



**WYŻSZA SZKOŁA INFORMATYKI I
UMIEJĘTNOŚCI Wydział Informatyki i Za-
rządzania KIERUNEK: Informatyka**

PRACA DYPLOMOWA INŻYNIERSKA

Algorytm modelowania geometrii
drzew o skrzeliowych w przestrzeni 3D

Imię i nazwisko: Kacper Pluta
Studia: pierwszego stopnia niestacjonarne
Specjalność: programowanie i bazy danych Nr albumu: 18942
Promotor: dr M. Janaszewski

Rok akademicki 2012/2013

Spis treści

1 Wstęp	5
1.2 Słownik pojęć związanych z topologią dyskretną.....	7
1.3 Układ pracy.....	10
2 Ilościowy opis oskrzeli na bazie obrazów tomograficznych 3D	11
2.1 Znaczenie prac nad algorytmami ilościowej analizy drzew oskrzelowych	12
2.2 Proces pomiaru lokalnych parametrów oskrzeli.....	13
3 Wprowadzenie do modelowania drzew oskrzelowych.....	14
3.1 Charakterystyka problemu testowania algorytmów ilościowej analizy drzew oskrzelowych.....	16
3.2 Główne cechy modelu rozszerzonego.....	17
4 Koncepcja podstawowego, trójwymiarowego modelu drzewa oskrzelowego.....	19
4.1 Algorytm generujący model podstawowy drzewa oskrzelowego	20
4.1.1 Reguły algorytmu podstawowego	21
5 Rozszerzony algorytm modelowania drzewa oskrzelowego.....	28
5.1 Wygięcia gałęzi.....	28
5.2 Zniekształcenia modelu wprowadzone w domenie grafiki objętościowej.....	30
6 Szczegółы implementacyjne	34
6.1 Generowanie płaszczyzny wyznaczającej granice drzewa	35
6.2 Gałęzie drzewa.....	36
6.2.1 Generowanie kolejnych rozwidleń	38
6.3 Podział obszaru, w którym znajduje się gałąź.....	39
6.4 Wykrywanie opuszczenia regionu przez gałąź.....	40
6.5 Konwersja modelu opisanego siatką wielokątową do grafiki objętościowej.....	41

6.6 Implementacja znieksztalceń w domenie grafiki powierzchniowej	44
6.7 Interfejs użytkownika.....	45
7 Wykorzystane narzędzia informatyki	47
7.1 Środowisko developerskie.....	47
7.1.2 Narzędzia służące do budowania projektu	48
7.1.3 Narzędzia ułatwiające znalezienie wadliwego kodu.....	50
7.1.4 System kontroli wersji.....	50
7.1.5 Narzędzia wspomagające generowanie dokumentacji.....	51
7.2.1 Wybrane operatory pakietu PINK użyte do zweryfikowania poprawności uzyskanych wyników.....	52
7.2.2 Wizualizacja weryfikowanych danych.....	52
8 Wizualizacja stereoskopowa	53
8.1 Technika anaglif	53
8.2 Technika side-by-side	54
9 Rezultaty komputerowych eksperymentów	56
9.1 Badanie topologii modelu rozszerzonego	56
10 Wnioski końcowe	59
Literatura.....	62
Dodatek A	67

Dziękuję mojemu promotorowi Panu doktorowi Marcinowi Janaszewskiemu, przede wszystkim za możliwość wspólnej pracy oraz poświęcony czas oraz, za pokazanie nieznanego mi wcześniej świata nauki.

Kiedyś usłyszałem, że dobry promotor to nie tylko ktoś kto pomaga w napisaniu pracy dyplomowej ale również, ktoś do kogo można udać się o pomoc i radę w każdej sprawie oraz, ktoś kto pomaga w wyborze dalszej drogi rozwoju. Po prostu przyjaciel. Z dumą mogę powiedzieć, że Pan doktor Marcin Janaszewski należy do tej kategorii.

Szczególne podziękowania należą się również Panu magistrowi inżynierowi Michałowi Postolskiemu za całą pomoc i poświęcony czas. Myślę również, że Pan magister inżynier Michał Postolski spełnia wszystkie wyżej wymienione kryteria dobrego promotora.

Ponadto chciałbym gorąco podziękować Panu Profesorowi doktorowi habilitowanemu inżynierowi Adamowi Pelikantowi za okazanie zainteresowania moimi poczynaniami. Zainteresowanie to działało i w dalszym ciągu działa na mnie niezwykle motywująco.

Na koniec chciałbym podziękować mojej przyjaciółce Dominice Dolacie za wsparcie i zainteresowanie tematyką niniejszej pracy.

1 Wstęp

Na przestrzeni ostatnich kilkudziesięciu lat obserwujemy burzliwy rozwój diagnostyki medycznej na bazie tomografii komputerowej (CT). W tym okresie dokonano ogromnego postępu, który objawia się nie tylko coraz lepszymi tomografią ale również coraz bardziej zaawansowanymi algorytmami przetwarzania i analizy obrazów tomograficznych (w skrócie obrazów CT). Postęp ten odnotowuje się również w diagnostyce chorób oskrzeli, gdzie aktualnie wykorzystuje się zaawansowane wielodetektorowe tomografy, potrafiące w krótkim czasie dokonać skanu klatki piersiowej pacjenta generując obraz przestrzenny wysokiej rozdzielczości.

Równolegle, na świecie, prowadzone są prace nad algorytmami przetwarzania i analizy obrazów tomograficznych klatki piersiowej w celu wyodrębnienia oraz dokonania ilościowej analizy drzew oskrzelowych. W wyniku prowadzonych prac powstało szereg metod przetwarzania i analizy tego typu obrazów.

Podstawowym problemem dotyczącym testowania algorytmów pomiaru lokalnych parametrów oskrzeli jest to, że trudno jest zweryfikować ilościowo wyniki generowane przez algorytm ponieważ po wykonaniu tomografii pacjenta nie można uzyskać informacji o wartościach poprawnych. Trzeba byłoby zoperować pacjenta, wyjąć oskrzela, przekroić je w wielu miejscach i dokonać pomiarów. Takie działanie jest oczywiście niemożliwe do przeprowadzenia, gdyż doprowadziłoby do śmierci pacjenta. Stąd istnieje potrzeba zbudowania komputerowego modelu oskrzeli, który odzwierciedlałby oskrzela uzyskane w wyniku segmentacji obrazów tomograficznych (w skrócie oskrzela CT). Mając taki model (w postaci trójwymiarowego obrazu) znane byłyby poprawne parametry lokalne stąd byłaby możliwa weryfikacja algorytmów dokonujących tych pomiarów.

Biorąc pod uwagę powyższe poniżej sformułowano trzy cele niniejszej pracy:

1. Prezentacja podstawowych zagadnień dotyczących algorytmów modelowania drzew oskrzelowych.

2. Opracowanie, zaimplementowanie oraz zbadanie algorytmu generowania modelu drzew oskrzelowych będących wynikiem segmentacji obrazów CT klatki piersiowej.
3. Wygenerowanie filmu i zdjęć stereoskopowych prezentujących modele drzew oskrzelowych.

Cel pierwszy stanowi część teoretyczną pracy zawartą w rozdziale 3. Cel drugi i trzeci stanowi część praktyczną pracy i jego realizację opisano w rozdziałach 5 i 8.

Niniejsza praca jest rozwinięciem prac wcześniej publikowanych na konferencjach naukowych [33] oraz [32], i opisuje nową koncepcję trójwymiarowego modelu drzewa oskrzelowego opracowanego na potrzeby weryfikacji algorytmów generacji ilościowego opisu drzew CT. Warto w tym miejscu nadmienić, iż zaprezentowany w tej pracy model drzewa oskrzelowego został wykorzystany do weryfikacji wyników w ramach rozprawy doktorskiej mgr inż. Michała Postolskiego.

Omawiany w tej pracy algorytm generowania trójwymiarowego modelu drzewa oskrzelowego stanowi rozwinięcie wcześniejszego algorytmu [18]. Należy zaznaczyć, iż dotychczas powstało kilka innych modeli, np.: [15] oraz [44]. Modele te powstały jednak do badania przepływu aerozoli w oskrzelach, i z powodów opisanych w dalszej części pracy nie nadawały się do testowania algorytmów analizy ilościowego opisu drzew oskrzelowych. Zaproponowany w niniejszej pracy algorytm wzbogaca model [18] o deformacje geometryczne oraz zaszumienie. Wprowadzenie tych elementów miało na celu przybliżenie modelu do drzew oskrzelowych uzyskiwanych z danych tomograficznych.

Należy również podkreślić, że niniejsza praca jest częścią większego projektu dotyczącego badania algorytmów ilościowej analizy drzew oskrzelowych na bazie obrazów CT klatki piersiowej. Głównym efektem niniejszej pracy jest algorytm generowania modelu drzewa oskrzelowego CT w postaci trójwymiarowego obrazu. Model ten jest konieczny do testowania algorytmów ilościowej analizy drzew oskrzelowych.

1.2 Słownik pojęć związanych z topologią dyskretną

Rozdział ten zawiera definicje wybranych pojęć użytych w dalszej części pracy, oraz pojęcia niezbędne do zrozumienia kolejnych rozdziałów mniejszej pracy.

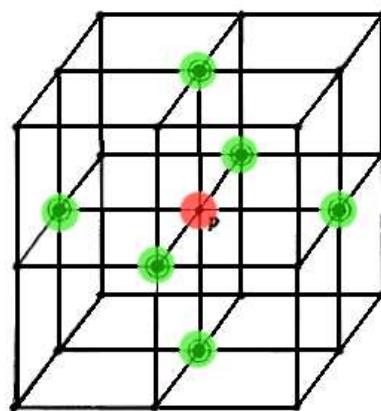
Sąsiedztwo punktu w przestrzeni trójwymiarowej: oznaczmy przez Z zbiór liczb całkowitych, przez N zbiór nieujemnych liczb całkowitych, przez N_+ zbiór dodatnich liczb całkowitych, przez R_+ dodatnich liczb rzeczywistych. Ponadto nadto, niech $E = Z^3$. Niech $X \subseteq E$, przez \bar{X} oznaczamy dopełnienie zbioru X czyli $\bar{X} = E \setminus X$. Punkt $p \in E$ jest zdefiniowany przez (p_1, p_2, p_3) , gdzie $p_i \in Z$. Dla dowolnego $p \in E$ rozważamy trzy typy sąsiedztwa N_6 , N_{18} , N_{26} zdefiniowane następująco:

$$N_6(p) = \{y \in E : |p_1 - y_1| + |p_2 - y_2| + |p_3 - y_3| \leq 1\},$$

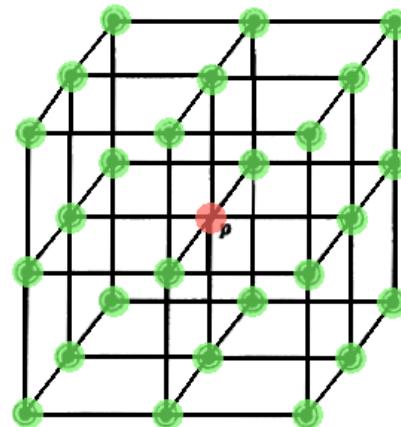
$$N_{18}(p) = \{y \in E : |p_1 - y_1| + |p_2 - y_2| + |p_3 - y_3| \leq 2\} \cap N_{26}(p),$$

$$N_{26}(p) = \{y \in E : \max(|p_1 - y_1|, |p_2 - y_2|, |p_3 - y_3|) \leq 1\}$$

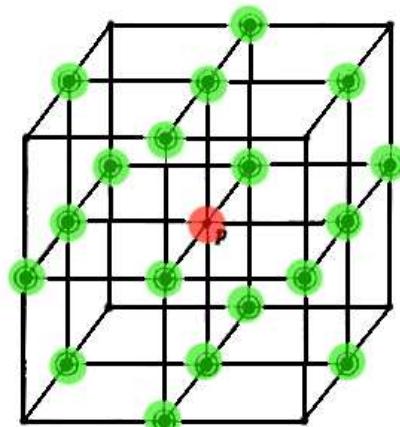
Zbiór $N_n(p)$ nazywany jest *n-sąsiedztwem* punktu p . Ponadto definiujemy $N_n^*(x) = N_n(x) \setminus \{x\}$, dla $n = 6, 18, 26$. Każdy punkt $y \in N_n^*(p)$ nazywany jest *n-przyległym* ($n = 6, 18, 26$) do p lub mówmy również, że y jest *n-sąsiadem* p . Wizualizacja sąsiedztwa punktu dla $n = 6, 18$ i 26 została przedstawiona odpowiednio na rys. 1, 2 i 3.



Rys. 1 Wizualizacja sąsiedztwa $N_6(p)$. Rysunek prezentuje wycinek siatki punktów przestrzeni Z^3 , gdzie zielone punkty należą do $N_6(p)$.



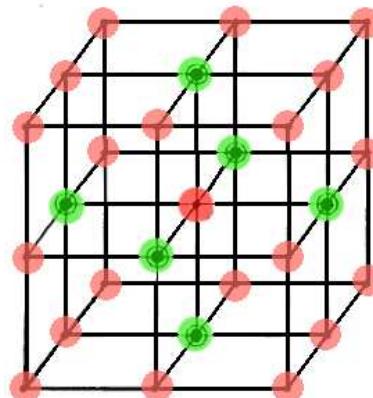
Rys. 2 Wizualizacja sąsiedztwa $N_{26}(p)$. Rysunek prezentuje wycinek siatki punktów przestrzeni Z^3 , gdzie zielone punkty należą do $N_{26}(p)$.



Rys. 3 Wizualizacja sąsiedztwa $N_{18}(p)$. Rysunek prezentuje wycinek siatki punktów przestrzeni Z^3 , gdzie zielone punkty należą do $N_{18}(p)$.

n-ścieżka s jest (istnieje możliwość, że pustą) sekwencją punktów $p_1, \dots p_k$, gdzie $p_i \in N_n^*(x_{i-1})$, dla $i = 1, 2, \dots, k$. Jeżeli s jest niepusta to długość s jest równa k . Jeżeli $p_1 = p_k$ to ścieżka s jest zamknięta.

Obiekt $X \subseteq E$ jest *n-połączony* jeżeli dla każdej pary punktów z X , istnieje *n-ścieżka* między tymi punktami, zawarta w X . Relacja n-połączeniowości jest relacją równoważności a jej klasami równoważności są *n-połączone* komponenty X . Jeżeli X jest zbiorem skończonym to nieskończony połączony komponent \bar{X} jest nazywany *tłem*, inne połączone komponenty \bar{X} nazywane są pustkami. Zbiór złożony ze wszystkich *n-połączonych* komponentów X oznaczamy $C_n(X)$. Ponadto, jeżeli używamy *6-połączeniowości* dla X to musimy użyć innej *m-połączeniowości* dla \bar{X} (i odwrotnie). Jest to konieczne aby uniknąć paradoksów połączeniowości. Na rys. 4 przedstawiono paradoks połączeniowości, gdy dla wszystkich punktów użyto *6-sąsiedztwa*.



Rys. 4 Paradoks połączeniowości dla 6-sąsiedztwa. Wszystkie zielone punkty są rozłączne, ale ciągle dzielą zbiór czerwonych punktów na dwa zbiory; jeden zawierający tylko jeden punkt i drugi zawierający całą resztę.

Tunel: w obiekcie $X \subseteq E$ istnieje tunel wtedy i tylko wtedy gdy, istnieje pętla punktów należących do X , której nie można iteracyjnie transformować wewnątrz obiektu z wykorzystaniem elementarnych lokalnych transformacji, do jednego punktu należącego do X .

Otwór: to skończony element tła zawarty w Z^2 .

Pustka: połączony element tła ograniczony przez obiekt zawarty w Z^3

1.3 Układ pracy

Niniejsza praca składa się z 10 rozdziałów. W rozdziale pierwszym zawarto wstępne informacje dotyczące celu pracy oraz zdefiniowano niezbędne do zrozumienia dalszych rozdziałów pojęcia. Rozdział drugi zawiera rozważania teoretyczne związane z przetwarzaniem danych tomograficznych oraz ich znaczenia dla lepszego diagnozowania chorób układu oddechowego. Rozdział trzeci zawiera wstęp do tematu modelowania drzew oskrzelowych oraz jego znaczenia dla weryfikacji algorytmów ilościowej analizy drzew oskrzelowych. W rozdziale 4 opisano bazowy algorytm modelowania drzewa oskrzelowego, natomiast w rozdziale 5 omówiono jego rozszerzoną wersję, której owa praca dotyczy. Rozdział 6 zawiera szczegóły implementacyjne omawianego algorytmu. W rozdziale 7 zawarto omówienie wykorzystanych narzędzi informatyki. Rozdział 8 stanowi omówienie wizualizacji omawianych modeli z wykorzystaniem technik stereoskopowych. W rozdziale 9 omówiono metodologię weryfikacji uzyskanych wyników. W ostatnim, 10 rozdziale zawarto wnioski końcowe.

2 Ilościowy opis oskrzeli na bazie obrazów tomograficznych 3D

Od szeregu lat na całym świecie prowadzone są prace mające na celu opracowanie algorytmów ilościowej analizy drzew oskrzelowych.

Pomimo wielu lat prac algorytmy analizy i przetwarzania obrazów tomograficznych oskrzeli wykazują ciągle pewne niedoskonałości, które powodują, że ich zastosowanie jest trudne i wymaga dużego doświadczenia oraz nakładu pracy ze strony lekarza. Typowe problemy jakie można tutaj odnotować to:

- Algorytmy zawodzą w wielu przypadkach podając informacje iż nie są w stanie dokonać pomiarów lub co gorsza generują błędne wyniki.
- Brak pełnej automatyzacji. Lekarz zmuszony jest do żmudnego wstępnie przygotowywania obrazu lub manualnego poprawiania wyników generowanych przez algorytm.
- Algorytmy często zwracają wyniki pomiarów informując jednocześnie o bardzo wysokim stopniu niepewności uzyskanego wyniku.
- Złożoność obliczeniowa stosowanych algorytmów jest często veryska co skutkuje długim czasem analizy obrazów.

Stąd promotor niniejszej pracy wraz z innymi naukowcami z kraju i zagranicy realizuje projekt naukowy dotyczący badań nad algorytmami ilościowej analizy drzew oskrzelowych na bazie obrazów CT klatki piersiowej. Celem tego projektu jest modyfikacja istniejących algorytmów lub opracowanie nowych tak aby wyeliminować powyżej opisane problemy. Niniejsza praca inżynierska jest częścią tego projektu a jej wyniki mają istotny wpływ na jego realizację.

2.1 Znaczenie prac nad algorytmami ilościowej analizy drzew oskrzelowych

Choroby oskrzeli są ważnym problemem społecznym. Najbardziej rozpoznawanymi chorobami oskrzeli są astma i przewlekła obturacyjna choroba płuc w skrócie POChP.

Astma jest przewlekłą chorobą zapalną dróg oddechowych, w której uczestniczy wiele komórek i substancji przez nie uwalnianych. Atakom astmy zwykle towarzyszy rozlane zwężenie oskrzeli o zmiennym nasileniu. Częstość występowania astmy zwiększa się we wszystkich krajach, zwłaszcza wśród dzieci.

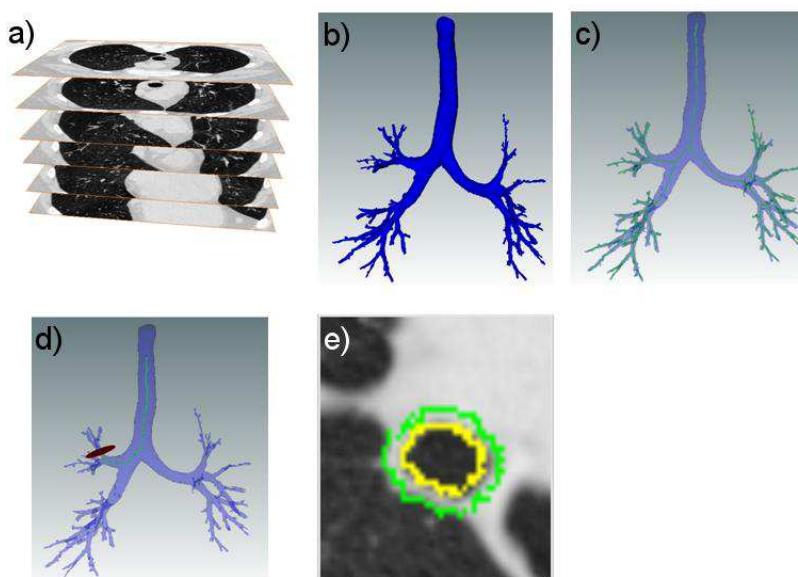
Jak wynika z badań przeprowadzonych w latach 2006-2008 4,6% ankietowanych deklaruje, że choruje na astmę, natomiast wystąpienie astmy w 2011 roku deklaruje 13,5%. W trakcie badania ambulatoryjnego lekarze, w oparciu o kryteria GINA [39], rozpoznali astmę u 10,6% badanych; częściej w ośrodkach miejskich(10,6% vs. 10,1%). Aż 66,9% chorych nie wiedziało wcześniej o swojej chorobie(astma nierozpoznana). Jednocześnie u 39% badanych, którzy deklarowali, że chorują na astmę, diagnozę tę zweryfikowano negatywnie(nadrozpoznawalność astmy) [21].

POChP jest jedną z najczęstszych chorób przewlekłych ze wszystkich i najczęstszą przewlekłą chorobą układu oddechowego gdzie zwężenie oskrzeli ma charakter postępujący i nie oddaje się leczeniu. Uważa się, że na POChP w Polsce choruje ok. 10% mieszkańców po 30 roku życia. Spośród chorych na POChP większą część stanowią osoby palące od wielu lat papierosy, a także osoby przebywające przez dłuższy okres czasu w zakurzonych, zapylonych pomieszczeniach [23].

