

Informatyka w Medycynie - Laboratorium	
Tomografia komputerowa - projekt	
Kierunek/semestr: Informatyka/6	Grupa: L16
Jakub Kwiatkowski 145356 Paweł Strzelczyk 145217	

1 Opis projektu.

Projekt został przygotowany w formie interaktywnego notatnika Jupyter Notebook. Symulowany tomograf jest tomografem typu stożkowego (fan-beam CT scanner). Do wykonania symulacji wykorzystano język Python 3 oraz biblioteki

- numpy
- matplotlib
- skimage
- OpenCV
- Pydicom
- ipython (ipywidgets, IPython)

2 Opis głównych funkcji programu.

2.1 Pozyskiwanie odczytów dla poszczególnych detektorów.

Aby pozyskać odczyty trzeba najpierw ustalić pozycję emitera oraz detektorów. Mając obraz o wymiarach $m \times n$ oraz położenie katowe emitera α możemy wyliczyć pozycję emitera na okręgu opisanym na obrazie:

$$\begin{aligned}
 radius &= \sqrt{\frac{m^2}{2^2} + \frac{n^2}{2^2}} \\
 emitter_x &= \cos \alpha * radius + \frac{m}{2} \\
 emitter_y &= \sin \alpha * radius + \frac{n}{2}
 \end{aligned}$$

Znając dodatkowo rozpiętość kątową wachlarza detektorów ϕ oraz ich liczbę D możemy obliczyć pozycję każdego z nich:

$$\begin{aligned} step &= \frac{\phi}{D-1} \\ start &= \alpha + \pi - \frac{\phi}{2} \\ \forall_{i \in D} \begin{cases} detector_x^i = \cos(start + i * step) * radius + \frac{m}{2} \\ detector_y^i = \sin(start + i * step) * radius + \frac{n}{2} \end{cases} \end{aligned}$$

Translacja o $\frac{m}{2}$ i $\frac{n}{2}$ wynika z faktu, że punkt centralny obrazu nie leży w środku układu współrzędnych.

Po wyznaczeniu współrzędnych emitera i detektorów, dla każdej pary emiter – detektor wyznaczamy piksele przez które przechodzi odcinek łączący tę parę (wykorzystujemy tu funkcję `line_nd` [listing 1] z pakietu `scikit-image`) i sumujemy wartości tych pikseli, dzięki czemu otrzymujemy sinogram obrazu.

```

1  def detector_positions(alpha, phi, count, shape):
2      """
3      Calculates locations of detectors in a fan-beam CT scanner.
4
5      :param alpha: Emitter's angular position angle (in radians).
6      :param phi: Detectors' angular span (in radians) [measured from circle's center
7      ].
8      :param count: Number of detectors.
9      :param shape: Image size (x, y).
10     :return: List of detector positions (x, y).
11     """
12
13     from numpy import pi as PI, cos, sin, sqrt, round, int64
14
15     w, h = shape
16     radius = round(sqrt(w ** 2 + h ** 2) / 2).astype(int64)
17
18     angular_step = phi / (count - 1)
19     start_angle = alpha + PI - (phi / 2)
20
21     x = lambda index: round(cos(start_angle + angular_step * index) * radius).
22     astype(int64) + (w // 2)
23     y = lambda index: round(sin(start_angle + angular_step * index) * radius).
24     astype(int64) + (h // 2)
25
26     return [(x(no), y(no)) for no in range(count)]
27
28 def emitter_position(alpha, shape) -> list[tuple[int, int]]:
29     """
30     Calculates emitter's location in a fan-beam CT scanner.
31
32     :param alpha: Emitter's angular position (in radians).
33     :param shape: Image size (x, y).
34     :return: Emitter's position (x, y).
35     """
36
37     from numpy import cos, sin, sqrt, round, int64
38
39     w, h = shape
40     radius = round(sqrt(w ** 2 + h ** 2) / 2).astype(int64)
41
42     x = round(cos(alpha) * radius).astype(int64) + (w // 2)

