Symulator tomografu komputerowego

Maciej A. Czyzewski
inf136698

Poznan University of Technology
Poland

1 Zastosowany model tomografu

Rownolegly (choc w projekcie zaimplementowany jest rowniez stozkowy).

2 Zastosowany język programowania oraz dodatkowe biblioteki

Python. Lista bibliotek (niektore sluza tylko do debugowania):

- watchdog==0.9.0
- numpy==1.17.2
- scipy==1.3.1
- numba==0.46.0
- pydicom==1.4.2
- ipython==7.13.0
- tqdm==4.36.1
- Unidecode==1.1.1
- Pillow==7.0.0
- scikit_image==0.15.0
- matplotlib==3.1.1
- skimage==0.0

3 Opis głównych funkcji programu

3.1 pozyskiwanie odczytów dla poszczególnych detektorów

Sinogram jest generowany funkcja `fn_tomograph` (ponizej fragment).

```
# dla kolejnych detektorow
for ray in range(0, rays):
    # funkcja `d2r` zamienia stopnie na radiany
    shift = -(l / 2) + ray * l / (rays - 1)

    # tutaj mamy wsplorzedna emitera
    if not cone:  # jaki model ma byc uzyty?
        x0 = size * np.cos(d2r(i - shift))
        y0 = size * np.sin(d2r(i - shift))
    else:
        x0 = size * np.cos(d2r(i))
        y0 = size * np.sin(d2r(i))
```

```
13
14      # po przeciwniej stronie chcemy detektor
15      x1 = (size) * np.cos(d2r(i + 180 + shift))
16      y1 = (size) * np.sin(d2r(i + 180 + shift))
17
18      # tricki:
19      #  - [diff_x0/diff_y0] dla przypadku
20      #    gdy wejscie to prostokat
21      #  - [size // 2] bo chcemy wysrodkowany
22      x0 = diff_x0 + int(x0) + size // 2
23      x1 = int(x1) + size // 2
24      y0 = diff_y0 + int(y0) + size // 2
25      y1 = int(y1) + size // 2
26
27      # generujemy punkty na linii
28      # oraz obcinamy te po za obrazkiem
29      line = fn_line(x0, y0, x1, y1, X=Sw, Y=Sh)
30
31      S = 0  # model addetywny
32      for p in line:
33          S += img[int(p[1]), int(p[0])]
34
35      linogram.append(S)  # --> czyli wiersz sinogramu
36      ray_lines.append([x0, y0, x1, y1])
```

3.2 filtrowanie sinogramu, zastosowany rozmiar maski

Filtorwanie jak i przeksztalcenie sinogramu odbywa sie w `fn_fbp` (ponizej fragment).

```
1 # znaleziony filtr:
2 # https://www.youtube.com/watch?v=pZ7JlXagT0w
3 f = fftfreq(sinogram.shape[0]).reshape(-1, 1)
4 # f - digital frequency
5 omega = 2 * np.pi * f  # angular frequency
6 fourier_filter = 2 * np.abs(f)  # ramp filter
7
8 projection = fft(sinogram, axis=0) * fourier_filter
9 sinogram = np.real(ifft(projection, axis=0))
```

Dlaczego akurat tak zrobilem, ilustruje w `showcase.ipynb`.

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

Ostatni postprocessing odbywa sie w `fn_clip`.

```python
def fn_clip(img, val=1.0, agressive=False):
    img = img.astype(float) * 255
    img[img < 0] = 0  # obcinamy ujemne (artefakty z ifft)
    img *= 1 / img.max() # chcemy [0, 1]

    if agressive:  # zbedny krok
        img = _fast_fn_noise_reduction(img)
        # obcinanie odchylen (najasniejszy, najciemniejszych)
        if CONFIG["rays"] >= 1000:
            p2, p98 = np.percentile(img, (0.5, 99.5))
        else:
            img = exposure.adjust_log(img, 1)
            p2, p98 = np.percentile(img, (2, 98))
        img = exposure.rescale_intensity(img, \
                                         in_range=(p2, p98))

    return img * 255
```

3.4 wyznaczanie wartości miary RMSE na podstawie obrazu źródłowego oraz wynikowego

