Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

3-D сегментація та візуалізація об'єктів на основі медичних зображень

Текстова частина до курсової роботи
за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи
ст. викл. Бучко О.А.

_____
(підпис)
“___” _____ 2020 р.

Виконав студент ІПЗ-16
Нгуєн С.Б.В.
“___” _____ 2020р.

Київ 2020

Календарний план виконання роботи

Тема: «3-D сегментація та візуалізація об'єктів на основі медичних зображень»

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	19.09.2019	
2.	Огляд технічної літератури за темою роботи.	грудень 2019	
3.	Пошук наявних алгоритмів та публічних наборів даних. Реалізація програми з мінімальним функціоналом.	січень 2020	
4.	Аналіз отриманих результатів з керівником.	21.02.2020	
5.	Покращення роботи алгоритму з урахуванням зауважень керівника.	лютий – березень 2020	
6.	Пошук інших способів реалізації застосунку, дослідження нових методів візуалізації.	березень 2020	
7.	Написання пояснювальної роботи.	березень 2020	
8.	Створення слайдів для доповіді та написання доповіді.		
9.	Захист роботи		

Студент *Нгуєн Сан Бинь Ванович*

Керівник *Бучко Олена Андріївна*

“___”_____

ЗМІСТ

Анотація

Мета курсової роботи написати застосунок, на вхід якого подається набір медичних зображень, що являють собою 2D зрізи певної ділянки людського тіла, а на виході отримати модель, проекцію якої можна переглянути у 3D просторі. Були розглянуті стандартні алгоритми для побудови 3D-моделей та сегментації. Наведені методи покращення, оптимізації та виявлені певні недоліки наведених алгоритмів. Реалізація, тестування та дослідження основної частини програми були проведені за допомогою мови програмування Python та написаних під неї бібліотек. Для візуалізації отриманих 3D-моделей було додатково створено застосунок під операційну систему iOS з використанням методів доповненої реальності за допомогою бібліотеки ARKit. Оглянуто готові рішення.

Вступ

Комп'ютерний зір (“Computer Vision”) одна з областей програмування, яка прагне розробити методи, що допомагають комп'ютерам “бачити” і розуміти зміст цифрових зображень, таких як фотографії та відео. Однак, наразі проблема комп'ютерного зору виявляється не такою простою та в значній мірі залишається невирішеною, яка пояснюється обмеженим розумінням біологічного зору, так і складністю сприйняття зору в динамічному і майже нескінченно різному фізичному світі.

Сьогодні комп’ютерний зір вирішує проблеми з різних галузей та застосовується у широкому спектрі задач. Широке використання цієї галузі можна простежити в медицині, де частою проблемою є візуалізація органів та тканин людини та пошук аномалій. Медична візуалізація - це техніка і процес створення візуальних уявлень про внутрішню частину тіла для клінічного аналізу та медичного втручання, а також візуального подання деяких органів або тканин.

Метою даної роботи є дослідження медичної візуалізації та перегляд існуючих програмних продуктів. Будуть розглянуті методи попередньої обробки зображення, алгоритми та способи покращення результатів тривимірної сегментації та візуалізації. Результатом дослідження стане програмний застосунок, який буде використовувати досліджені алгоритми.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Томографія

Томографія - це процес отримання секцій (зрізів) об'єкта за допомогою використання будь-якого виду проникаючої хвилі (радіохвилі, звукові хвилі тощо). Цей процес зустрічається в багатьох галузях науки, наприклад медицині, археології, астрономії, геології тощо. Під час сканування об’єкту, томограф – апарат, який здійснює цей процес – генерує серію зображень, зрізи об’єкту на деякій відстані одна від одної. Цю серію зрізів називають томограмою.

1.2 Види томографії

Зараз існує більше 30 типів томографії які використовують різні підходи та базуються на певних фізичних явищах, однак найпопулярніша та найпоширеніша зараз рентгенівська томографія, яку ще називають комп’ютерною томографією (КТ).

