Rozpoznawanie kości stawu biodrowo-krzyżowego

Projekt z przedmiotu Metody Rozpoznawania Obrazów

*Sylwia Kaleta, 113964*

*Kamil Wanat, 114021*

**Spis treści**

[1. Użyte technologie 4](#_Toc524081495)

[1.1. Python 4](#_Toc524081496)

[1.2. DICOM 5](#_Toc524081497)

[1.3. C++ w wersji 17 5](#_Toc524081498)

[1.4. PGM 7](#_Toc524081499)

[2. Pojęcia używane w pracy 7](#_Toc524081500)

[3. Środowisko 8](#_Toc524081501)

[3.1. O programie 3D Slicer 8](#_Toc524081502)

[3.2. Slicer 3D a przechowywanie obrazu DICOM 8](#_Toc524081503)

[3.3. CLion 9](#_Toc524081504)

[3.4. MinGW 9](#_Toc524081505)

[4. Prace początkowe 10](#_Toc524081506)

[4.1. Cel projektu 10](#_Toc524081507)

[4.2. Metoda działów wodnych 10](#_Toc524081508)

[4.3. Metoda baloników (metoda ustalania poziomu) 10](#_Toc524081509)

[4.4. Implementacja rozwiązania w środowisku Slicer 3D 11](#_Toc524081510)

[5. Nowa koncepcja prac 11](#_Toc524081511)

[5.1. Cel projektu 11](#_Toc524081512)

[5.2. Opis rozwiązania 11](#_Toc524081513)

[5.3. Testy 12](#_Toc524081514)

[5.4. Dobór liczby iteracji filtru Gaussa 13](#_Toc524081515)

[5.5. Poszukiwanie najlepszej wartości tresholdu 14](#_Toc524081516)

[5.6. Wybór punktów startowych 17](#_Toc524081517)

[5.7. Porównanie wyników na innych obrazach 19](#_Toc524081518)

[6. Wnioski 22](#_Toc524081519)

[7. Podsumowanie 24](#_Toc524081520)

[8. Spis rysunków 25](#_Toc524081521)

[9. Bibliografia 26](#_Toc524081522)

# Użyte technologie

## Python

Python jest stworzonym przez Guido van Rossuma [4] open source’owym językiem programowania ogólnego użytku optymalizowanym pod względem, jakości i wieloplatfor-mowości [7]. Zaliczany jest do grupy języków obiektowych, a dynamiczna typizacja sprawia, że jest on znacznie prostszy w użyciu niż języki obiektowe typu Java czy C++.

Jest to język interpretowany, dzięki czemu gotowy program jest łatwiejszy w modyfi-kacji niż gdyby był napisany w językach kompilowanych typu C czy C++, jednak z drugiej strony powoduje to, że napisane w nim aplikacje cechują się większą powolnością działania [8].

Kod źródłowy napisany w Pythonie w rzeczywistości najpierw jest kompilowany do postaci kodu bajtowego [4] – jest to kod języka obiektowego skompilowany do uruchomienia na wirtualnej maszynie specyficznej dla danego środowiska. Maszyna wirtualna tłumaczy taki kod do postaci zrozumiałej dla procesora. Kod bajtowy zawiera ciąg instrukcji, zwykle o dłu-gości jednego bajta, nie odpowiadają one bezpośrednio instrukcjom procesora, jednak są to opisy pojedynczych operacji [6].

Otwarta licencja umożliwia zastosowanie go również do komercyjnych projektów, a możliwość rozwijania języka przez programistów (open source) oraz dostępność na wielu platformach (np. GNU/Lunux, Windows, Mac OS [5]) sprawiają, że staje się on coraz bar-dziej popularny. Już w 2003 roku liczba użytkowników tego języka była oceniana na 1 milion i do tego grona należeli nie tylko użytkownicy indywidualni, ale również duże, znane korporacje, jak Google, Hewlett-Packard, IBM, Seagate czy Yahoo! [5]. Twórcy Pythona stara-ją się zawrzeć w nim najlepsze rozwiązania pochodzące z innych języków programowania [4].

Mimo swojej prostoty, a może właśnie dzięki niej, jest on wciąż wykorzystywany przez korporacje typu Nokia, IBM, Google, czy Cern, które wykorzystują go w projektach wartych wiele milionów dolarów. Także jeden z najbardziej znanych serwisów wideo – YouTube częściowo jest napisany w tym języku [5].

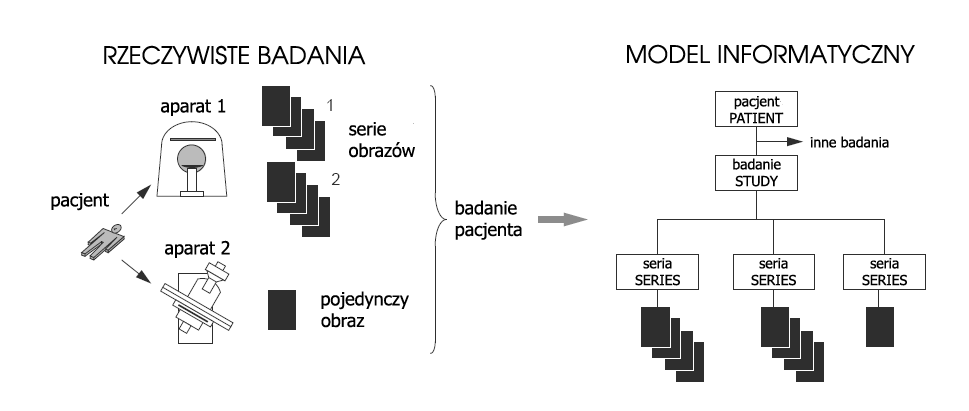
Język ten może być wykorzystywany do tworzenia przeróżnego rodzaju aplikacji, od skryptów mających na celu wykonanie konkretnego zadania, przez serwisy internetowe, a na aplikacjach desktopowych (w tym grach) kończąc. Warto również wspomnieć, że możemy się spotkać z implementacjami Pythona na różne platformy, na przykład: Jython na maszynę wir-tualną Javy lub IryonPython dla .NET [5].

## DICOM

Dicom (Digital Imaging and Communications In Medicine) – jest to międzynarodowy standard służący do wymiany, przechowywania, przetwarzania i wyświetlania informacji medycznych. Umożliwia on integrację różnych systemów i środowisk do przetwarzania obrazów medycznych. Ciągły rozwój i utrzymanie tego standardu sprawiają, że odpowiada on potrzebom współczesnej medycyny [2].

Do najistotniejszych informacji przechowywanych w tym standardzie można zaliczyć [3]:

* dane pacjenta [patient], wszystkie dane personalne pacjenta (data urodzenia, imię, nazwisko, PESEL itp.)
* badania [study], zawarte w nich są między innymi: dane dotyczące wizyt pacjenta, opis przeprowadzonych badań a także ich wyniki.
* serie danych – gromadzone w nich są informacje uzyskane podczas badania mogą to być np. obrazy uzyskane w czasie badania (w przypadku opisywanego projektu są to dwie sekwencje rezonansu magnetycznego STIR oraz T1).



Rysunek Przykład powiązania danych rzeczywistych z modelem DICOM[3].

## C++ w wersji 17

W kolejnych etapach pracy nad projektem wprowadzony został język C++ jako główne narzędzie programistyczne wykorzystane w implementacji algorytmu działów wodnych. C++ to język programowania ogólnego przeznaczenia, dzięki czemu możliwe jest jego wykorzystanie w różnych dziedzinach naukowych, do rozwiązywania różnorodnych problemów. C++ został zaprojektowany przez Bjarne Stroustrupa jako rozszerzenie języka C o obiektowe mechanizmy abstrakcji danych   
i silną statyczną kontrolę typów. Zachowanie zgodności z językiem C na poziomie kodu źródłowego pozostaje jednym z podstawowych celów projektowych kolejnych standardów języka. W latach 90. XX wieku język ten zdobył pozycje jednego z najbardziej popularnych języków programowania.   
W początkach wieku XXI liczbę programistów C++ szacowano na około 3mln. Język C++ jest standaryzowany przez ISO. Ostatnia, najnowsza z wersji standardu, nazywana C++17, opublikowana została w grudniu 2017 jako ISO/IEC 14882:2017[10]. Pierwszą wersję, C++98, opublikowano w 1998 roku, jako ISO/IEC 14882:1998[11]. Do najważniejszych właściwości języka C++ zaliczyć możemy między innymi:

- wieloparadygmatowość - Oznacza to, że można w nim stosować jednocześnie różne style programowania, w tym programowanie proceduralne, obiektowe, generyczne, jak również programować na poziomie asemblera.

- statyczna kontrola błędów z elementami kontroli dynamicznej

- bezpośrednie zarządzanie pamięcią.

