# Raport: Tomografia komputerowa.

## 1. Wstęp.

Program pozwala na rekonstrukcję obrazu wejściowego na zasadzie symulacji działania tomografu komputerowego przy podanych parametrach:

- Kąt θ, o jaki przesuwany jest emiter przy każdej iteracji,
- Liczba detektorów,
- Rozpiętość detektorów,
- Typ filtra.

Aplikacja okienkowa umożliwia również porównanie zrekonstruowanego obrazu z obrazem wejściowym oraz analizę wygenerowanego sinogramu i błędu średniokwadratowego dla każdej iteracji programu.

## 2. Realizacja.

Program został napisany w języku Python 3, przy użyciu pakietów: matplotlib oraz numpy. Główna część programu wykonuje transformatę Radona, a następnie odwrotną transformatę wygenerowanego wcześniej sinogramu użyciem filtrowania lub bez. Zastosowany został model stożkowy tomografu. Przetwarzany obraz, aby mógł zostać zrekonstruowany, przekształcany jest do kwadratu, w którym obraz wejściowy jest wpisany w okrąg, a pozostały obszar jest czarny. Umożliwia to odtworzenie każdego piksela obrazu wejściowego.

Algorytm transformaty Radona zaczyna się wyznaczeniem współrzędnych centrum obrazu oraz pierwszej pozycji emitera dla danych wymiarów obrazu. Następnie wyznaczane są na podstawie poszczególnych kątów oraz wcześniej wspomnianych współrzędnych, pozycje emitera, a później analogicznie wyznaczane są współrzędne detektorów znajdujących się naprzeciwko niego. Dla każdego detektora i odpowiadającego mu emitera uruchamiany jest algorytm Bresenhama, który zwraca współrzędne na obrazie, przez które przechodzi promień. Wartości na obrazie odpowiadające tym współrzędnym są sumowane i trafiają do odpowiedniego miejsca w sinogramie. Po przetworzeniu wszystkich możliwych układów emiter/detektor, otrzymujemy sinogram.

Algorytm odwrotnej transformaty Radona wykorzystuje podobne dane. Dla każdej projekcji w sinogramie dla danego kąta emitera oraz odpowiadających mu pozycji detektorów, pobierane są odpowiadające wartości z sinogramu, które następnie są nanoszone na początkowo czarny obraz przy użyciu algorytmu Bresenhama. Z każdą iteracją obraz ten coraz bardziej przypomina obraz wejściowy. Ostatecznie zostaje odtworzony z pewnym błędem. Zrekonstruowany obraz byłby pozbawiony błędu, jeśli wykonalibyśmy algorytm dla nieskończonej liczby kątów oraz detektorów.

Algorytm filtrowania polega na wygenerowaniu funkcji o kształcie odwróconego „V”, pomnożenie jej przez transformatę Fouriera sinogramu, a na otrzymanym produkcie wykonywana jest odwrotna transformata Fouriera, czego rezultatem jest odfiltrowany sinogram. Funkcja ma taki kształt, ponieważ zadaniem filtra jest podkreślenie wysokich, a niwelacja niskich częstotliwości na obrazie.

Dyskretna dwu-wymiarowa transformata Fouriera oraz odwrotna transformata zaimplementowane zostały na podstawie poniższych wzorów:

- Dwuwymiarowa dyskretna transformata Fouriera:

$$F[k,l]=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f[m,n]e^{-j2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)}$$