1.2.1 Комп’ютерна томографія

Комп'ютерна томографія - це вид томографії, який використовує рентгенівське випромінювання. При скануванні, томограф випускає рентгенівські проміні під різним кутом навколо об’єкту, а потім, в залежності від того, як промінь послабився і поглинувся після проходу об’єкту, будується його зріз (див. Рисунок 1.3.1 та додаток А). Винахідники комп’ютерного томографу – фізик Аллан М. Кормак та інженер-електрик Годфрі Н. Х'юнсфілд – отримали в 1979 році нобелівську премію з фізики або медицини. Наразі, комп’ютерна томографія широко використовується в медицині для діагностування різноманітних хвороб та в цілях профілактики. Варто зазначити, що КТ знайшло своє застосування не тільки в медицині. Археологи, щоб не пошкодити археологічні знахідки, використовують КТ. Так було досліджено багато саркофагів.

Обертаючись навколо об’єкта, рентгенівський генератор випускає промені, які потім приймаються рентгенівськими детекторами на протилежній стороні. Зловлені промені аналізуються, а потім будується серія зображень поперечного перерізу об’єкту. Кожен піксель на зображенні позначає послаблення випромінювання після проходження певної тканини. Таке послаблення вимірюється за шкалою Гаунсфілда. За цією шкалою найбільше послаблення мають кістки +3071 HU (одиниць Гаунсфілда), а найменше – повітря -1024 HU.

Таблиця 1.2.1.1 Середні денситометричні показники (шкала Гаунсфілда)

Речовина	**HU**
Повітря	-1000
Легені	-500
Жир	від -100 до 50
Вода	0
Спинномозкова рідина	15
Нирки	30
Кров	від +30 до +45
М’язи	від +10 до +40
Сіра речовина	від +37 до +45
Біла речовина	від +20 до +30
Печінка	від +40 до +60
М’які тканини, контраст	від +100 до +300
Кістка	від +700 (губчата речовина) до +3000 (кісткова речовина)

Шкала одиниць Гаунсфілда (денситометричних показників, англ. HU) — це шкала лінійного послаблення випромінювання по відношенню до дистильованої води, рентгенівська щільність якої була прийнята за 0 HU (при

стандартному тиску та температурі). Для деякого матеріалу X з лінійним коефіцієнтом послаблення μ_X, величина HU визначається згідно формули:

$$\frac{\mu_X - \mu_{\text{вода}}}{\mu_{\text{вода}} - \mu_{\text{повітря}}} \times 1000 \quad (1.1)$$

де $\mu_{\text{вода}}$ та $\mu_{\text{повітря}}$ лінійні коефіцієнти послаблення води та повітря при стандартних умовах.

Варто зауважити, що інколи, для виділення певних структур, як кровоносні судини, або для покращення чіткості контурів деяких органів може використовуватися контрастна речовина – радіоконтраст.

1.2.2 Магнітно-резонансна томографія

Іншим видом томографії є магнітно-резонансна томографія (МРТ), яка базується на явищі збудженні атомів водню на певні електромагнітні хвилі. Цей тип томографії дає набагато чіткіші зображення тканин серця, м’яз, та мозку, на відміну від комп’ютерної томографії, та широко використовується для пошуку новоутворень в організмі людини. Однак сфера застосування МРТ обмежується лише медициною. Принцип роботи дуже схожий на КТ.

Процес створення поперечних знімків за допомогою МРТ є дещо складнішим за КТ, але допомагає вирізнити різні структури організму. Це можливо через те, що МРТ дає можливість використовувати різні фізичні властивості магнітного поля, наприклад змінювати частоту та період сигналу, і таким чином ми будемо отримувати абсолютно різні знімки – на деяких знімкам буде краще видно сіру та білу речовину, на інших кров тощо. Такі конфігурації параметрів МРТ називають послідовність МРТ (англ. “MRI sequence”) - це конкретна конфігурація радіочастотних імпульсів та градієнтів, що призводить до появи конкретного зображення.

Так само як і в КТ, під час МРТ часто використовують контрастну речовину, яку вводять, наприклад у кров, щоб дати можливість краще вирізняти структури на зображені.

1.3 DICOM

Медичні знімки КТ, МРТ тощо зберігається у DICOM файлі – формат збереження даних, який є стандартом ISO. Формат був розроблений виробниками медичного електронного обладнання. У самому DICOM файлі зберігається дуже багато додаткової інформації, такі як демографічні дані пацієнта, вид обстеження та час, інформація про обладнання тощо.