- brak negatywnych wpływów wydajnościowych - Projekt języka zakłada, że żadna nowa (względem języka C) cecha języka C++ nie może mieć negatywnego wpływu na szybkość działania programu   
lub zapotrzebowanie na pamięć operacyjną. Dzięki temu dobrze napisany program w C++ jest   
z reguły co najmniej równie szybki, jak jego odpowiednik napisany w C; co więcej, dzięki możliwości zastosowania algorytmów generycznych w wielu przypadkach C++ jest wyraźnie szybszy od C.   
W wersji C++ 17 wprowadzone zostały następujące (najważniejsze zmiany):

* Instrukcje *if* oraz *switch* z inicjalizatorami zmiennych
* *constexpr if*
* Statyczne składowe *inline*
* Zagnieżdżone przestrzenie nazw
* Wyrażenia lambda w C++17  
  Dedukcja argumentów szablonu klasy
* Wyrażenia *fold*
* *Pack expansion* z deklaracją *using*
* Parametry szablonu z *auto*
* Rozszerzenia C++17 w bibliotece *type\_traits*
* Nowe narzędzia - *std::invoke*, itp.
* Klasa *std::string\_view*
* *std::optional*
* Typ wariantowy *std::variant* i mechanizm wizytacji
* Obiekty dynamiczne - *std::any*
* Typ *std::byte*
* Obsługa plików w C++17
* Nowe funkcji konwersji
* Rozszerzenia w bibliotece wielowątkowości - *std::scopedlock i std::sharedlock*

## PGM

Przenośny format szarej bitmapy pgm (portable graymap) jest to prosty format monochromatyczny. Nadaje się on do przechowywania obrazów zawierających wiele odcieni szarości. W podstawowej wersji tworzony plik jest plikiem tekstowym zawierającym informacje   
o pojedynczym obrazie. Plik zapisany w formacie pgm zawiera w kolejności:

* „Numer magiczny”, określający rodzaj pliku. Numer magiczny tekstowej wersji pliku pgm   
  to dwa pierwsze znaki, ustawione na „P2”.
* Co najmniej jeden znak biały (spacja, tabulacja, CR, LF).
* Szerokość obrazu (W), podana w pikselach jako liczba dziesiętna zapisana za pomocą znaków ASCII.
* Co najmniej jeden znak biały.
* Wysokość obrazu (H), znowu podana w pikselach jako liczba dziesiętna zapisana za pomocą znaków ASCII.
* Co najmniej jeden znak biały.
* Maksymalna wartość szarości (M), również podana jako liczba dziesiętna zapisana za pomocą znaków ASCII.
* Co najmniej jeden znak biały.
* Ciąg W×H wartości zawierających się między 0 a M i zapisanych jako liczby dziesiętne   
  w kodzie ASCII oddzielone od siebie białymi znakami. 0 oznacza czerń, M— biel, pozostałe wartości — odcienie pośrednie. Wartości kodują piksele obrazu począwszy od jego lewego górnego rogu w kierunku zgodnym z kierunkiem normalnego czytania.
* Linia zawierająca informacje o maksymalnej wartości szarości obrazu może być poprzedzona liniami komentarza rozpoczynającymi się znakiem „#” — linie te są ignorowane.
* Linie nie powinny być dłuższe niż 70 znaków

# Pojęcia używane w pracy

**T1W** – sekwencja rezonansu magnetycznego, Najczęściej wykorzystywana jest  
do otrzymywania obrazu mózgu oraz wątroby, jednak znajduje ona również zastosowanie w wykonywaniu zdjęć RM stawów krzyżowo biodrowych [8].

**STIR** - (short TI inversion recovery) – Zwykle wykorzystywana przy obrazowaniu oczodołu oraz narządów jamy brzusznej. Wykorzystywana także przy diagnozowaniu stanów zapalnych stawów. Wraz z sekwencją T1W pozwalają na zdiagnozowanie stanu zapalnego stawu krzyżowo-biodrowego [9].

# Środowisko

## O programie 3D Slicer

3D Slicer jest open-source’ową platformą służącą do przetwarzania oraz wizualizacji obrazów medycznych. Rozwijany przez ponad 20 lat przez szerokie grono deweloperów z całego świata oraz przy wsparciu National Institutes of Health dostarcza darmowe, wieloplatformowe środowisko wspomagające przetwarzanie i analizowanie obrazów medycznych. [1]

## Slicer 3D a przechowywanie obrazu DICOM

Program Slicer udostępnia interfejs do tworzenia rozszerzeń w języku Python, za pomocą odpowiednich modułów możemy pobrać obecnie wczytany w Slicerze obraz w formie skalarnej, za co odpowiada poniższy kod:

|  |
| --- |
| inputVolume = inputSelector.currentNode() vtkImage = inputVolume.GetImageData() vtkData = vtkImage.GetPointData().GetScalars() |
| Przykładowy kod umożliwiający pobranie obrazu z aktualnie wyświetlanego obrazu DICOM |

Gdzie:

* inputSelector – „wejście” programu pobiera dane dotyczące aktualnie wczytanego obrazu
* inputVolume – reprezentacja wczytanego obrazu w standardzie DICOM
* vtkImage – obraz przechowywany we wczytanym pliku DICOM
* vtkData – obraz przechowywany we wczytanym pliku DICOM w formie skalarnej

Za pomocą funkcji:

dim = vtkImage.GetDimensions()

możemy odczytać rozmiary obrazu, funkcja ta zwraca tablicę 3 elementów, które zwracają liczbę punktów na każdej z osi (x, y, z)

W celu sformatowania obrazu do postaci łatwej do przetwarzania można przekonwerterować go   
do tablicy numpy:

|  |
| --- |
| npVolume = vtk.util.numpy\_support.vtk\_to\_numpy(vtkData)  npVolume = npVolume.reshape(dim[2], dim[1], dim[0]) |
| Konwersja obrazu przechowywanego w pliku DICOM na obraz przechowywany w tabeli numpy |

Do tak przygotowanego obrazu możemy się odwoływać poprzez zwykłe operacje na indeksach tablicy trójwymiarowej:

npVolume[indeks przetwarzanego aktualnie obrazu w 2D][współrzędna x][współrzędna y]

Każdy z odczytanych w ten sposób pikseli przechowuje liczbę z zakresu [0, 255], która informuje o jego jasności.

## CLion

CLion to zintegrowane środowisko programistyczne (IDE) obsługujące język C oraz C++ oraz wspierające JavaScript, XML, HTML i CSS. Posiada szereg narzędzi i funkcji niezbędnych   
do efektywnej pracy jak graficzny debugger GDB, moduł do analizy kodu źródłowego. CLion został zintegrowany z systemem kontroli wersji Mercurial, Subversion, Git, CVS oraz Perforce. Edytor kodu obsługuje podkreślanie składni, tworzenie snippetów czy refaktoryzację. Środowisko posiada również wbudowany emulator terminala, oraz obsługę CMake. CLion obsługuje między innymi kompilatory GCC, clang oraz MinGW. CLion pozwala również na wygodne przeszukiwanie i eksplorowanie zawartości poszczególnych klas. Nie zabrakło także systemu automatycznie generującego różne konstrukcje jak konstruktory/destruktory, gettery/settery, równości, nierówności, relacje   
czy domykanie klamrą. Wszystko to przy użyciu prostych skrótów klawiszowych.

## MinGW

MinGW (Minimalist GNU for Windows) – port GCC, dostarczający darmowe i otwarte środowisko oraz narzędzia pozwalające na kompilację natywnych plików wykonywalnych   
dla platformy Windows. Podstawowymi składnikami MinGW są kompilatory GCC, zestaw programów binutils oraz zestaw plików nagłówkowych i bibliotek charakterystycznych dla platformy Windows. MinGW pozwala na kompilację kodu źródłowego napisanego w językach programowania C++, C, Fortran77, Java, Ada oraz Objective C do kodu wykonywalnego (EXE) lub, zależnie od potrzeb, plików obiektowych, bibliotek statycznych lub bibliotek dynamicznych (DLL). Przy użyciu MinGW można kompilować większość programów przeznaczonych dla systemu Windows   
i wykorzystujących różne zaawansowane biblioteki, zarówno komercyjne, jak i typu Open Source,   
np. w32API, Qt, DirectX, OpenGL, SDL czy wxWidgets. Programy skompilowane z użyciem MinGW do komunikacji z systemem operacyjnym wykorzystują bezpośrednio standardowe biblioteki dynamiczne systemu Windows i nie używają emulacji standardu POSIX. Ponadto MinGW korzysta   
z bibliotek Microsoft Visual C++ Runtime, a nie GNU C Library. Są to podstawowe cechy odróżniające MinGW od Cygwina. MinGW może być też używany do kompilowania i przenoszenia do systemu Windows bibliotek napisanych dla innych systemów operacyjnych, np. Linuksa, w tym bibliotek z projektu GNU.