Z powodu postępu cywilizacyjnego, a co za tym idzie zwiększonej zachorowalności na wymienione wyżej choroby układu oddechowego, praca nad lepszymi metodami wspomagającymi wczesne wykrywanie tych chorób oraz pozwalającymi na lepszą diagnostykę z roku na rok zyskuje na znaczeniu.

2.2 Proces pomiaru lokalnych parametrów oskrzeli

Ogólny przebieg procesu analizy danych tomograficznych w celu pomiaru lokalnych parametrów oskrzeli polega na rekonstrukcji obrazu trójwymiarowego z dwuwymiarowych obrazów uzyskanych w procesie kolejnych, wykonywanych po sobie skanowań klatki piersiowej (rys. 5a). Ten etap zwykle realizowany jest przez oprogramowanie wbudowane w tomograf. Następny krok polega na wyodrębnieniu (segmentacji) drzewa oskrzelowego [42], [34] z trójwymiarowego obrazu klatki piersiowej (rys 5b). Następnie generowany jest szkielet czyli cienka krzywa przebiegająca w środku każdej gałęzi oskrzeli (rys. 5c). Szkielet ten umożliwia na dalszym etapie zbudowanie płaszczyzn przekrojów prostopadłych do szkieletu a co za tym idzie prostopadłych do gałęzi oskrzeli (rys 5d). Wyznaczenie płaszczyzn prostopadłych jest warunkiem koniecznym dla dokonania dokładnych pomiarów grubości ściany i pola powierzchni prześwitu oskrzeli. Ostatni etap to dokonanie pomiarów na wyciętym za pomocą płaszczyzny prostopadłej fragmencie gałęzi oskrzela (rys 5e).

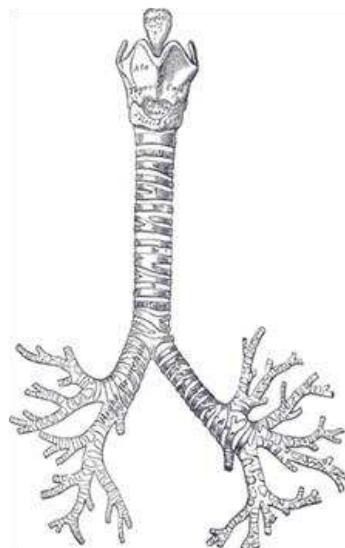


Rys.5 Proces analizy danych tomograficznych klatki piersiowej w celu uzyskania informacji o lokalnych parametrach drzewa oskrzelowego. a) złożenie dwuwymiarowych danych, b) segmentacja danych, c) wyznaczenie szkieletu drzewa oskrzelowego, d) wyodrębnienie interesującego odcinka oskrzela, e) analiza lokalnych parametrów oskrzela.

3 Wprowadzenie do modelowania drzew oskrzelowych

Oskrzela są częścią układu oddechowego i znajdują się pomiędzy tchawicą a oskrzelikami. Oskrzela mają kształt małych, połączonych ze sobą rurek. W zależności od wielkości gałęzi chrząstki podtrzymująca jej kształt przyjmują formę pierścieni, małych płytka bądź wysepek. U zdrowych ludzi, występują pojedyncze rozgałęzienia drzewa, to znaczy każda gałąź dzieli się na dwie kolejne. Warto dodać w tym miejscu, iż przeprowadzone w 2011 roku badanie ukazało potencjalne powiązanie z nienaturalną liczbą rozwidleń gałęzi a autyzmem [38]. Jeśli w wyniku kolejnych badań uda się wykluczyć zmiany genetyczne jako przyczynę powstawania nienaturalnych rozwidleń, to doprowadzić może to do powstania nowych metod wykrywania autyzmu na wczesnych etapach rozwoju dzieci.

Funkcją oskrzeli jest doprowadzanie i odprowadzanie powietrza do/z płuc. Oskrzela posiadają strukturę drzewa, w której wyróżnia się dwa oskrzela główne: lewe i prawe, oskrzela główne dzielą się następnie na kolejne. Na rys. 6 przedstawiono fragment drzewa oskrzelowego.



Rys. 6 Przedni widok chrzęstek krtani, tchawicy i oskrzeli. [12]

Geometria drzewa oskrzelowego jest ściśle powiązana z jego położeniem przestrzennym. Kolejne gałęzie dzielą dostępna przestrzeń na zaopatrywane przez nie obszary organu, co umożliwia powiązanie objętości transportowanego przez gałąź powietrza z jej położeniem wewnątrz organu. Kształt i objętość danego obszaru ma bezpośredni wpływ na wielkość gałęzi oraz kierunek, w którym gałąź jest skierowana.

W przeciągu ostatnich kilkunastu lat zaproponowano szereg modeli drzewa oskrzelowego różniących się pod względem swojej konstrukcji oraz dokładności, z jaką potrafią one odzwierciedlić budowę rzeczywistego drzewa oskrzelowego. Początkowo opracowane modele były ograniczone jedynie do przestrzeni jednowymiarowej [47], a także model [13], który w znacznym stopniu przyczynił się do powstania kolejnych modeli. Następnie powstały modele dwuwymiarowe [27], [24].

Szybki rozwój technologii trójwymiarowej, który nastąpił pod koniec lat 90 XX wieku i trwa do dziś, a także zapotrzebowanie, ze strony środowiska medycznego na trójwymiarowe obrazowanie ludzkich dróg oddechowych, doprowadziło do powstania kilku trójwymiarowych modeli drzewa oskrzelowego. Pierwsze modele odnoszące się do przestrzeni trójwymiarowej powstały już na początku lat 80 XX wieku [4]. Jednak pierwszym bardziej znaczący modelem trójwymiarowym, jest model opracowany przez [18]. Warto wspomnieć w tym miejscu o późniejszym modelu wykorzystującym metodę Monte Carlo autorstwa [15], a także o modelu [11] dzięki, któremu możliwe stało się przewidzenie obrazów transportu powietrza u pacjentów chorych na astmę z potencjalnymi skutkami klinicznymi.

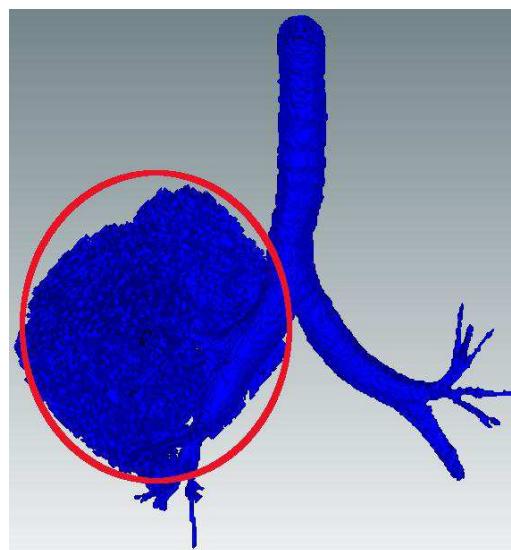
Omawiany w niniejszej pracy model drzewa oskrzelowego został opracowany na bazie wyżej wymienionego modelu [18] - dla uproszczenia model ten w dalszej części pracy nazywany będzie modelem podstawowym. Natomiast model, którego owa praca dotyczy modelem rozszerzonym.

Należy nadmienić w tym miejscu, że tym co odróżnia oba modele jest przede wszystkim klasa problemów, dla których zostały one opracowane. Model podstawowy został opracowany z myślą o badaniu przepływu powietrza w drzewie oskrzelowym, natomiast model rozszerzony został opracowany z myślą o testowa-

niu algorytmów przetwarzających drzewa oskrzelowe uzyskane w procesie segmentacji obrazów tomograficznych.

3.1 Charakterystyka problemu testowania algorytmów ilościowej analizy drzew oskrzelowych

Drzewo oskrzelowe uzyskane w procesie segmentacji obrazu uzyskanego z tomografu komputerowego jest obarczone wieloma zniekształceniami. Źródłem tych zniekształceń są na przykład; różnice pochłanianej ilości promieniowania rentgenowskiego pomiędzy oskrzelami a płucami, błędy i niedoskonałości algorytmów segmentacji, ruchy podczas akwizycji danych, akwizycja z wykorzystaniem niskiej dawki promieniowania czy nawet bicie serca, lub oddychanie przez pacjenta podczas wykonywania skanowania. Na rys. 7 przedstawiono jeden z powszechniejszych problemów związanych z segmentacją danych tomograficznych o wysokim zaszumieniu.



Rys. 7 Kolorem czerwonym zaznaczono „wyciek” algorytmu segmentacji - *region growing* [49] do mięśnia płucnego. Wyciek jest spowodowany przez silne zaszumienie obrazu tomograficznego oraz niewłaściwe dobranie parametrów progów, a także niską skuteczność algorytmu *region growing*.

Analiza lokalnych parametrów wysegmentowanego drzewa oskrzelowego pozwala na uzyskanie konkretnych informacji przydatnych podczas testowania nowych leków oraz diagnostyki stanu zdrowia pacjenta. Jako przykład można podać powiększoną grubość ściany oskrzeli u osób palących papierosy. Aby uzyskać dane o odpowiedniej wiarygodności niezbędne jest opracowanie algorytmów ilościowej analizy drzewa oskrzelowego, które pozwolą osiągnąć ten cel mimo szeregu wymienionych wyżej znieksztalcień. Testowanie, tychże algorytmów na obrazach rzeczywistych oskrzeli jest trudne ponieważ nie znamy poprawnych lokalnych wartości parametrów drzewa.

Jednym z rozwiązań, stosowanych wcześniej przez [31] jest zbudowanie fantomu z plastikowych rurek o stałej średnicy i regularnym kształcie, a następnie zeskanowanie go w tomografie.

Innym podejściem zaproponowanym w niniejszej pracy jest opracowanie modelu, którego lokalne parametry są znane. Należy w tym miejscu zaznaczyć, że model składający się jedynie z połączonych, prostych geometrycznych obiektów jest zbyt łatwy do analizy i nie zapewnia, że uzyskane wyniki będą miały odzwierciedlenie w rzeczywistych obrazach. Zastosowanie takiego modelu ogranicza się do wstępnych testów oraz weryfikacji koncepcji. Dlatego ważne jest by zaproponowany model, odzwierciedlał często znieksztalcone i zaszumione drzewo oskrzelowe z jednoczesnym zachowaniem informacji o jego parametrach.

3.2 Główne cechy modelu rozszerzonego

Tym co w głównej mierze odróżnia model podstawowy od omawianego w niniejszej pracy modelu rozszerzonego, jest jak to już wspomniano wcześniej klasa problemów, dla których opracowano oba modele.

Podeczas projektowania modelu rozszerzonego zdefiniowano trzy główne cechy, które powinien on posiadać.

1. *Stałosć średnicy gałęzi:* analiza średnicy gałęzi pozwala uzyskać szereg istotnych informacji na temat analizowanego drzewa. Dlatego ważne jest, aby wartość ta nie uległa zmianie podczas wykonywania transformacji na generowanym modelu. Dzięki temu założeniu znamy poprawną wartość

średnicy, co jest istotne podczas testowania algorytmów ilościowego opisu drzew oskrzelowych.

2. *Szumy i zniekształcenia geometryczne:* Jak wspomniano wcześniej dane uzyskane w procesie segmentacji zawierają szereg niedoskonałości, które z punktu widzenia zastosowania omawianego modelu powinny zostać w nim uwzględnione. W celu zobrazowania problemu, w dalszej części pracy przedstawiono analizę wpływu poziomu zaszumienia oraz jego braku na wyniki uzyskiwane przez algorytmy szkieletyzacji [30]. Zaszumieniu oraz zniekształceniom geometrycznym poświęcono odpowiednie podrozdziały rozdziału 5.
3. *Czas uzyskania informacji o średnicy gałęzi:* podczas analizy drzewa istotna wydaje się możliwość szybkiego i łatwego w realizacji porównywania uzyskiwanych przez testowane algorytmy wyników, z faktycznymi wartościami. Dlatego zdecydowano się, aby przechowywać informację o średnicy danej gałęzi w wartościach wokseli (punkt obrazu posiadający niezerową współrzędną osi Z), które do niej przynależą. Dane te zapisane są w niezawierającej szumów mapie drzewa.

4 Koncepcja podstawowego, trójwymiarowego modelu drzewa oskrzelowego

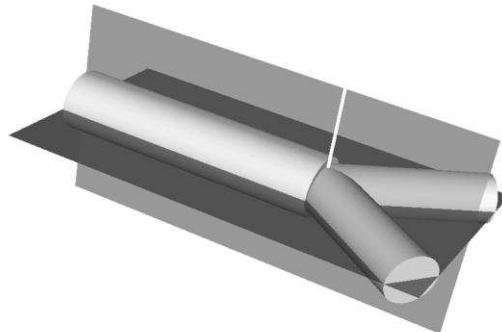
Niniejszy rozdział opracowano na postawie artykułu [18], który prezentuje szczegółowy opis algorytmu podstawowego i badań właściwości modelu drzewa generowanego tym algorytmem.

Algorytm podstawowy, został oparty na dwóch przedstawionych poniżej założeniach umożliwiających zrealizowanie równomiernego i efektywnego transportu powietrza wewnątrz całych płuc.

Pierwsze z założeń pozwala na połączenie powietrza transportowanego przez oskrzele z jego położeniem przestrzennym i mówi, że całkowita objętość tego powietrza jest proporcjonalna do objętości obszaru, który jest przez nie zaopatrywany. Ponadto przyjęto, że dla każdego rozwidlenia objętość dzieci jest proporcjonalna do objętości rodzica, a także, że całkowity przepływ powietrza transportowany przez dzieci jest proporcjonalny do przepływu powietrza transportowanego przez rodzica.

Drugie założenie mówi natomiast o tym, że końcowe gałęzie drzewa są homogenicznie rozmieszczone wewnątrz organu. Ponadto dla ludzkiego drzewa oskrzelowego gałęzie końcowe modelu, są traktowane jako końcowe oskrzeliki.

Przedstawione powyżej założenia pozwalają na zdefiniowanie opisanych dalej reguł procesu generowania kolejnych rozwidleń drzewa oskrzelowego. Rys. 8 przedstawia podstawowe parametry pojedynczego rozwidlenia.



Rys. 8 Pojedyncze rozwidlenie. Ciemniejsza płaszczyzna rozwidlenia, jaśniejsza płaszczyzna po-działu objętości. Obie płaszczyzny rozpościerają się do granic obszaru rodzica. Kolorem białym za- znaczono normalną płaszczyznę rozwidlenia zaczepioną w punkcie rozwidlenia.

4.1 Algorytm generujący model podstawowy drzewa oskrzelowego

Algorytm podstawowy został zbudowany w oparciu o dziewięć bazowych i cztery dodatkowe reguły, których kolejne wykonywanie umożliwia wygenerowanie drzewa oskrzelowego. Algorytm wymaga podania danych wejściowych, które stanowią zestaw parametrów opisujących korzeń drzewa – tchawicę oraz powierzchnię ograniczającą przestrzeń, w której generowane jest drzewo.

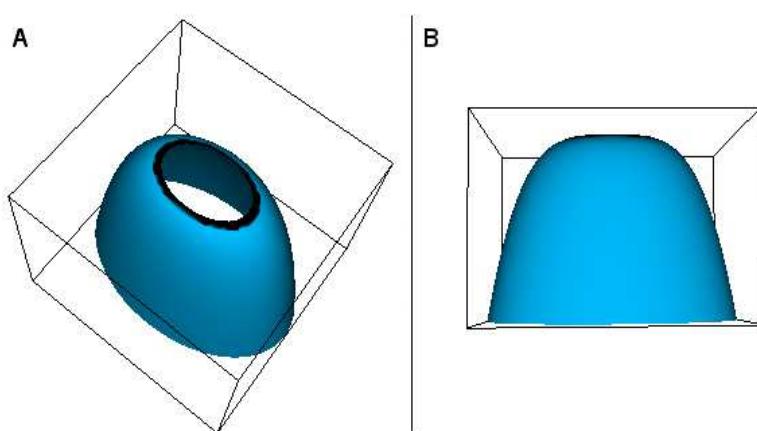
Tchawica: najistotniejszymi parametrami korzenia drzewa jest jego średnica, długość, oraz położenie w przestrzeni. Ważne jest, aby korzeń znajdował się w odpowiedniej pozycji względem powierzchni ograniczającej przestrzeń. Parametry tchawicy powinny zostać dobrane tak by były zgodne z wynikami badań morfologicznych. Koniec korzenia za pomocą, którego wyznaczony jest punkt pierwszego rozwidlenia, powinien znajdować się w około $\frac{1}{4}$ długości przestrzeni w kierunku osi

Z , w której nastąpi wygenerowanie drzewa.

Przestrzeń: jest ona ograniczona za pomocą powierzchni, która może zostać określona na kilka sposobów. Jednakże dla uproszczenia tej pracy przyjęto prostą postać powierzchni opisaną *równaniem 1.*

$$z = 2 * 15^{-3} (x^2 + (1,5y)^2)^2 \text{ dla } 0 \leq z \leq 30 \quad (1)$$

Tak zdefiniowana powierzchnia przedstawiona na rys. 9 w sposób wystarczający do dalszych rozważań opisuje granice organu.



Rys. 9 Powierzchnia ograniczająca przestrzeń, w której następuje wygenerowanie drzewa.

4.1.1 Reguły algorytmu podstawowego

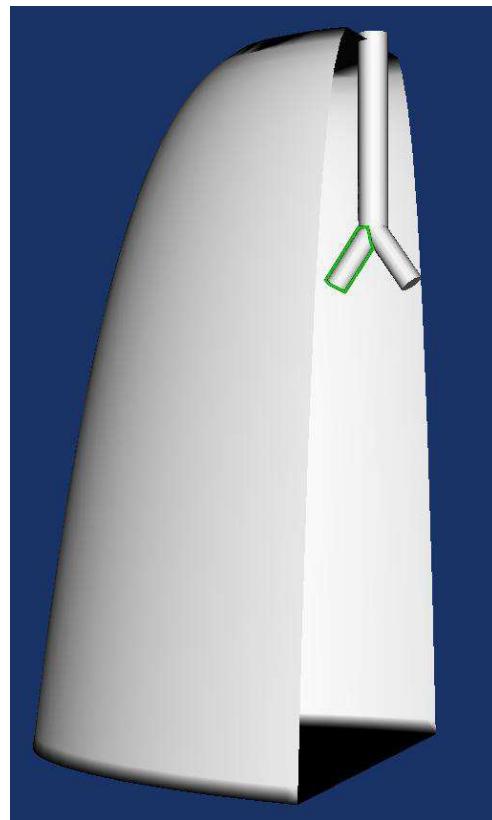
Przedstawione poniżej reguły definiują kolejne kroki, których sukcesywne wykonywanie, aż do spełnienia warunku końcowego – zdefiniowanego w *regule 9* umożliwia wygenerowanie drzewa oskrzelowego.

Reguła 1: każde rozwidlenie jest dychotomiczne, tzn. każdy rodzić dzieli się na dwoje dzieci.

Reguła 2: przekroje wzdłużne rodzica i jego dzieci leżą na tej samej płaszczyźnie, zwanej dalej płaszczyzną rozwidlenia.

Reguła 3: suma przepływu przez dzieci równa się całkowitemu przepływowi rodzica; $d_0^n = d_1^n + d_2^n$, gdzie d_0 , d_1 i d_2 oznaczają odpowiednio średnicę rodzica, pierwszego dziecka, drugiego dziecka. Bazując na zasadzie najmniejszego wydatku energii wartość n została zaproponowana jako 3 [16], [27]. Natomiast Horsfield, i Thurlbeck [14] bazując na danych morfologicznych czterech gatunków ssaków zaproponowali wartość n jako liczbę z przedziału 2,4 – 2,9. Jednakże na podstawie analizy danych opublikowanych przez Raabe i inni [36] przyjęto wartość n równą 2,8. Stosunek przepływu w gałęziach wyrażony jest następującą relacją: $d_1 \leq d_2$. Sam przepływ wyrażony przez znormalizowane równanie równy jest: $Q = (d)^n$

Reguła 4: obszar przestrzeni, który jest zaopatrywany przez rodzica, jest dzielony na dwa obszary dzieci za pomocą płaszczyzny podziału przestrzeni. Płaszczyzna ta jest prostopadła do płaszczyzny rozwidlenia i rozciąga się do granic segmentu rodzica. Na rys. 10 przedstawiono segment wyznaczony zgodnie z tą regułą dla jednej z gałęzi drzewa.



Rys. 10 Obszar przestrzeni wyznaczony dla gałęzi drzewa zaznaczony na obrazie obwódką.

Reguła 5: stosunek podziału przepływu r wykorzystany w równaniach z *reguły 6*(patrz dalej) jest równy stosunkowi podziału objętości, który definiuje się jako stosunek obszaru przestrzeni rodzica do obszaru mniejszego dziecka. Wartość r powinna zawierać się w przedziale $0 < r \leq 0,5$. Stosunek podziału objętości wyznaczany jest poprzez podział prostokątnego obszaru zawierającego przestrzeń rodzica na mniejsze prostokąty, których boki są 20 razy mniejsze niż boki prostokąta wewnętrznego którego są umieszczone. Następnie określany jest stosunek ilości tych prostokątów w obszarze rodzica do ich ilości w prostokątnym obszarze zawierającym obszar dziecka.