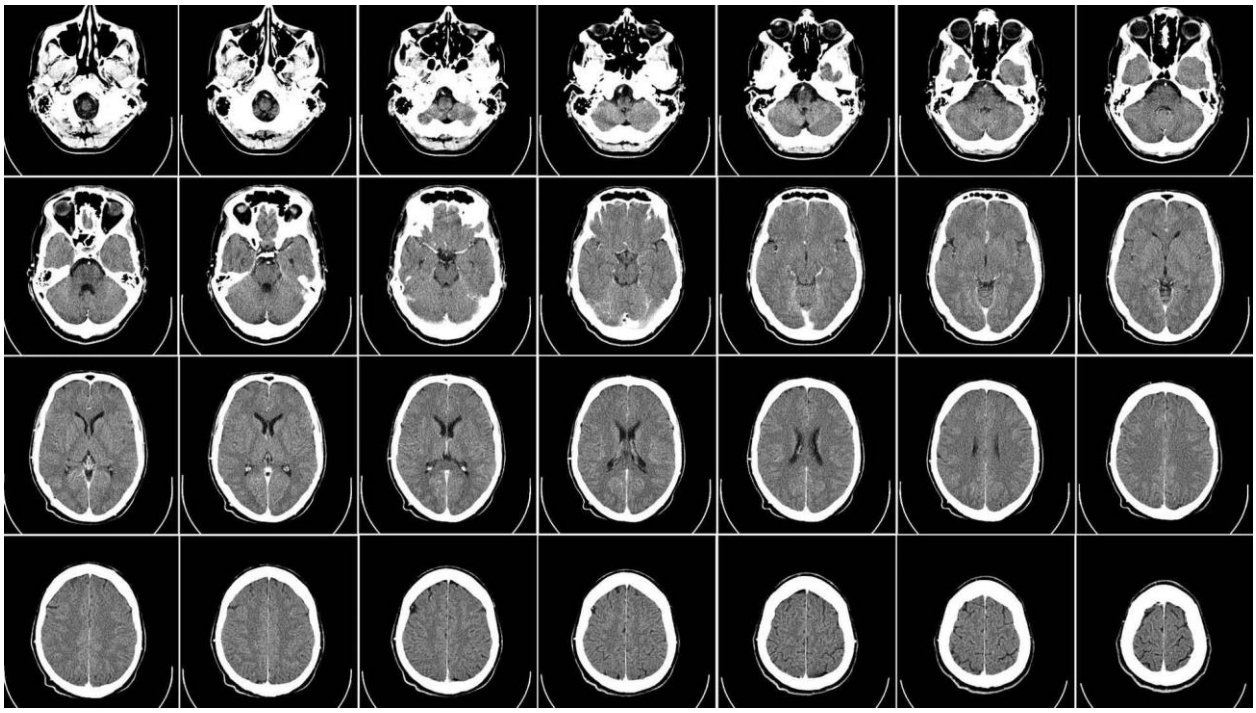


Рисунок 1.3.1 Приклад збереження КТ знімків голови у форматі DICOM

РОЗДІЛ 2. СЕГМЕНТАЦІЯ

2.1 Попередня обробка вхідних даних

Дуже важливим етапом перед застосуванням будь-якого алгоритму комп’ютерного зору, як правило, є попередня обробка зображення. Це необхідно для того, щоб покращити якість вхідного зображення, прибрати шум та зайві елементи, які погіршують результат та дослідження об’єктів і моделей. Також попередня обробка може знадобитися для виділення областей або фрагментів нашого інтересу, збільшення або зменшення їх, обертання, розтягування тощо. Також ми можемо збільшувати контраст і яскравість зображення, або навпаки зменшувати.

2.1.1 Фільтрація шуму

Шум на зображенні може бути спричинений багатьма причинами – нерівномірним або недостатнім освітленням, недосконалим апаратним забезпеченням або навколишніми чинниками. Наприклад, на знімках, які ми будемо досліджувати, нерідко трапляється шум через властивості проникнення хвиль через тіло людини. Коли хвиля виходить через дуже густу ділянку тіла, і потрапляє в менш густу, відбувається різкий перехід, тому може створюватися так звані артефакти – спотворення деяких пікселів. Тому, будемо застосовувати попередню фільтрацію для уникнення шумів. Існує багато різних фільтрів, але розглянемо найпопулярніші – медіанний та гаусівський фільтр.