# Prace początkowe

## Cel projektu

Celem projektu był projekt oraz implementacja rozwiązania pozwalającego na przetestowanie różnych metod segmentacji kości w obrazie rezonansu magnetycznego. Zakładaliśmy zaimplementowanie dwóch wybranych metod w środowisku Slicer 3D z wykorzystaniem języka programowania Python. Metodami wybranymi do testów były: algorytm działów wodnych   
oraz metoda baloników. Zadaniem programu miało być przeprowadzenie algorytmu segmentacji kości tworzących staw krzyżowo biodrowy, które to następnie miały być analizowane w ramach dalszych prac badawczych związanych z praca naukową. Implementacja wybranych metod w środowisku Slicer 3D odbywała się z wykorzystaniem bibliotek dostępnych jako moduły środowiska programistycznego.

## Metoda działów wodnych

Dział wodny (wododział, watershed) to granica pomiędzy zlewiskami (catchment basin) różnych rzek lub zbiorników wodnych (obszarami, z których opady wodne spływają do wspólnego zbiornika

wodnego). Wododziały są zawsze umiejscowione na grzbietach funkcji wysokości terenu.   
Linie rozgraniczającą zlewiska nazywamy wododziałami; Grzbiet to punkt, który nie jest pełnym

maksimum, ale w maksimum przynajmniej w jednym kierunku (w N-1 wymiarach). Oznacza to, iż są:

•maksimami w przypadku sygnału 1D

•liniami w przypadku sygnału 2D (rzeczywista rzeźba terenu)

•powierzchniami w przypadku funkcji 3D

W analizie i przetwarzaniu obrazów pod pojęciem działu wodnego rozumie się Operację morfologiczną na obszarze, którego centrum stanowi lokalne minimum w obrazie (jeśli kropla wody upuszczona na obraz spłynęłaby do lokalnego minimum, to granica oddziaływania grawitacyjnego na kroplę wyznaczałaby właśnie linię działu wodnego. Obszary o małej intensywności stanowią   
w obrazie lokalne „doliny”, zaś obszary o dużej intensywności – lokalne „wzniesienia” . Jako wysokość terenu przyjmuje się amplitudę gradientu intensywności pikseli lub samą intensywność,   
za wododział zaś – grzbiety tych funkcji. Liczba wyodrębnianych obszarów może być dobierana automatycznie lub deklarowana przez operatora.

## Metoda baloników (metoda ustalania poziomu)

Metoda ustalania poziomu – metoda segmentacji obrazu po raz pierwszy zaproponowana przez Oshera i Sethiana do segmentacji fal oceanicznych i płonącego ognia. Malladi następnie zastosował   
go do segmentacji obrazów medycznych. Metoda zakłada zastosowanie krzywych 2D na przestrzeni obrazów 3D. Metoda ustalania poziomu może być skutecznie wykorzystana do rozwiązywania problemów z topologią podczas ewolucji krzywych, podczas gdy poprzednie algorytmy nie mogą sobie z nimi radzić. W ostatnich latach istnieje wiele algorytmów segmentacji obrazu opartych   
na metodzie ustalania poziomu. W przypadku różnych zastosowań przetwarzania obrazu ludzie przedstawili odpowiednie rozwiązania, a duża liczba badaczy nadal poprawia i zwiększa efektywność i skuteczność tych algorytmów.

## Implementacja rozwiązania w środowisku Slicer 3D

Prace związane z przygotowaniem rozwiązania w języku Python trwały przez około 3 miesiące. Niestety poziom skomplikowania samego środowiska jak i bibliotek pozwalających na rozwiązanie postawionego problemu uniemożliwiły dokończenie projektu w formie zaproponowanej na początku prac nad projektem. W celu pozytywnego ukończenia implementacji wymagania musiały zostać zmienione, tak aby możliwe było pomyślne zakończenie prac oraz uzyskanie miarodajnych wniosków pozwalających na dalsze prace badawcze.

## Nowa koncepcja prac

## Cel projektu

Nowym celem projektu jest projekt oraz implementacja programu pozwalającego   
na przeprowadzenie segmentacji kości z obrazu pgm przy użyciu metody działów wodnych. Zmienione zostały zarówno dane wejściowe (DICOM -> PGM) jak i narzędzia użyte   
do przygotowania rozwiązania. Środowisko programistyczne zostało zmienione z Slicer 3D na CLion, z doinstalowanym kompilatorem MinGW oraz CMake. Został zmieniony również język programowania z Python na C++ w wersji 17. Wszystkie te narzędzia zostały szczegółowo pisane   
w poprzednich rozdziałach. Wszystkie te zmiany zostały wprowadzone ze względu na lepszą znajomość podanego środowiska oraz języka programowania, co ostatecznie przełożyło się na szybsze oraz efektywniejsze przeprowadzenie prac związanych z implementacja rozwiązania. W trakcie ustalania nowej koncepcji projektu odrzucona została możliwość wykorzystania bibliotek pomocniczych implementujących metodę działów wodnych. W tej wersji projektu chcieliśmy sami zaimplementować wybrana metodę w taki sposób w jaki rozumiemy jej działanie, oraz przetestować jej przydatność w projektowaniu całego komponentu.

## Opis rozwiązania

Rozwiązanie problemu segmentacji kości z obrazu pgm wykonywane jest etapowo. Każdy z nich odpowiada za pewna część przygotowania obrazu wejściowego do wykonania na nim algorytmu działów wodnych. Pierwszym z etapów jest kilkukrotne zastosowanie filtru Gaussa na obrazie wejściowym. Zabieg taki ma na celu zmniejszenie, a w najlepszym wypadku eliminacje szumu pojawiającego się na obrazie wejściowym. Ilość przebiegów filtru Gaussa definiowane jest przez użytkownika programu.

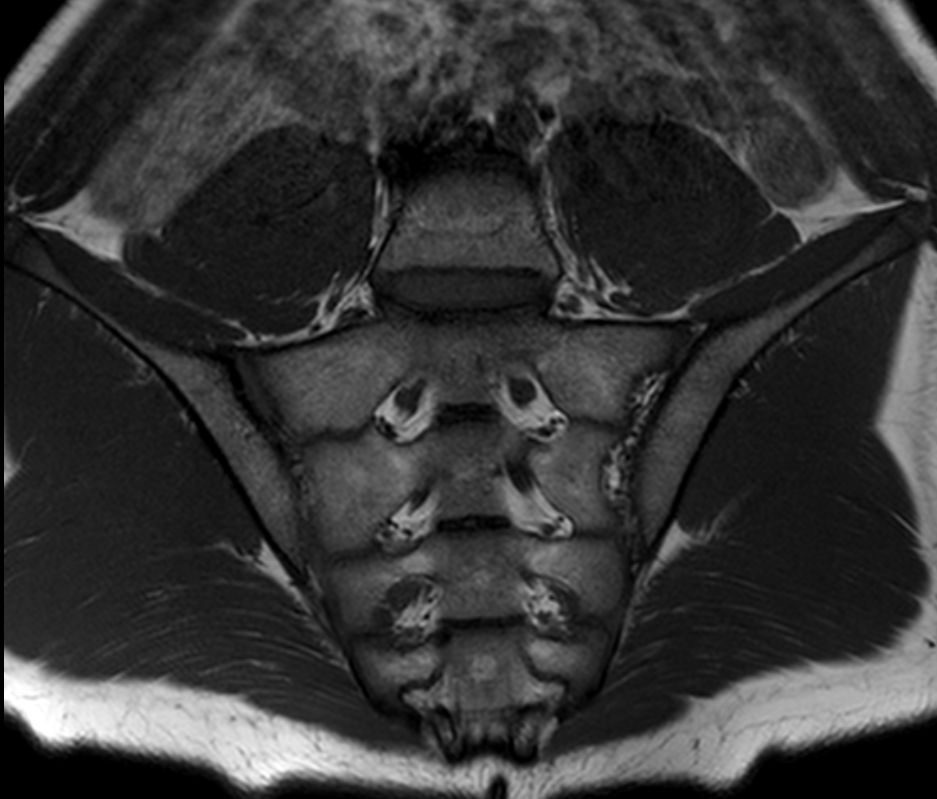
Kolejnym krokiem jest przygotowanie obrazu różnicowego w celu uzyskania konturów kości. Wynik taki możemy uzyskać poprzez wykonanie na obrazie przefiltrowanym procesu dylatacji   
oraz erozji. Następnie od wyniku dylatacji odejmowany jest wynik erozji. Dzięki temu w wyniku dostajemy kontury kości tworzących staw krzyżowo biodrowy.