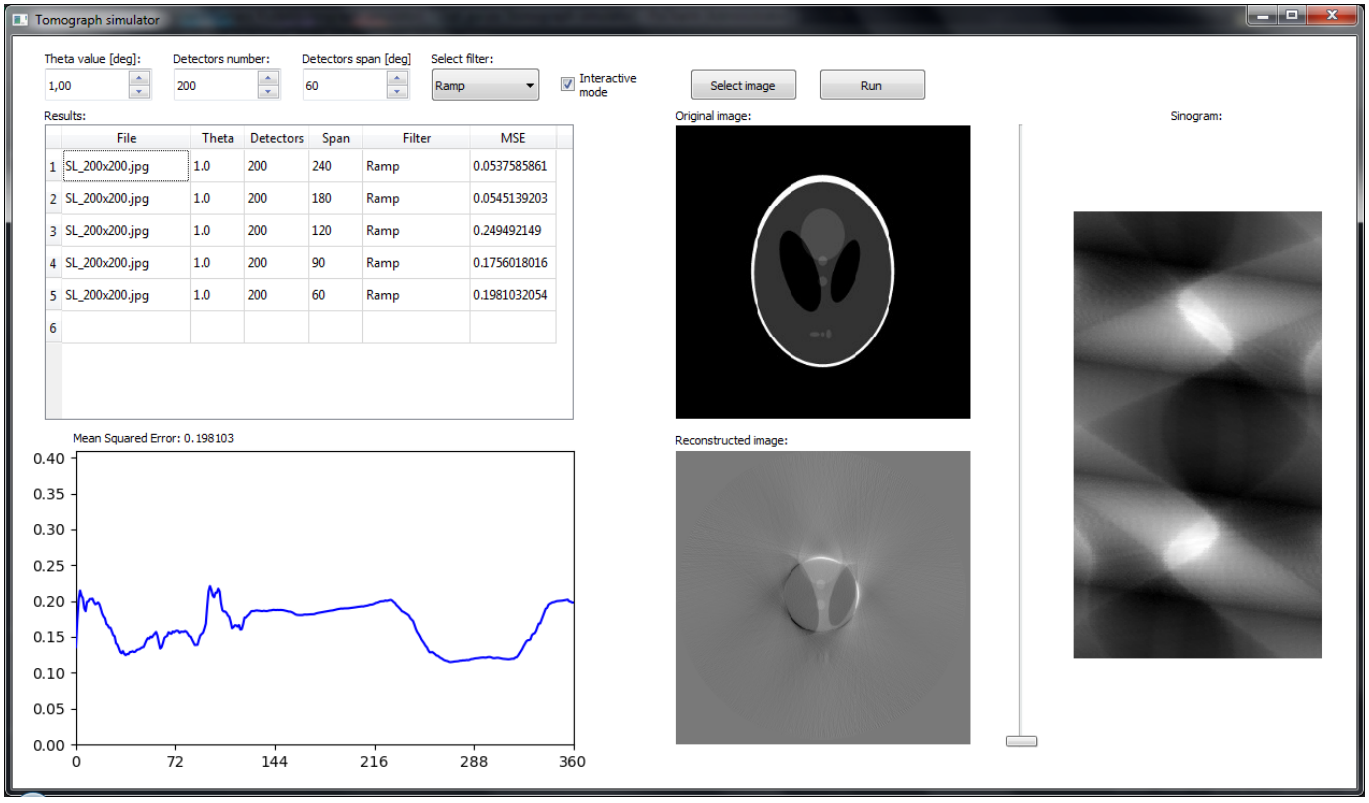
- Odwrotna dwuwymiarowa dyskretna transformata Fouriera:

$$f[m,n]=\frac{1}{MN}\sum_{k=0}^{M-1}\sum_{l=0}^{N-1} F[k,l]e^{j2\pi\left(\frac{k}{M}m+\frac{l}{N}n\right)}$$