Медіанний фільтр можна уявити як деяке вікно непарної розмірності (наприклад, 3 на 3 або 9 на 9), яке рухаючись по зображенню додає всі значення пікселів та присвоює центральному пікселю вікна їх середнє арифметичне. Якщо формалізувати, то вводиться таке вікно (англ. “kernel”), наприклад для вікна розміром 3 на 3:

$$\frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \quad (2.1)$$

і рухаючись по зображенню, значення зображення і вікна перемножуються поелементно і результат записується в центральну під вікном комірку.

Припустимо ми маємо такий кусочок зображення:

$$\begin{matrix} 255 & 200 & 230 \\ 0 & 250 & 234. \\ 0 & 0 & 255 \end{matrix}$$

Центральний елемент, при застосуванні медіанного фільтра буде замінений на (255 + 200 + 230 + 0 + 250 + 234 + 0 + 0 +255) / 9 = 158. Слід зазначити, що значення округлилось, бо використовується 8-бітне кодування зображення (значення пікселю належить проміжку [0,255]). Медіанний фільтр досить гарно долає імпульсні шуми (наприклад шуми типу “сіль та перець”), а також рівномірний цифровий шум.

Гаусівський фільтр базується на функції Гауса і достатньо гарно розмиває зображення. Функція Гауса для двовимірного випадку має такий вигляд:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}, \quad (2.2)$$

де x та y – це координати, а σ – стандартне відхилення.

Цю функцію ми використовуємо для заповнення відповідного вікна, де початком координат буде центральній елемент вікна. Наприклад, для $\sigma = 1$, матриця розміром 3 на 3 буде мати вигляд:

$$\begin{matrix} 0.077 & 0.123 & 0.077 \\ 0.123 & 0.195 & 0.123. \\ 0.077 & 0.123 & 0.077 \end{matrix}$$

Процес застосування фільтру аналогічний до медіанного фільтру.

2.1.2 Пошук границь та градієнту

Одним із можливих кроків сегментації зображення може бути пошук границь. Границя – це точки на цифровому зображенні, де яскравість зображення різко змінюється. Припустимо ми маємо такий шматочок зображення:

$$\begin{matrix} 255 & 255 & 255 \\ 0 & 255 & 255. \\ 0 & 0 & 255 \end{matrix}$$

Ми чітко розділити нулі від 255. Саме тут і буде пролягати границя.

Ще одним важливим терміном є градієнт. Градієнт – це векторна величина, що показує напрямок найшвидшого зростання деякої величини. У випадку цифрових зображень, градієнт показує напрямок зростання яскравості зображення. Зазвичай, перед пошуком границь відбувається пошук градієнтів зображення. Градієнт може знаходитися і як незалежний крок для покращення зображення. Наприклад, ми можемо використати оператор Лапласа – знайти лапласіан (зображення, до якого використали цей оператор), та відняти від оригінального зображення. Контури об’єктів на такому зображені стануть більш чуткими.

Одним з найефективніших алгоритмів знаходження границь є алгоритм Кенні (іноді Канні) або “оператор Кенні”, який був запропонований австралійським вченим Джоном Кенні (John F. Canny) в 1986 році. Зараз цей алгоритм використовується майже всюди і залишається найбільш ефективним.

Алгоритм складається з п'яти окремих кроків:

1) Згладжування. Розмиття зображення для видалення шуму.
   Зазвичай для розмиття використовують гаусів фільтр.
2) Пошук градієнтів. Межі відзначаються там, де градієнт зображення набуває максимальне значення.
   Для цієї задачі використовують оператор Собеля.
3) Придушення не-максимумів (англ. “non-maximum suppression”). Тільки локальні максимуми відзначаються як границі.

4) Подвійна порогова фільтрація. Потенційні границі визначаються за певними порогами (англ. “threshold”).
5) Трасування області неоднозначності. Підсумкові межі визначаються шляхом придушення всіх країв, незв'язаних з певними (сильними) межами.

