**Raport z realizacji projektu Tomograf Komputerowy**

1. **Skład grupy –** Jakub Raczyński (136787), Julia Tadej (136820)
2. **Zastosowany model tomografu –** równoległy
3. **Zastosowany język programowania oraz dodatkowe biblioteki –** Projekt został w całości zrealizowany w języku programowania Python. Do zaprojektowania GUI wykorzystaliśmy bibliotekę Qt. Poza tym, w programie wykorzystaliśmy biblioteki takie jak:

**-** Numpy

**-** Scikit Image

**-** Pydicom

1. **Opis głównych funkcji programu**
2. Pozyskiwanie odczytów dla poszczególnych detektorów – Zaczynamy od wyliczenia współrzędnych emitera i detektora w danej iteracji, mając dany kąt położenia oraz promień okręgu, na którym układ się znajduje:

Wynik musi być zaokrąglony do liczby całkowitej, gdyż pracujemy na dyskretnym obrazie składającym się z pikseli.

Następnie iterujemy się wzdłuż pikseli należących do prostej przechodzącej przez emiter i detektor. Działanie to symuluje promień przechodzący przez obrazek. Do liniowego przejścia po kolejnych pikselach obrazu dyskretnego wykorzystaliśmy algorytm Bresenhama. Zdecydowaliśmy się na addytywne modelowanie pochłaniania promieniowania, więc promień na początku jest zerowy, a wraz z napotkaniem jasnych pikseli obrazu (przeszkody) staje się on coraz silniejszy. Jako że algorytm Bresenhama różni się nieznacznie dla czterech różnych przedziałów nachyleń prostej po której przechodzimy, poniżej załączamy kod jedynie dla prostych o współczynniku kierunkowym z przedziału (0,1).

 Powyższy kod to po prostu implementacja algorytmu Bresenhama. Jako że dla współczynników kierunkowych z tego przedziału współrzędna x zmienia się szybciej niż y, iterujemy po x-ach. Warto zwrócić uwagę, że cały obrazek jest „przesunięty” o połowę długości i szerokości tak, aby jego środek znajdował się w punkcie o współrzędnych (0,0). W ostateczności po wykonaniu tego algorytmu, w pikselu odpowiadającym danemu sinogramowi w danej pozycji otrzymujemy sumę jasności pikseli, przez które przeszedł nasz promień.

1. Filtrowanie sinogramu, zastosowany rozmiar maski – Maska wykorzystywana w naszym algorytmie ma długość 41 elementów. Nie było sensu wydłużać jej bardziej, gdyż wartości skrajne były bardzo małe i nie miały wpływ na wynik przetwarzania. Sinogram przefiltrowaliśmy wykonując na nim operację splotu z maską. Poniżej zamieszczamy fragment kodu, który odpowiadał za splot:



Przy wykonywaniu splotu trzeba było mieć na uwadze, żeby przesunąć maskę o połowę jej długości, gdyż jest ona symetryczna i piksel aktualnie przetwarzany powinien być przemnażany przez jej środkowy element.

1. Ustalanie jasności poszczególnych punktów obrazu wynikowego oraz jego przetwarzanie końcowe – Jasność poszczególnych punktów obrazu wynikowego jest równa wynikowi odwrotnej transformaty Radona dla danego piksela. Odwrotna transformata Radona przebiegała analogicznie do transformaty pierwotnej. Jedyna różnica względem transformaty pierwotnej, to dodawanie do piksela obrazu wynikowego (przez który przechodził promień, który został pochłonięty przez dany detektor) wartości odpowiadającej mocy promienia odczytanej przez dany detektor w danej pozycji zamiast dodawania do piksela sinogramu jasności piksela obrazu wejściowego. Poniżej

zamieszczam kod dla prostych o współczynnikach kierunkowych z przedziału (-1,0):



W ramach końcowego przetwarzania normalizowaliśmy obraz wynikowy i odrzucaliśmy najjaśniejsze oraz najciemniejsze piksele. Kod odpowiadający za końcowe przetwarzanie zamieszczamy poniżej:



1. Wyznaczanie wartości miary RMSE na podstawie obrazu źródłowego oraz wynikowego – Miarę tę po prostu wyznaczaliśmy iterując po kolejnych pikselach obrazów i podnosząc różnicę jasności między odpowiadającymi pikselami do kwadratu. Następnie różnice te były sumowane, suma była podzielona przez ilość pikseli, a następnie pierwiastkowana. Kod zamieszczamy poniżej:

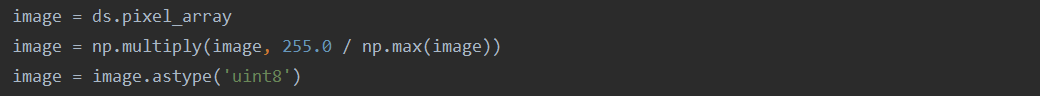


1. odczyt i zapis plików DICOM – Odczyt plików DICOM polega na pobraniu z otwartego obiektu [FileDataset](https://pydicom.github.io/pydicom/dev/reference/generated/pydicom.dataset.FileDataset.html#pydicom.dataset.FileDataset) wszystkich interesujących nas właściwości. W przypadku naszego programu odczytujemy dane o imieniu, nazwisku, ID i płci pacjenta, a także datę badania oraz komentarze. Są to te same informacje, których zapisanie umożliwia interfejs naszego programu. Wczytane dane ustawiane są w odpowiednich polach interfejsu graficznego, aby umożliwić ich modyfikację lub uzupełnienie. Zamieszczony poniżej fragment kodu prezentuje operację odczytania jednej z własności – ID pacjenta. Zupełnie analogicznie przebiega ona dla pozostałych wymienionych własności.





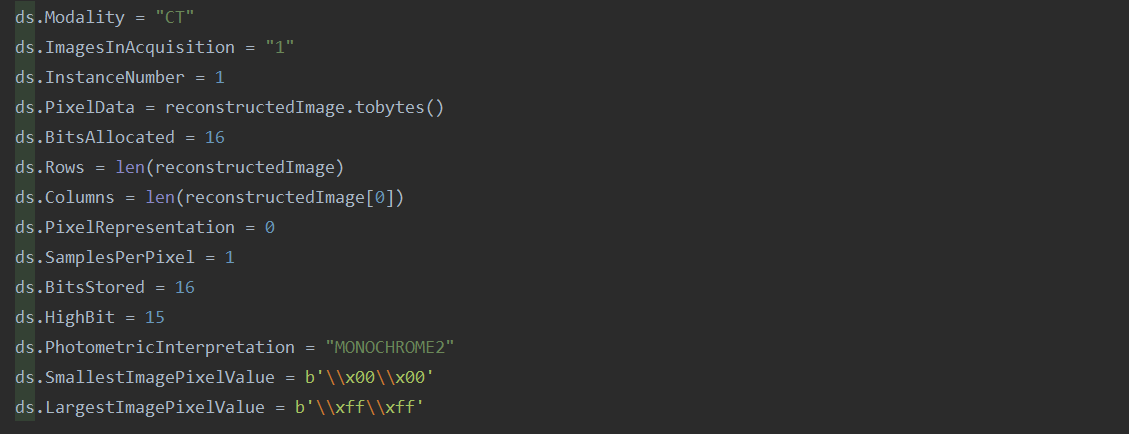
Po wczytaniu obrazu potrzebna jest dodatkowa konwersja do postaci użytecznej dla naszego programu tzn. spójnej z tą, w której wczytujemy obrazy niezapisane w standardzie DICOM.



Zapis plików w formacie DICOM to nieco bardziej skomplikowane zadanie. Aby zachować poprawność standardu koniecznie jest utworzenie obiektu przechowującego metadane. UID są generowane przy wykorzystaniu funkcji z biblioteki pydicom. W rzeczywistych zastosowaniach w numerach tych może być zaszyta informacja np. o podmiocie wykonującym badanie. Zapisywane są także wszystkie informacje o pacjencie pobrane z GUI. Dodatkowym elementem jest ID badania, generowane z pozyskanych informacji. Jest to jednocześnie nazwa zapisywanego pliku z rozszerzeniem .dcm.

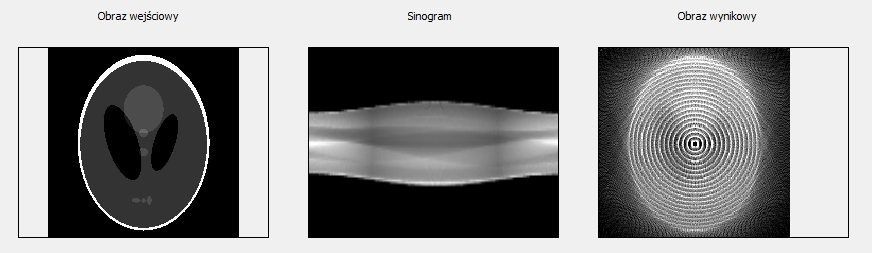


Aby móc poprawnie otworzyć zapisany obraz konieczne jest uzupełnienie szeregu parametrów z nim związanych. Przechowywanie obrazu w formie tablicy pikseli wymaga podania m.in. jego wymiarów, modelu kolorystycznego, sposobu reprezentacji pikseli (unsigned int lub U2) czy liczby bitów zaalokowanych oraz przechowywanych dla każdego z pikseli. Ustawiamy także informacje wskazujące na rodzaj przechowywanego obrazu – skan wykonany przez tomograf.