$$\text{where}\quad k=0,1,\ldots,M-1\quad\text{and}\quad l=0,1,\ldots,N-1$$

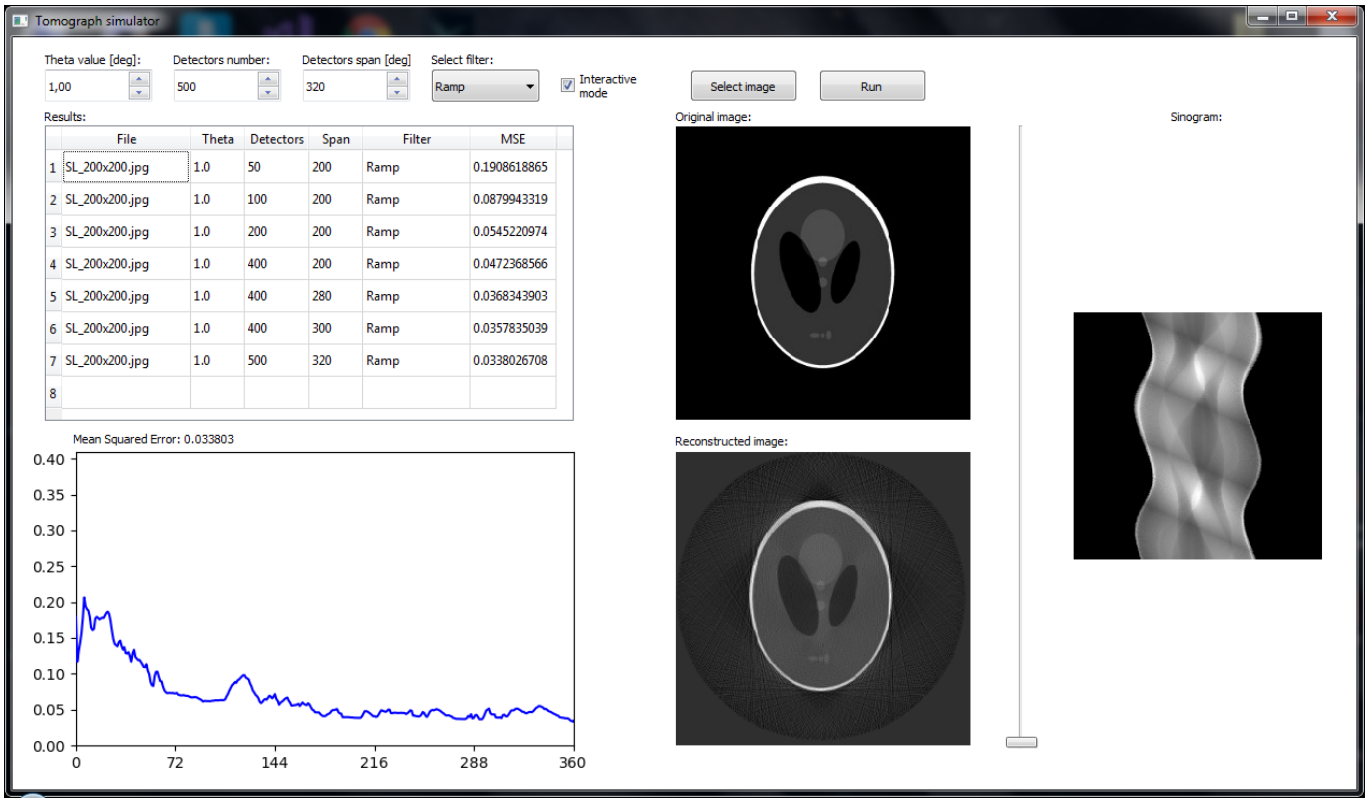
## 3. Przykładowe instancje.

Zaprezentowane poniżej wykresy prezentują przykładowe działanie aplikacji oraz mają na celu zaprezentowanie zależności parametrów od błędu średniokwadratowego. Wszelkie parametry wejściowe widoczne są na obrazach. Wyniki poszczególnych instancji widoczne są w tabeli.