Ми будемо використовувати пошук границь для кращої сегментації зображень.

2.1.3 Морфологічні перетворення

Морфологічні перетворення це прості операції, які застосовуються над бінарними зображеннями. На вхід подається зображення та віконце (kernel), яке застосовується на зображенні. Дві основних операції, на яких будуються всі інші – це ерозія і дилатація.

Ерозія – це процес, коли ми навмисно прибираємо зафарбовані пікселі (пікселі зі значенням 1) на границях. Піксель у вихідному зображенні буде зафарбований лише у тому випадку, якщо всі пікселі під вікном дорівнюють 1, інакше ми його прибираємо.

Дилатація – процес зворотний до ерозії, ми додаємо нові зафарбовані пікселі на границях. Піксель у вихідному зображенні буде зафарбований у тому випадку, якщо хоча б один піксель під вікном дорівнює 1.

Ці операції є базовими, а послідовне їх використання формує нові види морфологічних перетворень. Наприклад “відкривання” - це процес послідовного застосування спочатку ерозії, а потім дилатації. Цей процес допомагає прибрати шум на зображені. А ось “закривання” навпаки - це процес послідовного застосування спочатку дилатації, а потім ерозії. Цей процес допомагає закрити дірки на зображені. Ми будемо використовувати “закривання” і “відкривання” для покращення масок, які будуть використовуватися для виділення потрібних ділянок зображень.

2.1.4 Матричні просторові перетворення

Маючи задані точки у просторі, ми можемо робити перетворення над ними, такі як переміщення, обертання навколо прямої, масштабування відносно заданої точки тощо. Ці операції дуже зручно вводити за допомогою матриць. Наприклад ми маємо точку $A(x, y, z)$, ми хочемо перемістити її на вектор $T(T_x, T_y, T_z)$. Тоді ми можемо зробити наступне:

$$\begin{bmatrix} x' \\ y' \\ z' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & T_x \\ 0 & 1 & 0 & T_y \\ 0 & 0 & 1 & T_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \times \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \tag{2.3}$$

де вектор $A'(x', y', z')$ буде шуканим.

Так само водяться і інші матриці. Матриця масштабування з фактором розширення $S(S_x, S_y, S_z)$ відносно заданої точки (x_p, y_p, z_p) має наступний вигляд:

$$\begin{bmatrix} S_x & 0 & 0 & x_p(1 - S_x) \\ 0 & S_y & 0 & y_p(1 - S_y) \\ 0 & 0 & S_z & z_p(1 - S_z) \\ 0 & 0 & 0 & 1 \end{bmatrix}. \tag{2.4}$$

Матриця обертання навколо вектору $\bar{v}(a, b, c)$ на кут α має наступний вигляд:

$$\begin{bmatrix} a^2K + cos\alpha & abK - csin\alpha & acK + bsin\alpha & 0 \\ abK + csin\alpha & b^2K + cos\alpha & bcK - asin\alpha & 0 \\ acK - bsin\alpha & bcK + asin\alpha & c^2K + cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.5}$$

де $K = 1 - cos\alpha$ та $|\bar{v}| = 1$.

За допомогою цих матриць ми можемо легко та швидко маніпулювати даними, наприклад надати нашій моделі реалістичні пропорції, бо знімки часто робляться з різним проміжком від 1 до 3 мм, тому щоб відновити їх треба скористатися матрицею масштабування. Також щоб відцентрувати модель, ми будемо використовувати матрицю зсуву. А ось матрицю обертання ми будемо

використовувати, щоб перевернути модель (людину) знизу-вверх або з горизонтального положення в вертикальне.

2.1.5 Застосування масок

Часто, щоб виділити певні області на зображенні використовуються маски. Маска це певне бінарне зображення, де зафарбовані пікселі представляють пікселі оригінального зображення, які мають залишитися або не залишитися на вибір. Замість пікселів, які прибираються з оригінального зображення можна поставити 0 або деяке інше значення (наприклад для КТ знімків будемо ставити -1000, що відповідає повітрю). Маски зручно використовувати як до сегментації зображення, так і після. Ми будемо використовувати маски для виділення певних органів або тканин на знімках (див. додаток Е).

