### **Autorzy**

Małgorzata Wachulec, Aleksandra Wichrowska

### **Temat**

*Zobacz kotku, co mam w środku, czyli jak działa tomografia komputerowa.*

### **Szczegółowy opis zagadnienia**

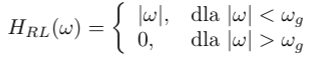
Tomografia komputerowa przebiega w następujący sposób: przez badany obiekt przepuszczane jest promieniowanie rentgenowskie (pod pewnym ustalonym kątem), następnie znajdujące się po przeciwległej stronie detektory przesyłają do komputera informację o pochłoniętym przez tkanki promieniowaniu. Po powtórzeniu tej procedury dla wielu różnych kątów możliwe jest odtworzenie obrazu badanego obiektu poprzez zsumowanie wielu jednowymiarowych "przekrojów" - do tego służy algorytm filtrowanej projekcji wstecznej, który będziemy implementować w ramach tego projektu.

Podstawowa wersja algorytmu stosowanego do zrekonstruowania obrazu na podstawie projekcji (metoda projekcji wstecznej) nie zakłada stosowania funkcji filtrujących. Uzyskane w ten sposób rezultaty są niewyraźne i rozmazane. Rozszerzeniem i ulepszeniem tej metody jest omawiany w pracy algorytm filtrowanej projekcji wstecznej.

Algorytm filtrowanej projekcji wstecznej:

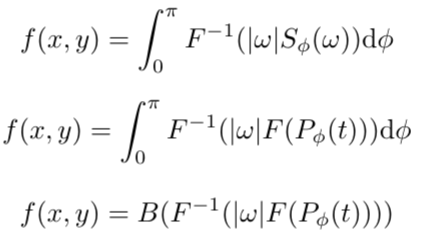
1. Projekcja obrazu
2. Transformacja Fouriera
3. Przemnożenie wyniku przez funkcję filtrującą

Dobór odpowiedniego filtra pozwala usunąć rozmycie obecne na projekcjach. Najczęściej stosowanym filtrem jest filtr Ram-Lak, który zakłada ograniczenie pasma do pulsacji granicznej. Możemy go przedstawić wzorem:

gdzie to pulsacja graniczna zależna od .

Inne używane filtry to między innymi filtr Sheppa-Logana, filtr Hamminga oraz niskopasmowy filtr cosinusowy. [[2]](#_pufz6l1n99ks)

1. Odwrotna transformacja Fouriera
2. Projekcja wsteczna

Powyższy algorytm możemy opisać poprzez następujące równanie:

gdzie:

- dwuwymiarowa funkcja pochłaniania promieniowania, którą chcemy odtworzyć

- transformata Fouriera

- moduł pulsacji

- projekcja

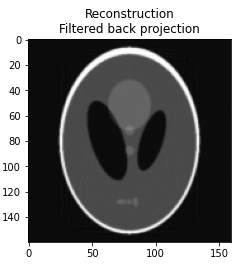
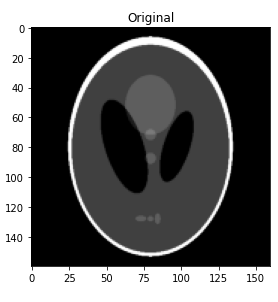
- projekcja wsteczna

Aby móc zaimplementować i przetestować algorytm filtrowanej projekcji wstecznej, musimy mieć projekcje obrazów (na przykład fantomu Sheppa-Logana), na których będziemy testować nasz algorytm. Uzyskamy je za pomocą funkcji *radon* z pakietu *skimage.transform* [[1]](#_2ldk7h42qat0)dostępnego w języku Python, w którym zrealizujemy nasz projekt. Funkcja ta przyjmuje obraz oraz wektor kątów, pod którymi obraz będzie "prześwietlony". Możliwość regulowania liczby i kątów nachylenia projekcji będzie przydatna podczas testów naszej implementacji algorytmu filtrowanej projekcji wstecznej.