```

```

41     y = round(sin(alpha) * radius).astype(int64) + (h // 2)
42
43     return (x, y)
44
45 def scanlines(bounds, emitter, detectors):
46     """
47     Calculates scanlines (pixels in ray path) for a CT scanner.
48
49     :param bounds: Image's bounds (x, y).
50     :param emitter: Emitter's position (x, y).
51     :param detectors: List of detector positions (x, y).
52
53     :return: List of scanlines (list of lists of pixels in ray path (one for every
54             detector)).
55
56     This function automatically removes pixels not in the image's bounds.
57     """
58     from skimage.draw import line_nd
59
60     w, h = bounds
61
62     in_bounds = lambda point: point[0] >= 0 and point[0] < w and point[1] >= 0 and
63         point[1] < h
64
65     ex, ey = emitter
66
67     # line_nd uses transposed coordinates (y, x), so we need to swap them.
68     lines = [list(filter(in_bounds, zip(*line_nd((d[1], d[0]), (ey, ex))))) for d in
69         detectors]
70
71     return [[x for x in zip(*line)] for line in lines]
72
73 from numpy import ndarray
74
75 def radon_for(image, alpha, phi, count):
76     """
77     Calculates Radon transform of an image for given alpha angle.
78
79     :param alpha: Emitter's angular position (in radians)
80     :param phi: Detectors' angular span [measured from circle's center] (in radians).
81     :param count: Number of detectors.
82     """
83
84     from numpy import sum
85
86     w, h = image.shape[:2]
87
88     emitters = emitter_position(alpha, (w, h))
89     detectors = detector_positions(alpha, phi, count, (w, h))
90     lines = scanlines((w, h), emitters, detectors)
91
92     row = []
93     for line in lines:
94         if len(line) == 0:
95             row.append(0)
96             continue
97         r, c = line
98         row.append(sum(image[r, c]))
99
100    return row

```

```

101
102
103 def radon(image, phi, step, count) -> ndarray:
104     """
105     Calculates Radon transform of an image for given alpha angle.
106
107     :param phi: Detectors' angular span [measured from circle's center] (in degrees).
108     :param step: Step size (in degrees).
109     :param count: Number of detectors.
110     """
111
112     from numpy import deg2rad, arange, array, pi
113
114     phi = deg2rad(phi)
115     step = deg2rad(step)
116
117     sinogram = []
118
119     for angle in arange(0, pi * 2, step):
120         sinogram.append([angle] + radon_for(image, angle, phi, count))
121
122     return array(sinogram)

```

Listing 1: Wylizanie sinogramu

2.2 Filtrowanie sinogramu.

Wykorzystano i porównano dwa sposoby filtrowania.

Pierwsze filtrowanie jest prostym splotem nazwanym filtrem medianowym.

Aby obraz nie został zbyt rozmyty zastosowaliśmy jądro o wymiarach 3×3 piksele.

Drugi filtr to splot całkowy z wykorzystaniem maski opisanej wzorem:

$$h[k] = \begin{cases} 1 & k = 0 \\ 0 & k \bmod 2 = 0, k \neq 0 \\ \frac{-4}{\pi^2 k^2} & k \bmod 2 = 1, k \neq 0 \end{cases}$$

2.3 Ustalanie jasności poszczególnych punktów obrazu wynikowego oraz jego przetwarzanie końcowe.

Po wykonaniu odwrotnej transformacji Radona za pomocą techniki *back-projection* otrzymany obraz zawiera piksele o jasnościach wykraczających poza ramy kanału 8-bitowego, dlatego obraz musi zostać znormalizowany.

Normalizacja transformuje obraz z nieograniczonej prawostronnie reprezentacji całkowitoliczbowej do reprezentacji zmiennoprzecinkowej w zakresie $[0.0; 1.0]$. Normalizacja przebiega według schematu:

$$\forall_{p \in P(x,y)} p' = \frac{p - \min(P)}{\max(P) - \min(P)}$$

2.4 Wyznaczanie wartości miary RMSE.

Miara RMSE jest obliczana poprzez wyliczenie odchylenia standardowego różnicy obrazów – wejściowego i wyjściowego.

2.5 Odczyt i zapis plików DICOM.

Do obsługi standardu DICOM wykorzystano bibliotekę Pydicom.