Wyznaczanie miary RMSE odbywa sie w `fn_calc_rmse`.

```python
def fn_calc_rmse(A, B):
    A, B = fn_clip(A), fn_clip(B)
    errors = np.asarray(A - B) / 255
    return np.sqrt(np.mean(np.square(errors)))
```

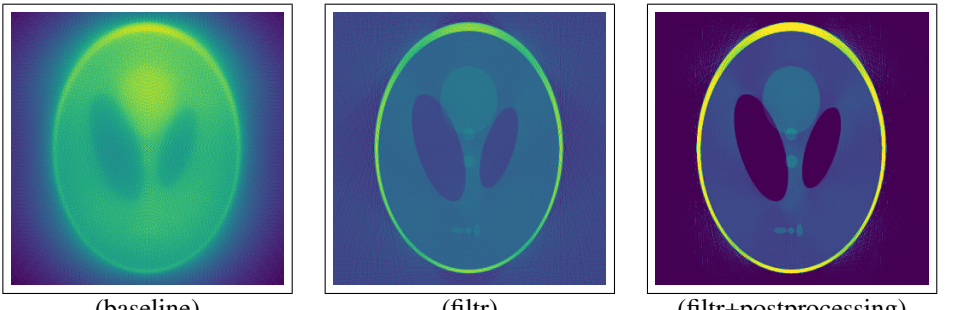
3.5 odczyt i zapis plików DICOM

Najlepiej sprawdzic dzialanie w pliku `showcase.ipynb`.

```python
def do_dicom(PatientName=None, ImageComments=None, StudyDate=Non
    # edycja danych w pliku DICOM
    fn_save_dicom(
            "data/test.dcm",
            data={
                "PatientName": PatientName,
                "ImageComments": ImageComments,
                "StudyDate": StudyDate,
            },
        )

    # tutaj laduje i wyswietla dane z pliku (prefix "===")
```

```
    img = fn_load_dicom("data/test.dcm")

    # obslugujemy te pliki
    # mozna z nich robic Radon-a
    plt.figure()
    plt.imshow(img)

# interaktywne okienko / zgodnie z wymaganiami
interact_manual(do_dicom, \
        PatientName="Maciej A. Czyzewski", \
        ImageComments="Obrazek testowy, prawda?", \
        StudyDate="20200119")
```

4 Filtrowanie



(baseline) (filtr) (filtr+postprocessing)

Figure 1: Wplyw filtrowania na jakosc wizualna wyniku.

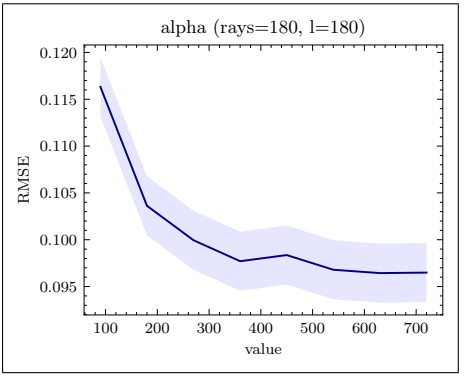
Wyraznie widac ze filtr jak i odpowieni postprocessing jest waznym elementem procesu aby osiagnac wizualnie poprawnie obraz. Wiecej na temat zjawiska ktore jest powodem "rozmazania" w `showcase.ipynb`.

Method	RMSE
baseline	0.5087
filtr	0.1142
filtr+postprocessing	**0.0916**

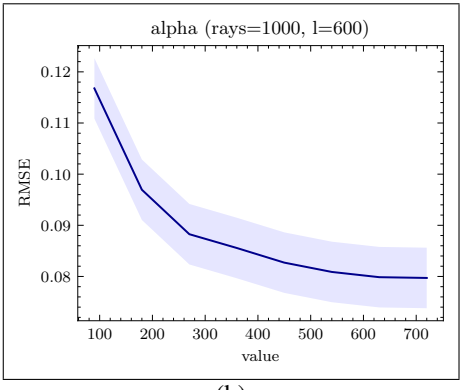
Table 1: Wplyw filtrowania na RMSE.

5 Wynik eksperymentu

Tak. Wnioski wynikające z przebiegow są zgodne z oceną subiektywną jakości obrazu.