2.2 Порогування

Найпростішим видом сегментації є порогування (англ. “thresholding”) – процес, коли ми представляємо зображення в бінарному вигляді. Існує багато різновидів цього процесу, але основним принципом цього перетворення є те, що обирається певний поріг T, а потім всі пікселі, значення яких менше за поріг T замінюються на 0, всі інші на 1 (або на 255, в залежності від формату збереження зображення). Часто такий спосіб називають “бінарним порогуванням” або “простим порогуванням”.

Іншим різновидом є “адаптивне порогування”, яке підлаштовує визначений поріг T в залежності від значень пікселів, що знаходяться навколо нього. Так спосіб покращує результат зображень, що мають нерівномірне освітлення.

Існують також методи порогування, які автоматично підбирають поріг T, наприклад “бінаризація Оцу” (англ. “Otsu's binarization”), яка спирається на побудові гістограми зображення та поділом її на дві частини. Такий спосіб оптимальний, коли ми не знаємо розподіл значень на зображенні.

2.3 Алгоритм “ріст регіонів”

Ріст регіонів (англ. “Region Growing”) достатньо простий, але досить ефективний алгоритм сегментації, який застосовується як і для двовимірних зображень, так і для тривимірних. Цей алгоритм має декілька різновидів, які по-різному працюють в різних ситуаціях. Основними принципами цього сімейства алгоритмів наступні:

- копіюємо структуру оригінального зображення, в якій будемо відмічати у відповідність пікселі, які нас цікавлять – назвемо “маскою” зображення;
- обираємо значення порогу T;
- обираємо початковий піксель (воксель у тривимірному випадку), який називається зерном (англ. “seed”);
- створюємо чергу, в яку додаємо наше зерно;
- для кожного пікселя в черзі перевіряємо його сусідів – чи різниця між поточним пікселем і сусідом не перевищує поріг T;
- якщо поріг не перевищує, то додаємо перевіреного сусіда в чергу і помічаємо його в масці;
- повторюємо, поки черга не стане порожньою.

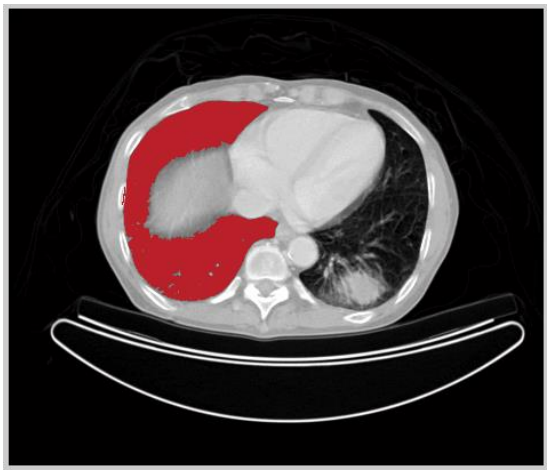


Рисунок 2.3.1 Приклад роботи алгоритму “ріст регіонів” - виділення лівої частини легені

Варто пояснити деякі кроки алгоритму. Оскільки алгоритм “ріст регіонів” може сегментувати не все зображення, а лише виокремити той об’єкт, який нас цікавить, то відповідно зерно треба обирати в межах цього об’єкту - навколо цього зерна буде розростатися область сегментації. Можна обирати декілька початкових зерен для однієї області, що може дозволити покращити результат. Додаємо в чергу ми лише ті пікселі, які ще не помічені. Під кінець виконання алгоритму пікселі не зможуть подолати поріг і черга не буде поповнюватися. Поріг в алгоритмі залежить від значень вокселів, і залежить від розподілу значень в об’єкті, який нас цікавить. Яких саме сусідів ми будемо перевіряти залежить від ситуації – для тривимірного зображення це можуть буди лише 6 пікселів, які безпосередню є сусідами вокселя, або можна перевіряти сусідніх 24 пікселя. Звичайно це збільшить кількість перевірок, але може дати і кращий результат.