Dla uproszczenia odczytu i zapisu stworzone zostały funkcje pomocnicze przedstawione na listingu 2.

```
1 def read_from_dicom(filename) -> ndarray:
2     """
3     Reads pixel data from DICOM file and returns it as ndarray.
4
5     WARNING: Currently this function assumes that image is single and in grayscale.
6     !!! COLOR & MULTIPLANAR IMAGES ARE NOT SUPPORTED !!!
7     """
8
9     from pydicom import dcmread
10
11     contents = dcmread(filename)
12
13     return contents.pixel_array
14
15
16 def write_to_dicom(filename, image: ndarray, data: dict):
17     """
18     Writes data to DICOM file.
19
20     :param image: image data in form of ndarray.
21     WARNING: Currently this function assumes that image is single and in grayscale.
22     !!! COLOR & MULTIPLANAR IMAGES ARE NOT SUPPORTED !!!
23
24     :param data: Dictionary with keys "id", "name", "date", "comments"
25     """
26     from skimage.util import img_as_ubyte
27     from pydicom.uid import generate_uid, ExplicitVRLittleEndian
28     from pydicom._storage_sopclass_uids import CTImageStorage
29     from pydicom.dataset import validate_file_meta
30     from pydicom import Dataset, FileDataset
31
32     image = img_as_ubyte(image)
33
34     # Populate required values for file meta information
35     meta = Dataset()
36     meta.MediaStorageSOPClassUID = CTImageStorage
37     meta.MediaStorageSOPInstanceUID = generate_uid()
38     meta.TransferSyntaxUID = ExplicitVRLittleEndian
39
40     ds = FileDataset(None, {}, preamble = b"\0" * 128)
41     ds.file_meta = meta
42
43     ds.is_little_endian = True
44     ds.is_implicit_VR = False
45
46     ds.SOPClassUID = CTImageStorage
47     ds.SOPInstanceUID = meta.MediaStorageSOPInstanceUID
48
49     ds.PatientID = data["id"]
50     ds.PatientName = data["name"]
51     ds.StudyDate = data["date"]
52     ds.ImageComments = data["comments"]
53
54     ds.Modality = "CT"
55     ds.SeriesInstanceUID = generate_uid()
56     ds.StudyInstanceUID = generate_uid()
```

```

57 ds.FrameOfReferenceUID = generate_uid()
58
59 ds.BitsStored = 8
60 ds.BitsAllocated = 8
61 ds.SamplesPerPixel = 1
62 ds.HighBit = 7
63
64 ds.ImagesInAcquisition = 1
65 ds.InstanceNumber = 1
66
67 ds.Rows, ds.Columns = image.shape
68
69 ds.ImageType = r"ORIGINAL\PRIMARY\AXIAL"
70
71 ds.PhotometricInterpretation = "MONOCHROME2"
72 ds.PixelRepresentation = 0
73
74 validate_file_meta(ds.file_meta, enforce_standard = True)
75
76 ds.PixelData = image.tobytes()
77
78 ds.save_as(filename, write_like_original = False)

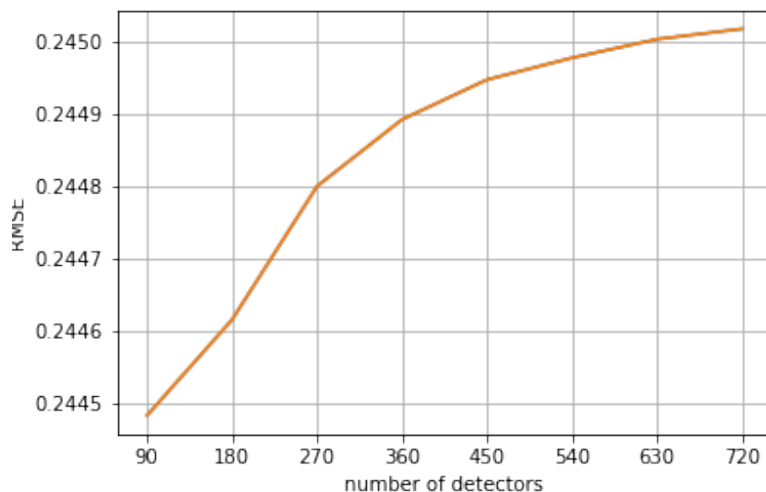
```