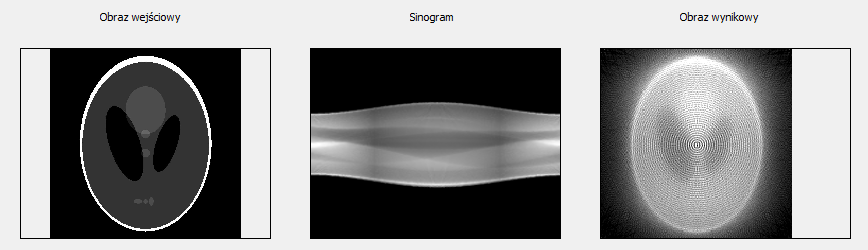


1. **Wynik eksperymentu sprawdzającego wpływ poszczególnych parametrów (liczba detektorów, liczba skanów, rozpiętość stożka/wachlarza z detektorami) na jakość obrazu wynikowego wyrażoną za pomocą miary RMSE –** Zgodnie z poleceniem przeprowadziliśmy eksperyment sprawdzający wpływ parametrów na jakość obrazu wynikowego wyrażoną subiektywnie oraz za pomocą miary RMSE. Wyniki zamieszczamy poniżej.

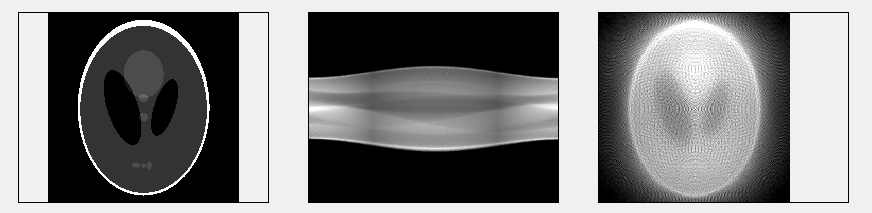
**ZMIANA LICZBY DETEKTORÓW:**



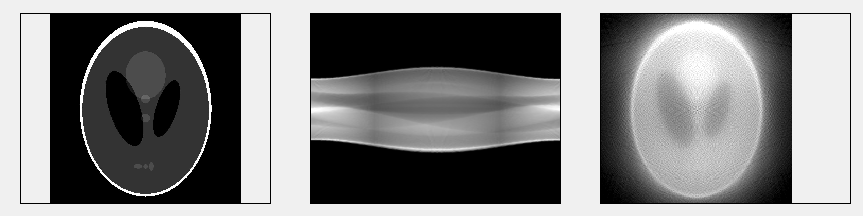
90 detektorów



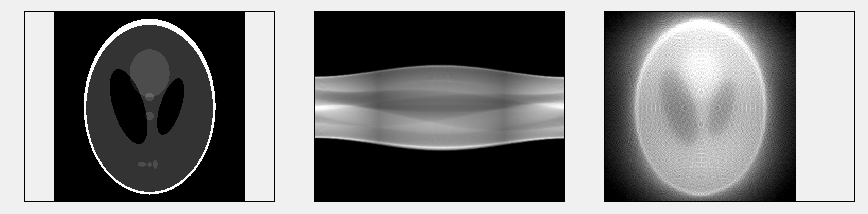
180 detektorów



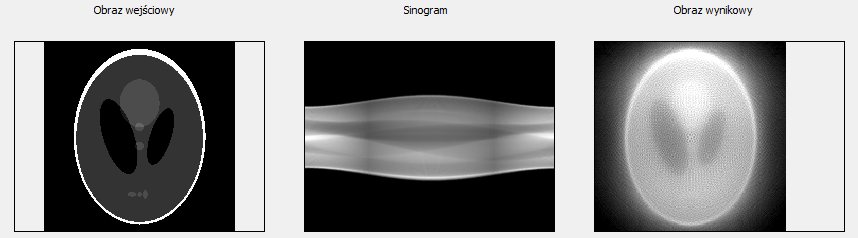
270 detektorów



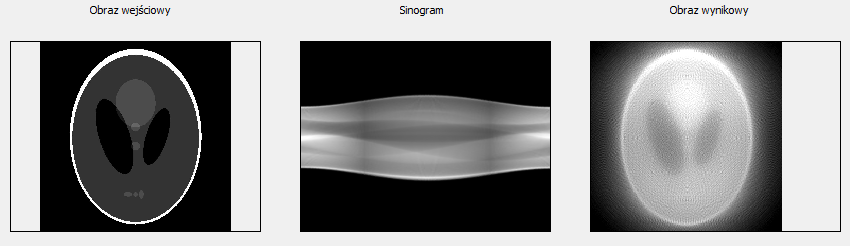
360 detektorów



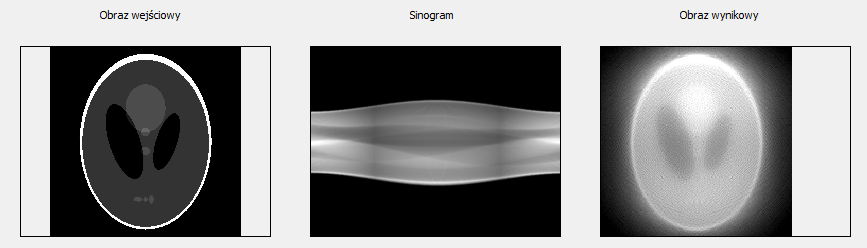
450 detektorów



540 detektorów



630 detektorów

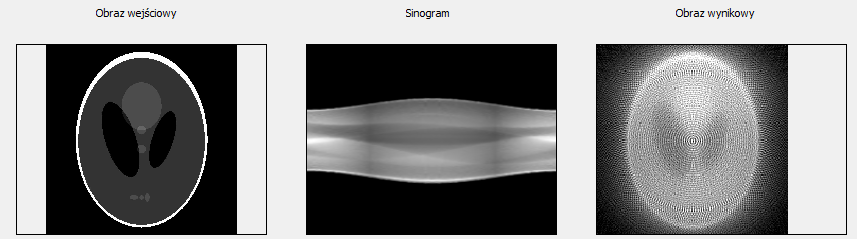


720 detektorów