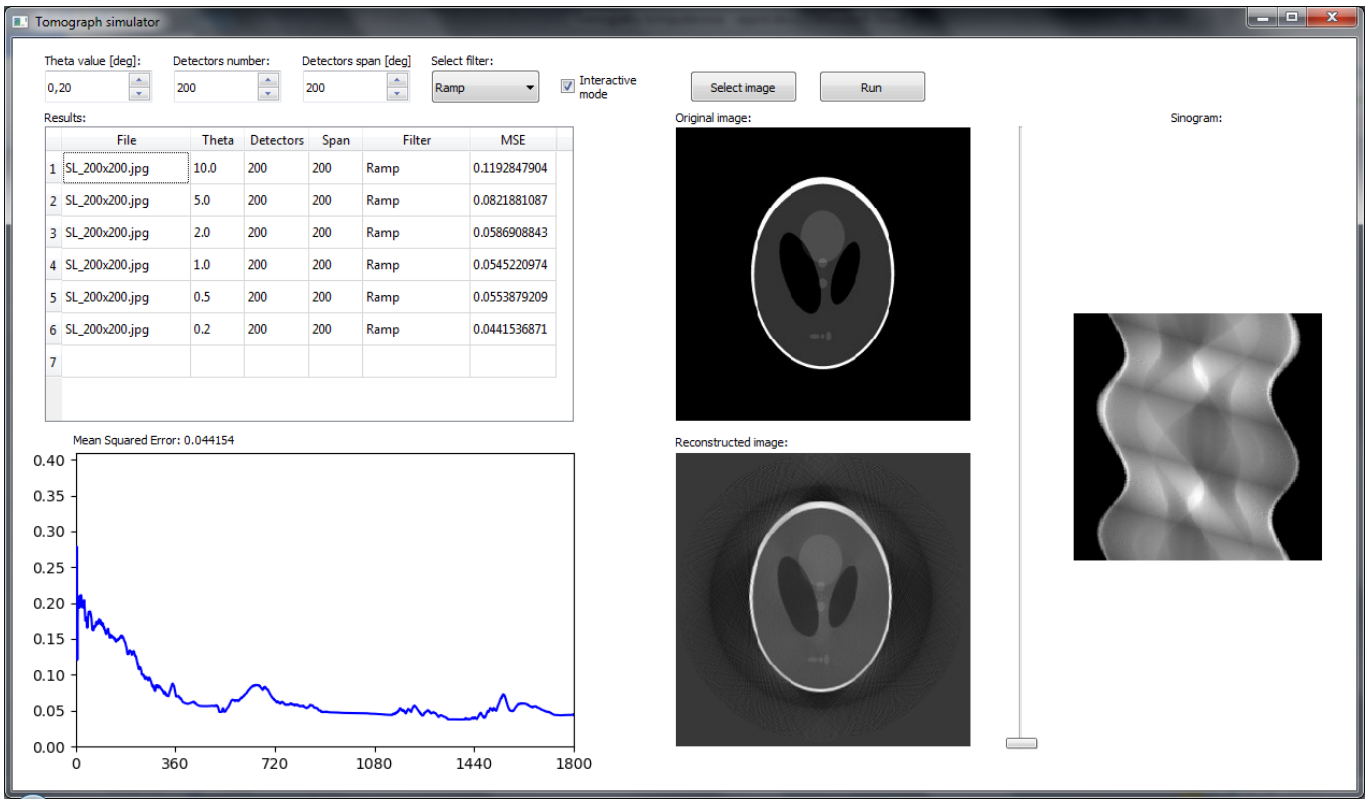



| | File | Theta | Detectors | Span | Filter | MSE |
|---|---|---|---|---|---|---|
| 1 | SL_200x200.jpg | 1.0 | 200 | 240 | Ramp | 0.0537585861 |
| 2 | SL_200x200.jpg | 1.0 | 200 | 180 | Ramp | 0.0545139203 |
| 3 | SL_200x200.jpg | 1.0 | 200 | 120 | Ramp | 0.249492149 |
| 4 | SL_200x200.jpg | 1.0 | 200 | 90 | Ramp | 0.1756018016 |
| 5 | SL_200x200.jpg | 1.0 | 200 | 60 | Ramp | 0.1981032054 |
| 6 | | | | | | |