Listing 2: Obsługa DICOM

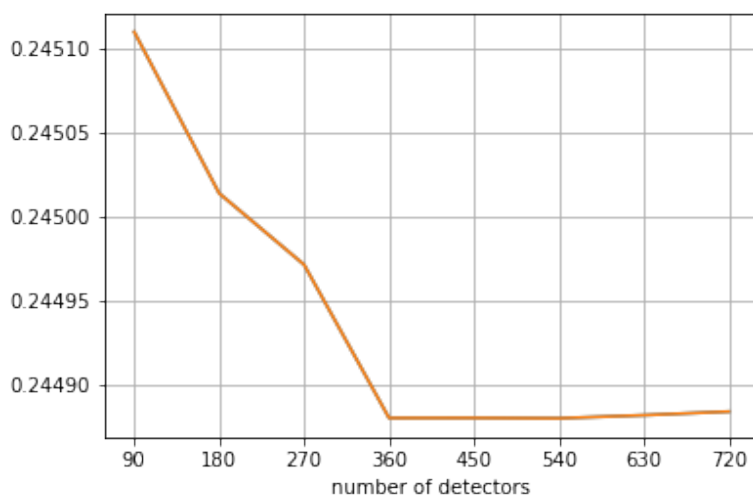
3 Wpływ poszczególnych parametrów na jakość obrazu wynikowego.

3.1 Liczba detektorów.

Liczba detektorów jest zmieniana w przedziale $\langle 90, 720 \rangle$ z krokiem 90.



Z wykorzystaniem filtracji splotowej.

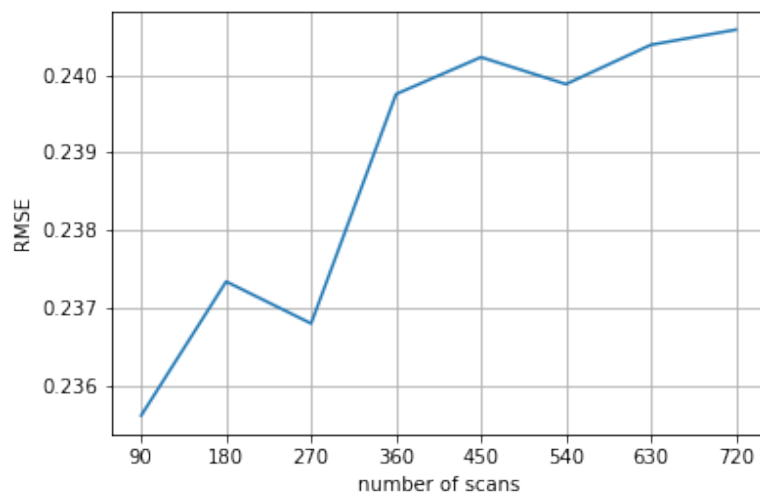


Z wykorzystaniem filtracji medianowej.

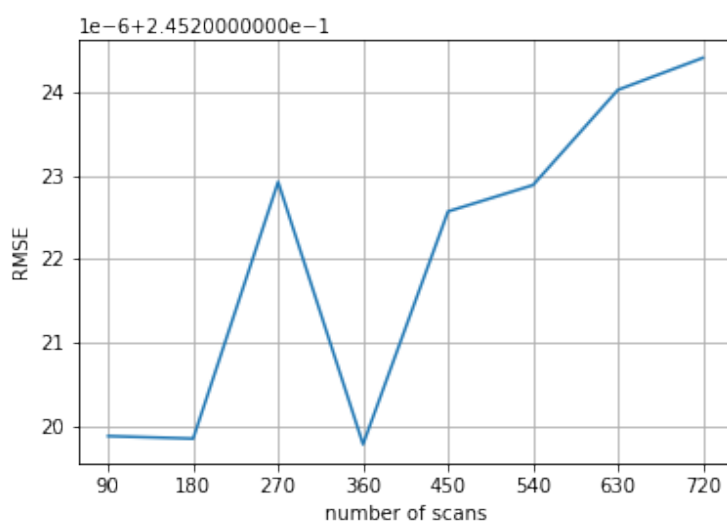
Podczas rekonstrukcji obrazu z wykorzystaniem filtrowania splotowego, liczba detektorów jest pewnego rodzaju współczynnikiem skalowania jasności. Im jest ich więcej, tym silniejsze będzie odwzorowanie na całym obszarze skanu, a co za tym idzie różnica statystyczna będzie większa, ale perceptualnie zrekonstruowany obraz będzie posiadał bardziej wyraźne szczegóły oraz lepiej odwzorowane kolory i kształty obiektów. W przypadku filtracji medianowej nie dochodzi zwiększania kontrastu, lecz do rozmycia, stąd większa liczba detektorów może przekładać się na dokładność odtworzenia ogólnego zarysu i układu elementów na obrazie, a co za tym idzie, do zmniejszenia błędu.