Reguła 6: średnica dzieci oraz kąt rozwidlenia między nimi wyrażona jest odpowiednio przez *równanie 2*, oraz *równanie 3*.

$$\begin{aligned} d_1 &= d_0 r^{\frac{1}{n}} \\ d_2 &= d_0 (1-r)^{\frac{1}{n}} \end{aligned} \quad (2)$$

$$\cos \Theta_1 = \frac{[1+r^{\frac{4}{n}} - (1-r)^{\frac{4}{n}}]}{2r^{\frac{2}{n}}} \quad (3)$$

$$\cos \Theta_2 = \frac{[1+(1-r)^{\frac{4}{n}} - r^{\frac{4}{n}}]}{2(1-r)^{\frac{2}{n}}}$$

gdzie:

d_0 – średnica rodzica,
 d_1, d_2 – średnica dzieci,
 $n = 2,8$

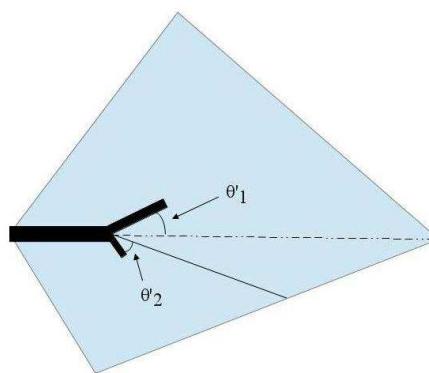
Reguła 7: długość każdej z gałęzi jest trzykrotnością jej średnicy.

Reguła 8: kontynuacja generowania nowych gałęzi w danym kierunku powoduje, że dzieci stają się nowymi rodzicami, a ich płaszczyzna rozwidlenia zostaje ustawiona prostopadle do płaszczyzny rozwidlenia ich rodzica.

Reguła 9: generowanie nowych gałęzi w danym kierunku trwa do momentu przekroczenia wartości progu określającego minimalną wartość przepływu przez gałąź lub w momencie, gdy opuściła ona swój segment.

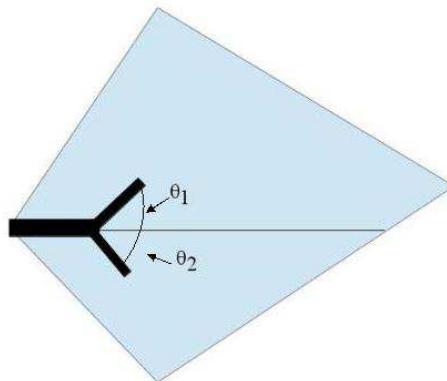
Kolejne reguły rozszerzają część wcześniej wymienionych reguł i służą do korekcji parametrów generowanego rozwidlenia w zależności od wystąpienia szczególnych warunków.

Reguła 4a: w przypadku gdy $|\Theta_1 - \Theta_2| \geq 10^\circ$ (patrz rys. 11) oznacza, to że



Rys. 11 Przedstawia rodzica z dwójką dzieci w przypadku gdy $|\Theta_1 - \Theta_2| \geq 10^\circ$. Kolorem niebieskim zaznaczono płaszczyznę rozwidlenia, linią ciągłą zaznaczono dodatkową półpłaszczyznę podziału określoną zgodnie z regułą 4a i zwróconą prostopadle do płaszczyzny rozwidlenia, linią przerywaną zaznaczono wcześniejszą zdefiniowaną płaszczyznę podziału ograniczoną do końca rodzica.

współczynnik podziału objętości nie jest wystarczająco izotropowy, przez co jest on nieoptymalny, a w konsekwencji kąt rozwidlenia oraz średnica dzieci są nieoptymalne. W celu uzyskania lepszego stosunku podziału należy ograniczyć płaszczyznę podziału do końca rodzica oraz wyznaczyć nową półplaszczyznę podziału, która zaczepiona jest na końcu rodzica i rozchodzi się do końca jego obszaru oraz skierowana jest w kierunku dwusiecznej kąta rozwidlenia. Po skorygowaniu obszarów dzieci oraz współczynnika podziału za pomocą obu półplaszczyzn podziału *reguły 5 i 6* muszą zostać ponownie wykonane. Na rys. 11 zaznaczono nowe kąty rozwidlania. Reguła ta stosowana jest tylko raz dla każdego rozwidlenia. Na rys. 12 przedstawiono natomiast rozwidlenie dla, którego nie jest konieczne wykonanie *reguły 4a*.

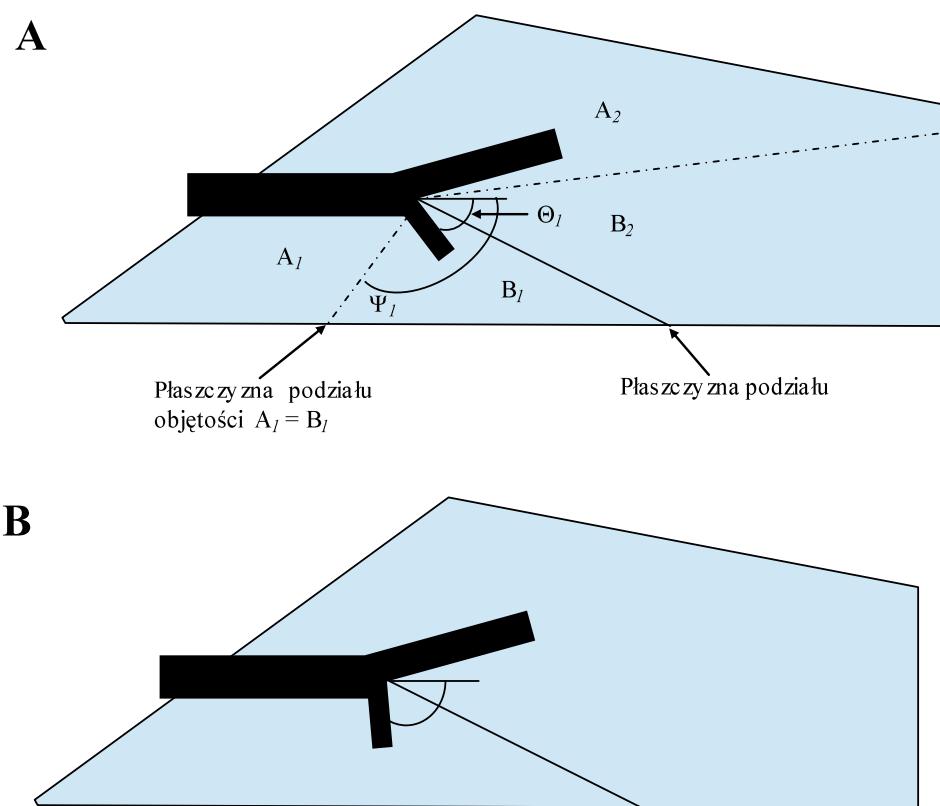


Rys. 12 Przedstawia rodzica z dwójgiem dzieci w przypadku gdy $|\Theta_1 - \Theta_2| < 10^\circ$. Kolorem niebieskim zaznaczono płaszczyznę rozwidlenia, linią zaznaczono płaszczyznę podziału zwróconą prostopadle do płaszczyzny rozwidlenia.

Reguła 6a: zgodnie z równaniem 3 maksymalny kąt każdego rozwidlenia wynosi 90° . Kiedy współczynnik podziału przepływu r przyjmuje małą wartość np. 0,01 wtedy mniejszy kąt rozwidlenia przyjmuje wartość około 78° . Zatem równanie 3 przewiduje, że prawie każdy kąt rozwidlenia jest mniejszy od 80° . Jednakże dopuszczalne jest, aby drzewo posiadało kilka gałęzi o kącie zbliżonym do kąta prostego. Taka sytuacja występuje, gdy obszar takiej gałęzi rozszerza się w kierunku wstecznym. W tym wypadku równanie 3 nie zapewnia odpowiedniego kąta dla takiej gałęzi. Aby sytuacji takiej przeciwdziałać po wykonaniu reguły 4a, należy

określić płaszczyznę dzielącą obszar dziecka na dwa zawierające identyczną objętość oznaczone dalej jako A oraz B . Płaszczyzna ta skierowana jest prostopadle do płaszczyzny rozwidlenia i zaczepiona w punkcie rozwidlenia (patrz rys. 13). Tak zdefiniowane płaszczyzny dla obu dzieci wyznaczają razem z rodzicem kąt ψ . Jeżeli kąt θ jest mniejszy od kąta ψ , wtedy wyznaczany jest nowy kąt θ równy wartości średniej obu kątów (*równanie 4*).

$$\Theta' = \frac{(\Psi_1 + \Theta_1)}{2} \quad (4)$$



Rys. 13 Na rysunku A zaznaczono najistotniejsze parametry pojedynczego rozwidlenia z punktu widzenia reguły 6a. Kąt $\Psi_2 < \theta_2$ objętość $A_2 = B_2$. Na rysunku B przedstawiono rozwidlenie po skorygowaniu kąta rozwidlenia zgodnie z równaniem 4.

Natomiast jeśli wartość średnia obu kątów przekracza 90° , wtedy należy ją zmniejszyć do wartości 90° .

Reguła 7a: jak wspomniano wcześniej, współczynnik stosunku długości względem średnicy powinien wynosić 3. Jednakże czasami jest to wartość nieod-

powiednia i gałąź wykracza poza obszar rodzica albo jej koniec znajduje się zbyt blisko granicy obszaru. W celu korekcji stosunku długości wprowadzono stosunek dystansu do długości. Stosunek ten wyrażony jest *równaniem 5*. Korekcja stosunku długości do średnicy odbywa się w krokach co 0,25 aż do przekroczenia dolnego lub górnego progu stosunku dystansu do długości odpowiednio 3 i 6. Nowa wartość stosunku długości do średnicy nie powinna być mniejsza od 1. Rys. 14 przedstawia parametry rozwidlenia zgodnie z *regułą 7a*.

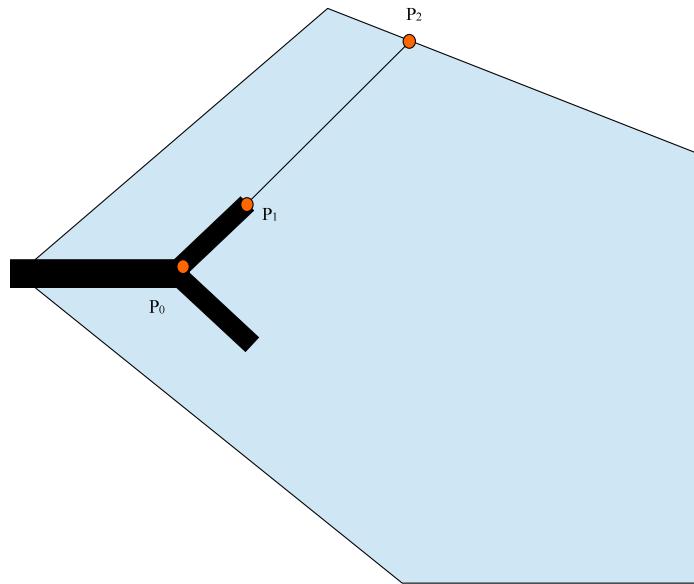
$$\frac{d(P_0, P_2)}{d(P_0, P_1)} \quad (5)$$

gdzie:

P_0 – początek gałęzi

P_1 – koniec gałęzi,

P_2 – punkt na granicy obszaru wyznaczony przez wektor kierunkowy gałęzi



Rys. 14 Na rysunku przedstawiono punkty znajdujące się kolejno na początku gałęzi, na końcu gałęzi, oraz na granicy obszaru. Linią zaznaczono odległość pomiędzy punktem na końcu gałęzi, a punktem znajdującym się na granicy obszaru.

Reguła 8a: Kąt obrotu płaszczyzny rozwidlenia nowego rodzica względem jego rodzica został zdefiniowany wcześniej jako 90° . Jednakże jeśli współczynnik podziału objętości r przyjmuje małą wartość, średnica mniejszego dziecka staje się nierealnie mała. Dlatego konieczne jest wprowadzenie dolnej granicy wartości tego współczynnika. Kiedy wartość r dla kąta prostego jest mniejsza od progu, kąt ten zostaje skorygowany poprzez zmniejszenie lub zwiększenie w krokach co 9° aż do momentu osiągnięcia przez współczynnik podziału objętości wartości wyższej od granicznej. Wartość graniczna równa jest 0,05. Jednakże gdy przepływ gałęzi jest 1,5 razy mniejszy od wartości granicznej przepływu, wartość graniczna r powinna zostać zwiększona do 0,35.

5 Rozszerzony algorytm modelowania drzewa oskrzelowego

Niniejszy rozdział przedstawia autorskie rozwiązania dotyczące rozszerzeń modelu podstawowego. Omówione dalej wygięcia gałęzi pozwalają w sposób bardziej realistyczny przedstawić gałęzie drzewa oskrzelowego, dzięki czemu umożliwiają dokładniejsze testowanie algorytmów analizujących obrazy prawdziwych drzew oskrzelowych. Natomiast zaszumienie obrazu pozwala na odwzorowanie niedoskonałości drzew oskrzelowych uzyskanych w procesie segmentacji danych tomograficznych. Należy zwrócić uwagę, iż sam proces generowania modelu w pierwszej kolejności odbywa się w grafice powierzchniowej – wielokątowa siatka reprezentująca zewnętrzne granice obiektów. Dopiero gdy zostanie wygenerowany model oparty o reguły modelu podstawowego w raz, z wygięciami następuje opisana szczegółowo w kolejnych rozdziałach konwersja do grafiki objętościowej reprezentowanej przez woksele. W dalszych etapach generowania modelu, wszelkie kroki dodawane są już w domenie grafiki objętościowej. Za takim podejściem przemawia nie tylko specyfika dobranych algorytmów dodających szum, lecz przede wszystkim dużo większa dokładność transformacji dokonywanych na grafice powierzchniowej. Dokładność ta ma istotny wpływ podczas wykonywania reguł modelu podstawowego.

5.1 Wygięcia gałęzi

Wygięcia zostały opracowane poprzez odpowiedni dobór wartości iteracyjnego równania spirali. Odrysowanie linii na bazie, której zostanie wygenerowana gałąź odbywa się przez wylosowanie odpowiedniego wariantu funkcji f_1 i f_2 w *równaniu 6*, która może przyjąć postać \sin , $-\sin$, \cos , $-\cos$ - kolejne warianty *równania 6* różnią się znakiem oraz typem funkcji tak jak *sinus* lub *cosinus* dla współrzędnych w osi X i Z . Następnie na podstawie wybranego wariantu *równania 6* obliczane są współrzędne kolejnych punktów. Na rys. 15 przedstawiono wpływ parametrów równania na generowaną gałąź.

$$\begin{aligned} z_k &= r \cdot f_1(2\pi x \frac{i}{n-1}), \\ x_k &= r \cdot f_2(2\pi x \frac{i}{n-1}), \\ y_k &= h \frac{i}{n} \end{aligned} \quad (6)$$

gdzie:

x – współczynnik obrotu. Określa liczbę zawinięć spirali. Wartość tego współczynnika dobierana jest w zależności od stopnia deformacji gałęzi. Przyjęto $x=1$

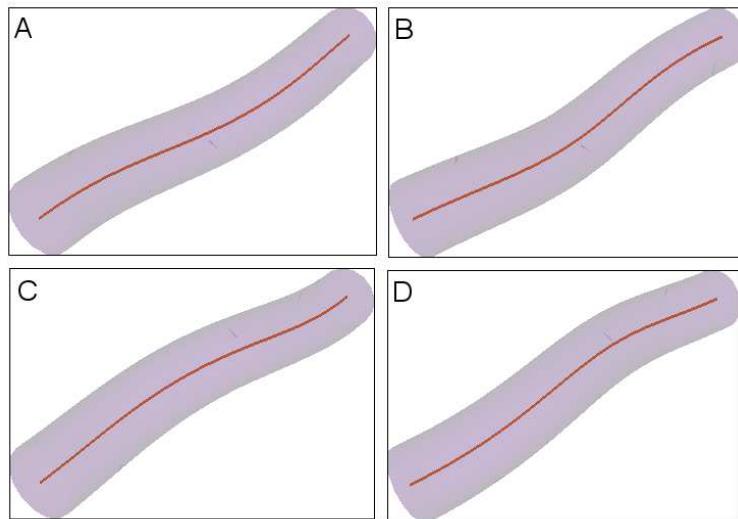
r – promień obrotu. Jego wartość może zostać określona na kilka sposobów w zależności od spodziewanego efektu. Może to być np. wartość losowana z przedziału od 0,1 do 0,9 lub promień gałęzi podzielony przez np. 3,5

n – liczba punktów tworzących gałąź

k – kty punkt linii opisującej gałąź

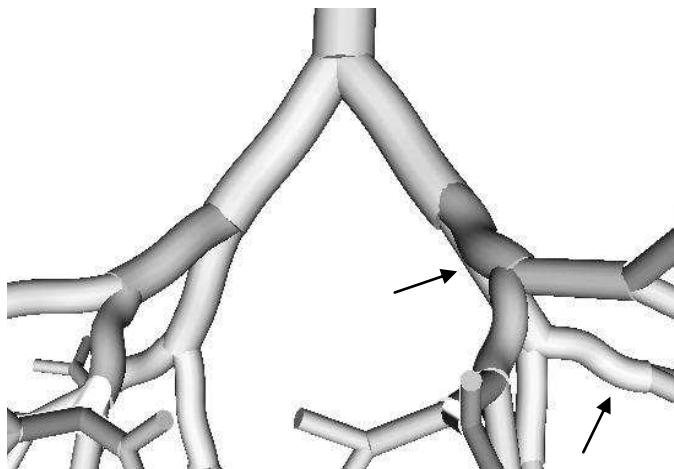
h – długość gałęzi

i – wartość iteracji



Rys. 15 Wpływ funkcji f równania 6 na generowaną gałąź. a) $f_1=\sin$, $f_2=\cos$; b) $f_1=-\sin$, $f_2=\cos$; c) $f_1=\sin$, $f_2=-\cos$; d) $f_1=-\sin$, $f_2=-\cos$. Ponadto dla wszystkich obrazów $r=2,25$; $x=1$; $h=120$, $n=1,3h$.

W celu wygenerowania gałęzi nieposiadającej wygięć należy za wartości *równania 6* w osiach *X* i *Z* podstawić 0. Przykładowy efekt zastosowania wygięć przedstawiono na rys. 16. Na koniec tego podrozdziału należy wspomnieć, iż z obserwacji wynika, że wygięcia zrealizowane z wykorzystaniem *równania 6* najlepiej ograniczyć do mniej więcej piątego poziomu rozgałęzień.



Rys. 16 Fragment modelu drzewa z widocznymi powyginanymi gałęziami.

Warto zaznaczyć, że w przypadku dobrania wartości, które w znacznym stopniu wyginają gałąź należy pamiętać, że spowoduje to przesunięcie punktu rozwidlania, co doprowadzi do sytuacji, w której dzieci w nienaturalny sposób będą w punkcie rozwidlania wystawały ponad rodzica. Jednym, ze sposobów zapobiegania takiej sytuacji jest wygenerowanie swego rodzaju łączników gałęzi. Łączniki te można wygenerować poprzez zdefiniowanie, kilku punktów o wartościach zbliżonych do 0 w osiach *X* i *Z* na początku i końcu wygiętych gałęzi.

5.2 Zniekształcenia modelu wprowadzone w domenie grafiki objętościowej

Zniekształcenia generowane są w dwóch krokach. Pierwszy z nich polega na dodawaniu lub usuwaniu wokseli do/z obiektu za pomocą zmodyfikowanego algorytmu EDEN [7] zwanego topologicznym EDEN [3]. Algorytm ten zapewnia zachowanie topologii modyfikowanego obiektu, dzięki usuwaniu jedynie, tak zwanych punktów prostych, których dodanie lub usunięcie do/z obiektu nie zmienia je-

go topologii. Algorytm do wykrywania punktów prostych w przestrzeni trójwymiarowej został zaprezentowany wraz z dowodem matematycznym przez [2].

W przypadku drzew oskrzelowych zachowanie topologii jest bardzo istotne, ponieważ daje nam pewność, że podczas modyfikowania drzewa, żadne dwie gałęzie nie zostaną połączone.

Procedura topologicznego EDEN erodująca dany obiekt może zostać opisana za pomocą następujących kroków przedstawionych na *listing 1*.

```

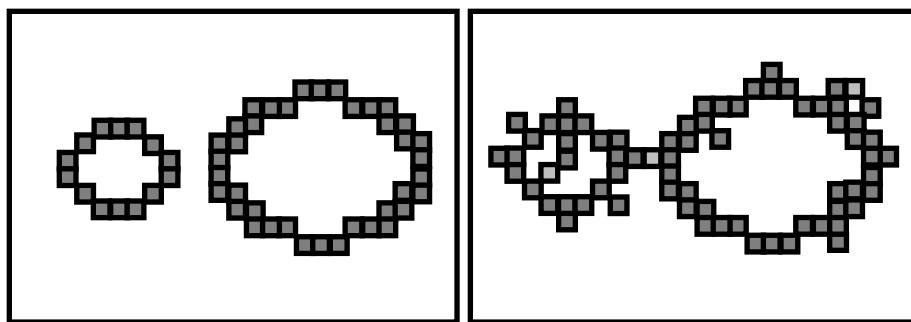
1. topoEDEN( wejście: obraz I, liczba iteracji n, wyjście: obraz O)
2.   = I
3. Utwórz listę L wokseli brzegowych 0
4. for i=0; i<n; i++
5.   wybierz losowo jeden weksel x z L.
6.   jeśli x jest punktem prostym wtedy zmień jego wartość w 0 na 0.
7.   usuń x z L, i zaktualizuj L.
8. zwróć O

```

Listing 1 Pseudokod algorytmu EDEN - erozja

Analogicznie definiuje się procedurę topologicznego EDEN, która powiększa przetwarzany obiekt, w tym przypadku tworzona jest lista punktów, z brzegu tła. Następnie losuje się woksel z listy i jeśli jest on prosty to dodawany jest on do obiektu. Możliwe jest, również zdefiniowanie procedury wykonującej obie operacje równocześnie.