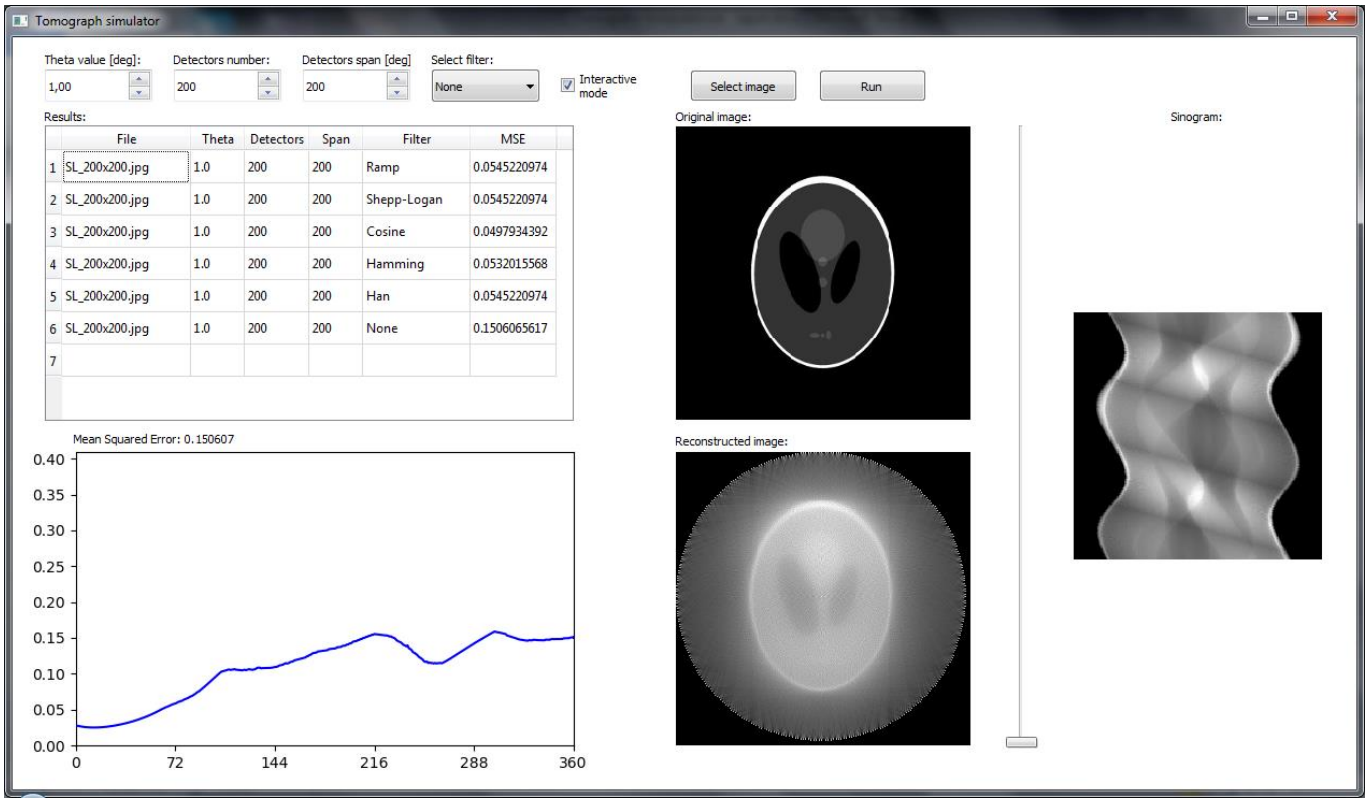
Dla każdej instancji zmieniana jest wartość rozpiętości detektorów. Można wywnioskować, że jakość zrekonstruowanego obrazu jest lepsza wraz ze wzrostem rozpiętości. Poniżej 180 stopni można zaobserwować ucinanie obrazu, co powoduje większy błąd.