3.2 Liczba skanów.

Liczba skanów jest zmieniana w przedziale $\langle 90, 720 \rangle$ z krokiem 90.



Z wykorzystaniem filtracji splotowej.

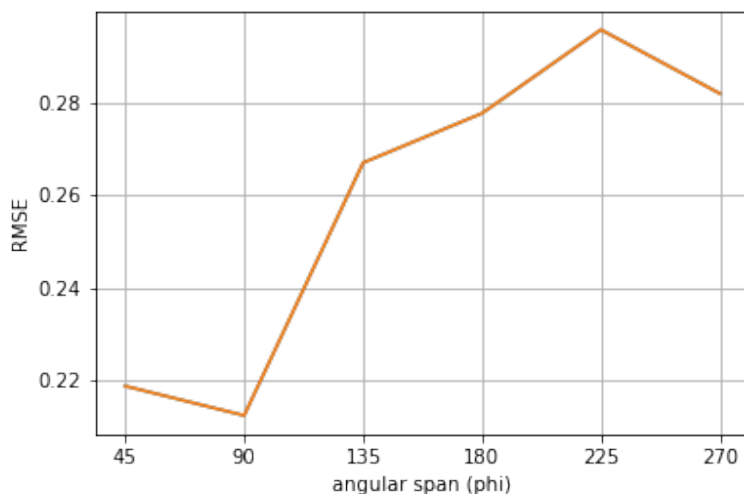


Z wykorzystaniem filtracji medianowej.

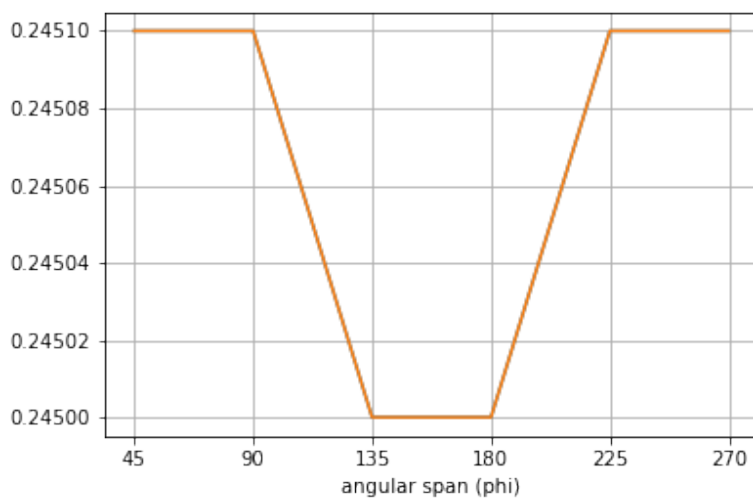
Najmniejsze wartości błędu otrzymano przy liczbie skanów równej 90, 180 oraz 360 (bez użycia filtru splotowego). Podczas generowania obrazów oceniono, że najlepsze rezultaty są osiągnięte dla wartości 180 i 360. RMSE dla rekonstrukcji z wykorzystaniem filtru splotowego jest wyższy po 360 skanach niż po 180, lecz nie zauważono różnicy w dokładności odwzorowania kształtów. W przypadku obydwu filtrów wartość błędu rośnie powyżej 360 skanów. Warto także zauważyć, że przy wykorzystaniu filtru medianowego, zmiana błędu w zależności od liczby skanów jest bardzo mała.

3.3 Rozpiętość wachlarza.

Rozpiętość wachlarza jest zmieniana w przedziale $\langle 45, 270 \rangle$ z krokiem 45.



Z wykorzystaniem filtracji splotowej.



Z wykorzystaniem filtracji medianowej.

