|  |  |  |  |
| --- | --- | --- | --- |
| **Laboratorium z przedmiotu Informatyka w Medycynie (IWM)** | | | |
|  | | | |
| Projekt nr 1 | | | |
|  | | | |
| ***Temat: Tomograf*** | | | |
|  | | | |
| *Prowadzący:*  *mgr inż. Iwo Błądek* | *Autor*  Tobiasz Gruszczyński 145333  Sebastian Grabowski 145248 | *Grupa dziekańska:* | *L15* |
| ***Ocena:*** |  |
|  | | | |

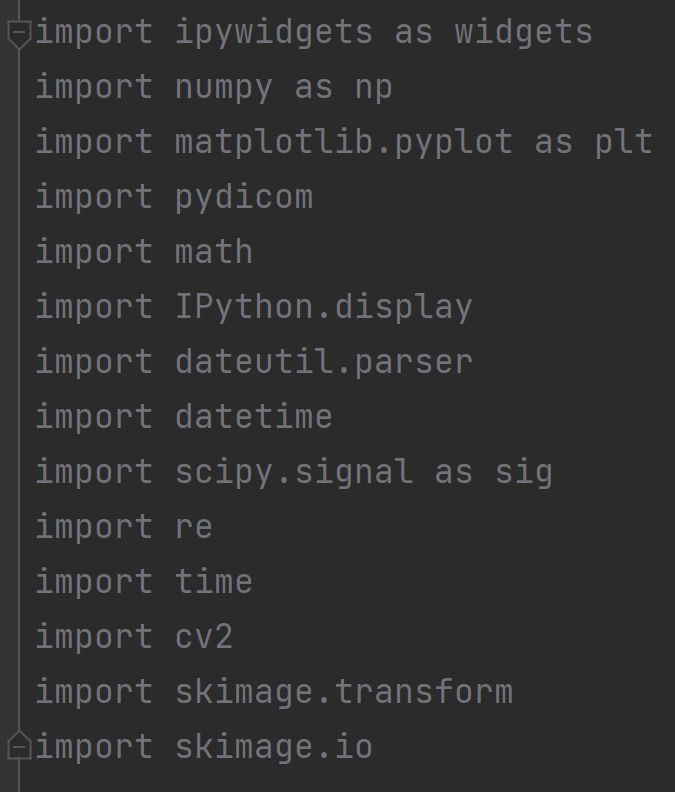
**Zastosowany model tomografu**

Przy wykonywaniu projektu wykorzystaliśmy model stożkowy.

**Zastosowany język programowania**

Zaimplementowaliśmy działanie tomografu w języku Python oraz wykorzystaliśmy Jupyther Notebook