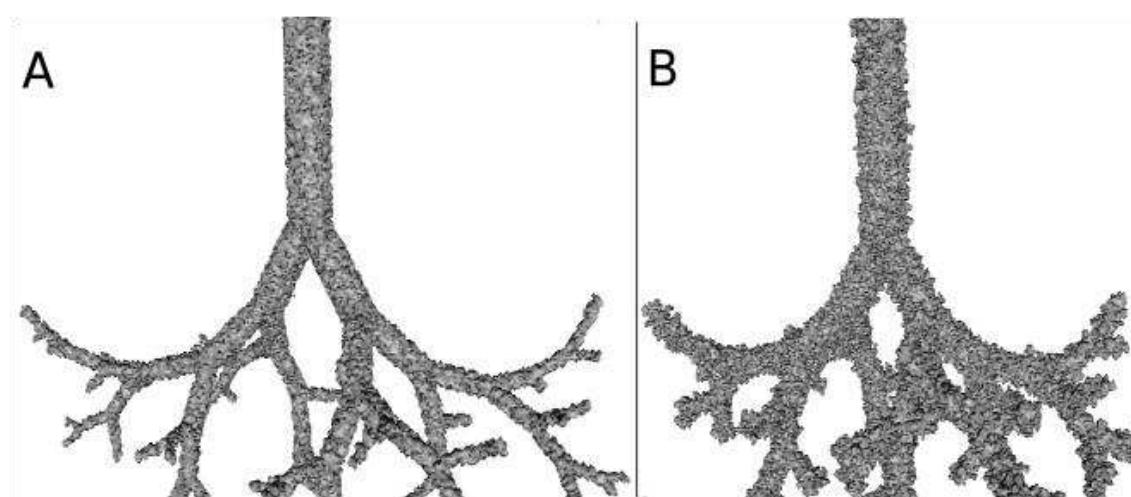
Przykładowe działanie algorytmu topologicznego EDEN zostało przedstawione na rys. 17



Rys. 17 Przykładowy efekt działania topoEDEN (wersja dodająca punkty do obiektu) dla hipotetycznego obiektu 2D z lewej strony. Piksele są reprezentowane przez ciemnoszare kwadraty. Z prawej wynik działania topoEDEN po kilku iteracjach. Żaden jasnoszarych pikseli nie może być dodany przez topoEDEN w kolejnych iteracjach, ponieważ zmieniłoby to topologię obiektu.

Obraz wejściowy z lewej strony zawiera dwa obiekty, a każdy z nich zawiera wewnątrz po jednym otworze. Obraz z prawej prezentuje wynik działania topoEDEN po kilku iteracjach (ciemnoszare piksele). Żaden jasnoszarych pikseli nie może być dodany przez topoEDEN w kolejnych iteracjach, ponieważ zmieniłoby to topologię obiektów. Żaden z tych pikseli nie jest punktem prostym, a ich dodanie spowodowałoby powstanie nowych otworów oraz połączenie obiektów.

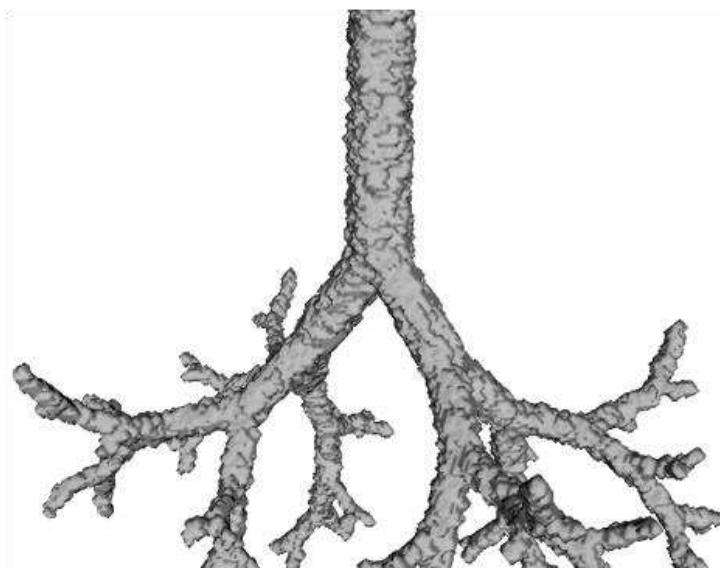
Przykładowy efekt działania topoEDEN w przestrzeni trójwymiarowej został zaprezentowany na rys. 18.



Rys. 18 Przykładowy wynik zastosowania topoEDEN na drzewie oskrzelowym. A) erozja, B) rozszerzanie.

Zaprezentowany na rys. 18 efekt działania topoEDEN nie jest satysfakcjonujący w znaczeniu podobieństwa do drzew oskrzelowych uzyskanych z segmentacji danych tomograficznych np. tego zaprezentowanego na rys 27a. Z tego powodu w kolejnym kroku przetwarzania, stosuje się iteracyjnie algorytm wygładzania ASFT (ang. *Alternate Sequential Filter controlled by Topology*) [5]. W tej metodzie wygładzanie uzyskiwane jest poprzez morfologiczną operację otwierania-zamykania [37] z wykorzystaniem sfery o zmiennym promieniu. Zastosowanie ASFT na obrazie uzyskanym w wyniku zastosowania procedury topoEDEN daje w

efekcie dużo lepiej wyglądającą powierzchnię drzewa oskrzelowego. Przykładowy efekt zaprezentowano na rys. 19.



Rys. 19 Efekt uzyskany poprzez zastosowanie topoEDEN (wersja dodająca punkty), a następnie zastosowanie wygładzania ASFT.

W końcu rozszerzony algorytm może zostać opisany za pomocą kilku następujących kroków:

1. Generowanie tchawicy oraz powierzchni wyznaczającej granice drzewa.
2. Generowanie gałęzi drzewa z wykorzystaniem algorytmu podstawowego.
3. Wygięcie poszczególnych gałęzi drzewa.
Konwersja modelu drzewa do domeny grafiki objętościowej.
4. Iteracyjne, naprzemienne zastosowanie operacji EDEN oraz ASFT.

6 Szczegóły implementacyjne

Rozdział ten został w całości poświęcony kwestiom implementacyjnym omawianego algorytmu generowania drzew oskrzelowych. Początek tego rozdziału skupia się na sposobie generowania modelu podstawowego, oraz innych operacji generowanych w domenie grafiki powierzchniowej. Następnie omówiono realizację konwersji, takiego modelu do grafiki objętościowej, oraz sposób dodawania szumu.

Implementacja algorytmu podstawowego została w pełni zrealizowana z wykorzystaniem klas wchodzących w skład biblioteki klas VisualizationToolKit – VTK [20]. Za wyborem VTK na tle konkurencyjnych rozwiązań przemawia przede wszystkim bogactwo klas wchodzących w skład pakietu, przenośność, bardzo dobra dokumentacja oraz dostępność VTK w formie Wolnego Oprogramowania. Poniżej przedstawiono ścieżkę przetwarzania obrazu w VTK.



Rys. 20 Główna ścieżka wizualizacyjna w VTK

6.1 Generowanie płaszczyzny wyznaczającej granice drzewa

Generowanie płaszczyzny ograniczającej granice generowanego drzewa, za-implementowane zostało na bazie abstrakcyjnej klasy `vtkImplicitFunction`. W celu wyliczenia odpowiedniej powierzchni należy funkcję opisującą tę powierzchnię zdefiniować w metodzie `EvaluateFunction`. Część kodu klasy implementującej wyliczanie powierzchni ograniczającej przestrzeń przedstawiono na *listing 2*.

```
1. class SimpleOrganShape : public vtkImplicitFunction, public BasicObject{
2. private:
3.     vtkSampleFunction *imlFunction;
4.     vtkContourFilter *shapeContour;
5. public:
6.     //! Makro tworzy odpowiednie metody dla klasy dziedziczącej z klas bi-
blioteki VTK
7.     vtkTypeMacro(SimpleOrganShape,vtkImplicitFunction)
8.     //! Definicja tej metody generowana jest przez makro vtkStandardNewMa-
cro()
9.     staticSimpleOrganShape *New();
10.    //! Dokonuje obliczeń powierzchni oraz koniecznych konwersji.
11.    void Evaluate(){
12.        imlFunction->SetImplicitFunction(this);
13.        // Wymiary powierzchni.
14.        imlFunction->SetModelBounds(-20, 20, -20, 20, 0, 30);
15.        imlFunction->CappingOn();
16.        /* Klasa vtkImplicitFunction domyślnie tworzy na wyjściu
17.        obiekt klasy vtkImageData. W celu uzyskania danych vtkPolyData
18.        użytkownik zatrzymie się.*/
19.        shapeContour->SetInputConnection(imlFunction->GetOutputPort());
20.        shapeContour->GenerateValues(1, 1, 1);
21.        shapeContour->Update();
22.        //! Na bazie konturu dokonywana jest inicjalizacja danych klasy BasicOb-
ject
23.        BasicObject::SetPipeline(shapeContour);
24.    }
25. protected:
26.     //! Wyliczanie funkcji gradient nie zostało zaimplementowane.
27.     virtual void EvaluateGradient(double x[3], double g[3]){}
28.     //! Metoda służy do wyliczenia równania opisującego powierzchnię.
29.     virtualdouble EvaluateFunction(double x[3]){
30.         //! Równanie płaszczyzny
31.         return (2 * pow(15,-3)*(pow((pow(x[0],2) + pow((1.5*x[1]),2)),2)),2))/x[2];
32.     }
```

```
33. };
```

Listing 2 Częściowa definicja klasy implementującej powierzchnię ograniczającą obszar generowania drzewa.

Należy zauważyć, że klasa ta odpowiada również za wyliczenie płaszczyzny dla konkretnych wartości, oraz przekonwertowanie obliczonej płaszczyzny do triangulacyjnej siatki za pomocą obiektu klasy `vtkContourFilter`. Obiekty klasy `SimpleOrganShape` dziedziczą również po klasie `BasicObject`, która z racji silniejszego związku z innymi częściami kodu została opisana w dalszej części pracy.

6.2 Gałęzie drzewa

Jak wspomniano w podrozdziale 5.1 każda gałąź opisana jest przez parametryczne równanie spirali. Po wygenerowaniu odpowiedniej ilości punktów za pomocą tego równania, konieczne jest wygenerowanie z nich linii opisującej gałąź, następnie tak wygenerowaną linie, oraz promień gałęzi przekazuje się do klasy `vtkTubeFilter`, która generuje tubę o zadanym promieniu wzduż zadanej linii. Przykładowy kod demonstrujący działanie tego filtra znajduje się na *listing 3*.

```
1. Branch::Branch(double r, double p_h, twisting p_twist) : BasicObject(),  
    h(p_h){  
2.     points = vtkPoints::New();  
3.     if(p_twist == TWIST_ON){  
4.         nV = h + 10 // dla niektórych gałęzi przydatne jest zwiększenie o kilka  
        punktów ich długości;  
5.         rS = r / 4.0 // promień spirali;  
6.         nCyc = 1.0 // liczba skręceń spirali - gałęzi;  
7.         EquationRandomizer *equationRandomizer = new EquationRandomizer();  
8.         equationRandomizer->SearchAnotherEquation();  
9.         boost::tuple<double, double> result;  
10.        for(int i = 0; i < nV; i++){  
11.            result = equationRandomizer->GetValue(rS, nCyc, nV, i);  
12.            vX = boost::get<0>(result);  
13.            vZ = boost::get<1>(result);  
14.            vY = h * i / nV;  
15.            points->InsertPoint(i, vX, vY, vZ);  
16.        }  
17.    } else{  
18.        nV = static_cast<int>(SHAPE_RES*r);  
19.        rS = 0.0;
```

```
20.    nCyc = 1.0;
21.   for(inti = 0; i<nV; i++){
22. // Współrzędne spirali
23.     vX = 0;
24.     vZ = 0;
25.     vY = h * i / nV;
26.     points->InsertPoint(i, vX, vY, vZ);
27.   }
28. }
29. // Dodanie kolejnych punktów do linii
30. lines = vtkCellArray::New();
31. lines->InsertNextCell(nV);

32. for (inti = 0; i<nV; i++){
33. lines->InsertCellPoint(i);
34. }

35. polyData = vtkPolyData::New();
36. polyData->SetPoints(points);
37. polyData->SetLines(lines);

38. // vtkTubeFilter jest wrażliwy na zdublowane punkty, należy je usunąć
39. vtkCleanPolyData *cleaner = vtkCleanPolyData::New();
40. cleaner->SetInput(polyData);

41. //Stworzenie tuby dookoła linii
42. tubeFilter = vtkTubeFilter::New();
43. tubeFilter->SetInputConnection(cleaner->GetOutputPort());
44. tubeFilter->SetRadius(r);
45. //! Ustawia rozdzielcość gałęzi. Zobacz @ref AlgorithmConstData.h
46. tubeFilter->SetNumberOfSides(SHAPE_RES);
47. tubeFilter->CappingOn();

48. triFilter = vtkTriangleFilter::New();
49. triFilter->SetInput(tubeFilter->GetOutput());
50. triFilter->Update();

51. cleaner->Delete();
52. delete equationRandomizer;
53. BasicObject::SetPipeline(triFilter);
54. };
```

Listing 3 Konstruktor korzenia drzewa oskrzelowego.

Jak widać na *listing 3* gałąź po wygenerowaniu poddawana jest działaniu filtra konwertującego reprezentację komórek siatki do reprezentacji, trójkątnej. Należy w tym miejscu naznaczyć, iż wiele opracowanych dotychczas algorytmów działających w domenie grafiki powierzchniowej potrafi operować jedynie na siatkach opisanych za pomocą trójkątów, stąd na etapie generowania gałęzi jest ona od razu konwertowana na typ pożądanego przez resztę kodu.

6.2.1 Generowanie kolejnych rozwidleń

Jednym z ważniejszych aspektów generowania drzewa są współrzędne nowych gałęzi. Współrzędne nowej pary gałęzi są za każdym razem tak określone, by były zgodne z punktem rozwidlenia wyznaczonym przez ich rodzica.

Klasa `vtkTransform` umożliwia zdefiniowanie hierarchii transformacji, w lokalnym układzie współrzędnych rodzica co ułatwia dokonywanie geometrycznych przekształceń na jego dzieciach. Hierarchia ta pozwala także manipulować obiektami nadzewnętrznymi bez konieczności ręcznej aktualizacji współrzędnych obiektów podrzędnych. Aby uprościć zarządzanie hierarchią transformacji opracowano klasę `BasicObject`. Zdefiniowanie odpowiednich połączeń dokonywane jest za pomocą metody `SetPipeline` przedstawionej na *listing 4*. Zdefiniowanie hierarchii transformacji pozwala na znaczne uproszczenie procesu generowania nowych gałęzi.

```
1. //! \parm *objShpe – np. gałąź opisana za pomocą grafiki powierzchniowej
2. void SetPipeline(vtkAlgorithm *objShape){
3.   // Przefiltrowanie typu siatki obiektu na trójkątny
4.   polyDataTransFilter->SetInputConnection(objShape->GetOutputPort());
5.   polyDataTransFilter->SetTransform(objTransform);
6. }
7. //! \parm *relativeTrs – transformacja nadzawanego obiektu
8. void SetPipeline(vtkTransform *relativeTrs, vtkAlgorithm *objShape){
9.   //! Powiązanie transformacji obiektów
10.  objTransform->SetInput(relativeTrs);
11.  objTransform->Update();
12. //! Powiązanie obiektu z transformacjami
13.  polyDataTransFilter->SetInputConnection(objShape->GetOutputPort());
14.  polyDataTransFilter->SetTransform(objTransform);
15. }
```

Listing 4 Metoda ustawiająca hierarchię transformacji pomiędzy gałęziami drzewa

Klasa `vtkTransformPolyDataFilter` (patrz *listing 4*) umożliwia zastosowanie odpowiedniej transformacji bezpośrednio na zestawie danych opisujących obiekt, bez konieczności odwoływania się do klasy `vtkActor`, która spełnia to zadanie w klasycznej ścieżce wizualizacyjnej wykorzystywanej w VTK (patrz rys. 20).

6.3 Podział obszaru, w którym znajduje się gałąź

Zgodnie z *regułą 4* obszar gałęzi-rodzica dzielony jest na dwa obszary gałęzi-dzieci. Aby podzielić obszar rodzica w pierwszej kolejności niezbędne jest zdefiniowanie płaszczyzny dzielącej ten obszar (patrz rys. 9). Wspomniana wcześniej klasa `vtkTransform` umożliwia uzyskanie normalnej transformowanego obiektu, posiadając normalną obiektu możemy za pomocą klasy `vtkPlane` określić odpowiednią płaszczyznę. Przykładowy kod definiujący odpowiednią płaszczyznę został przedstawiony na *listing 5*.

```
1. void TreeBranchProducer::ComputeCutPlane(boost::shared_ptr< Branch > branch, float
degrees){  
2.     /* Płaszczyzna podziału obszaru zorientowana jest o 90°(reguła 8 i 8a) względem *
normalnej obiektu dlatego należy przed pobraniem normalnej obrócić obiekt względem
osi Y  
3.     */  
4.     //! Ustawienie normalnej w odpowiedniej pozycji  
5.     branch->GetTransform()->RotateY(-degrees);  
6.     //! Zaczepienie płaszczyzny na środku gałęzi  
7.     cutPlane->SetOrigin(branch->GetCenter());  
8.     //! Ustawienie normalnej płaszczyzny zgodnie z normalą gałęzi po wydruku  
9.     branch->GetTransform()->TransformNormal(cutPlane->GetNormal(),normal);  
10.    cutPlane->SetNormal(normal);  
11.    //! Odtworzenie obrotu gałęzi sprzed stanu przed pobraniem normalnej  
12.    branch->GetTransform()->RotateY(degrees);  
13. }
```

Listing 5 Wyznaczanie płaszczyzny podziału obszaru gałęzi

Kolejnym krokiem po zdefiniowaniu płaszczyzny podziału jest podział danego obszaru. Odpowiednia metoda klasy `SpaceDivision` została zaprezentowana na *listing 6*.

```
void SpaceDivision::ClipBoundaries(vtkPlane *plane, vtkPolyData* surfaces,
vtkPolyData *tmp[4]){
    //Przetnij obiekt zgodnie z płaszczyzną.
    clipper->SetInput(surfaces);
```

```

clipper->SetClipFunction(plane);
clipper->Update();
//Tylko jeśli siatka wyjście zawiera jakieś wielokąty – dokonano podziału na
//dwa nowe obiekty
if(clipper->GetOutput()->GetNumberOfPolys() > 0 && clipper->GetClippedOutput()->GetNumberOfPolys() > 0){
    /**
     * for(int i = 0; i< 4;i++) {
     *     tmp[i] = vtkPolyData::New();
     * }
     * tmp[0]->DeepCopy(AddBoundary(clipper->GetOutput()));
     * tmp[1]->DeepCopy(clipper->GetOutput());
     * tmp[3]->DeepCopy(clipper->GetClippedOutput());
     //Ostatni region ze ścianą
     * tmp[2]->DeepCopy(AddBoundary(tmp[3]));
    } else {
        for(int i = 0; i< 4;i++)
            tmp[i] = NULL;
    }
}

```

Listing 6 Podział obszaru na dwa nowe, za pomocą płaszczyzny podziału

Metoda ta, na początku inicjalizuje obiekt klasy `vtkClipPolyData` –`clipper` następnie dokonywane jest sprawdzenie poprawności danych uzyskanych na wyjściu obiektu `clipper`. Jak wynika z *listing 6* metoda ta kopiuje do tablicy zawierającej nowe obszary, obszar zawierający ścianę w miejscu przecięcia oraz jego odpowiednik nie posiadający tejże ściany. Konieczność posiadania obszaru zawierającego wspomnianą ścianę wiąże się z kolejnymi podziałami. Natomiast wersja jej nie posiadająca, niezbędna jest, jak zostanie to pokazane w dalszej części pracy do zrealizowania 9 *reguły* algorytmu podstawowego. Na rys. 10 przedstawiono obszar uzyskany za pomocą kodu z *listing 6*.

6.4 Wykrywanie opuszczenia regionu przez gałąź

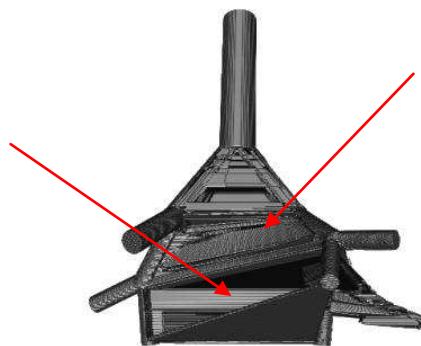
Jak wspomniano w rozdziale 4.1.1, zgodnie z *regułą 9* zakończenie generowania nowych gałęzi w danym kierunku powinno zakończyć się w momencie, gdy gałąź opuści swój segment. W celu zrealizowania tego założenia przeprowadzono wykrywanie kolizji pomiędzy gałęziami a ścianami ich segmentów. Odpowiednie algorytmy wykrywania kolizji niestety nie znajdują się bezpośrednio w VTK, dlatego wykorzystano rozwijaną niezależnie klasę `vtkCollisionDetectionFil-`

ter [22]. Klasa `vtkCollisionDetectionFilter` jest filtrem, który na wejściu przyjmuje dwa obiekty zbudowane z siatki wielokątowej. Wykrywanie kolizji zostało oparte o zorientowane bryły ograniczające (ang.*oriented bounding box*) [9].