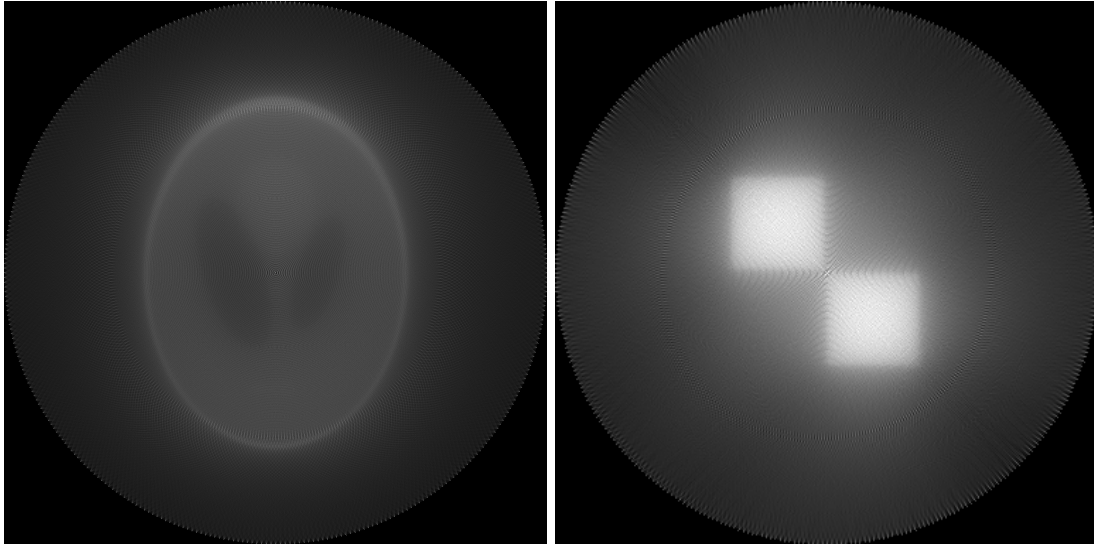
Przy niezmienniej liczbie detektorów większy kąt rozwarcia powoduje utworzenie większych przerw pomiędzy skanami oraz objęcie większego fragmentu przestrzeni, która nie musi w całości być zajęta przez badany obiekt. W przypadku filtru splotowego najmniejszy błąd występuje przy małych kątach rozwarcia - 45° , 90° , ponieważ w tym zakresie z reguły zawiera się znaczna część skanowanego obiektu i niewielka część otoczenia. Dla większych kątów sinogram zawiera zbyt dużo informacji o otoczeniu, które podczas splotu będą generować szum (szare piksele) i powodować wzrost RMSE.

3.4 Filtr splotowy.



Obraz testowy do obliczeń z wykorzystaniem filtra.

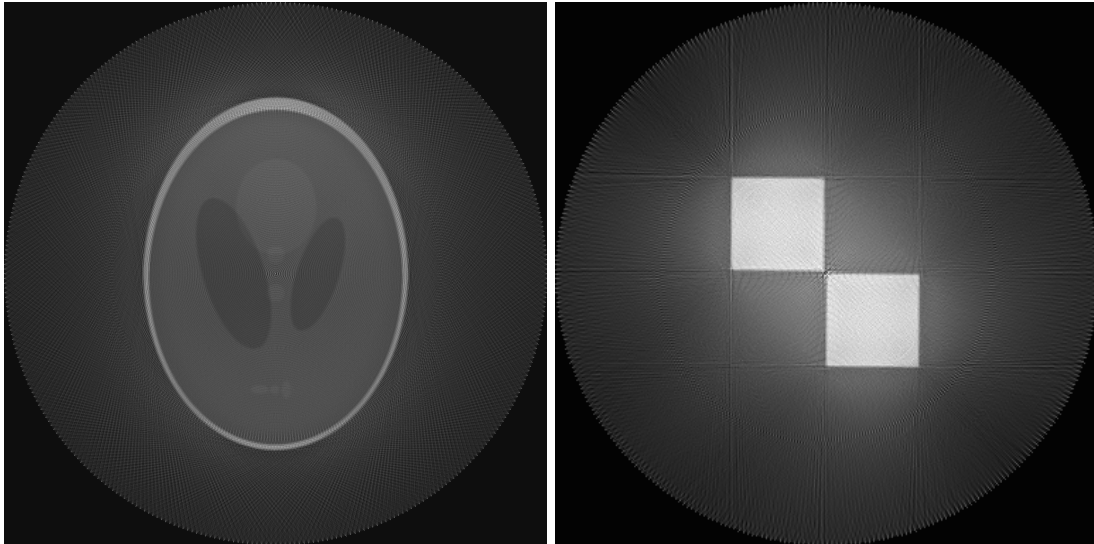
Rekonstrukcja bez filtrowania.