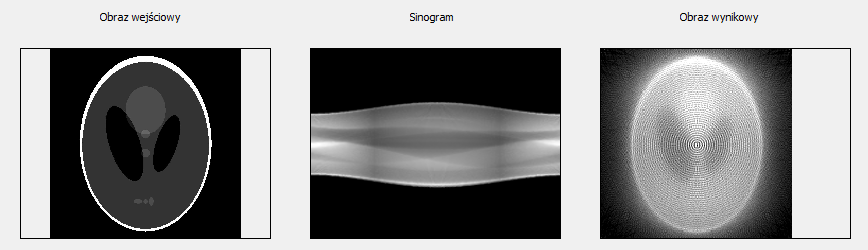
Niestety wzrost RMSE wraz z wzrostem liczby detektorów stoi w sprzeczności z wrażeniami subiektywnymi. Wynika to prawdopodobnie z tego, że obraz wynikowy jest ogółem jaśniejszy od obrazu wejściowego, a obrazy wygenerowane przez układy składające się z mniejszej liczby detektorów posiada ciemne prążki, co zmniejsza RMSE. Jeśli chodzi o wrażenie subiektywne, to im więcej detektorów, tym mniej artefaktów (prążków) posiada obraz wynikowy. Największą różnicę można zauważyć od 90 do 360 detektorów.

Mimo to, że sam wzrost RMSE pokazuje jak zła jest to miara, to można zauważyć, że im większa różnica RMSE między kolejnymi punktami wykresu, tym większa jest różnica w subiektywnym odbiorze jakości obrazu. Wykres ten dobrze pokazuje, że dla rozpiętości kątowej układu detektorów 180 stopni oraz 180 iteracjach nie ma sensu wykorzystywać więcej, niż 360 detektorów.