Następnie obraz poddawany jest filtracji przy pomocy filtru Gaussa. Etap ten jest konieczny   
ze względu na szum powstały w wyniku przeprowadzenia operacji uzyskania obrazu różnicowego. Końcowym etapem przygotowania obrazu jest jego 10-krotne rozjaśnienie. Poprawia to znacząco wyrazistość konturów, dzięki czemu algorytm właściwy działa dokładniej.

Ostatnim etapem przetwarzania obrazu jest wykonanie procesu segmentacji obszarów kości   
przy pomocy algorytmu działów wodnych. Metoda rozpoczyna swoje działanie w miejscu określonym przez użytkownika za pomocą współrzędnych (x,y). Algorytm wykonuje zalewanie piksel po pikselu obrazu aż do miejsca w którym jasność piksela zalewanego nie przekroczy wartości ustalonej przez treshold. Dodatkowo sprawdzane jest sąsiedztwo zalewanego piksela. Jeżeli nie ma on żadnych zalanych pikseli to sam także nie może zostać zalany poprzez algorytm działów wodnych. Wyklucza to możliwość przekroczenia granicy zalewania, dzięki czemu mamy pewność, że wysegmentowane zostaną tylko te obszary które nas interesują. Algorytm przebiega sprawnie. Cały program wykonuje się w kilka sekund, co pozwala na szybkie przetestowanie jego działania.

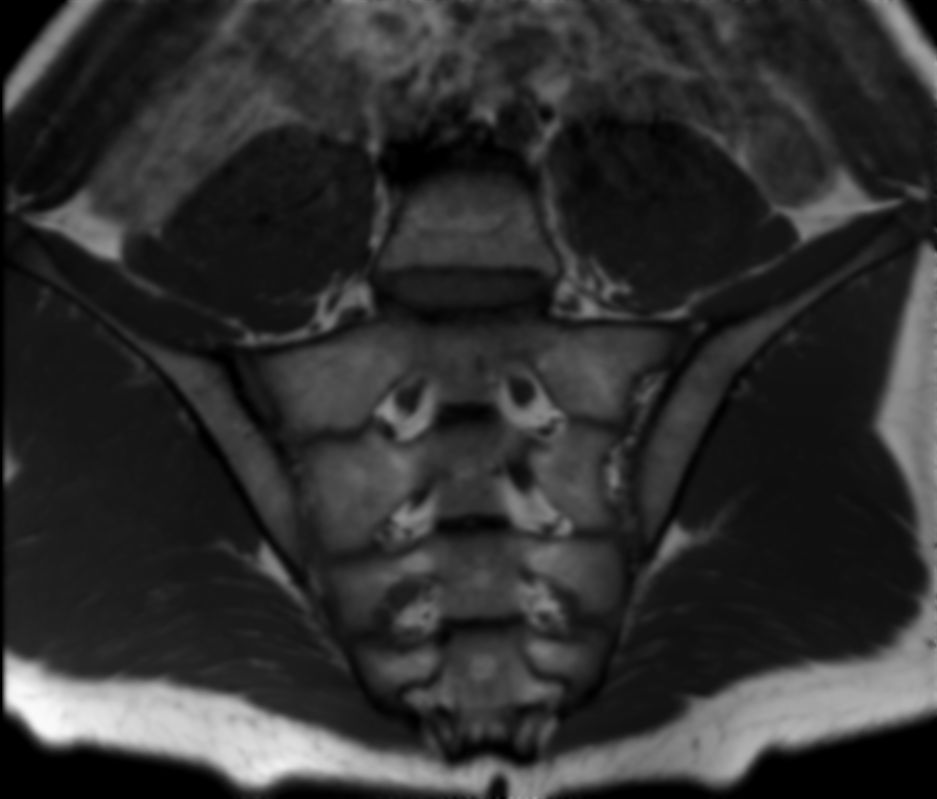
## Testy

Po zaimplementowaniu programu, przyszedł czas na najistotniejszą część projektu jaką jest testowanie wykonanego rozwiązania. Etap testów był niezwykle ważny, ponieważ odpowiadał   
na podstawowe pytanie: czy możliwe jest wykorzystanie algorytmu działów wodnych w procesie segmentacji kości stawu krzyżowo-biodrowego z obrazu rezonansu magnetycznego.   
Opis poszczególnych testów został przedstawiony poniżej. Do ustalania parametrów wejściowych wybrany został obraz wejściowy widoczny na poniższym zrzucie ekranu.

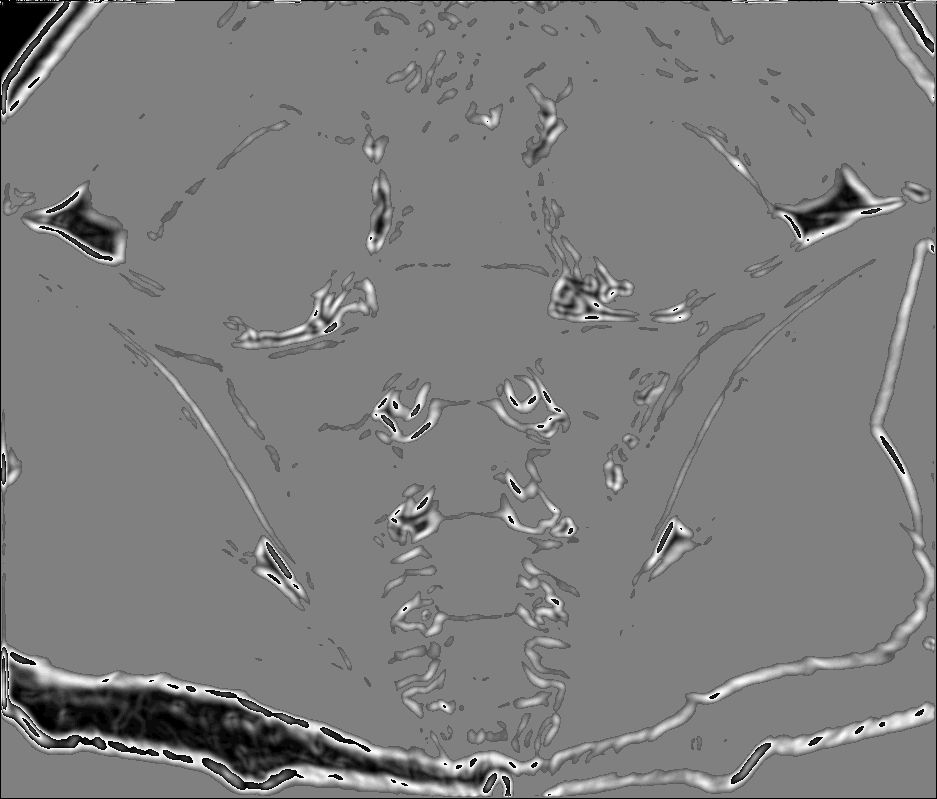


Rysunek Testowy obraz wejściowy.

## Dobór liczby iteracji filtru Gaussa

Pierwszym etapem testowania rozwiązania był dobór odpowiedniej liczby przebiegów filtru Gaussa. Dobrze dobrany parametr pozwoli na eliminację szumu występującego w obrazie, jednocześnie nie pozwalając na zbytnie rozmycie krawędzi kości, które mogłoby doprowadzić  
do błędnego działania programu. Jeżeli parametr byłby zbyt niski obraz wyjściowy w dalszym ciągu byłby mocno zaszumiony. Z drugiej jednak strony zbytnie zwiększanie liczby iteracji funkcji Gaussa sprawi ze obraz wyjściowy będzie niedokładny, co z punktu widzenia dalszych badań   
jest niedopuszczalne. Z tego względu należało znaleźć „złoty środek” pozwalający na uzyskanie zadowalającego efektu. Testy polegały na ustawieniu stałego tresholdu na poziomie 80   
oraz wywoływaniu programu z kolejnymi iteracjami filtru Gaussa od 0 do 10. Następnie dokonano analizy obrazów przefiltrowanych oraz wynikowych, na podstawie której wysnute zostały wnioski. Ostatecznie do dalszych prac przyjęto liczbę sześciu iteracji filtru Gaussa jako najbardziej optymalną. W dokumentacji zawarto jedynie obrazy przedstawiające wynik filtrowania oraz segmentacji   
dla optymalnej wartości liczby iteracji filtru Gaussa. Wszystkie obrazy dostępne są w katalogu TEST1. 

Rysunek Obraz po procesie 6-krotnej filtracji.



Rysunek Obraz wynikowy.