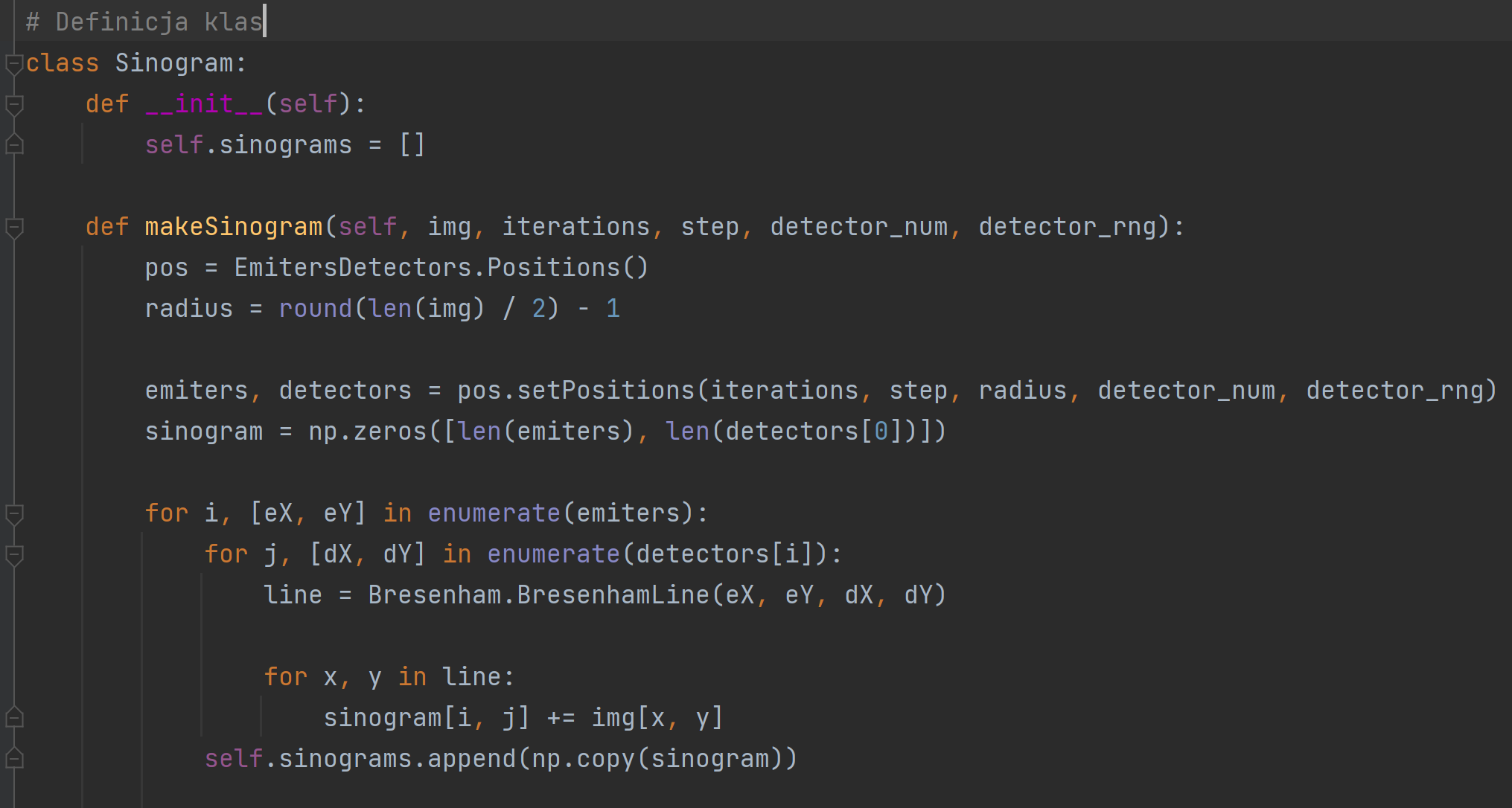
***Wykorzystane biblioteki:***



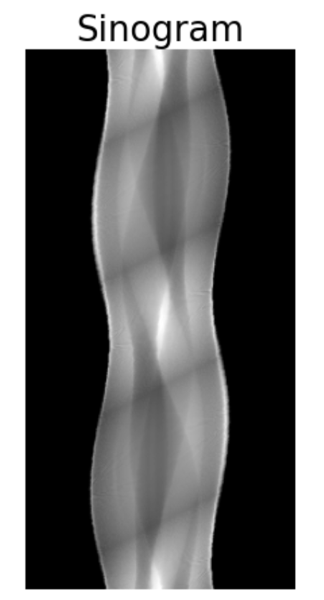
**Rys.1 Biblioteki**

**Opis głównych funkcji programu**

***Pozyskiwanie odczytów dla poszczególnych detektorów:***

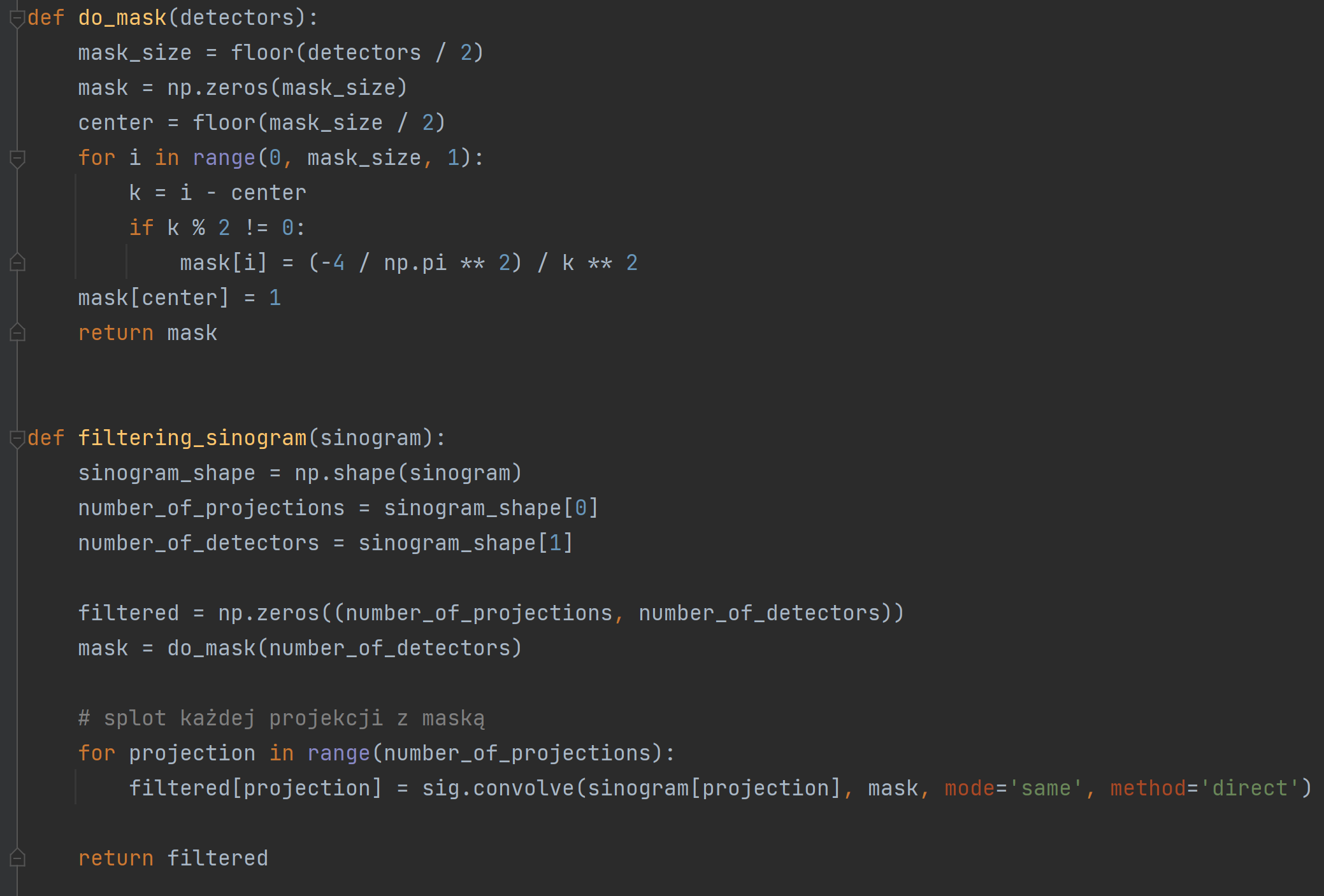
 **Rys.2 Tworzenie sinogramu**

Powyższa funkcja odpowiada za rozmieszczenie emiterów i detektorów. Następnie podczas kolejnych iteracji pętli iterujemy po wszystkich emiterach i dla każdego emitera po wszystkich detektorach. Kolejnym korkiem jest wykorzystanie algorytmu Bresehmana, z którego otrzymujemy linie sinogramu, która dodajemy do tablicy wyjściowej. Analogicznie odbywa się przetwarzanie sinogramu na obraz wyjściowy.