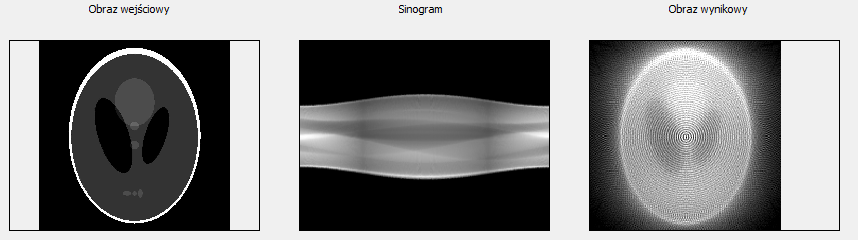
**ZMIANA LICZBY SKANÓW:**



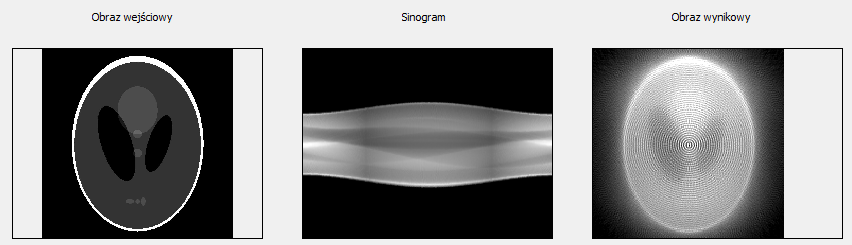
90 skanów



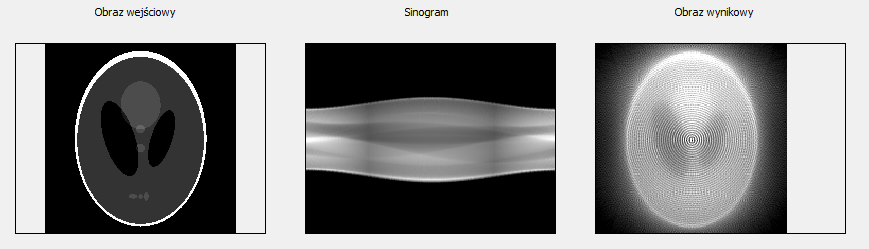
180 skanów



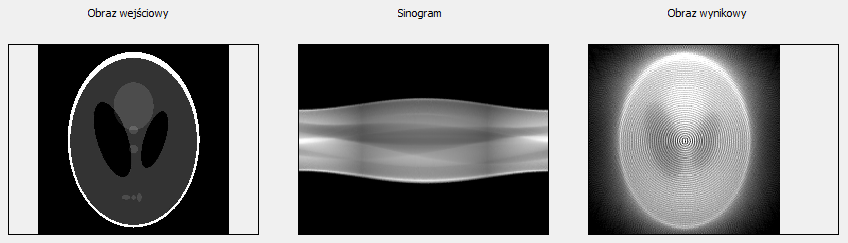
270 skanów



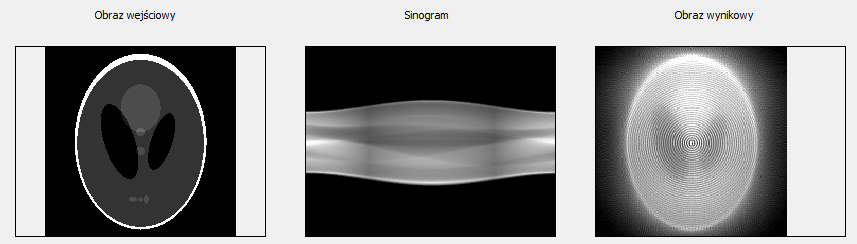
360 skanów



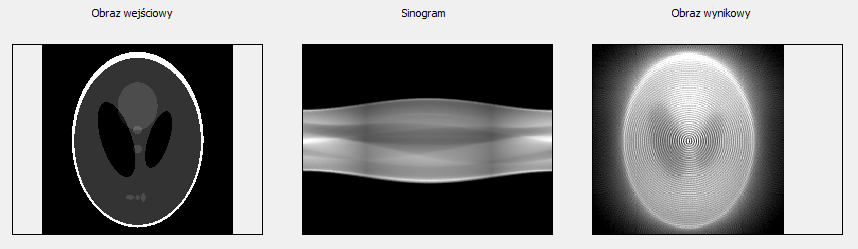
450 skanów



540 skanów



630 skanów



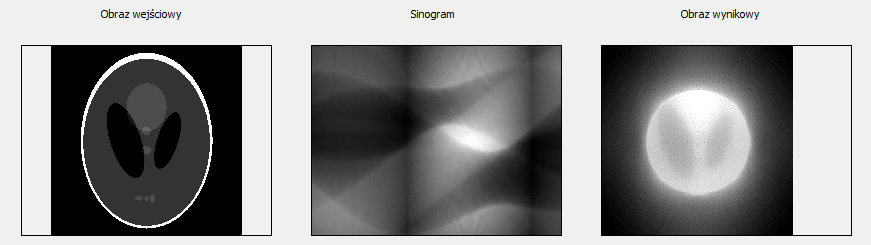
720 skanów

Niestety ponownie wzrost RMSE wraz z wzrostem liczby detektorów stoi w sprzeczności z wrażeniami subiektywnymi. Sytuacja wygląda bardzo analogicznie do zmiany liczby detektorów. Mimo to, różnice w wartościach RMSE nawet dla niewielkiej liczby skanów (90,180,270) są stosunkowo niewielkie.