## Poszukiwanie najlepszej wartości tresholdu

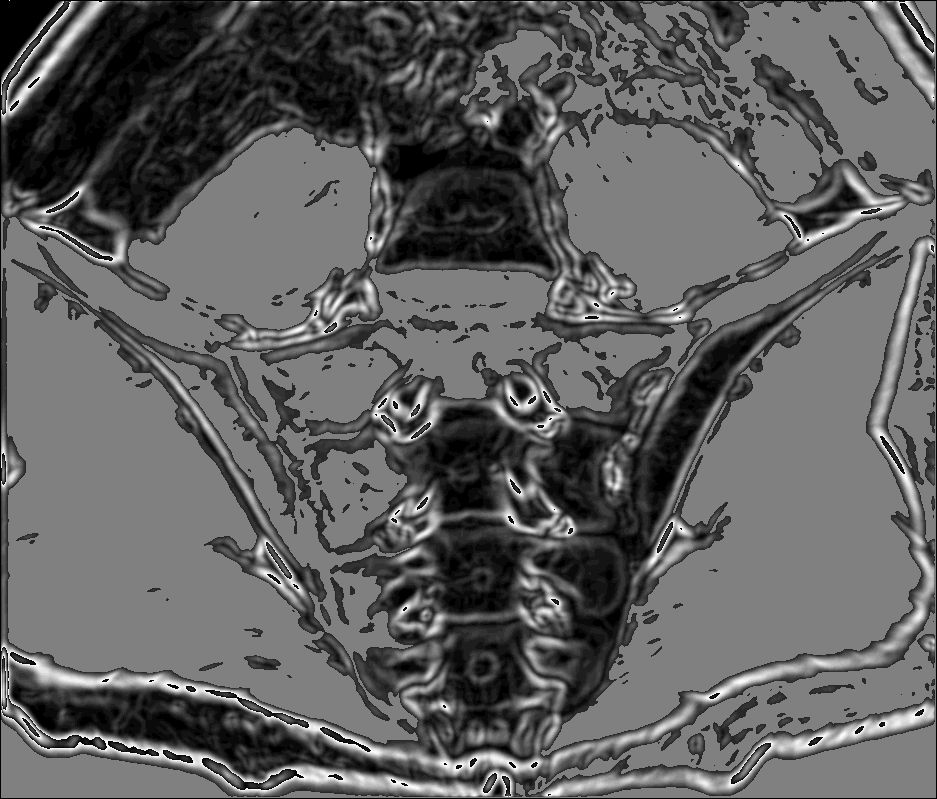
W kolejnym kroku testów skupiliśmy się na dobraniu odpowiedniego poziomu tresholdu.   
To właśnie na tym etapie doszliśmy do najważniejszych wniosków krytycznych dla całego projektu. W fazie testowania poziomu zalewania, sprawdziliśmy poprawność działania algorytmu.   
Dzięki testom oraz poklatkowemu zapisowi plików pośrednich udało się uzyskać odpowiedź   
na pytanie czy algorytm działów wodnych może mieć zastosowanie w tego typu obrazach z jakimi mamy do czynienia. Niestety odpowiedź jest negatywna a jej wyjaśnienie znajduje się w następnym rozdziale. Testy poziomu zalewania polegały na stopniowym zalewaniu obrazu wejściowego   
od poziomu 0 aż do 100, z krokiem 2. Każdy obraz pośredni zostaje także zapisany do pliku. Dzięki temu możemy stwierdzić który z poziomów jest dla nas dostateczny. Na poniższych slajdach pokazane zostało kilka przykładowych obrazów pozwalających na obserwację procesu zalewania obrazu.   
Więcej przykładów wraz z animacją zalewania przykładowego obrazu od 2 do 100 znajduje się   
w katalogu TEST2.



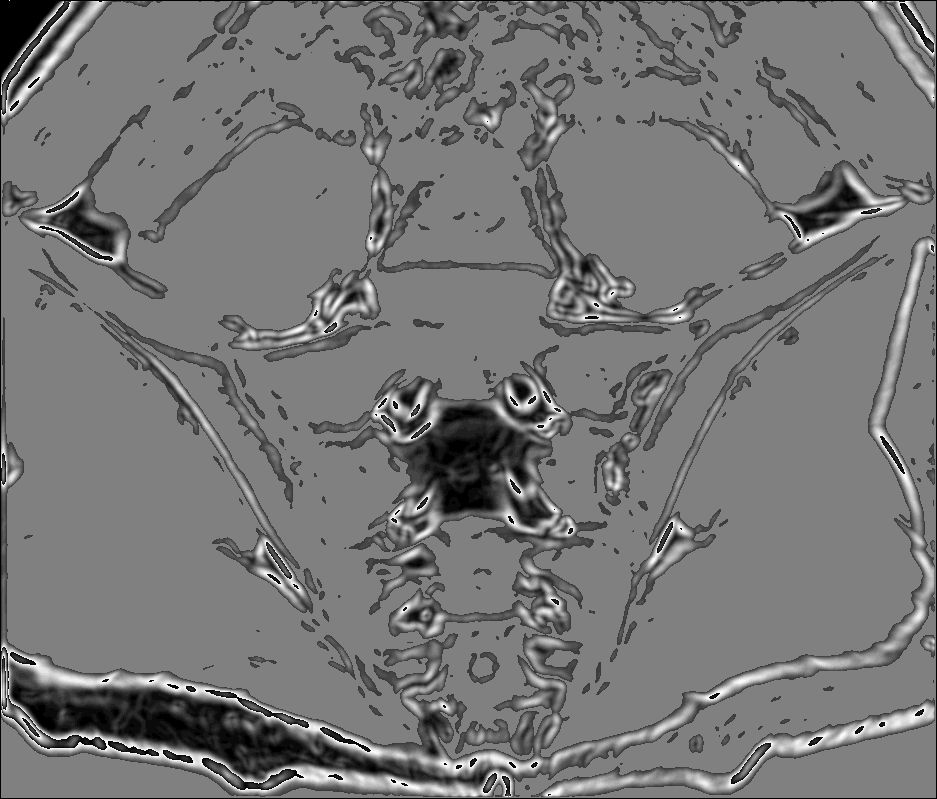
Rysunek Obraz przed zalewaniem z zaznaczonym punktem startowym.



Rysunek Poziom zalania=28.



Rysunek Poziom zalania=30.



Rysunek Poziom zalania=56.

## Wybór punktów startowych

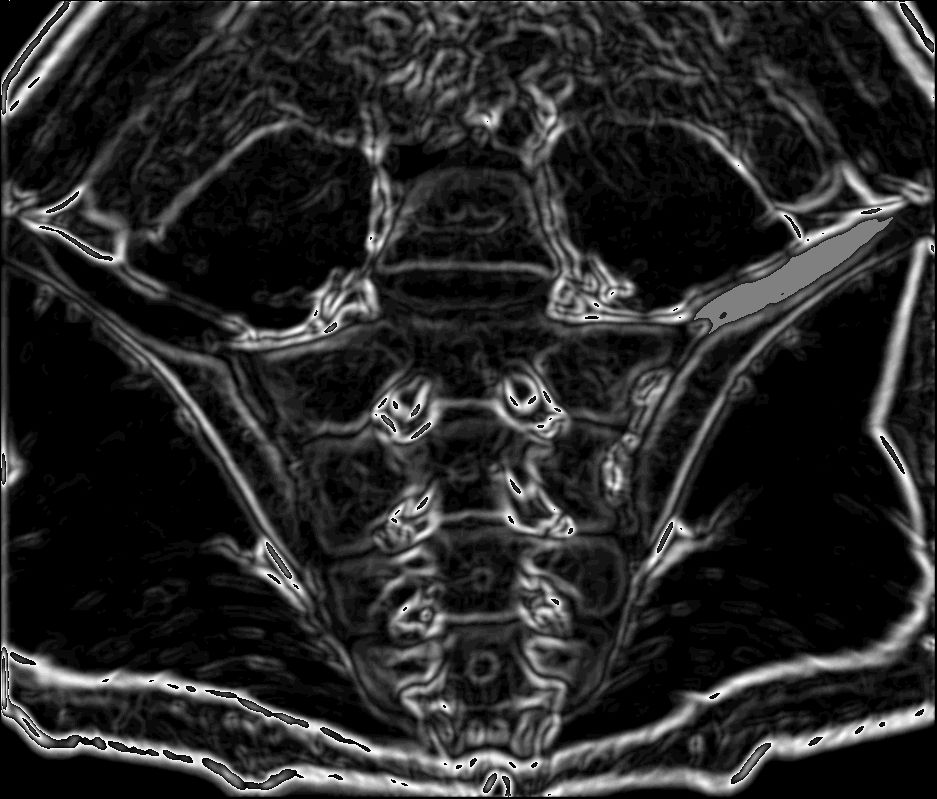
Ważnym elementem testów był wybór miejsca w którym algorytm ma rozpocząć swoją pracę. Oczywiście dla każdej osobnej kości powinniśmy przeprowadzić osobną segmentację gdyż algorytm   
z założenia nie powinien rozlać się poza krawędź jednej kości. W tym celu wybraliśmy kilka punktów dla których przeprowadziliśmy proces segmentacji. Wyniki testów przedstawione zostały na kilku zrzutach ekranu.



