

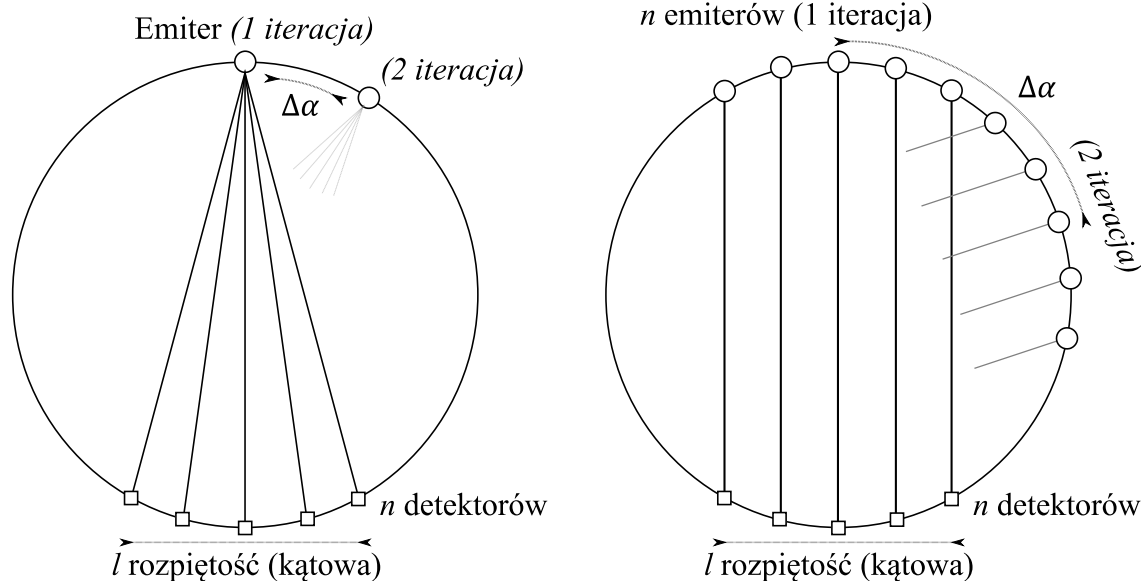
Symulator tomografu komputerowego

Słowa kluczowe: symulacja, wizualizacja, analiza danych, DICOM

Opis: Implementacja aplikacji symulującej działanie tomografu komputerowego (symulacja dwuwymiarowa). Zobacz: <https://www.youtube.com/watch?v=tgNP-n2z3po>

Wymagania (obowiązkowe):

- Aplikacja desktopowa przy wykorzystaniu dowolnego języka programowania (Java, C#, Python – tutaj może trochę wolniej przetwarzać obrazy, itp.)
- Wejściowy format obrazu: bitmapa. Obraz kwadratowy. Zakładamy, że badany obiekt w całości znajduje się w okręgu wpisanym w kwadrat wyznaczony przez granice obrazu. Pracujemy tylko na obrazach o czarno-białej (w odcieniach szarości) paletce barw. Ale nie binarnej. Każdy piksel może mieć dowolny kolor szarości z zakresu 0-255 (przynajmniej).
- Aplikacja musi dokonać transformacji Radona obraz wejściowy \rightarrow sinogram i odwrotnej transformacji sinogram \rightarrow obraz wyjściowy. Wymagana jest wizualizacja wyników (obraz wejściowy, sinogram, obraz wyjściowy).
- **Niedozwolone jest** wykorzystywać wbudowanych magicznych funkcji typu *doTomography()* itp. Obliczenia muszą zostać własnoręcznie zamodelowane. Ponadto nie wolno założyć stałej pozycji emitery(detektorów) i symulować obrót dokonując rotacji obrazu. Ruch emitery(detektorów) należy zamodelować samemu (funkcja kąta).
- Aplikacja powinna umożliwić wygenerowanie sinogramu i obrazu wyjściowego bez pokazania kroków pośrednich oraz z pokazywaniem (iteracyjnie), czyli z, np., suwakiem, którym regulujemy postęp obrotu emitery(detektorów) i aktualizujemy wyniki na bieżąco.
- Należy wykorzystać jeden z dwóch modeli emitery(detektor): stożkowy lub równoległy.
- Aplikacja powinna móc pozwolić konfigurować następujące elementy:
 - a) Krok $\Delta\alpha$ układu emitery(detektor).
 - b) Dla jednego układu emitery(detektor) liczbę (próbkowanie) rzutów na detektory (n).
 - c) Rozwartość (układ stożkowy)/rozpiętość (układ równoległy) układu emitery(detektor) (l).



- Należy wykorzystać algorytm Bresenhama do linowego przejścia po kolejnych pikselach obrazu dyskretnego.

- Symulację pochłaniania promieniowania można zasymulować na jeden ze sposobów: addytywny/substraktywny/ilorazowy. Należy sobie poradzić z normalizacją wyników.
- Należy zastosować proste filtrowanie (splot) by zredukować szum powstały przez niedokładną (dyskretną/skończoną) odwrotną transformację (link poniżej).
- Należy dokonać prostej analizy statystycznej w oparciu o jedną zdefiniowaną miarę jakości. Np. mając obraz wejściowy i wyjściowy można policzyć błąd średniokwadratowy (lub lepiej – jego pierwiastek; po wszystkich pikselach różnicy – wartość bezwzględna – obrazu wejściowego i wyjściowego). Taka analiza powinna uwzględnić następujące elementy:
 - Błąd średniokwadratowy w funkcji iteracji (jego spadek).
 - Pokazać spadek błędu średniokwadratowego przy zwiększaniu dokładności próbkowania (trzy uprzednio wymienione parametry modelu emiter/detektor).
 - Błąd średniokwadratowy a zastosowane filtrowanie/jego brak. Pokazać zysk.

Linki

- Iteracyjna symulacja: <https://www.youtube.com/watch?v=tgNP-n2z3po>
- Algorytm Bresenhama: https://pl.wikipedia.org/wiki/Algorytm_Bresenhama
- Filtrowanie: <http://www.dspguide.com/ch25/5.htm>
- Splot: [https://pl.wikipedia.org/wiki/Splot_\(analiza_matematyczna\)](https://pl.wikipedia.org/wiki/Splot_(analiza_matematyczna))
- DICOM: <https://pl.wikipedia.org/wiki/DICOM>