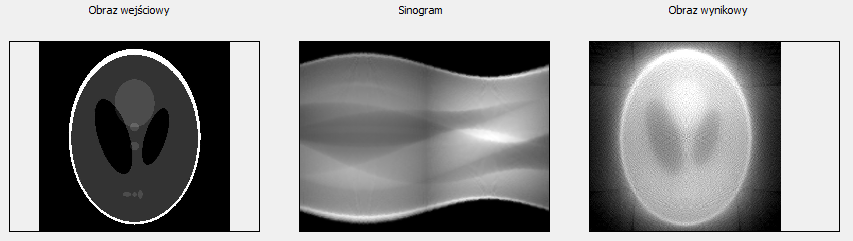
Jeśli chodzi o subiektywne wrażenie, to ponownie największa poprawa w jakości obrazu jest przy zwiększaniu liczby skanów z 90 na 180 i ze 180 na 270. Oznacza to, że w zasadzie nie warto wykonywać więcej niż 360 skanów, gdyż narzut czasowy jest dużo większy od korzyści w postaci poprawy jakości obrazu.

Różnice między kolejnymi wartościami RMSE ponownie dobrze odzwierciedlają subiektywne odczucie różnicy jakości rekonstrukcji.

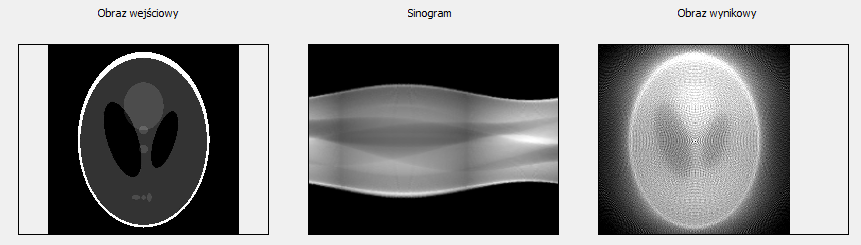
**ZMIANA ROZPIĘTOŚCI DETEKTORÓW/EMITERÓW:**



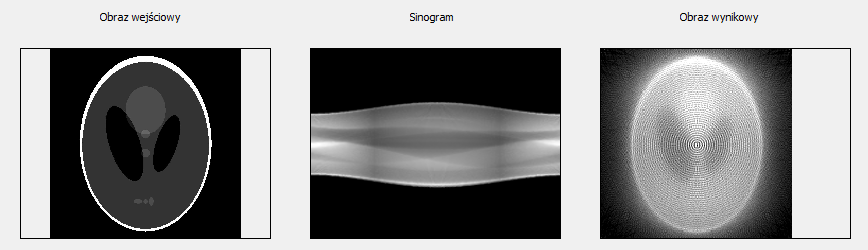
45 stopni



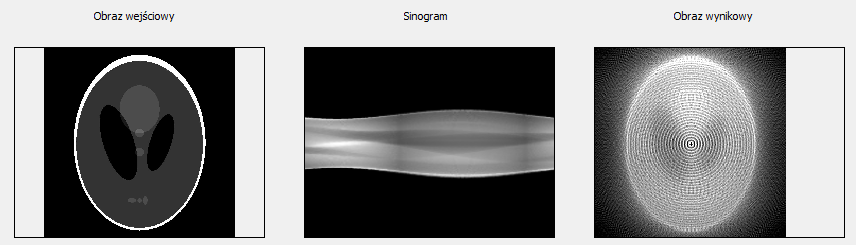
90 stopni



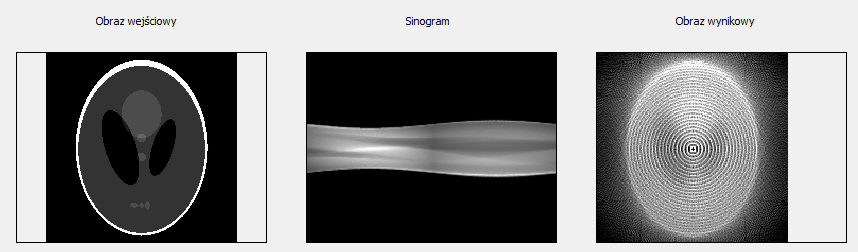
135 stopni



180 stopni



225 stopni



270 stopni

Zależność RMSE od rozpiętości detektorów nie jest już taka prosta. Okazuje się, że najmniejsze RMSE otrzymujemy dla 45 stopni, gwałtownie rośnie ono dla 90 stopni, a następnie zaczyna stopniowo maleć.