Rysunek Punkt startowy w lewej dolnej kości.



Rysunek Punkt startowy w lewej górnej kości.



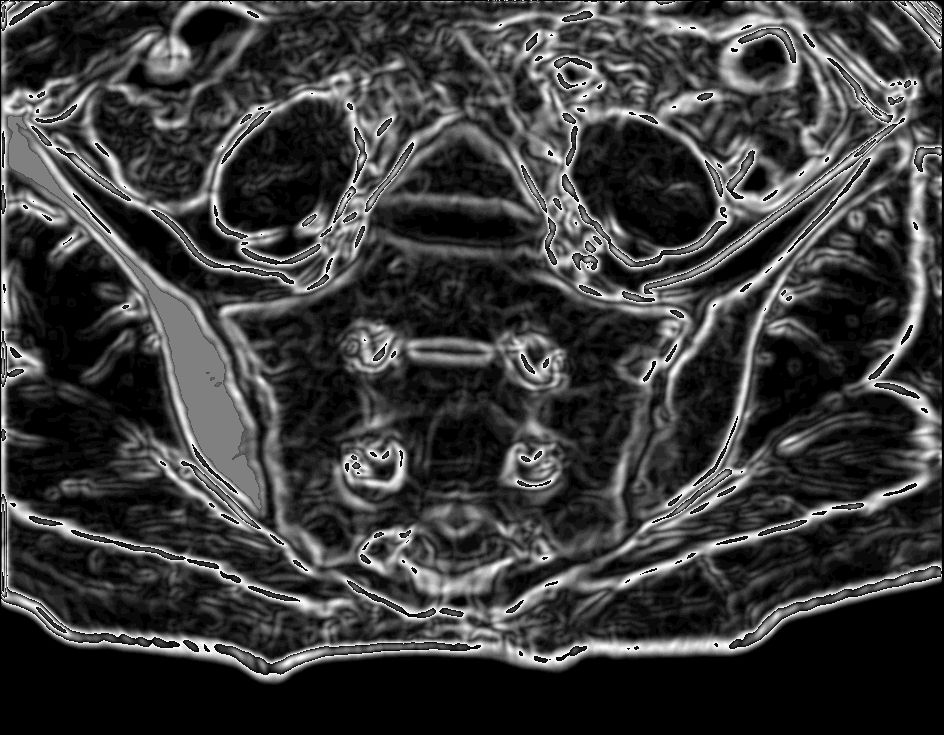
Rysunek Punkt startowy w prawej kości.



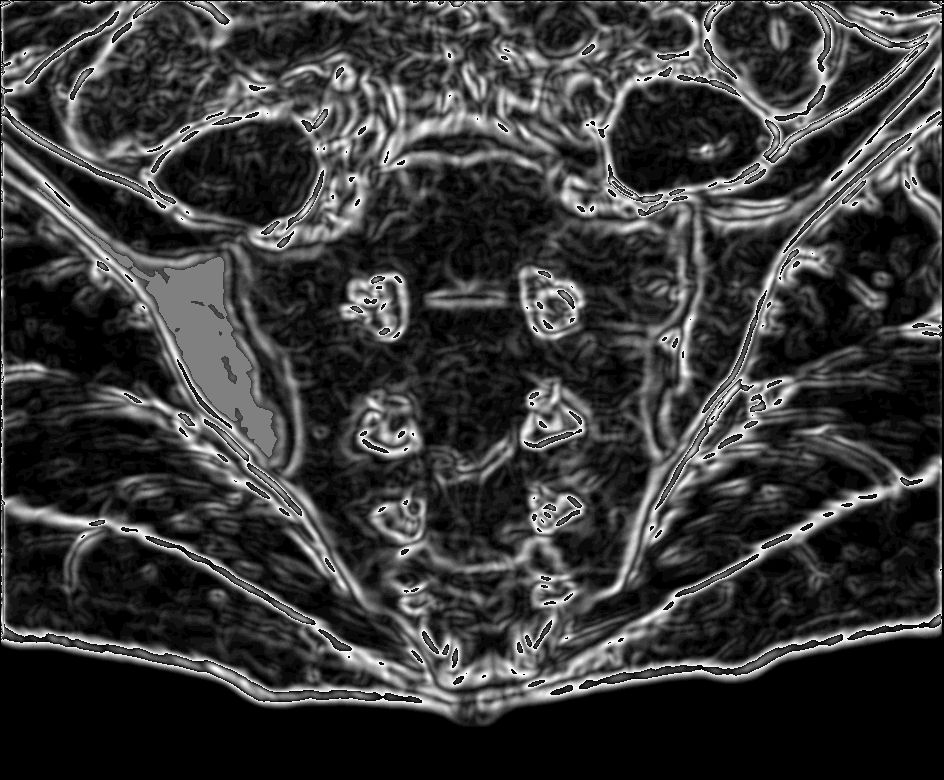
Rysunek Punkt startowy w kości kręgosłupa.

## Porównanie wyników na innych obrazach

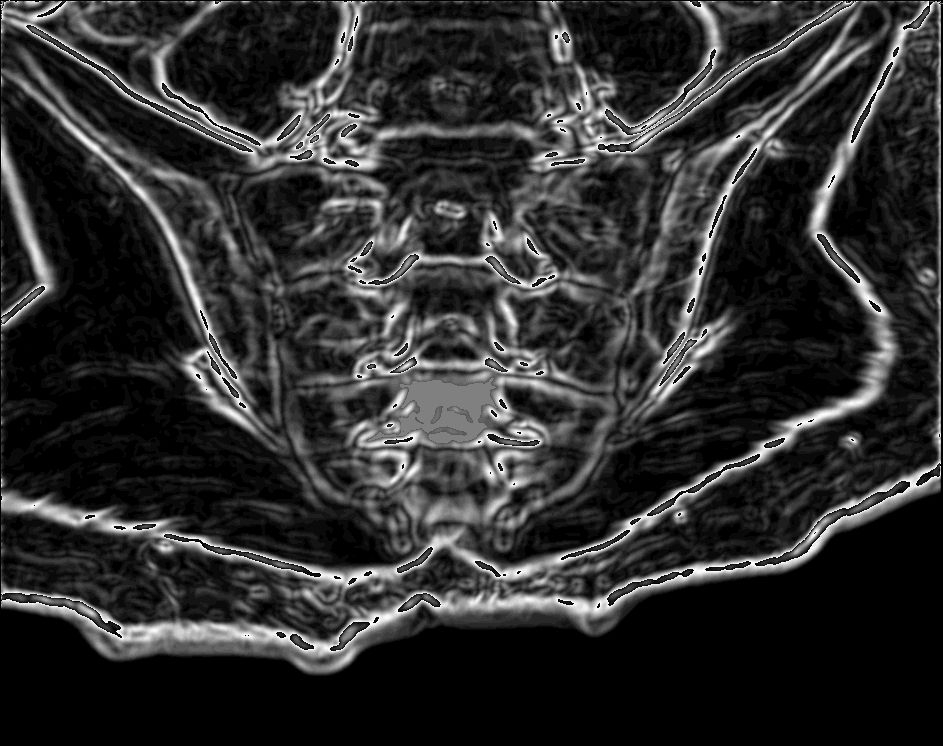
Ostatnim testem było wykonanie segmentacji z wykorzystaniem metody działów wodnych na innych obrazach testowych. W tym celu przygotowany został zestaw 20 kolejnych obrazów które zostały poddane segmentacji. Wyniki kilku z nich zostaną zaprezentowane poniżej. Całość materiału dostępna jest w katalogu TEST4.