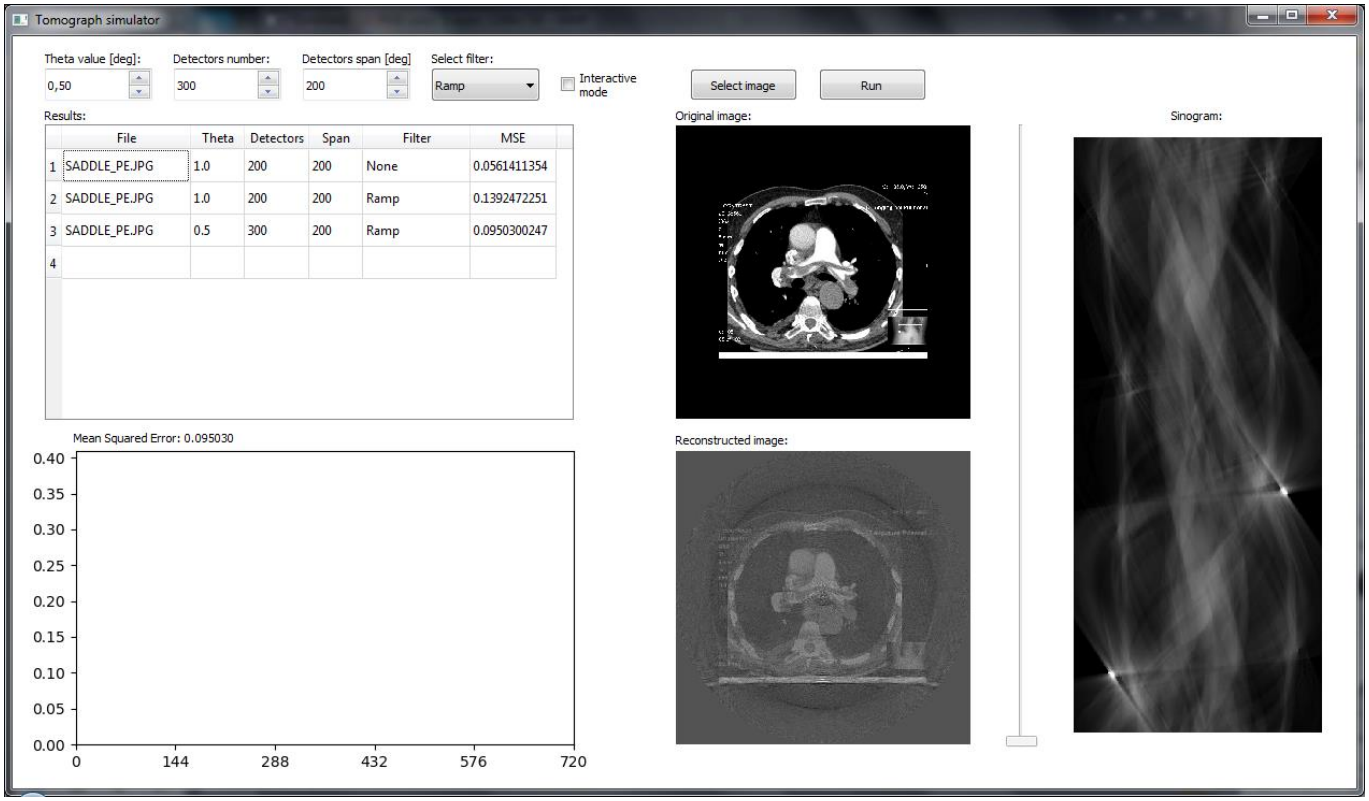

Dla każdej instancji zmieniana jest ilość detektorów. Można wywnioskować, że jakość zrekonstruowanego obrazu jest lepsza wraz ze wzrostem ilości detektorów. Co więcej, należy zauważyć, że duża rozpiętość przy dużej ilości detektorów potęguje efektywność rekonstrukcji.