6.5 Konwersja modelu opisanego siatką wielokątową do grafiki objętościowej

Konwersja modelu powierzchniowego do modelu objętościowego może zostać zrealizowana na kilka sposobów. Jednak z punktu widzenia dalszych przekształceń wygenerowanego modelu interesujące są te metody, które zapewniają wypełnienie wnętrza konwertowanego obiektu. Metody te można dalej podzielić na dwie grupy w zależności od tego czy dany obiekt da się przedstawić za pomocą innych obiektów, czy też należy go rozpatrywać jako całość. Rozpatrywanie drzewa oskrzelowego, jako jednego niepodzielnego obiektu niesie za sobą kilka wspomnianych dalej problemów.

Najwydajniejszą metodą konwersji dostępną w VTK jest metoda zaimplementowana w klasie `vtkPolyDataToImageStencil`. Metoda ta polega na przeszukiwaniu za pomocą promienia wzdłuż osi X kolejnych ścian obiektu wyznaczonych przez płaszczyznę opisaną na osiach Y, Z . Promień ten wypełnia wokselami przestrzeń zawartą pomiędzy dwoma dowolnymi ścianami obiektu, nie dokonując przy tym wystarczających sprawdzeń. Dlatego w momencie, gdy promień zostanie załączony na granicy obiektu posiadającego „zatokę” i trafi on w przeciwną stronę ścianę przestrzeń ta zostanie wypełniona. Błędy, które powstają podczas takiej konwersji, zostały przedstawione na rys. 21.



Rys. 21 Błędy powstające podczas konwersji drzewa algorytmem z klasy `vtkPolyDataToImageStencil`.

W VTK znaleźć można jeszcze jedną obiecującą metodę opartą o funkcję matematyczną opisującą obiekt. Metoda ta niestety nie jest wystarczająco wydajna, a ponadto może w niej dochodzić do „wycieków” wokseli. Warto w tym miejscu wspomnieć o możliwości konwersji obiektu algorytmem „promienia przeszywającego” [29] zaimplementowanego w programie binvox [26], która ze wspomnianych do tej pory metod oferuje najlepsze wyniki.

Zadowalające wyniki można uzyskać również poprzez konwersję drzewa „gałąź po gałęzi” z zastosowaniem wcześniej wspomnianej metody konwersji zaimplementowanej w klasie `vtkPolyDataToImageStencil`. Metoda ta pozwala także na przypisanie już na etapie konwersji odpowiednich wartości wokselom konwertowanej gałęzi, bez późniejszej klasyfikacji wokseli, którą należałyby wykonać po konwersji całego drzewa traktowanego jako niepodzielny obiekt. Właśnie ta metoda została wykorzystana na potrzeby niniejszej pracy.

Całym procesem konwersji zarządza klasa `TreeBranchVoxelizer`. Klasa ta wykorzystuje bibliotekę Boost.Thread [50] do uruchamiania konwersji gałęzi na oddzielnych wątkach procesora. *Listing 7* prezentuje metodę wspomnianej klasy, która uruchamia wielowątkową konwersję. Jak widać na wspomnianym listingu, metoda `Start` tworzy kolejne obiekty klasy `TreeBranchVoxelizer` wiążąc metodę `Generate` z nowym wątkiem. Do zabezpieczenia sekcji krytycznej wykorzystano mechanizm muteksów.

```
1. /* Metoda uruchamia proces konwersji obiektu w grafice powierzchniowej na
   obiekt w grafice
2. objętościowej. Jeśli lista zawiera więcej elementów niż jeden uruchom kon-
   wersje na wątkach.
3. */
4. void TreeBranchVoxelizer::Start() {
5.     int i;
6.     //Stworzenie i przypisanie wspólnego dla wszystkich wątków Iteratora;
7.     ListIterator = new std::list<boost::shared_ptr<Branch>>::iterator;
8.     *(ListIterator) = BranchList.begin();
9.     if( BranchList.size() == 1 ) {
10.         Generate();
11.     } else {
12.         if(BranchList.size() >= MAX_THREAD){
```

```

13.     i = MAX_THREAD;
14. } else {
15.     i = BranchList.size();
16. }
17. for(; i > 0; i--) {
18.     t_group.create_thread(boost::bind(&TreeBranchVoxelizer::Generate, new
    TreeBranchVoxelizer(*this)));
19. }
20. t_group.join_all();
21. }
22. delete ListIterator;
23. }
```

Listing 7 Uruchamianie konwersji modelu do grafiki objętościowej na wątkach

```

1. void TreeBranchVoxelizer::Generate() {
2.     while(true) {
3.     /*
4.     * Obiekty z STL nie są bezpieczne podczas użytkowania na wątkach.
5.     * Dlatego nie usuwamy nic z listy w tym miejscu by nie zmienić jej rozmiaru
6.     * co doprowadziłoby do nieoczekiwanej zakończenia się programu, gdyż Iterator
    wspólny jest dla wszystkich wątków.
7.     */
8.     mutex.lock();
9.     if(*ListIterator) != BranchList.end()) {
10.         actualConvertBranch = *(*ListIterator);
11.         *(*ListIterator)++;
12.     } else {
13.         mutex.unlock();
14.         break;
15.     }
16.     mutex.unlock();
17.     Convert();
18.     double offset[3];
19. // Wyliczenie przesunięcia obrazu zawierającego gałąź względem obrazu wynikowego
20.     offset[0] = origin[0] > origin2[0] ? fabs(origin[0] - origin2[0])/spacing :
    fabs(origin2[0] - origin[0])/spacing;
21.     offset[1] = origin[1] > origin2[1] ? fabs(origin[1] - origin2[1])/spacing :
    fabs(origin2[1] - origin[1])/spacing;
22.     offset[2] = origin[2] > origin2[2] ? fabs(origin[2] - origin2[2])/spacing :
    fabs(origin2[2] - origin[2])/spacing;
23.     int dim2[3];
24.     tmpIMG->GetDimensions(dim2);
25.     for (register unsigned int z = 0; z < dim2[2]; z++) {
26.         for (register unsigned int y = 0; y < dim2[1]; y++) {
27.             for (register unsigned int x = 0; x < dim2[0]; x++) {
```

```
28.         if( *( static_cast<unsigned char *>( tmpIMG->GetScalarPointer( x, y, z ) ) !=  
    BACKGROUND_VAL) {  
29.             *( static_cast<unsigned char *>( rawTree->GetScalarPointer( (x + offset[0]), (y + offset[1]), (z + offset[2]) )  
    ) ) = *( static_cast<unsigned char *>( tmpIMG->GetScalarPointer( x, y, z ) ) );  
30.         }  
31.     }  
32. }  
33. }  
34. }  
35. if(tmpIMG != NULL) {  
36.     tmpIMG->Delete();  
37. }  
38. }  
39. }
```

Listing 8 Metoda konwertująca kolejne gałęzie.

6.6 Implementacja znieksztalceń w domenie grafiki powierzchniowej

Wspomniane w rozdziale 5.2 znieksztalcenia zostały zaimplementowane z wykorzystaniem biblioteki PINK [8], która zawiera około 200 algorytmów filtracji oraz segmentacji obrazu. Biblioteka ta powstała do celów badawczych oraz edukacyjnych, i jest rozpowszechniania jako Wolne Oprogramowanie.

Na potrzeby omawianej pracy wszystkie wykorzystane funkcje wchodzące w skład biblioteki PINK zostały obudowane adapterami, dziedziczącymi po klasie QThread, która pozwala na uruchamianie fragmentów kodu na oddzielnych wątkach. Klasa ta pochodzi z zestawu bibliotek Qt [6]. Definicja przykładowego adaptora została przedstawiona na *listing 9*. Należy zaznaczyć w tym miejscu, iż wykorzystanie, dwóch różnych bibliotek, tak jak Boost.Thread oraz QThread do realizacji tego samego zadania związane jest z ewolucją programu realizującego opisywany w tej pracy algorytm. Na początkowym etapie rozwoju program ten nie posiadał graficznego interfejsu użytkownika, a ponadto postanowiono wykorzystać biblioteki Boost do realizacji zadań takich jak wielowątkowość. W raz z rozwojem programu powstał opisany, w kolejnym rozdziale interfejs graficzny oparty o Qt, a z czasem część nowych klas została tak napisana by korzystać już tylko z Qt.

```
1. class pinkAsft : public QThread {  
2.     Q_OBJECT  
3.     private:  
4.         structxvimage *image;  
5.         int32_t rayonmax;  
6.         int32_t connex;  
7.         boost::tuple<std::string, int, int, std::string>&argv;  
8.     protected:  
9.         virtual void run();  
10.    public:  
11.        pinkAsft(boost::tuple<std::string, int, int, std::string>&);  
12.        virtual ~pinkAsft();  
13.    signals:  
14.        QThread *endCompute(QThread *);  
15.    };
```

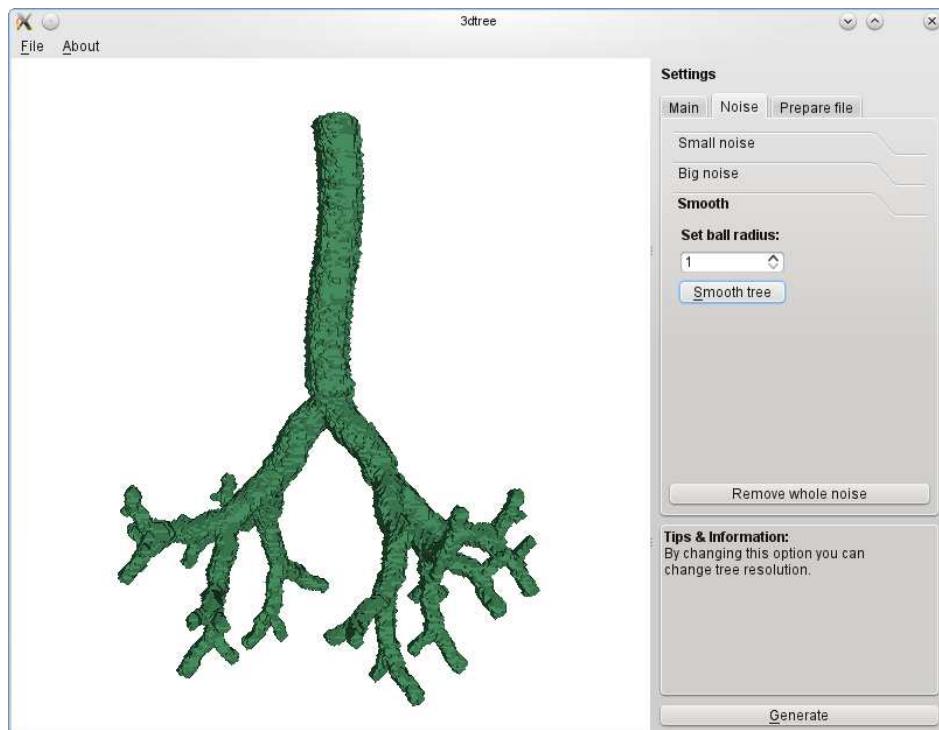
Listing 9 Definicja klasy, adaptera, który przesłania algorytm ASFT z biblioteki PINK

6.7 Interfejs użytkownika

Interfejs użytkownika został, jak wcześniej wspomniano zbudowany z wykorzystaniem bibliotek wchodzących w skład pakietu Qt. Na rys. 22 przedstawiono główne okno aplikacji. Na prawej stronie interfejsu aplikacji umieszczono podzielony na zakładki panel pozwalający w wygodny sposób na zdefiniowanie takich parametrów drzewa jak wielkość woksela w obrazie wynikowym, minimalną średnicę gałęzi jaką może dalej poddać rozwidlaniu, a także parametry szumu. Zakładka związana z zaszumieniem pozwala na określenie odpowiedniej ilości zaszumienia, a także jego rodzaju. Na panelu umieszczono również, zakładkę, z której można wygenerować plik zawierający losowe wartości wokseli drzewa oskrzelowego w wybranym przez użytkownika przedziale wartości.

Jak wspomniano wcześniej fragmenty kodu uruchamiane z poziomu głównego okna, zostały tak napisane by wykorzystywać mechanizm wielowątkowości, dzięki czemu podczas generowania modelu, i dodawania zaszumienia główne okno cały czas jest aktywne i umożliwia podgląd danych wygenerowanych wcześniej.

Warto zaznaczyć, iż program posiada w obecnej wersji podstawowe wsparcie dla widoku stereoskopowego, widok ten można aktywować poprzez wybranie z klawiatury klawisza o numerze 3. Aktualnie wspieraną techniką widoku stereoskopowego jest Anaglif [48] Więcej na temat widoku stereoskopowego napisano w rozdziale 7.



Rys. 22 Interfejs użytkownika.

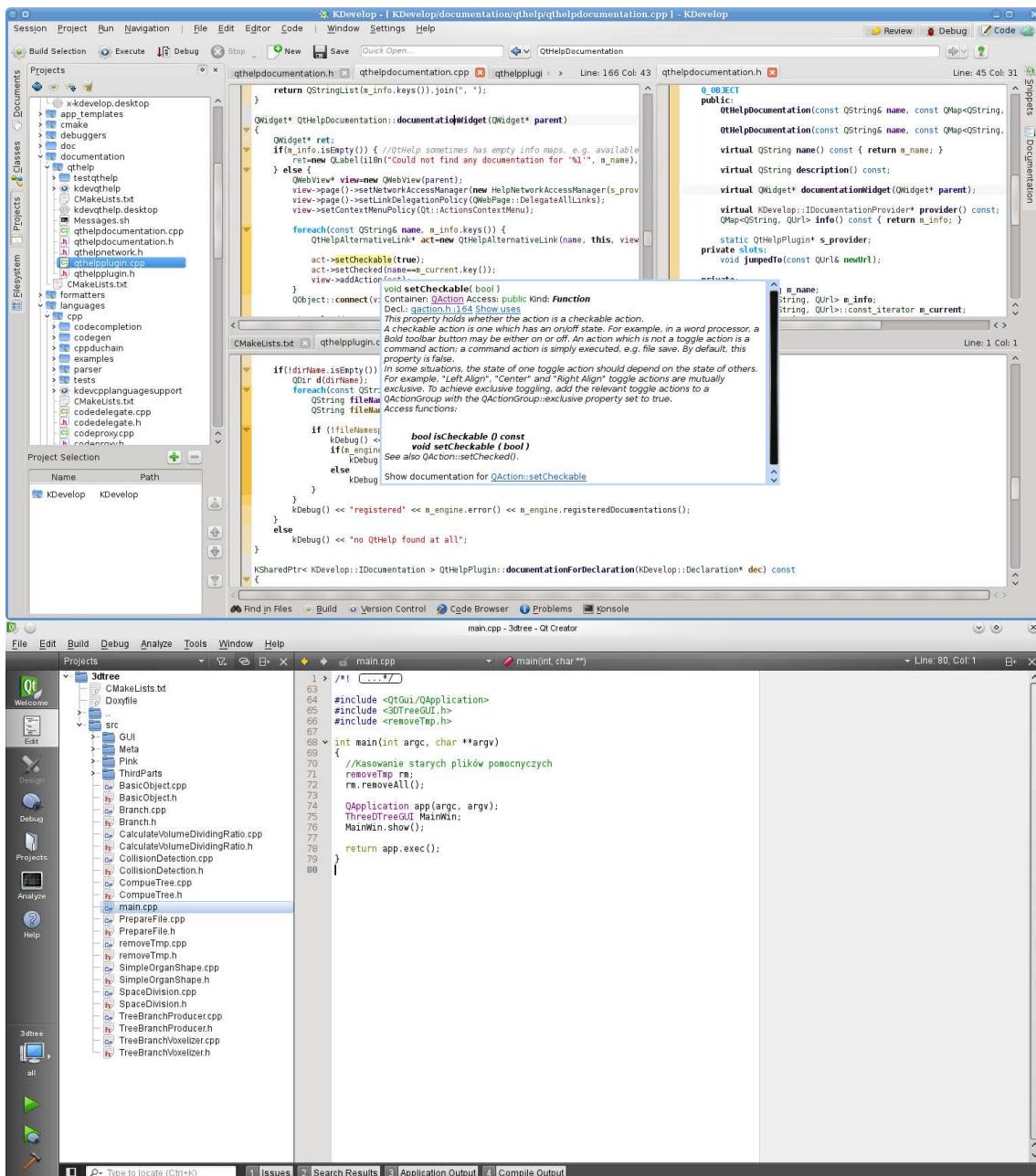
Główna część interfejsu zajmuje obraz generowany przez VTK, wyświetlany na oknie za pomocą klasy **QVTKWidget**. Należy zwrócić uwagę, iż nie należy umieszczać w konstruktorze okna korzystającego z **QVTKWidget** kodu związanego z wyświetlaniem komponentów VTK. Kod taki należy umieścić w metodzie **showEvent**, która wywoływana jest zaraz po konstruktorze. Za takim podejściem przemawia, przede wszystkim, fakt iż **QVTKWidget** może nie być w całości zainicjowany podczas tworzenia komponentu, na którym ma być rysowany.

7 Wykorzystane narzędzia informatyki

Rozdział ten skupia się na omówieniu wykorzystanych narzędzi informatycznych, które posłużyły do zbudowania, oraz zweryfikowania modelu rozszerzonego. W pierwszej części tego rozdziału skupiono się na omówieniu środowiska developerskiego - IDE (ang. *Integrated Development Environment*), narzędzi CASE, etc. Druga część natomiast została poświęcona narzędziom, które pozwoliły na weryfikowanie uzyskiwanych wyników. Warto w tym miejscu zaznaczyć, iż bardzo ważne było, takie napisanie kodu tworzonego oprogramowania by możliwe było skompilowanie, i uruchomienie go niezależnie od platformy systemowej. Co też udało się osiągnąć poprzez odpowiedni dobór wykorzystanych narzędzi oraz bibliotek. Dzięki temu, omawiana w niniejszej pracy implementacja działa na systemach z rodziny GNU/Linux oraz Windows. Niestety, z braku dostępu do odpowiednich zasobów sprzętowych nie udało się przeprowadzić testów na systemie OS X.

7.1 Środowisko developerskie

Na początku prowadzonych prac, gdy projekt był w fazie weryfikacji założeń wykorzystywano środowisko Kdevelop [40], wchodzące w skład narzędzi developerskich opracowanych dla środowiska graficznego KDE. Z czasem gdy zdecydowano się na opracowanie interfejsu graficznego projekt został przeniesiony do środowiska QtCreator, które wchodzi w skład SDK (ang. *Software Development Kit*) opracowanego do rozwoju oprogramowania wykorzystującego zestaw bibliotek Qt. Innym, lecz mniej istotnym powodem zmiany środowiska developerskiego był narzut na zasoby systemowe ze strony Kdevelop. Dzięki wykorzystaniu opisanego w dalszej części pracy narzędzia CMake [19], proces zmiany środowiska developerskiego nie był obarczony żadnymi problemami, i utrudnieniami. Warto wspomnieć, że oba środowiska posiadają wiele przydatnych funkcji; rozbudowane narzędzia do refaktoryzacji kodu, podpowiadanie składni, podświetlanie składni, bogate możliwości konfiguracji oraz spójny i intuicyjny interfejs. Na rys. 23 zaprezentowano interfejs obu środowisk.



Rys. 23 Na powyższych rysunkach przedstawiono; na górze KDevelop oraz na dole QtCreator

7.1.2 Narzędzia służące do budowania projektu

Narzędzie tego typu są niezbędne podczas pracy z większymi projektami, i praktycznie mało kto może się bez nich obyć. Każde środowisko IDE, wykorzystuje jakieś narzędzia do budowy tworzonych w nim projektów. Należy jednak zwrócić

uwagę, iż nie zawsze jest to narzędzie, które zapewni nam odpowiedni poziom niezależności od tegoż właśnie środowiska. Warto więc sięgać po rozwiązania uniwersalne, pozwalające na łatwą zmianę, środowiska developerskiego, lub nawet pozwalające na zbudowanie projektu poza nim.