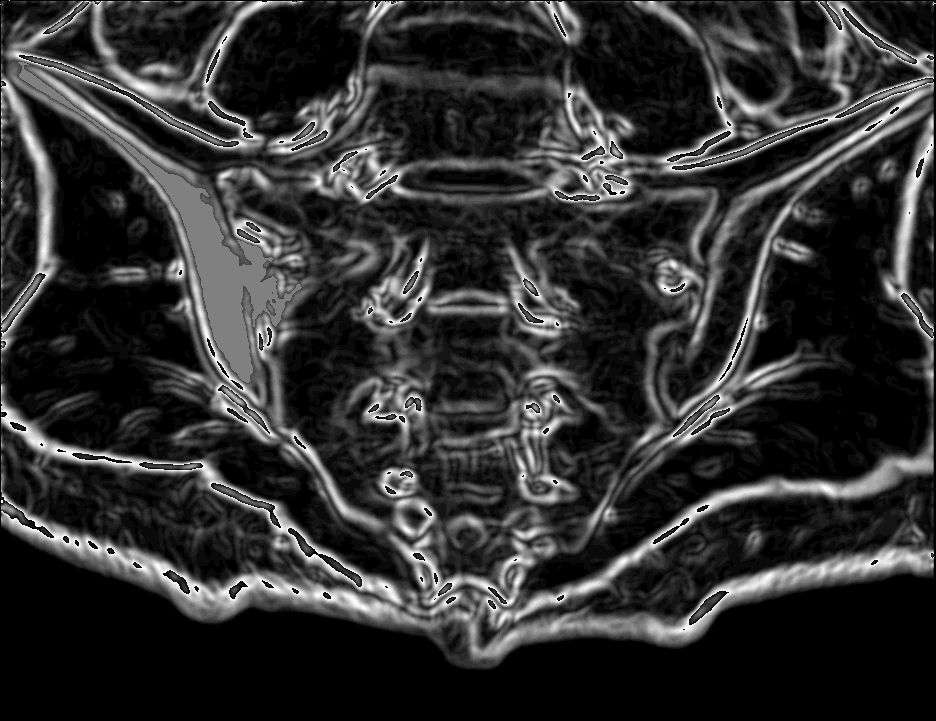
Rysunek Obraz po segmentacji (1).



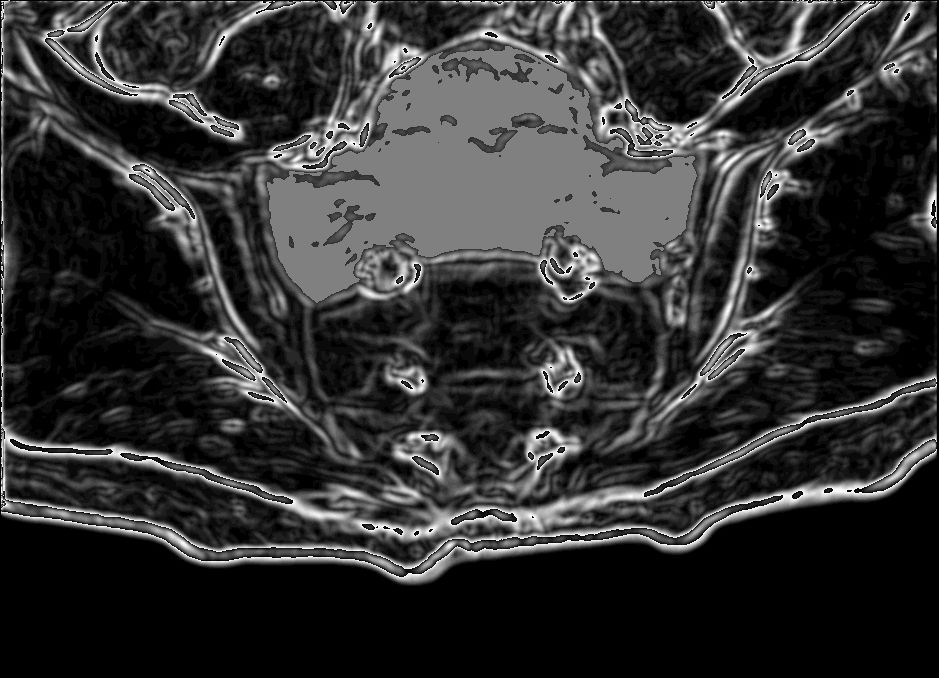
Rysunek Obraz po segmentacji (2).



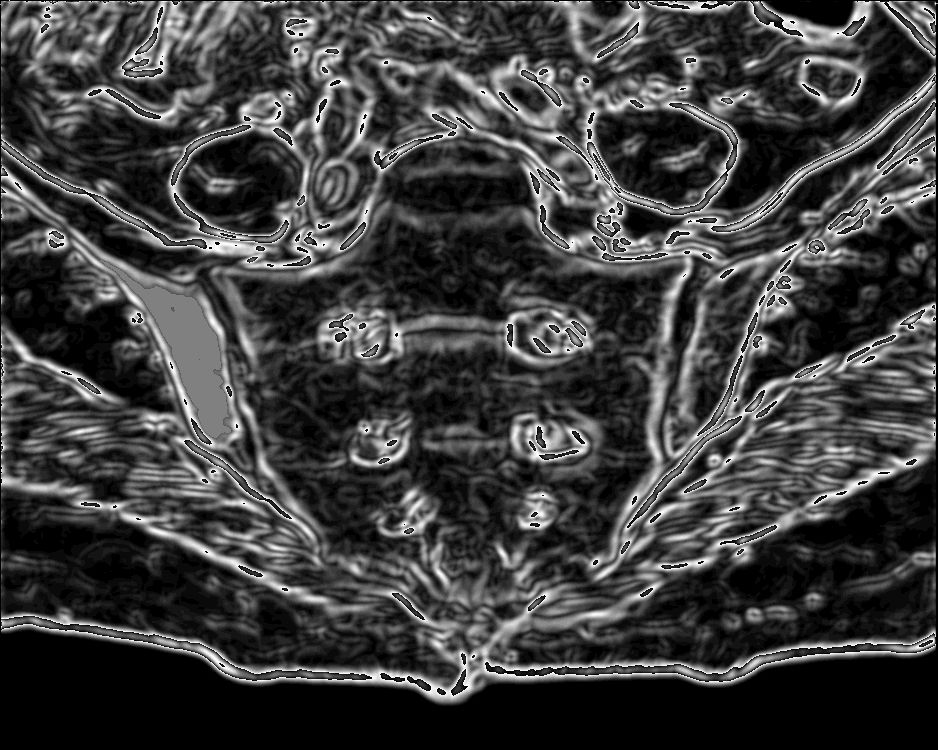
Rysunek Obraz po segmentacji (3).



Rysunek Obraz po segmentacji (4).



Rysunek Obraz po segmentacji (5).

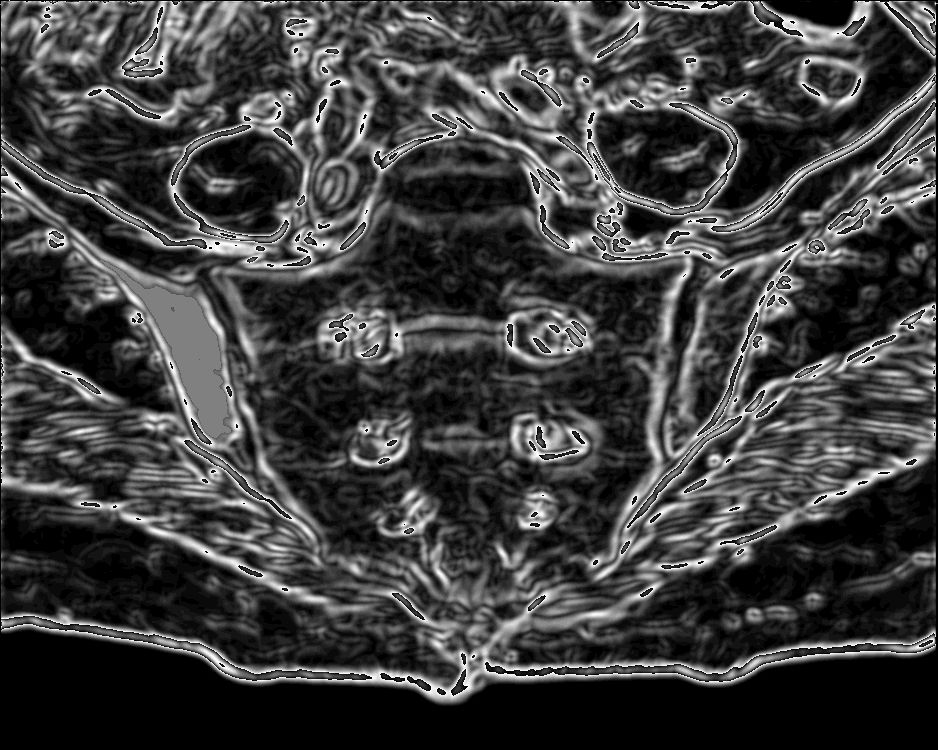


Rysunek Obraz po segmentacji (6).

## Wnioski

Po przeprowadzonych testach których część wyników można było zaobserwować w poprzednim punkcie możemy dojść do pewnych wniosków, których omówienie pomogło nam w ukierunkowaniu dalszych prac badawczych związanych z przetwarzaniem obrazu rezonansu magnetycznego. Pierwszy wniosek który mocno nasuwa się po kilkukrotnym odpaleniu programu na różnych obrazach wejściowych jest oczywisty. Metoda działów wodnych nie znajduje zastosowania w przypadku obrazu rezonansu magnetycznego, gdyż większość próbek podanych na wejście programu zwraca błędny obraz wynikowy. Pomimo zmiany parametrów tresholdu, punktu startowego oraz ilości przebiegów filtru gaussa, nie jest możliwe przeprowadzenie poprawnej segmentacji kości.

