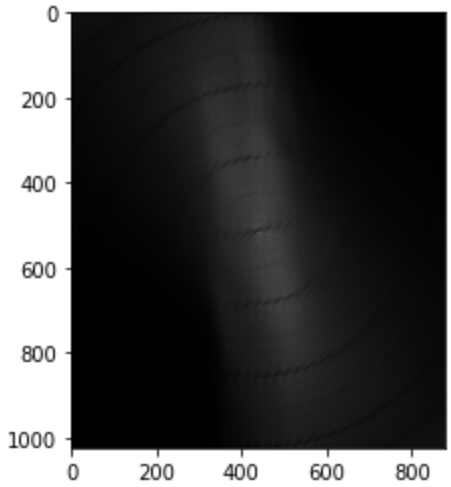
Odtworzony obraz - wyniki pośrednie i ostateczny.

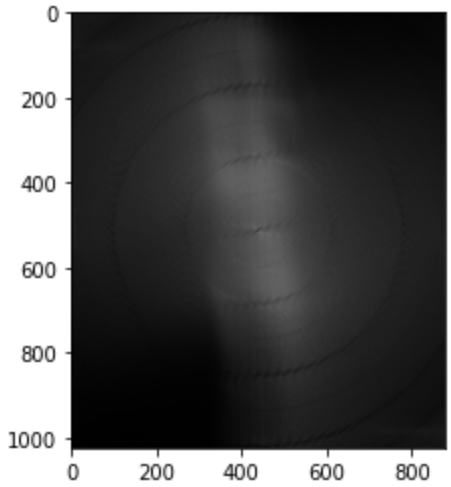


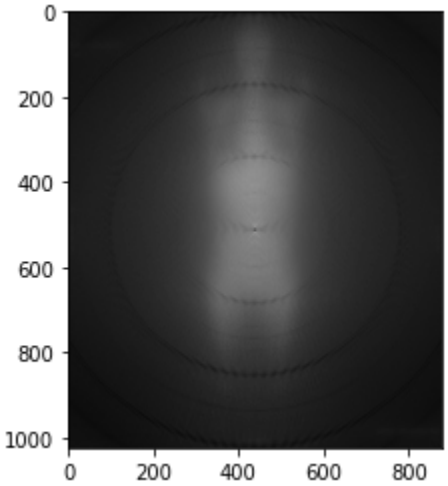
Oryginalny obraz.



Sinogram - wyniki pośrednie i ostateczny.







Odtworzony obraz - wyniki pośrednie i ostateczny.