Przykładem dobrego narzędzia posiadającego wymienione wyżej cechy przedstawić jest CMake.[19] Narzędzie to pozwala na wygenerowanie odpowiednich plików Makefile programu make, plików projektu dla wielu środowisk programowania, w tym Visual Studio. Dzięki temu, uniezależnia tworzony projekt, od wykorzystywanych przez developerów środowisk, oraz w pewnym stopniu narzędzi. Po stworzeniu projektu dla CMake możemy w bardzo łatwy sposób wygenerować projekt dowolnego wspieranego środowiska. Na *listing 10* przedstawiono przykładowy plik CMakeLists.txt.

```
1. project(demo) # Definicja projektu
2. cmake_minimum_required(VERSION 2.8) #Minimalna wersja CMake z jaką zgodny
   jest projekt
3. aux_source_directory(. SRC_LIST) #Zdefiniwanie plików źródłowych projektu
4. add_executable(${PROJECT_NAME} ${SRC_LIST}) # Określenie nazwy pliku wynikowego
   oraz plików, z których ma zostać on zbudowany
```

Listing 10 Przykładowy plik CMakeLists.txt

Jako narzędzie zarządzania budowaniem projektu posłużył popularny w środowisku GNU/Linux Make. Należy w tym miejscu zaznaczyć, iż narzędzie to często kojarzone jest tylko z procesem budowania projektów programistycznych, natomiast umożliwia ono wiele więcej. Program Make pozwala po prostu na zdefiniowanie przekształcenia, które ma zostać wykonane na pliku źródłowym tak by uzyskać plik wynikowy. Przykładem innego zastosowaniem Make może być Makefile, który służy do automatycznego i cyklicznego generowania plików audio zawierających wyjście syntezatora mowy dla, którego źródłem są strony WWW (patrz *listing 11* i *12*). Należy wspomnieć, że narzędzie Make doczekało się wielu implementacji, i dostępne jest na szeroki wachlarz systemów operacyjnych.

```
1. SCRIPT = ./scripts/tts.sh
2. .SUFFIXES: .ogg .html
3. .html.ogg:
   a. $(SCRIPT) $<
4. all: index.ogg moons.ogg moons2.ogg missions.ogg missions2.ogg missions3.ogg
       missions4.ogg geo.ogg basicinfo.ogg
```

Listing 11 Makefile konwertujący pliki html do plików audio.

```
#!/bin/bash
1. var=`echo $1 | cut -f1 -d .` #Wydobycie nazwy pliku z pominięciem rozszerzenia

#Jeśli plik posiada sekcję TTS następuje przekonwertowanie pliku html na plik txt, na bazie,
którego generowany jest plik audio z wykorzystaniem syntezatora mowy.
2. sed -n '/<!--start TTS -->/,/<!-- end TTS -->/p' $var.html | tr 'h1' 'span' | html2text >>
$var.txt; milena_book -o $var.txt; rm -rf $var.txt;
```

Listing 12 Wykorzystany w *listing 11* skrypt generujący pliki audio z plików html z wykorzystaniem syntezatora mowy MILENA [25].

7.1.3 Narzędzia ułatwiające znalezienie wadliwego kodu

Wykorzystywanie oprogramowania służącego do debuggowania kodu jest niemalże koniecznością podczas procesu tworzenia oraz konserwacji oprogramowania. Narzędzia tej klasy dzieli się na grupy. Z punktu widzenia niniejszej pracy wykorzystano klasyczny debugger służący do krokowego uruchamiania oprogramowania, w raz z możliwością podglądu zawartości pamięci, wykonywaniem skoków, czy ustawnianiem punktów zatrzymania. Podczas realizacji niniejszej pracy wykorzystano GDB (*GNU DeBugger*) [10]. GDB jest dojrzałym projektem, dla którego wymienione wcześniej środowiska developerskie oferują wsparcie; między innymi oferują dla niego wygodny interfejs graficzny.

Następną klasą wykorzystanych debuggerów, stanowią analizatory pamięci, które umożliwiają odnalezienie niezwolnionej przez program w trakcie jego wykonywania pamięci. Jednym, z lepszych tego typu narzędzi jest Valgrind [43]. Program ten oferuje szereg przydatnych opcji, oraz generuje szczegółowe logi.

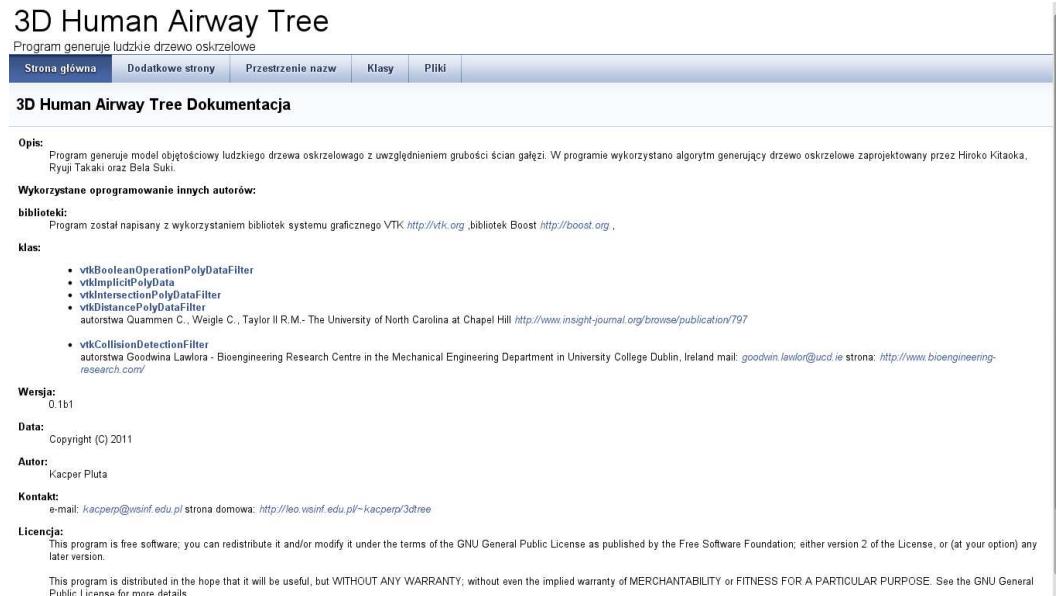
7.1.4 System kontroli wersji

Systemy zarządzania wersjami pozwalają nie tylko na tworzenie oprogramowania w zespołach ale również na wygodne testowanie koncepcji i ewentualny

powrót do wcześniejszej wersji bez konieczności ręcznego przywracania kopii kodu. Na rynku dostępnych jest wiele systemów zarządzania wersjami. Do realizacji niniejszej pracy wykorzystano popularny i niezawodny system git [41]. System ten był wielokrotnie wykorzystywany podczas realizacji dużych przedsięwzięć informatycznych (Oryginalnie został zaprojektowany przez Linusa Torvaldsa do zarządzania projektem Linux).

7.1.5 Narzędzia wspomagające generowanie dokumentacji

Obecnie istnieje wiele różnych narzędzi wspomagających tworzenie dokumentacji oprogramowania wprost z kodu źródłowego. Polega to na dodawaniu do komentarzy specjalnych, rozpoznawanych przez dany generator znaczników. Podczas realizacji niniejszego projektu zdecydowano się na wybór popularnego Doxygena [45]. Doxygen jest dość elastyczny i pozwala na używanie znaczników innych popularnych generatorów takich jak JavaDoc oraz Qtdoc. Ponadto oferuje on możliwość wygenerowania dokumentacji w wielu formatach np. HTML(patrz rys. 24), LaTeX.



Rys. 24 Dokumentacja html projektu wygenerowana przez Doxygen.

7.2.1 Wybrane operatory pakietu PINK użyte do zweryfikowania poprawności uzyskanych wyników

Biblioteka PINK zawiera narzędzia umożliwiające badanie topologii obiektów 3D. W rozdziale 9.1 omówiono szerzej zagadnienie topologii drzewa oskrzelowego. Należy w tym miejscu zaznaczyć, iż poznanie informacji na temat posiadowanych przez obiekt tuneli i pustek (ang. *cavity*) nie jest trywialne w przestrzeni dyskretnej gdzie obiekt reprezentowany jest za pomocą zbioru wokseli. Dlatego, też badanie topologii obiektu 3D przez wchodzący w skład PINK program 3dInvariants, odbywa się po wcześniejszej konwersji obiektu 3D do przestrzeni Khalimsky'ego [17]. Badanie występowania tuneli w wygenerowanym drzewie oskrzelowym realizowano za pomocą algorytmu zamykania tuneli [1]. Następnie odejmowano od wyniku tego operatora obraz źródłowy. W ten sposób otrzymywano obraz, w którym każdy tunel występujący w drzewie oskrzelowym jest reprezentowany przez lata, która go zamyka. Po naniesieniu tychże danych na wygenerowany model np. z wykorzystaniem opisanego dalej oprogramowania do wizualizacji Amira [20] możliwe było dokładne przestudiowanie miejsc występowania problemów.

7.2.2 Wizualizacja weryfikowanych danych

We wcześniejszym rozdziale wspomniano o wizualizacji dwóch obrazów poprzez naniesienie ich na siebie w celu zobrazowania miejsc, w których w jednym z tych obrazów powstały problemy opisane drugim obrazem. Te i wiele innych wizualizacji można zrealizować w wielu aplikacjach specjalnie w tym celu zaprojektowanych. Jednym z nich jest Amira [20], która jest kompletnym narzędziem służącym do pracy z trójwymiarowymi obrazami, w tym obrazami wolumetrycznymi. W jej skład wchodzą moduły umożliwiające przetwarzanie, analizę oraz wizualizację tego typu obrazów.

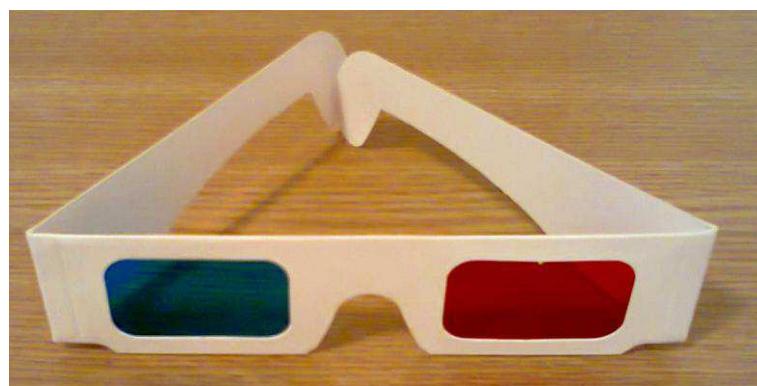
8 Wizualizacja stereoskopowa

Stereoskopia to technika obrazowania oddającą wrażenie głębi przestrzennej, bazująca na sposobie „składania” przez ludzki mózg obrazów w korze wzrokowej. Każde oko widzi zawsze trochę inny obraz, i aby uzyskać w obserwowanym obrazie wrażenie głębi przestrzennej należy dostarczyć do mózgu obraz widziany z perspektywy lewego i prawego oka podzielony na dwa niemal identyczne obrazy nazywane stereoparami. Jak już wspomniano stereopary są niemal identyczne i różnią się nieznacznie kątem widzenia obiektów, oraz szczegółami wzajemnego przesłanania się obiektów w obrazie.

Pierwsze obrazy stereoskopowe powstały w latach 40 XIX wieku za sprawą wynalezienia przez Sir Charlesa Wheatstonea stereoskopu zwierciadlanego. Na przestrzeni lat powstało wiele technik stereoskopowych pozwalających na uzyskanie różnej jakości efektu. W niniejszej pracy zostaną omówione dwie techniki: anaglif(ang. *anaglyph*) [48] oraz *side-by-side*.

8.1 Technika anaglif

Technika ta polega na nałożeniu na siebie przesuniętych poziomie identycznych obrazów posiadających różny dla każdego oka filtr barwny, najczęściej chromatycznie przeciwwstawnych. Popularnie stosowane są filtry czerwono-cyjanowe. Kiedy obserwator spojrzy na tak zakodowany obraz przez odpowiednie okulary zaprezentowane na rys 25 każdy z filtrów będzie dobierany tylko przez jedno oko. Dzięki czemu do kory wzrokowej zostanie dostarczona odpowiednia informacja po-



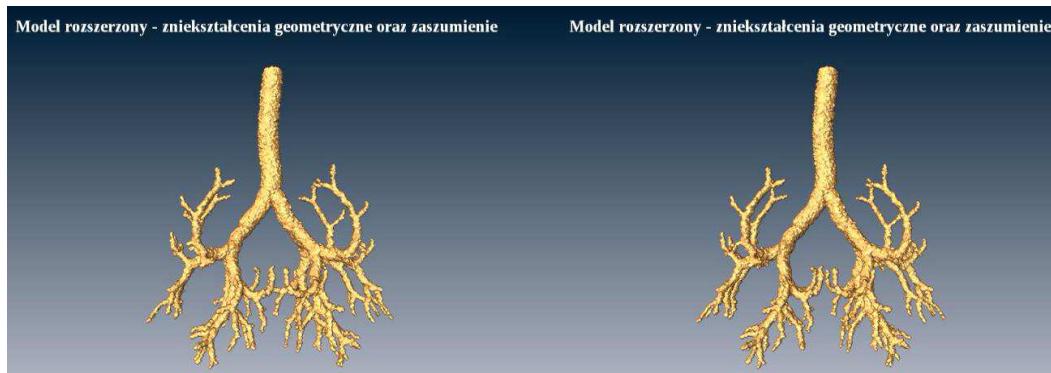
Rys. 25 Okulary służące do oglądania obrazów stereoskopowych wygenerowanych z wykorzystaniem techniki anaglif.

zwalająca na uzyskanie wrażenia głębi przestrzennej w widzianym obrazie.

Technika anaglif zyskała na popularność dzięki niskiej cenie oraz łatwości wykonania w tej technice obrazów. Do wad natomiast należy zaliczyć przebarwienia spowodowane przez zastosowanie filtrów, oraz niskie przestrzenne odwzorowanie szczegółów. W dodatku A niniejszej pracy przedstawiono wydruki modelu podstutowego i rozszerzonego w technice anaglif. W celu wygenerowania odpowiednich obrazów zamieszczonych w dodatku A wykorzystano zaimplementowany w klasie `vtkRenderer` filtr renderowanego na monitorze obrazu. Filtr ten wymaga jedynie wywołania odpowiedniej dla danej techniki stereoskopowej metody.

8.2 Technika side-by-side

Technika side-by-side w porównaniu z anaglif nie wykorzystuje filtrów barwnych, stereopary natomiast reprezentują obraz widziany z dwóch perspektyw równych lub prawie równych temu w jaki sposób, zostałyby one zarejestrowane przez każde oko oddzielnie. Technika ta pozwala na oglądanie obrazu bez zastosowania dodatkowego sprzętu, ale wymaga to odpowiednich ćwiczeń oczu oraz jest dla nich dość mącejące. Ponadto podczas oglądania bez specjalistycznego sprzętu stereopary nie mogą być od siebie zbyt oddalone. Na rynku istnieje szereg specjalistycznych okularów, które z odpowiednią częstotliwością zasłaniają naprzemiennie jedno oko. Na potrzeby prezentacji modelu rozszerzonego podczas konferencji SŁOK 2012 wykorzystano rozwiązanie 3D Vision firmy nVidia. W skład pakietu 3D Vision wchodzą okulary migawkowe dostosowane do współpracy z monitormi LCD oraz niezbędne oprogramowanie. Korzystając z DemoMakera wbudowanego w oprogramowanie Amira wygenerowano odpowiedni film prezentujący obracające się kolejno warianty modelu. Tak wygenerowany film należy odtworzyć oprogramowaniu posiadającym wsparcie dla techniki side-by-side. W skład pakietu 3D Vision wchodzi odpowiedni odtwarzacz, który niestety w użytkowanej wersji nie zawsze pozwala na płynne odtwarzanie materiału wideo. Na rys. 26 zaprezentowano klatkę ze wspomnianego wcześniej filmu.

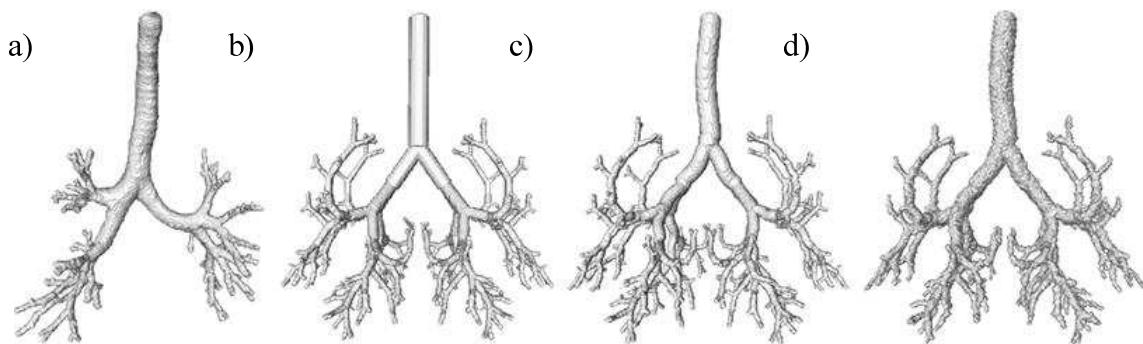


Rys. 26 Klatka z filmu zrealizowanego z wykorzystaniem techniki side-by-side prezentująca model rozszerzony.

9 Rezultaty komputerowych eksperymentów

W rozdziale tym przedstawiono test topologii modelu rozszerzonego oraz porównano model podstawowy, model rozszerzony oraz drzewo CT. Ponadto przedstawiono wyniki testów algorytmów szkieletyzacji [30].

Na rys.27 przedstawiono przykładowe drzewa oskrzelowe wygenerowane z wykorzystaniem opisywanych w niniejszej pracy algorytmów. Dla porównania pokazano również przykładowe drzewo CT. Jak widać gałęzie drzewa CT oraz gałęzie modelu rozszerzonego są wygięte. Ponadto powierzchnia modelu rozszerzonego i drzewa CT nie jest gładka, w odróżnieniu od powierzchnia modelu podstawowego. Małe, lokalne nieregularności na powierzchni modelu rozszerzonego czynią go bardziej podobnym do drzewa CT.



Rys. 27 Drzewa oskrzelowe wygenerowane przez różne modele; a) drzewo oskrzelowe uzyskane z segmentacji danych tomograficznych b) drzewo oskrzelowego wygenerowane z wykorzystaniem reguł modelu podstawowego c) model podstawowy z wygiętymi gałęziami d) model rozszerzony

9.1 Badanie topologii modelu rozszerzonego

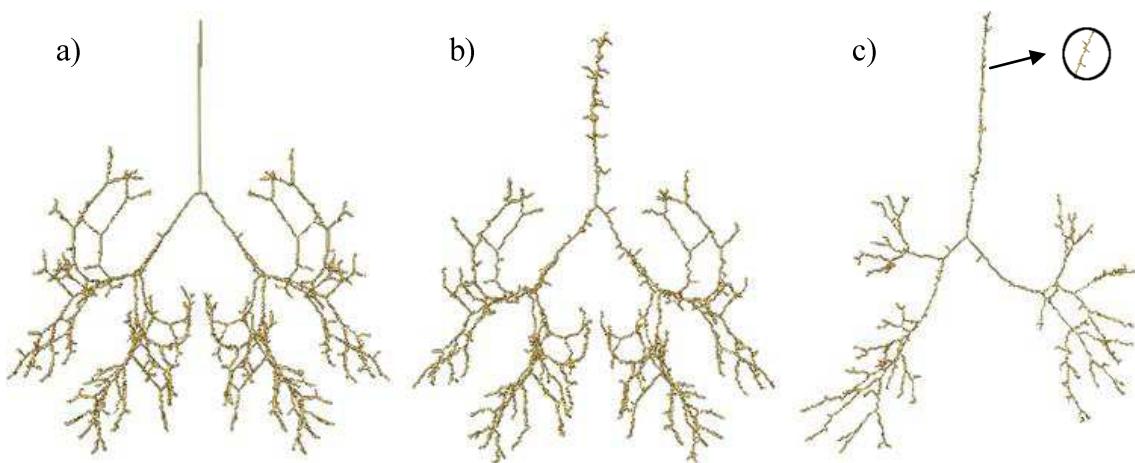
Następny eksperyment związany jest z topografią modelu rozszerzonego. Należy zaznaczyć, że model rozszerzony reprezentuje wynik segmentacji wnętrza drzewa oskrzelowego stąd powinniśmy otrzymać jeden spójny obiekt, który nie posiada żadnych pustek, oraz tuneli. Błędy podczas generowania drzewa oskrzelowego będą miały efekty w niewłaściwej jego topologii.

Przykładowo jeśli dojdzie do połączenia dwóch gałęzi, to powstanie tunel i topologia zostanie zmieniona. Jeżeli któraś z gałęzi będzie przerwana w wygenerowanym modelu to drzewo będzie złożone z więcej niż jednego komponentu, co oznacza zmianę topologii. Pojawienie się pustki w gałęzi również zmienia topologię drzewa.

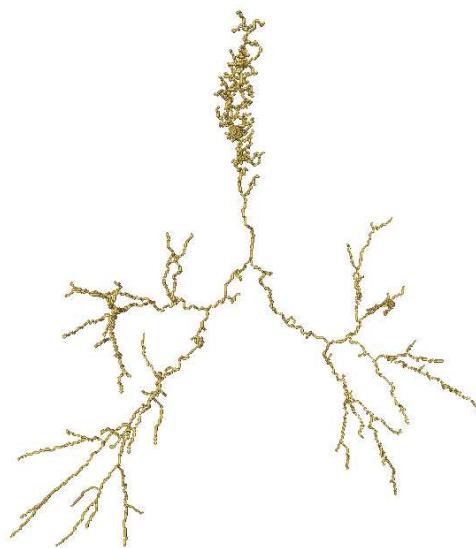
W celu przetestowania topologii modelu rozszerzonego, wygenerowano 10 wariantów modelu różniących się parametrami, wygięć, procedury topoEDEN oraz algorytmu ASFT. Następnie dla każdego drzewa, obliczono automatycznie ilość połączonych komponentów oraz liczbę pustek.

W celu sprawdzenia czy drzewo nie posiada pustki lub czy dwie gałęzie nie połączyły się, wykorzystano algorytm zamykania tuneli [1]. Żadne dwie gałęzie nie są połączone jeśli algorytm zamykania tuneli nie wygeneruje żadnej łyaty zamykającej tunel. Podobnie jeżeli drzewo zawiera pustkę to algorytm zamykania tuneli wygeneruje obiekt ją wypełniający. Dla wszystkich 10 wygenerowanych modeli algorytm zamykania tuneli nie wygenerował żadnego obiektu co oznacza, że drzewa nie posiadały żadnej pustki ani żadnego tunelu. Ponadto każde drzewo stanowiło jeden połączony komponent. Podsumowując dla 10 wygenerowanych modeli rozszerzonych, wszystkie posiadają tę samą topologię, odpowiednią dla wnętrza drzewa oskrzelowego.