Sama implementacja algorytmu będzie przyjmować sinogram (czyli zbiór wektorów projekcji dla różnych kątów prześwietlenia) i będzie zwracać zrekonstruowany obraz: macierz o wartościach numerycznych oznaczających stopień zaciemnienia piksela, gdzie położenie danego piksela jest określone przez jego indeks w macierzy.

### **Oczekiwane rezultaty**

W Pythonie jest wiele bibliotek, które implementują algorytm filtrowanej projekcji wstecznej. Pozwoli to na sprawdzenie, czego powinniśmy oczekiwać od naszej implementacji. We wspomnianej wcześniej bibliotece *skimage.transform* [[1]](#_2ldk7h42qat0) jest dostępna funkcja *iradon*, która zwraca zrekonstruowany obraz "prześwietlonego" obiektu. Wizualizacja oryginalnego obrazu - fantomu Sheppa-Logana - oraz jego zrekonstruowany obraz są przedstawione na [Rys.1](#_l9lxuiagjkd6). Jak widać, można rozpoznać zwizualizowany obraz gołym okiem, lecz jest on trochę rozmazany. Rozmazanie to można mierzyć za pomocą pierwiastkowanego błędu średniokwadratowego, który w wypadku rekonstrukcji przedstawionej na [Rys.1](#_l9lxuiagjkd6) jest równy 0.0283. Tego typu rezultatów oczekujemy od naszej implementacji - możliwości rozpoznania obrazu gołym okiem oraz względnie małego błędu średniokwadratowego.



##### Rysunek 1. Fantom Sheppa-Logana oraz jego rekonstrukcja uzyskana za pomocą funkcji iradon z pakietu skimage.transform w Pythonie

Będziemy także badać zachowanie algorytmu w zależności od liczby projekcji - spodziewamy się, że im więcej projekcji tym dokładniejsza będzie otrzymana rekonstrukcja badanego obiektu.

### **Projekt testów numerycznych**

Po zaimplementowaniu algorytmu zamierzamy wypróbować go na fantomie Sheppa-Logana i porównać uzyskane rezultaty z wynikiem funkcji *iradon* z pakietu *skimage.transform*. Będziemy porównywać obie metody dla różnej liczby projekcji, uzyskanych pod różnymi kątami i mierzyć ich wyniki za pomocą pierwiastkowanego błędu średniokwadratowego. (Liczba takich prób będzie zależeć od szybkości uzyskanego rozwiązania.) Pozwoli to na ocenę jakości naszego rozwiązania w zależności od liczby projekcji (spodziewamy się, że im mniej projekcji tym gorszy wynik uzyskamy, chodzi raczej o sprawdzenie czy nasz algorytm będzie sobie radził równie dobrze jak rozwiązanie już zaimplementowane w Pythonie niezależnie od liczby projekcji). Dodatkowo będziemy przedstawiać uzyskane wyniki na wykresie, pozwalając na wizualną ocenę jakości naszego algorytmu, poprzez porównanie ich do oryginalnego obrazu.

Na koniec chcemy przetestować nasz algorytm na przykładowym zdjęciu tomograficznym, żeby zobaczyć, czy mógłby być on wykorzystany w praktyce - tu także wyniki będą podlegać wizualnej ocenie i zostanie wyznaczony pierwiastkowany błąd średniokwadratowy. Pozwoli to ocenić czy projekcje fantomu są prostsze czy trudniejsze w rekonstrukcji od rzeczywistych prześwietleń uzyskanych w trakcie tomografii.

### **Literatura**

#### Pakiet scikit-image

#### <https://scikit-image.org/docs/dev/auto_examples/transform/plot_radon_transform.html>

#### Krystian Przybyła, *Badanie algorytmów rekonstrukcji obrazu tomograficznego*

*Potwierdzamy samodzielność powyższej pracy oraz niekorzystanie przez nas z niedozwolonych źródeł.*

Małgorzata Wachulec

Aleksandra Wichrowska