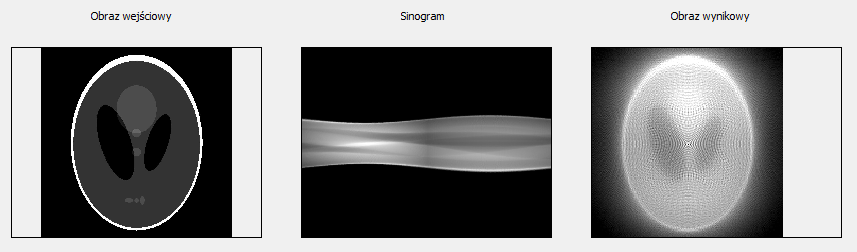
Patrząc na wrażenia subiektywne, to widać, że dla 45 stopni zrekonstruowany obraz jest jakby „niepełny”. Dopiero od 90 stopni obraz jest w 100% zrekonstruowany. Okazuje się jednak, że wraz z wzrostem rozpiętości kątowej jakość obrazu pogarsza się i staje się on coraz bardziej „prążkowany”.

Warto jednak zauważyć, że patrząc na wyniki eksperymentu, nie można brać pod uwagę tylko jednego parametru. Kluczowe jest odpowiednie dobranie wszystkich, np. liczba detektorów powinna być około 2 razy większa od rozpiętości detektorów w stopniach – dzięki temu sinogram mieści się na obrazie i stanowi jego dużą część, a rekonstrukcja jest dokładniejsza. Nie wolno jednak za bardzo zmniejszyć rozpiętości kątowej detektorów, gdyż sinogram „nie zmieści się” na obrazie i w związku z tym rekonstrukcja będzie niepełna.

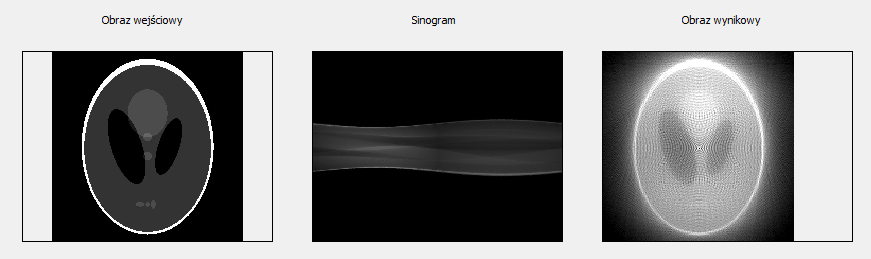
Porównując wartości RMSE z wrażeniami subiektywnymi okazuje się, że RMSE idealnie modelowałoby wrażenia subiektywne… Gdyby tylko większa wartość RMSE oznaczała wyższą jakość obrazu, a jest odwrotnie. Po raz trzeci otrzymaliśmy dowód na to, że RMSE jest bardzo słabą miarą błędu.

**Wpływ filtrowania sinogramu na wygląd rekonstrukcji:**

W tej części eksperymentu wykorzystaliśmy obrazy shepplogan oraz saddle\_pe. Poniżej załączamy screenshoty prezentujące wyniki eksperymentu:

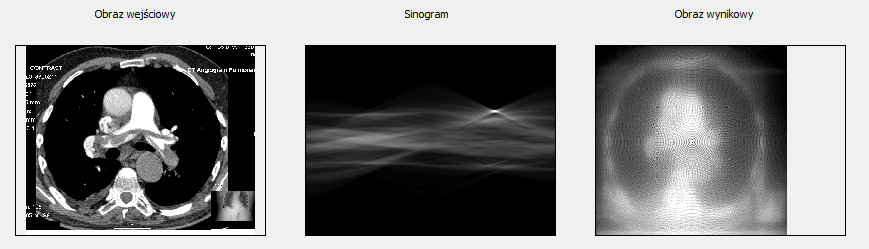


Bez filtra, RMSE: 0,439

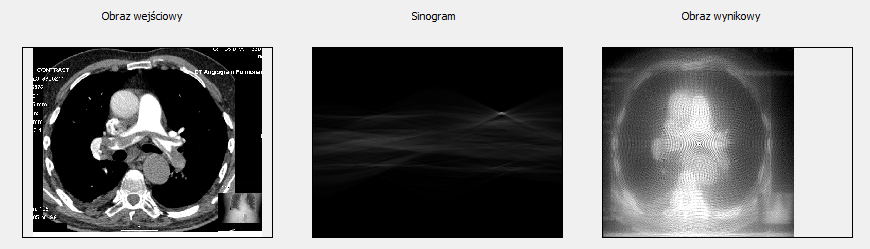


Z filtrem, RMSE: 0,417

Rekonstrukcja utworzona z filtrowanego sinogramu ma niższe RMSE. Jeśli chodzi o subiektywne wrażenie, to trudno zauważyć jakiekolwiek efekty filtrowania. Obraz jest minimalnie mniej ziarnisty oraz bardziej wyostrzony. Spodziewaliśmy się jednak bardziej spektakularnego efektu.