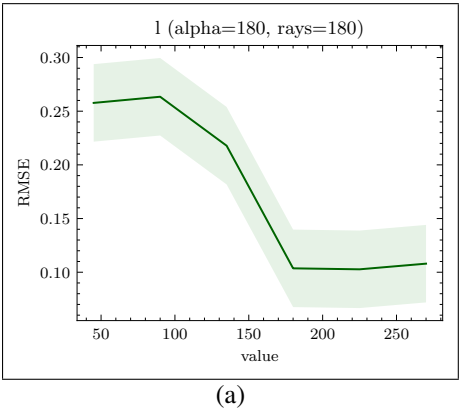


(a)

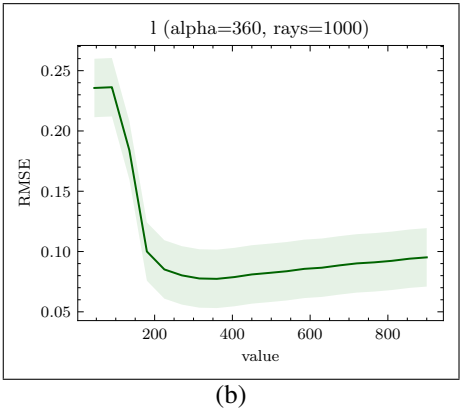


(b)

Figure 2: Wpływ kroku $\Delta\alpha$ układu emiter/detektor na RMSE.

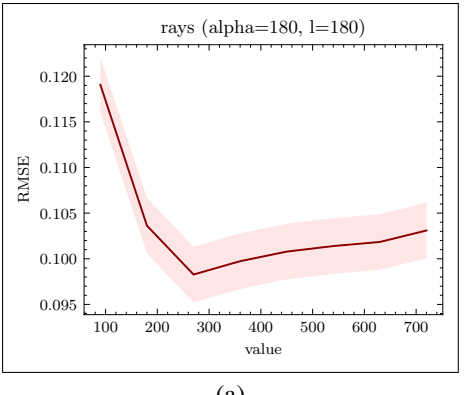


(a)

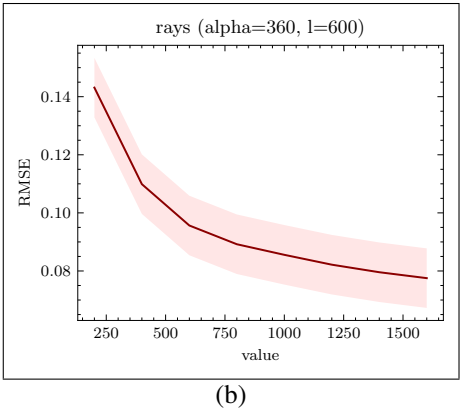


(b)

Figure 3: Wpływ rozwartości/rozpiętości układu emiter/detektor na RMSE.



(a)



(b)

Figure 4: Wpływ liczby detektorów układu emiter/detektor na RMSE.

Przebiegi dla eksperymentu z wartosciami poczatkowymi ustawionymi na 180 na kazdym parametrze byly niesatyskacjunujace (a); dlatego zostaly dla porownania wygenerowane dla duzo wiekszy wartosci/rozdzielczosci (b).

Zgodnie z oczekiwaniami zwiekszenie liczby detektorow zwieksza jakosc obrazu (wymiar Y sinogramu sie zwieksza) wiec mamy zakodowane wieksza ilosc informacji. Podobnie jest z parametrem α ktory natomiast powoduje ze mamy coraz wieksza gestosc pomiarow w wymiarze X (czyli na polowe pelnego obrotu). Parametr rozpietosci; l jest troche trudniejszy do interpretacji, nie powinnien on monotonicznie wplywac na RMSE, gdy zsumujemy wartosci promienii dla pustego obrazka zobaczymy “pregi”. Najlepiej wybrac takie l ktore dla danych pozostalych parametrow stworzy jak najmniej takich nieporzadanych kolek. Dodatkowo zbyt male l nie bedzie zbierac informacji z calego obrazka, zbyt duze zacznie oslabiac nam znaczenie detektorow zewnetrznych.