Kolejny wniosek związany jest z samą metodą działów wodnych. Na pewnych obrazach widoczne jest prawidłowe jej działanie co pokazują również dołączone do projektu pliki gif. Udowadnia   
to poprawną implementacje oraz działanie przedstawionego algorytmu. Dzięki przeprowadzonym testom możliwe było stwierdzenie czy zadanie implementacyjnie zostało wykonane z należytą dokładnością. Dokładny opis testów znajduje się w poprzednim punkcie. Na poniższym zrzucie prezentowany jest obraz prawidłowo zalanego obszaru.

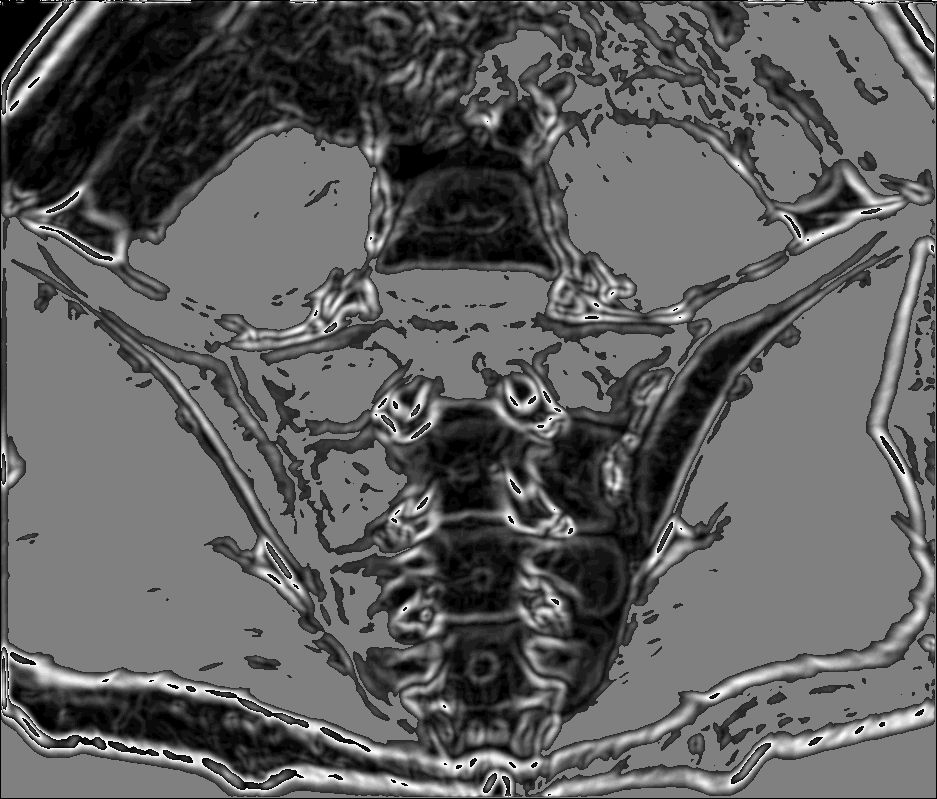


Rysunek Poprawnie zalany obszar.

Algorytm został poprawnie zaimplementowany oraz przetestowany. Jednak w swoim działaniu nie może zostać wykorzystany do obrazów rezonansu magnetycznego. Jest to wiązane   
z niedokładnymi konturami kości. Jeżeli kontur kości zostanie w którymkolwiek miejscu przerwany, algorytm zacznie się rozlewać poza obszar kości. Można to zaobserwować na rysunkach poniżej gdzie na pierwszym z nich zalewana jest kość, a gdy podniesiemy poziom jasności o kolejny stopień, algorytm rozleje nam się na większość rysunku przekraczając granice kości.



Rysunek Obraz częściowo zalany.



Rysunek Obraz przelany (wyciek).

## Podsumowanie

Projekt mimo komplikacji związanych z trudnościami implementacyjnymi w środowisku Slicer 3D udało się doprowadzić do końca czego efektem jest działające narzędzie pozwalające na segmentację metodą działów wodnych obrazów w skali szarości. Niestety pomimo przedstawienia działającego programu, niemożliwe jest jego wykorzystanie w dalszych pracach. Jego działanie nie jest poprawne dla zaproponowanych danych wejściowych, co nie pozwala na jego dalszy rozwój w kontekście przetwarzania obrazów rezonansu magnetycznego. Kolejnym etapem prac badawczych będzie zatem implementacja kolejnej metody segmentacji kości w oparciu o już zdobyte doświadczenie.

# Spis rysunków

[Rysunek 1 Przykład powiązania danych rzeczywistych z modelem DICOM[3]. 5](#_Toc524081446)

[Rysunek 2 Testowy obraz wejściowy. 12](#_Toc524081447)

[Rysunek 3 Obraz po procesie 6-krotnej filtracji. 13](#_Toc524081448)

[Rysunek 4 Obraz wynikowy. 14](#_Toc524081449)

[Rysunek 5 Obraz przed zalewaniem z zaznaczonym punktem startowym. 15](#_Toc524081450)

[Rysunek 6 Poziom zalania=28. 15](#_Toc524081451)

[Rysunek 7 Poziom zalania=30. 16](#_Toc524081452)

[Rysunek 8 Poziom zalania=56. 16](#_Toc524081453)

[Rysunek 9 Punkt startowy w lewej dolnej kości. 17](#_Toc524081454)

[Rysunek 10 Punkt startowy w lewej górnej kości. 17](#_Toc524081455)

[Rysunek 11 Punkt startowy w prawej kości. 18](#_Toc524081456)

[Rysunek 12 Punkt startowy w kości kręgosłupa. 18](#_Toc524081457)

[Rysunek 13 Obraz po segmentacji (1). 19](#_Toc524081458)

[Rysunek 14 Obraz po segmentacji (2). 19](#_Toc524081459)

[Rysunek 15 Obraz po segmentacji (3). 20](#_Toc524081460)

[Rysunek 16 Obraz po segmentacji (4). 20](#_Toc524081461)

[Rysunek 17 Obraz po segmentacji (5). 21](#_Toc524081462)

[Rysunek 18 Obraz po segmentacji (6). 21](#_Toc524081463)

[Rysunek 19 Poprawnie zalany obszar. 22](#_Toc524081464)

[Rysunek 20 Obraz częściowo zalany. 23](#_Toc524081465)

[Rysunek 21 Obraz przelany (wyciek). 23](#_Toc524081466)

# Bibliografia

1. *3DSlicer* [online], 2018 [dostęp 4.06.2018], dostępny w Internecie: https://www.slicer.org/
2. *DICOM Standard* [online], 2018 [dostęp 4.06.2018], dostępny w Internecie: https://www.dicomstandard.org/
3. *DICOM Digital Imaging and Communications in Medicine,   
   standard cyfrowej wymiany informacji medycznych* [online]. 2005 [dostęp 4.06.2018], dostępny w Internecie:https://www.google.com/url?sa=t&rct=j&q=&esrc=s&source=web&cd=6&ved=0ahUKEwirrbOzoLrbAhUsK8AKHV7LCp8QFghXMAU&url=http%3A%2F%2Fhome.agh.edu.pl%2F~socha%2Fpmwiki%2Fpmwiki.php%2FDICOM%2FCaly&usg=AOvVaw2js-iGbBwZctXbeTYfMZ-e
4. *O języku Python* [online], 2008 [dostęp 19.10.2017], dostępny w Internecie: https://pl.python.org/o,jezyku,python.html
5. *Co to jest Python i do czego można go użyć*, 2009 [dostęp 19.10.2017], dostępny w In-ternecie: http://www.python.rk.edu.pl/w/p/python-co-jest-i-do-czego-mozna-go-uzyc/
6. *Bytecode* [online], 2005 [dostęp 13.12.2017], dostępny w Internecie: http://whatis.techtarget.com/definition/bytecode
7. Patka J., *Podręcznik programowania w języku Python* [online], [dostęp: 28.11.2017] http://astronomia.zagan.pl/pliki/python/Podrecznik\_Pythona.pdf
8. *T1-weighted images* [online], [dostęp 4.06.2018], dostępny w Internecie: https://medical-dictionary.thefreedictionary.com/T1W
9. *Short Tau Inversion Recovery* [online], [dostęp 4.06.2018], dostępny w Internecie: <https://radiopaedia.org/articles/short-tau-inversion-recovery>
10. [ISO/IEC 14882:2017](https://www.iso.org/standard/68564.html) (ang.). International Organization for Standardization. [dostęp 2017-12-18].
11. [ISO/IEC 14882:1998](http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?ics1=35&ics2=60&ics3=&csnumber=25845) (ang.). International Organization for Standardization. [dostęp 2017-12-18].