Bez filtra, RMSE: 0,323

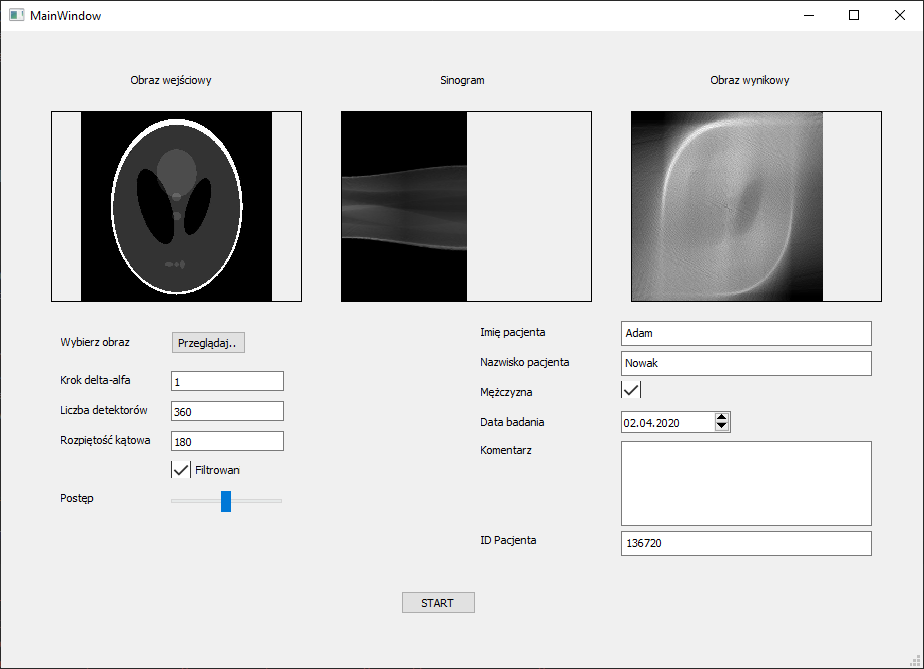


Z filtrem, RMSE: 0,302

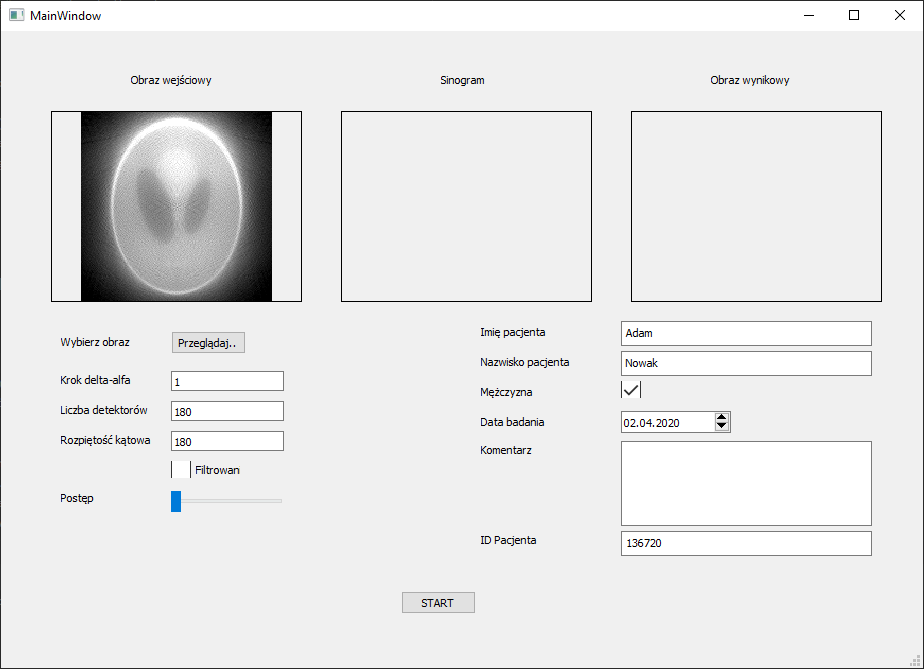
Efekt wykorzystania filtracji sinogramu przyniósł podobne efekty jak w wypadku obrazu shepplogan. Błąd RMSE został zmniejszony, a obraz stał się minimalnie ostrzejszy. Ponadto wydaje się, że filtrowanie pozytywnie wpłynęło (zwiększyło) kontrast rekonstrukcji. Mimo to, efekt nadal nie powala.

1. **Nasza aplikacja**

Screen jednego z kroków – w przybliżeniu w połowie obrotu układu detektorów



Odczyt utworzonego pliku DICOM



1. **Zapisany plik odczytany w przeglądarce plików DICOM**