$$RMSE_1 = 0,20693420547948$$

$$RMSE_2 = 0,266156126913852$$

Rekonstrukcja z filtrem splotowym.



$$RMSE_1 = 0,249729691096470$$

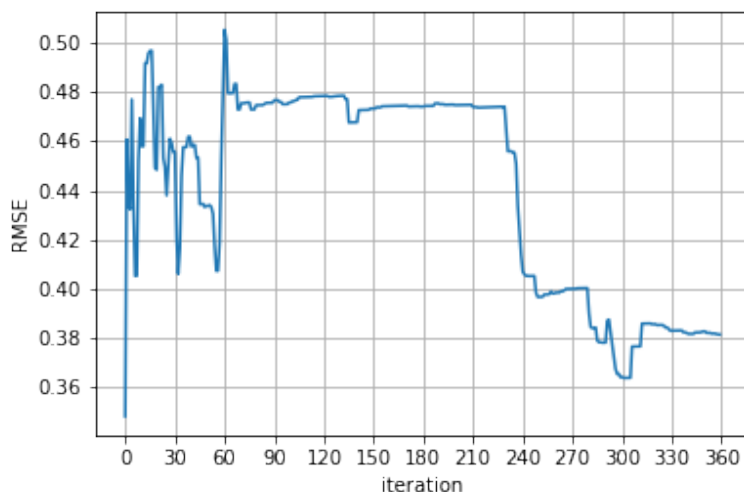
$$RMSE_2 = 0,26778570956508$$

Na powyższym przykładzie można zauważyć, że większa wartość błędu RMSE nie oznacza mniejszej czytelności wynikowego obrazu. Pierwszy obraz ma widocznie ostrzejsze kontury, widoczne są małe elipsy w dolnej części obiektu. Drugi obraz również charakteryzuje się ostrzejszymi konturami, a także widocznym przyciemnieniem niektórych obszarów w obrębie skanu - co miało wpływ na zmniejszenie różnicy błędu.

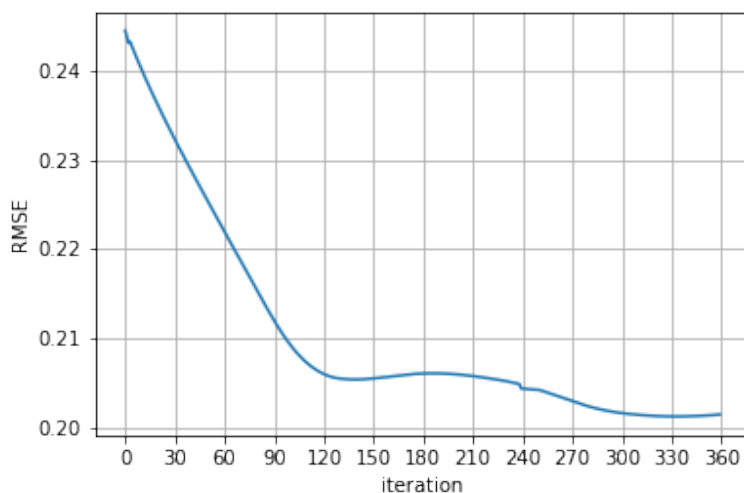
4 Zmiana RMSE podczas wykonywania kolejnych iteracji odwrotnej transformaty Radona.

Parametry transformacji:

- liczba detektorów - 180
- rozpiętość wachlarza - 180°
- łączna liczba skanów - 360



Z wykorzystaniem filtracji splotowej.



Z wykorzystaniem filtracji medianowej.

Podczas pierwszych 60 iteracji błąd dynamicznie rośnie, po czym oscyluje w okolicach 0.45 - odtwarzany jest wtedy jedynie fragment krawędzi obiektu, całe otoczenie znajdujące się po drugiej stronie jest niewiadomą, co powoduje generowanie silnego szumu przez filtr splotowy. W trakcie kolejnych 180 iteracji tworzy się zarys badanego obiektu, natomiast cały obraz pozostaje silnie rozjaśniony i zaszumiony. W okolicach 230. - 240. iteracji kończy się odtwarzanie głównego obramowania obiektu i cały obraz zostaje przyciemniony. Pod koniec rekonstrukcji utrwalane są pozostałe kontury oraz uzupełniane są informacje o całym otoczeniu.