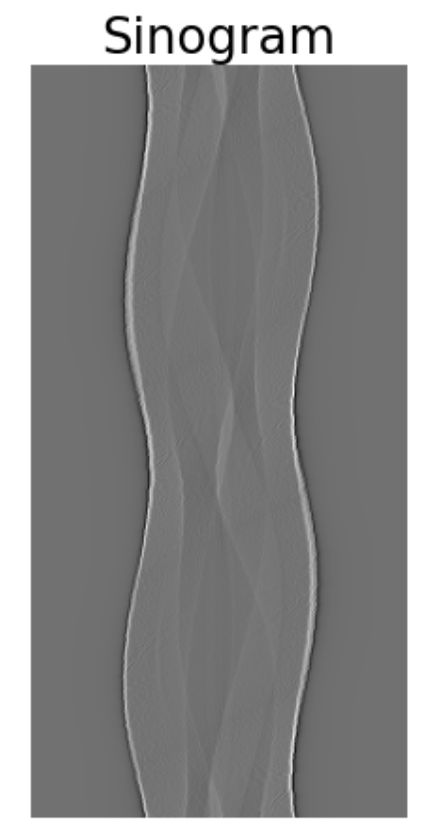
**Rys.3 Przykładowy wynik działania funkcji**

***Filtrowanie sinogramu, zastosowany rozmiar maski:***

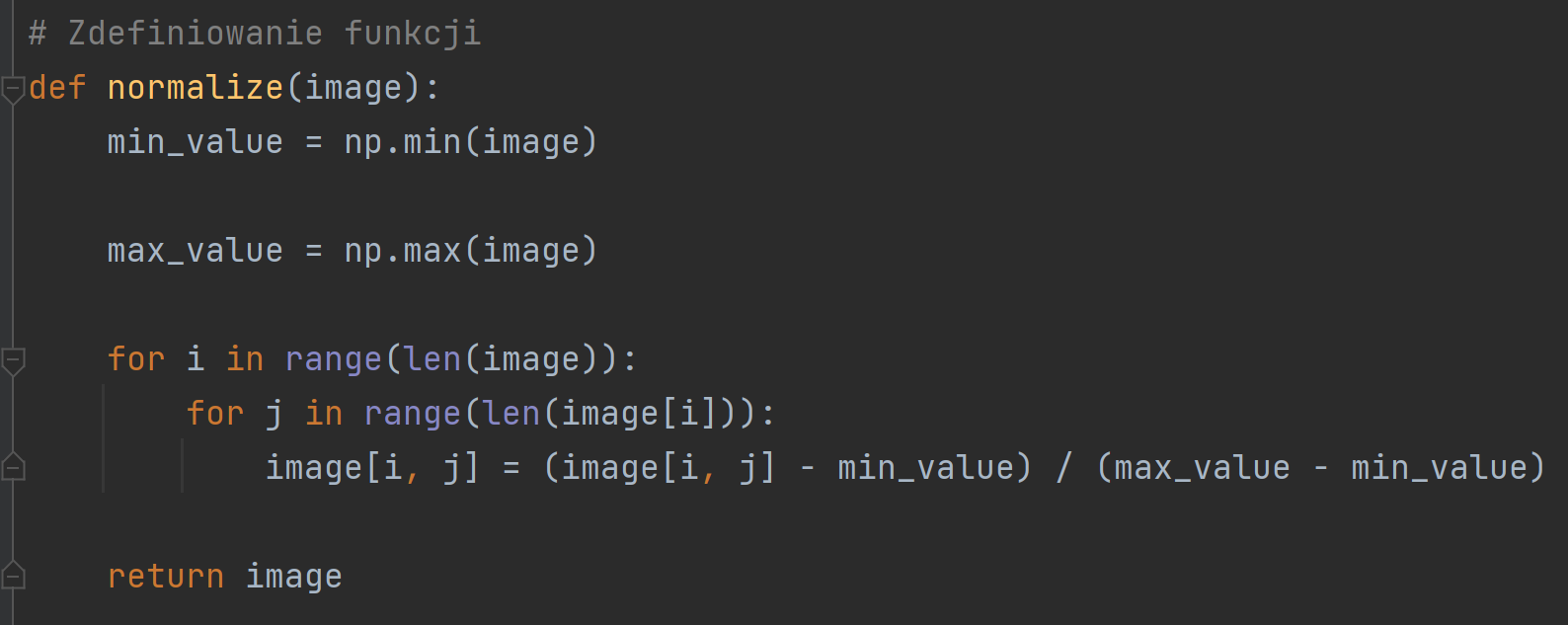


**Rys.4 Poprawione filtrowanie sinogramu**

Ten fragment kodu odpowiada za filtrowanie sinogramu. Najpierw odczytujemy liczbę projekcji oraz detektorów. Następnie w zależności od liczby detektorów dobieramy maskę (funkcja ***do\_mask***). Gdy już maska zostanie obliczona, to dla każdej projekcji odbywa się jej splot z maską.