Rys. 28 oraz rys. 29 przedstawia rezultaty uzyskane przez algorytm szkieletyzacji zastosowany do modelu podstawowego, modelu rozszerzonego oraz drzew oskrzelowego uzyskanego w procesie segmentacji danych tomograficznych. Jak możemy zaobserwować, algorytm szkieletyzacji zadziałał prawie idealnie dla modelu podstawowego, przy czym wyniki uzyskane dla modelu rozszerzonego i drzew CT są znacznie gorsze. Należy zwrócić szczególną uwagę na nieuchciane gałęzie widoczne na rys. 28b) - c)



Rys. 28 Szkielety drzewa wygenerowane dla; a) model podstawowy b) model rozszerzony c) drzewo oskrzelowe wysegmentowane z danych tomograficznych. W okręgu przedstawiono odcepy widoczne na drzewie w widoku bocznym.



Rys. 29 Szkielety drzewa CT o wysokiej liczbie niechcianych gałęzi

Wczesniejsze obserwacje zostały potwierdzone przez analizę ilościową. Najpierw wygenerowano model podstawowy, następnie wygenerowano 10 wariantów modelu rozszerzonego różniących się parametrami algorytmu topoEDEN oraz ASFT. Dla przykładu obliczono ilość niechcianych gałęzi dla drzewa uzyskanego z danych tomograficznych oraz modelu podstawowego. Dla 10 wariantów modelu

rozszerzonego obliczono średnią wartość niechcianych gałęzi. Wszystkie drzewa posiadają 64 końcowe gałęzie co determinuje tą samą ilość wszystkich gałęzi. Wyniki zaprezentowane w *tabela 1* pokazują, że model podstawowy nie nadaje się do testów algorytmów szkieletyzacji wnętrza drzew CT. Model podstawowy jest zbyt „uproszczony”, nie odzwierciedla wygięć i „niedoskonałości” powierzchni zewnętrznej drzewa CT co powoduje, że testowany algorytm szkieletyzacji generuje znacznie mniej niechcianych gałęzi niż dla drzewa CT. Natomiast średnia liczba niechcianych gałęzi dla modelu rozszerzonego jest zbliżona do liczby niechcianych gałęzi generowanych dla drzewa CT. Dlatego też model rozszerzony można z powodzeniem wykorzystać do testowania algorytmów szkieletyzacji.

Tabela 1Liczba niechcianych gałęzi

	Wysegmentowane drzewo	Model podstawowy	Model podstawowy z wygięciami	Model rozszerzony
Liczba nie chcianych gałęzi	56(rys. 28 c), 105(rys. 29)	6	5	57,3*

* średnia wartość uzyskana dla 10 wariantów modelu

10 Wnioski końcowe

W niniejszej pracy zaprezentowano nowy algorytm generowania trójwymiarowego modelu drzewa oskrzelowego. Model ten jest rozszerzeniem dobrze znaneego podejścia do generowania trójwymiarowego drzewa oskrzelowego [18], zaproponowanego do badania przepływu powietrza.

Omówione we wcześniejszych rozdziałach rozszerzenia wyżej wspomnianego algorytmu wprowadzają geometryczne deformacje oraz szum w celu przybliżenia modelu do drzew oskrzelowych uzyskanych w procesie segmentacji danych tomograficznych. W rozdziale 9 pokazano eksperymentalnie, że model rozszerzony drzewa oskrzelowego wykazuje dużo większe podobieństwo do drzew CT, aniżeli model podstawowy. Wyniki eksperymentów (patrz rozdział 9.1) pokazują, że drzewa generowane algorytmem rozszerzonym mają poprawną topologię (stanowią je-

den połączony komponent, nie posiadają pustki ani tunelu). Ponadto wyniki testowania algorytmu szkieletyzacji pokazały, że liczba niechcianych gałęzi szkieletu jest zbliżona dla modelu rozszerzonego i drzewa CT. Co wykazuje, że algorytmy szkieletyzacji zachowują się podobnie dla modelu rozszerzonego jak i dla drzew CT tzn., generują znaczną ilość niechcianych gałęzi, które w kolejnym kroku przetwarzania trzeba filtrować. W przypadku modelu podstawowego algorytm szkieletyzacji wygenerował jedynie 6 niechcianych gałęzi co oznacza, że model ten jest zbyt „uproszczony” a jego zastosowanie nie pokazuje istotnego problemu związanego z generacją dużej liczby niechcianych gałęzi przez algorytm szkieletyzacji. Stąd model rozszerzony wprowadza istotne, z punktu widzenia badania algorytmów ilościowej analizy drzew oskrzelowych, cechy nie występujące w algorytmie podstawowym.

Ponadto należy podkreślić, iż model rozszerzony stworzono pod kierunkiem pulmonologów, którzy zaakceptowali go do wydruku 3D w celu uzyskania fantomu przydatnego do testów oprogramowania do ilościowej analizy drzew oskrzelowych bazujących na obrazach tomograficznych płuc.

Podsumowując należy podkreślić, że wszystkie cele postawione w rozdziale 1 niniejszej pracy zostały osiągnięte a zgodnie z wiedzą autora stworzony algorytm generowania modelu rozszerzonego jest pierwszym tego typu algorytmem, a generowane z jego wykorzystaniem modele drzew można zastosować do badania algorytmów ilościowej oceny lokalnych parametrów drzew oskrzelowych na bazie obrazów CT klatki piersiowej.

Przyszłe badania związane z modelem rozszerzony skupią się na wykorzystaniu go do testowania algorytmów wyznaczających lokalną wartość średnicy prześwitu gałęzi drzewa.

Na koniec warto zaznaczyć, iż dzięki doświadczeniu i wiedzy zdobytej podczas realizacji niniejszej pracy udało się poznać wady wcześniej wykorzystywanych narzędzi informatycznych, a także zupełnie nowe oprogramowanie, co pozwoliło

wyselekcionować grupę komponentów środowiska developerskiego, które autor z powodzeniem wykorzystuje podczas prac nad nowymi zagadnieniami.

Literatura

- [1] Aktouf, Z., G. Bertrand, i L. Perrotin. „A three-dimensional holes closing algorithm.” *Pattern Recognition Letters*, 2002: 523-531.
- [2] Bertrand, G. „Simple points, topological numbers and geodesic neighborhoods in cubic grids.” *Pattern Recognition Letters*, 1994: 1003–1011.
- [3] Chaussard, J., M. Couprie, i H. Talbot. „Robust skeletonization using the discrete lambda-medial axis.” *Pattern Recognition Letters*, 2011: 1384-1394.
- [4] Chen, W. J., D. S. Shiah, i C. S. Wang. „A three-dimensional model of the upper tracheobronchial tree.” *Bulletin of Mathematical Biology*, 1980: 847-859.
- [5] Couprie, M., i G. Bertrand. „Topology preserving alternating sequential filter for smoothing 2D and 3D objects.” *Journal of Electronic Imaging*, 2004: 720-730.
- [6] Digia. *Qt home page*. <http://qt.digia.com> (data uzyskania dostępu: Grudzień 27, 2012).
- [7] Eden, M. „A two-dimensional growth process.” *In Proceedings of the 4th Berkeley Symposium on Mathematical Statistics and Probabilities*. University of California Press, 1961. 223–239.
- [8] ESIEE Engineering. *Pink Library*. <https://www.pinkhq.com/homepage/> (data uzyskania dostępu: Grudzień 27, 2012).
- [9] Foley, J., A. van Dam, J. Hughes, i R. Philips. *Wprowadzenie do grafiki komputerowej*. Warszawa: Wydawnictwo Naukowo-Techniczne, 2001.
- [10] Free Software Fundation. <http://www.gnu.org/software/gdb/> (data uzyskania dostępu: Styczeń 1, 2013).

- [11] Gillis, H. L., i K. R. Lutchen. „How heterogeneous bronchoconstriction affects ventilation distribution in human lungs: a morphometric model.” *Annals of Biomedical Engineering*, 1999: 1–13.
- [12] Gray, H. *Anatomy of the Human Body*. Philadelphia: 1918, Lea & Febiger.
- [13] Horsfield, K., G. Dart, D. E. Olson, G. F. Filley, i G. Cumming. „Models of the human bronchial tree.” *Journal of Applied Physiology*, 1971: 202-217.
- [14] Horsfield, K., i A. Thurlbeck. „Relation between diameter and flow in branches of the bronchial tree.” *Bulletin of Mathematical Biology*, 1981: 681–691.
- [15] Howatson, T. M., A. J. Pullan, i P. J. Hunter. „Generation of an Anatomically Based Three-Dimensional Model of the Conducting Airways.” *Annals of Biomedical Engineering*, 2000: 793-802.
- [16] Kamiya, A., T. Togawa, i A. Yamamoto. „Theoretical relationship between the optimal models of the vascular tree.” *Bulletin of Mathematical Biology*, 1974: 311–323.
- [17] Khalimsky, E. „Topological structures in computer science.” *Journal of Applied Mathematics and Simulation*, 1987: 25-41.
- [18] Kitaoka, H., R. Takaki, i B. Suki. „A three-dimensional model of the human airway tree.” *Journal of Applied Physiology*, 1999: 2207-2217.
- [19] KitWare. Strona programu CMake. <http://cmake.org/> (data uzyskania dostępu: Styczeń 1, 2013).
- [20] —. VTK - The Visualization Toolkit. <http://www.vtk.org> (data uzyskania dostępu: Grudzień 23, 2012).

- [21] Komorowski, J. „Epidemiologia astmy w Polsce w oparciu o wyniki badania ECAP.” Warszawa: Zakład Profilaktyki Zagrożeń Środowiskowych i Alergologii Wydział Nauki o Zdrowiu Warszawski Uniwersytet Medyczny, 2012.
- [22] Lawlor, G. *vtkbioeng*. <http://www.bioengineering-research.com/software/vtkbioeng> (data uzyskania dostępu: Grudzień 25, 2012).
- [23] Linemed. *Linemed*. 18 Listopad 2008.
http://linemed.pl/nhealth_guide/details/cId,23,id,1082 (data uzyskania dostępu: Styczeń 1, 2013).
- [24] Martonen, T. B., Y. Yang, i M. Dolovich. „Definition of airway composition within gamma camera images.” *Journal of Thoracic Imaging*, 1994: 188–197.
- [25] „Milena polski syntezator mowy dla systemu Linux.”
<http://milena.polip.com> (data uzyskania dostępu: Luty 7, 2013).
- [26] Min, P. *[binvox] 3D mesh voxelizer*.
<http://www.cs.princeton.edu/~min/binvox/> (data uzyskania dostępu: Grudzień 27, 2012).
- [27] Nelson, T. R., i D. K. Manchester. „Modeling of lung morphogenesis using fractal geometries.” *IEEE Transactions on Medical Imaging*, 1988: 321–327.
- [28] Nikolaidis, N., i I. Pitas. *3-D Image Processing Algorithms*. John Wiley & Sons Inc., 2001.
- [29] Nooruddin, F., i G. Turk. „Simplification and Repair of Polygonal Models Using Volumetric Techniques.” *IEEE Transactions on Visualization and Computer Graphics*, 2 Kwiecień 2003: 191-205.
- [30] Palagi, K., i A. Kuba. „A 3D 6-subiteration thinning algorithm for extracting medial lines.” *Pattern Recognition Letters*, 1998: 613–627.

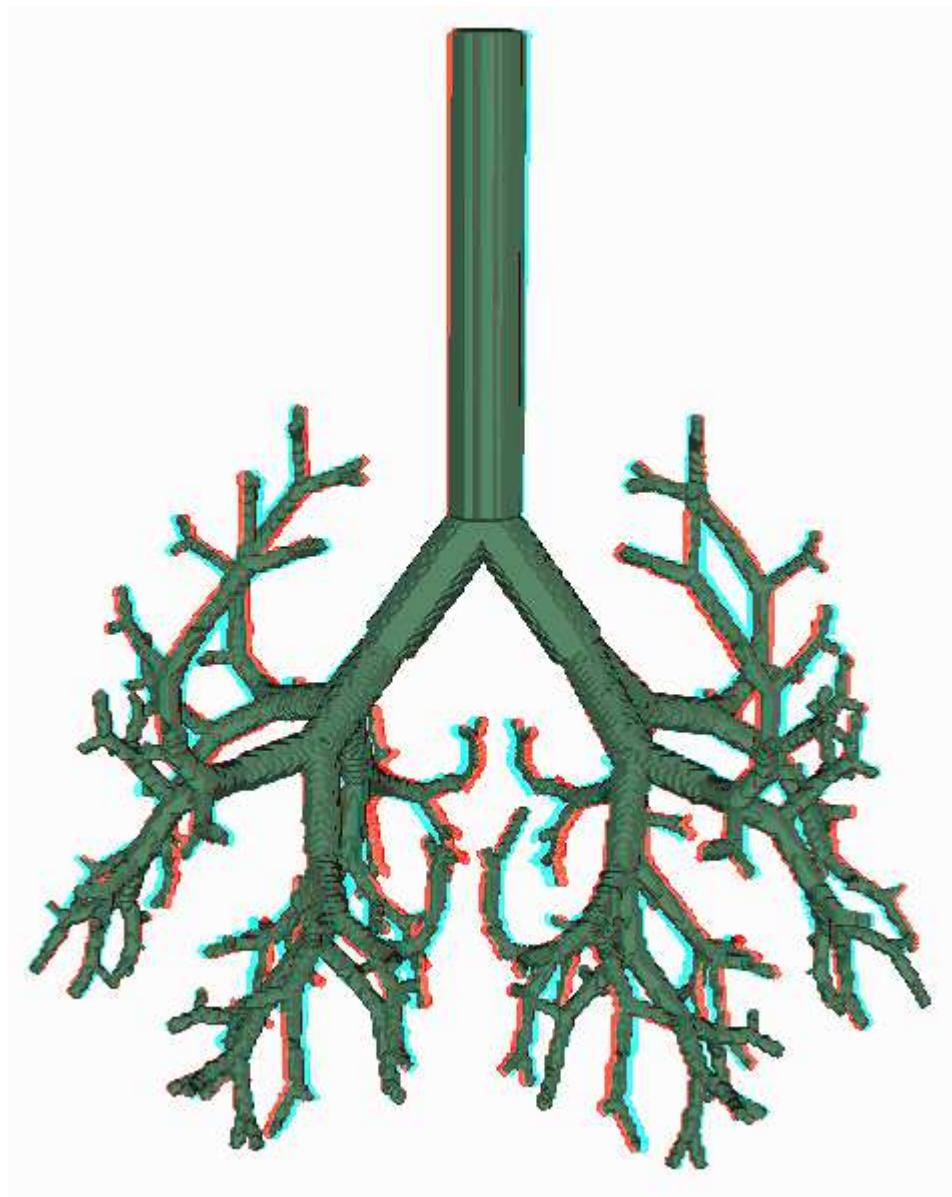
- [31] Palagi, K., J. Tscherren, E. A. Hoffman, i M. Sonka. „Quantitative analysis of pulmonary airway tree structures.” *Computers in Biology and Medicine*, 2006: 974-996.
- [32] Pluta, K., M. Janaszewski, i M. Postolski. „New algorithm for modeling of bronchial trees.” *Zeszytach Naukowych Akademii Górnictwo-Hutniczej - "Automatyka"*. Kraków: Akademia Górnictwo-Hutnicza, Zatwierdzno do druku.
- [33] Pluta, K., M. Postolski, i M. Janaszewski. „Algorytmy modelowania drzewa oskrzelowego.” *Zeszyt Naukowy Wyższej Szkoły Informatyki w Łodzi*. Łódź: Wyższa Szkoła Informatyki w Łodzi, 2012. 152-170.
- [34] Postolski, M., i inni. „Reliable Airway Tree Segmentation Based on Hole Closing in Bronchial Walls.” W *Computer Recognition Systems 3*, autor: M. Kurzyński i M. Woźniak, 389-396. Springer, 2009.
- [35] Postolski, M., M. Janaszewski, Y. Kenmochi, i J. O. Lachaud. „Tangent Estimation Along 3D Digital Curves.” *21st International Conference on Pattern Recognition*. Tsukuba, 2012.
- [36] Raabe, O. G., H. C. Yeh, H. M. Schum, i R. F. Phalen. *Tracheobronchial Geometry - Human, Dog, Rat, Hamster*. LF-53, Albuquerque: Lovelace Foundation for Medical Evaluation and Research, 1976.
- [37] Serra, J. „Image analysis and mathematical morphology.” *Academic Press*, 1982.
- [38] Stewart, B. „Can Bronchoscopic Airway Anatomy Be an Indicator of Autism?” *CHEST*. Honolulu: American College of Chest Physicians, 2011.
- [39] The Global Initiative for Asthma. <http://www.ginasthma.org> (data uzyskania dostępu: Styczeń 1, 2013).
- [40] The KDevelop Team. <http://www.kdevelop.org/> (data uzyskania dostępu: Styczeń 1, 2013).

- [41] Torvalds, L., i J. Hamano. <http://git-scm.com/> (data uzyskania dostępu: Styczeń 1, 2013).
- [42] Tschirren, J., E. Hoffman, G. McLennan, i M. Sonka. „Intrathoracic airway trees: segmentation and airway morphology analysis from low-dose CT scans.” *IEEE Transactions on Medical Imaging*, 2005: 1529-1539.
- [43] Valgrind Developers. <http://valgrind.org/> (data uzyskania dostępu: Styczeń 1, 2013).
- [44] van Erbruggen, C., Ch. Hirsch, i M. Paiva. „Anatomically based three-dimensional model of airways to simulate flow.” *Journal of Applied Physiology*, 2005: 970-980.
- [45] van Heesch, D. <http://www.doxxygen.org/> (data uzyskania dostępu: Styczeń 1, 2013).
- [46] Visualization Sciences Group. „Amira home page.” <http://amira.com/> (data uzyskania dostępu: Styczeń 5, 2013).
- [47] Weibel, E. R. *Morphometry of the Human Lung*. New York: Academic Press, 1963.
- [48] „Wikipedia.” *Wikipedia, the free encyclopedia*. https://en.wikipedia.org/wiki/Anaglyph_3D (data uzyskania dostępu: Luty 7, 2013).
- [49] Wikipedia. „Wikipedia.” https://en.wikipedia.org/wiki/Region_growing (data uzyskania dostępu: Luty 12, 2013).
- [50] Williams, A. http://www.boost.org/doc/libs/1_46_0/doc/html/thread.html (data uzyskania dostępu: Grudzień 27, 2012).

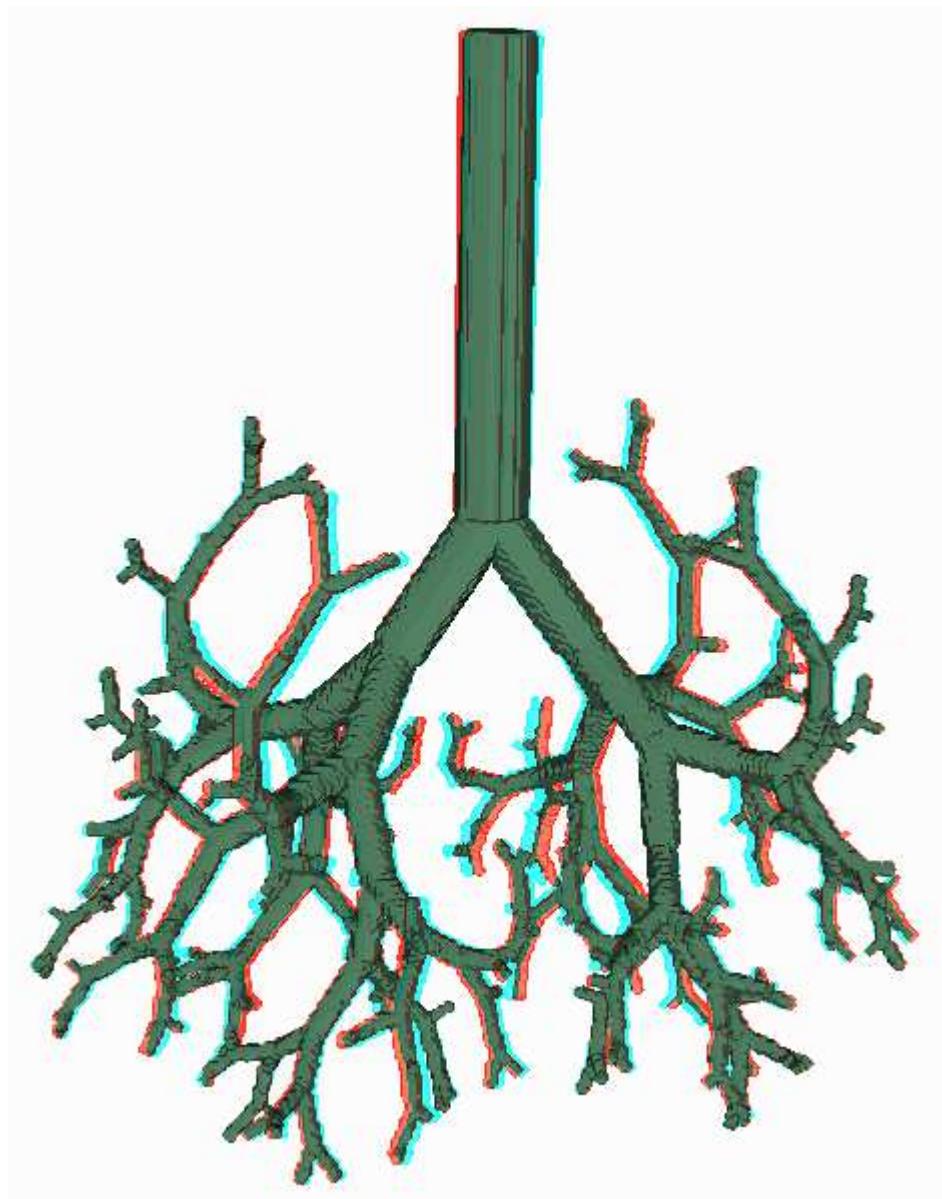
Dodatek A

W przypadku problemów z filtrowaniem stereopar, proszę odtworzyć wersję elektroniczną niniejszej pracy.