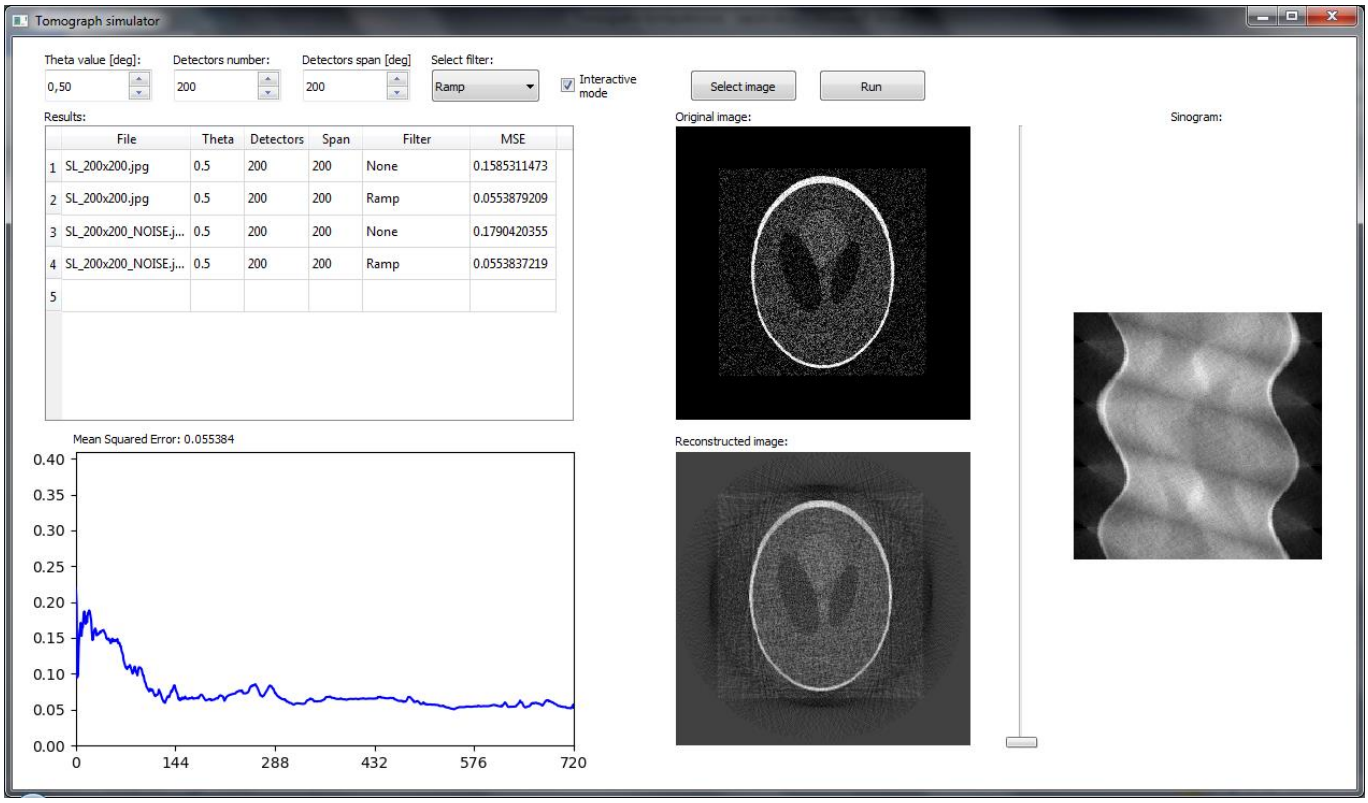

Dla każdej instancji zmieniany jest kąt kroku w pojedynczej iteracji. Można wywnioskować, że jakość zrekonstruowanego obrazu jest lepsza wraz z doborem mniejszego kąta. Więcej iteracji pozwala dokładniej odwzorować obraz.




Dla każdej instancji zmieniany jest filtr. Można zaobserwować, że jakość zrekonstruowanego obrazu jest podobna dla różnych rodzajów filtru. Dla filtru „Cosine” występuje niski błąd, prawdopodobnie ze względu na rozmycie jakie powoduje na zrekonstruowanym obrazie. Rekonstrukcja obrazu bez użycia filtra jest obarczona dużym błędem ze względu na rozmycie i niski kontrast.




W powyższym przykładzie zastosowane zostało zdjęcie, w którym występuje dużo mniejszych, trudniejszych do odwzorowania elementów. Przy ustawieniu odpowiednich parametrów i filtrowania udaje się całkiem dobrze zrekonstruować obraz.




W powyższym przykładzie użyto obrazu z wygenerowanym szumem. Można zaobserwować, że błąd jest nieco większy dla obrazu z szumem zrekonstruowanego bez użycia filtra, natomiast z użyciem filtra błąd jest podobny. Mimo tego porównując do poprzednich przykładów, wyraźnie widać, że zrekonstruowany obraz jest gorszej jakości. Wynika to z właściwości filtra.