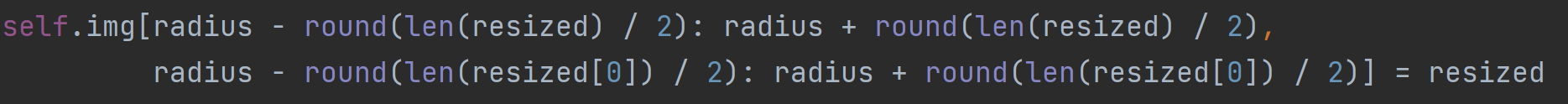
 **Rys.5 Przykładowy wynik filtrowania**

***Ustalenie jasności poszczególnych punktów obrazu oraz jego przetwarzanie końcowe (np. uśrednianie, normalizacja)***



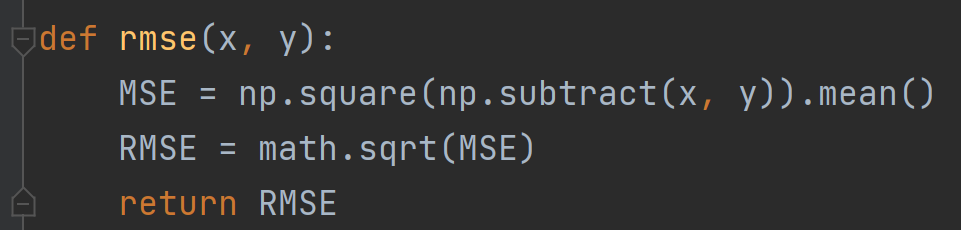
**Rys.6 Normalizacja**

Zadaniem funkcji ***normalize*** jest normalizacja otrzymanych wyników sinogramu oraz odwróconej transformaty sinogramu. Z kolei w celu uśrednienia obrazy wykorzystaliśmy poniższy fragment kodu.



**Rys.7 Uśrednianie**

***Wyznaczanie wartości miary RMSE na podstawie obrazu wejściowego oraz wyjściowego***



**Rys.8 Uśrednianie**

Funkcja ***rmse*** jest odpowiedzialna za obliczenie błędu średniokwadratowego. Najpierw odbywa się obliczenie MSE, czyli uśredniona suma potęgi różnic, a na końcu pierwiastkujemy poprzedni wynik otrzymując RMSE.

***Odczyt i zapis plików DICOM***

**Wynik eksperymentu sprawdzającego wpływ poszczególnych parametrów (liczba detektorów, liczba skanów, rozpiętość stożka/wachlarza z detektorami) na jakość obrazu wynikowego wyrażoną za pomocą miary RMSE. Jako wartości domyślne proszę przyjąć 180 detektorów, 180 skanów oraz rozpiętość wachlarza równą 180 stopni (PI)**

***Liczba detektorów zmienia się od 90 do 720 z krokiem 90***

***Liczba skanów(iteracji) zmienia się od 90 do 720 z krokiem 90***

***Rozpiętość wachlarza zmienia się od 45 do 270 z krokiem 45 stopni***

***Dla każdego wariantu proszę przedstawić wykres pokazujący zależność RMSE (oś Y) od aktualnej wartości zmienianego parametru (oś X) oraz krótko skomentować zaobserwowany przebieg***

***Dla dwóch wybranych obrazów oraz następujących parametrów: liczba detektorów = 360, liczba skanów = 360, rozpiętość wachlarza = 270 stopni, proszę wykonać dwa warianty obliczeń -- z włączonym i wyłączonym filtrowaniem sinogramu. Dla każdego obrazu proszę zaprezentować RMSE dla obrazu bez filtrowania i z filtrowaniem oraz krótko skomentować różnice w jakości między obrazami.***