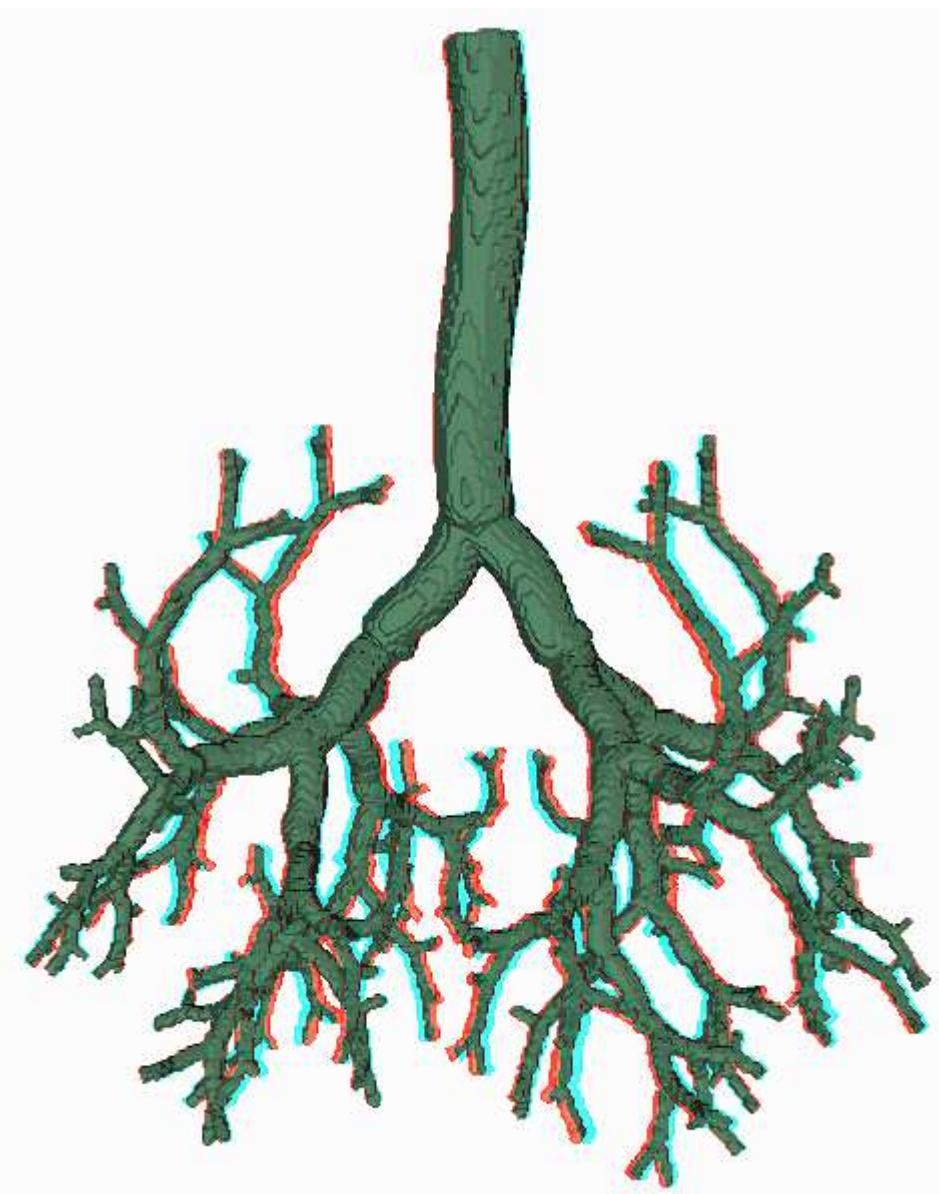
W tym miejscu powinna znajdować się koperta zawierająca parę okularów pozwalających na pełne wykorzystanie efektów wizyjnych stereoskopii w technice anaglif.



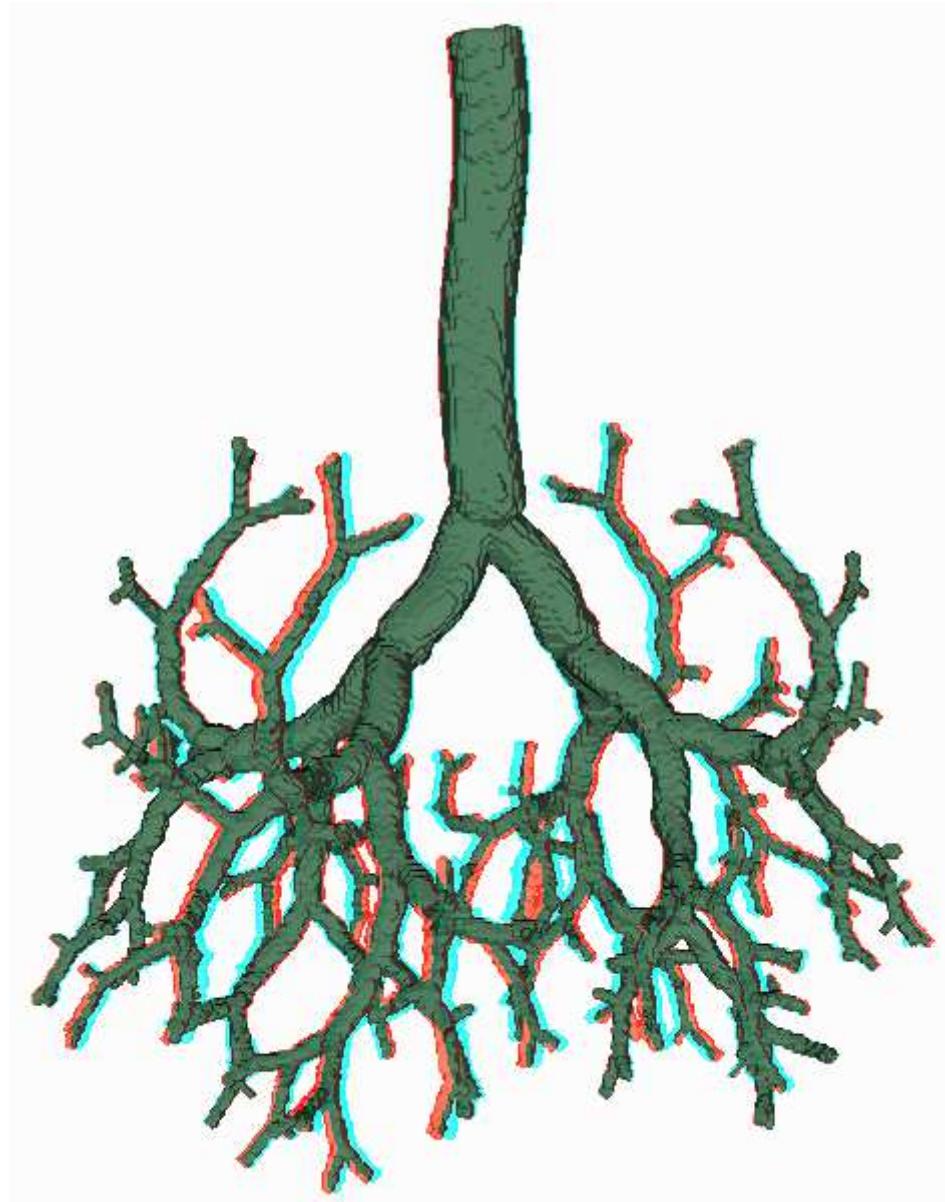
Dodatek A rysunek 1 Model podstawowy.



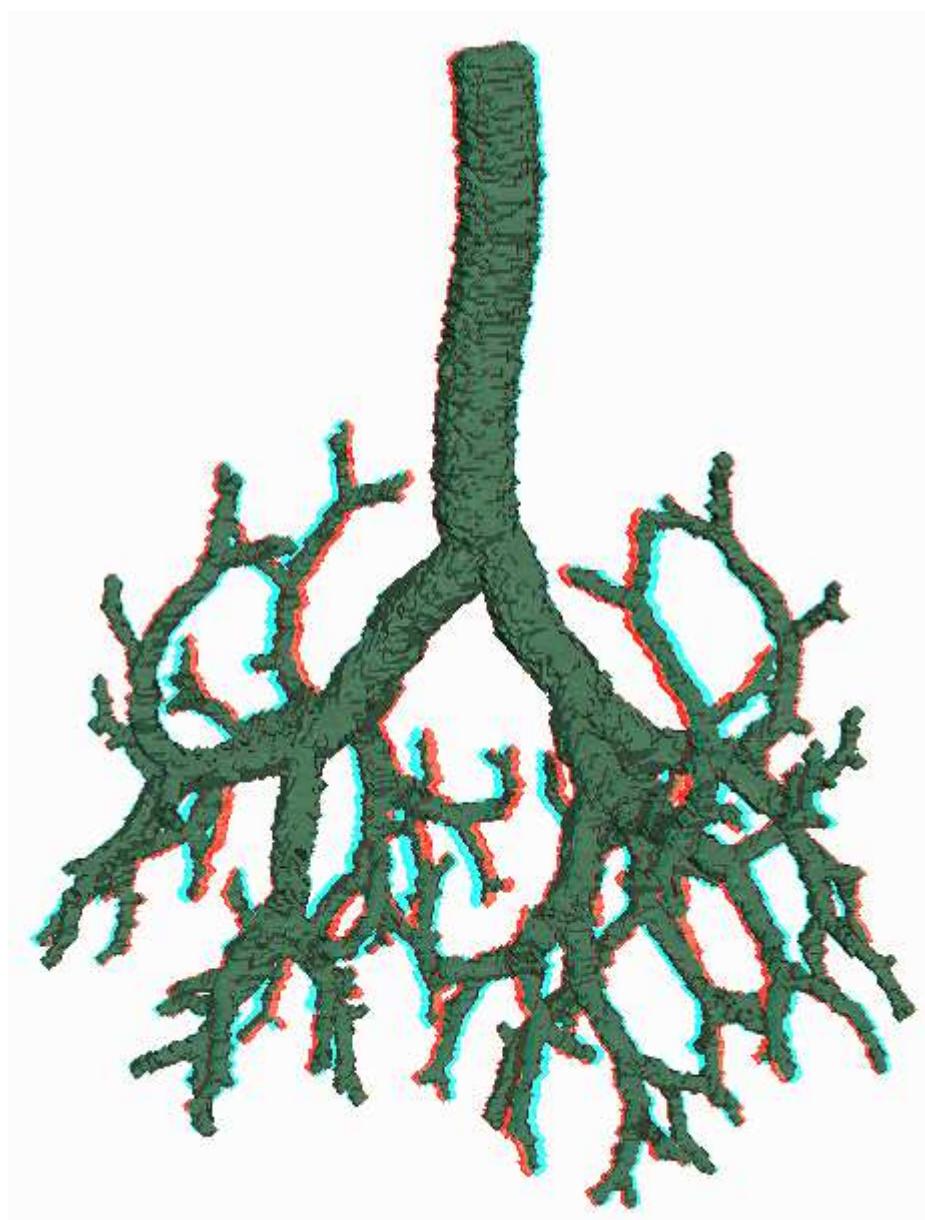
Dodatek A rysunek 2 Model podstawowy.



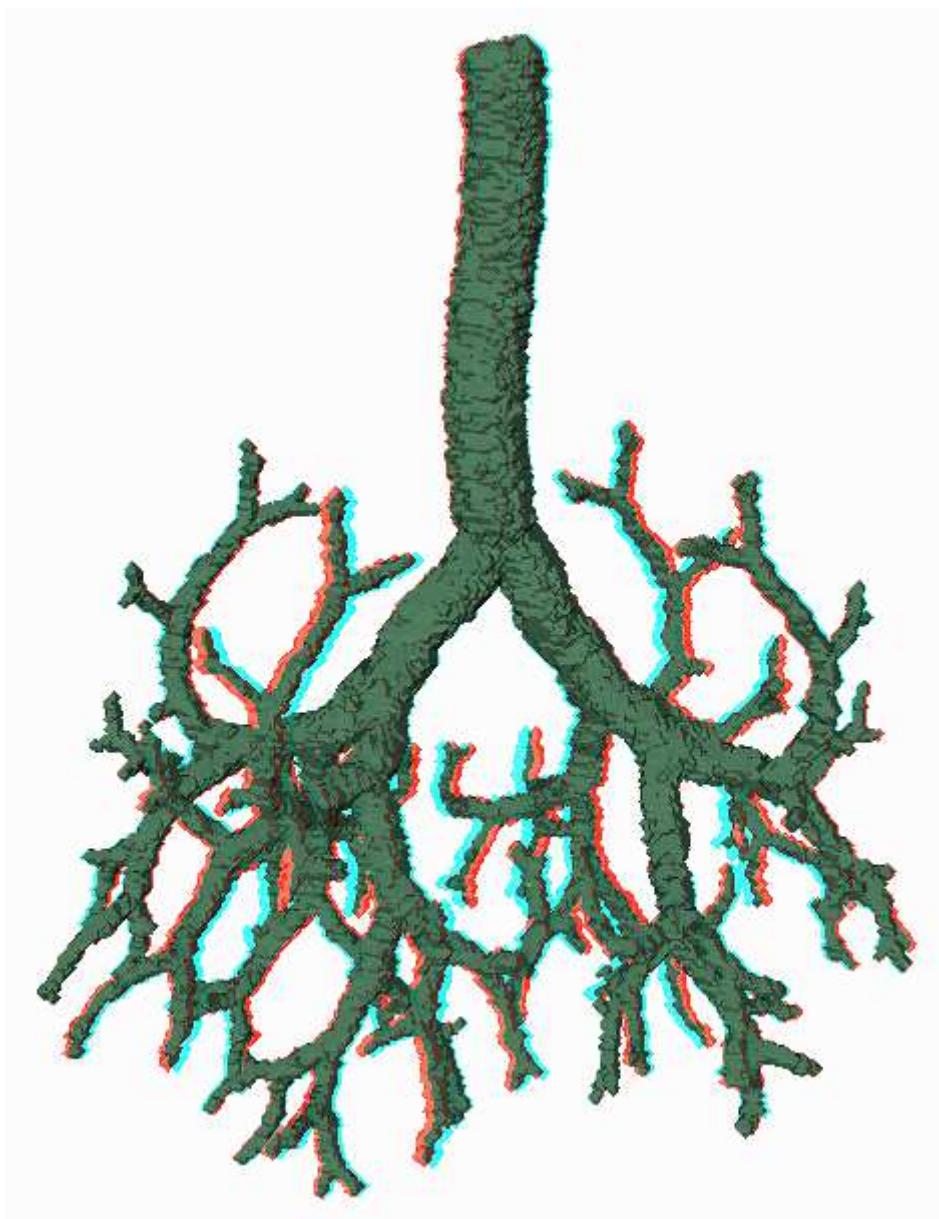
Dodatek A rysunek 3 Model rozszerzony zawierający jedynie powyginane gałęzie.



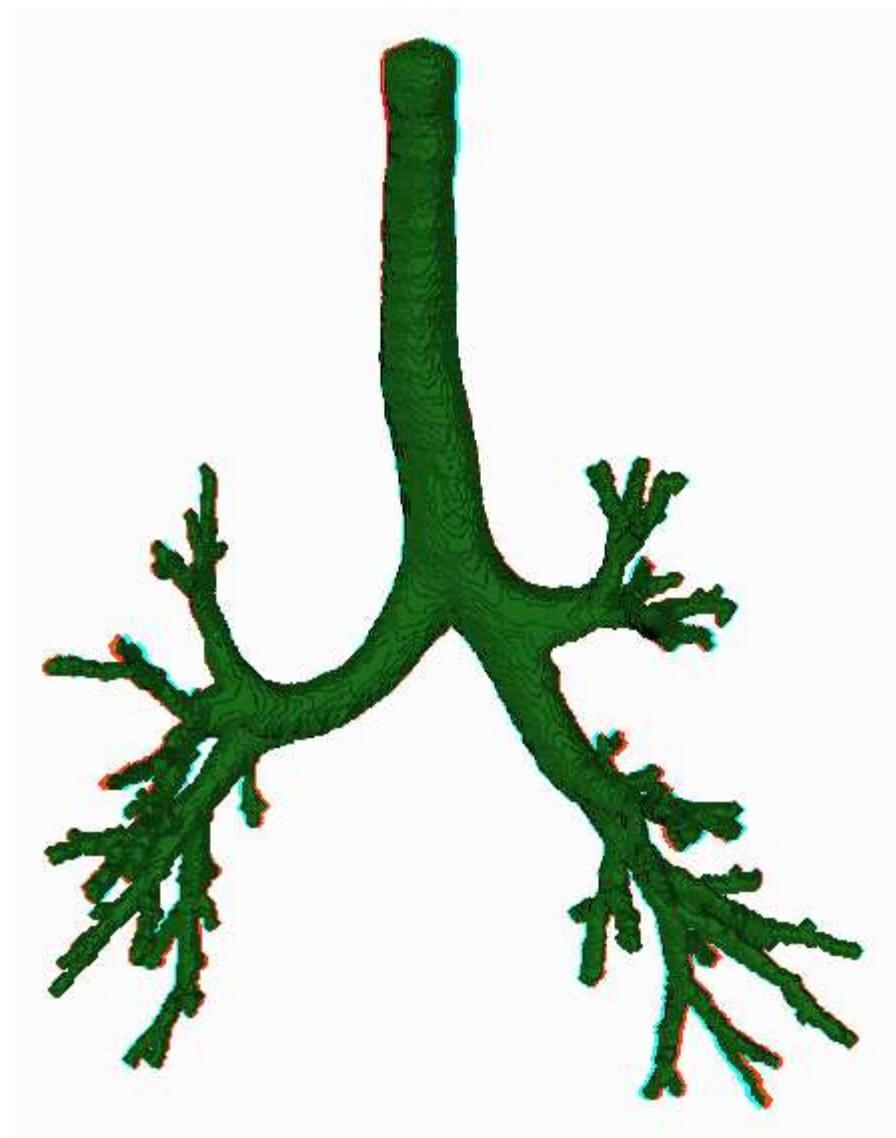
Dodatek A rysunek 4 Model rozszerzony zawierający jedynie powyginane gałęzie.



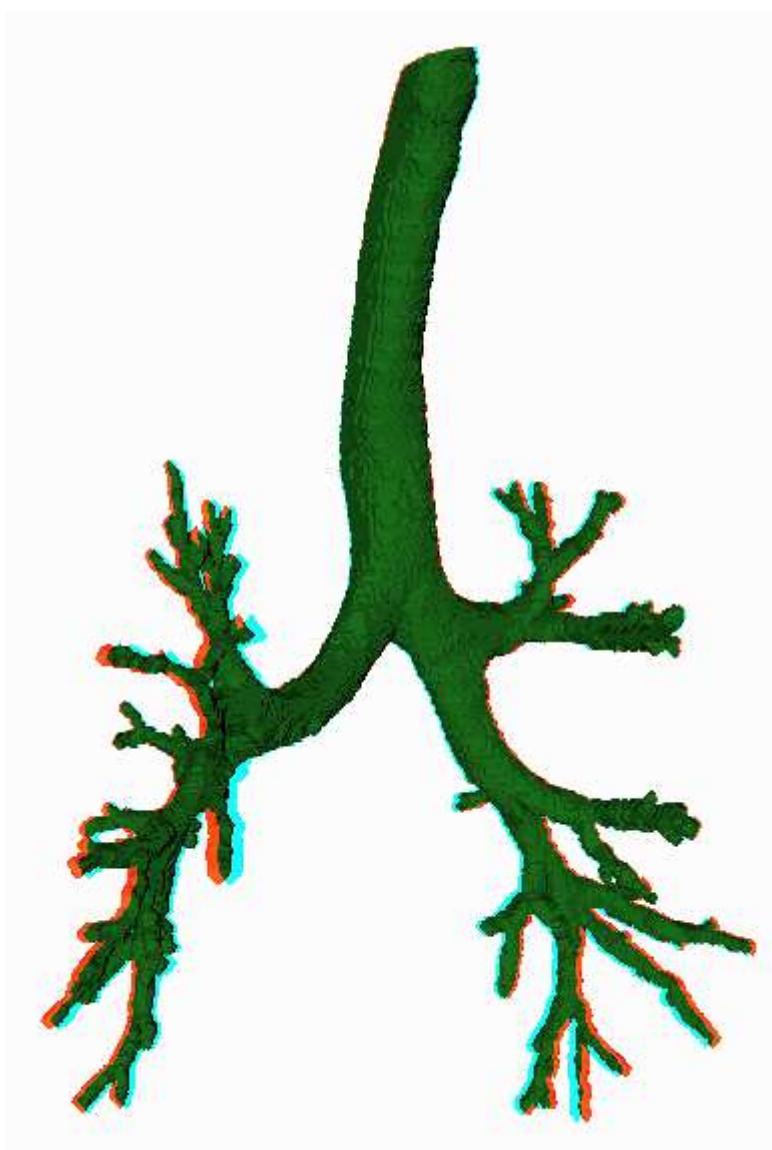
Dodatek A rysunek 4 Model rozszerzony.



Dodatek A rysunek 5 Model rozszerzony.



Dodatek A rysunek 5 Drzewo CT



Dodatek A rysunek 6 Drzewo CT