Головною перевагою алгоритму “ріст регіонів” є те, що він застосовується в тривимірному просторі і в нашому випадку буде зв’язувати елементи на різних зрізах томограми. Тому нам не треба перевіряти яка область зі зрізу А належить області на зрізі Б, який є сусіднім зрізом А. Дуже легко запускати алгоритм до тих пір, поки не залишиться непомічених вокселів і в результаті ми отримаємо багато сегментованих областей.

Існують такі різновиди алгоритму:

- “зв’язний поріг” (англ. “connected threshold”) – який перевіряє сусідні вокселі, і включає їх, якщо вони в межах визначеного порогу $T(t_{min}, t_{max})$;
- “впевнено зв’язний поріг” (англ. “confidence connected”) – який зв’язує сусідні вокселі, спираючись на стандартному розподілі $N(\mu, \sigma) = \mu \pm c\sigma)$, де $\mu -$ середнє значення усіх зерен, $\sigma -$ дисперсія цих значень, і $c -$ задана константа;
- “зв’язні сусіди” (англ. “neighborhood connected”) – який перевіряє, чи всі сусіди даного пікселя долають заданий поріг T, і тільки тоді помічає даний піксель у маску;

РОЗДІЛ 3. ВІЗУАЛІЗАЦІЯ

3.1 Загальні відомості

Для двовимірних растрових зображень найдрібнішою одиницею вимірювання є піксель, яку можна представити як точку або квадрат. Тоді, якщо об’єднати декілька растрових зображень, накладаючи одне зображення на інше, то пікселі вже будуть знаходитися у тривимірному просторі і будуть представляти з себе не квадрати, а куби. Ось такі куби, прийнято називати вокселями, а сукупність вокселей часто називають сіткою вокселей, що являє собою тривимірне скалярне поле. Саме вокселі ми отримаємо після об’єднання наших томографічних знімків в єдине ціле, а сама модель буде називатися воксельною. Також, воксельні моделі використовують під час тривимірного друку моделей.

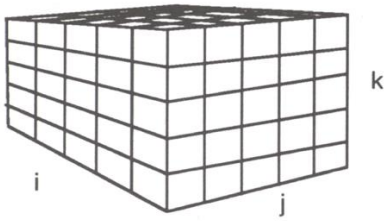


Рисунок 3.1.1 Воксельна модель і її представлення у вигляді кубів

Тепер, по аналогії з пікселем і вокселем, введемо поняття ізолінії та ізоповерхні. Ізолінія, або лінія рівня, або контурна лінія – це така лінія, всі точки якої мають однакове значення. Такі лінії можна побачити на топографічних картах, де лінії представляють певну висоту ландшафту. Також такі лінії можна побачити на різних схемах, кресленнях і у графіці. Ізолініями можна вважати спосіб представлення функції від двох змінних на площині. Тепер розглянемо поняття ізоповерхні. Ізоповерхня, або поверхня рівня – це поверхня, всі точки якої мають однакове значення. Ізоповерхні мають широке використання у таких сферах як обчислювальна гідродинаміка, геофізика, метеорологія та медицині. При створенні тривимірної моделі, ми будемо виділяти ізоповерхню з воксельної моделі об’єкта. Ця поверхня буде лежати між вокселями,

які належать об’єкту нашого інтересу, та зовнішнім простором (повітря або вокселі зі значенням 0). При цьому, сама ізоповерхня буде представлена полігональною сіткою (англ. “mesh”)– набір вершин, ребер і граней (грані в моделі будемо представляти трикутниками). Щоб виділити таку ізоповерхню, можна використовувати винайдені алгоритми: “крокуючі кубики” (“marching cubes”), крокуючі тетраедри (“marching tetrahedra”), “поверхнева мережа” (“surface nets”), “асимптотичний вирішальник” (“asymptotic decider”) та “подвійний контур” (“dual contouring”), однак найпопулярнішим і найпростішим способом залишається алгоритм “крокуючі кубики”.

3.2 Об'ємний рендеринг

Об’ємний рендеринг це процес візуалізації двовимірних даних у просторі. Прикладами таких даних можуть бути не тільки топографічні знімки (КТ, МРТ), а ще знімки мікроскопії, ультразвук та сейсмічні дані. Зазвичай вхідні дані об’ємного рендерингу є множина зрізів рівного розміру і товщини, які складаються з однакової кількості пікселів. Існує два типи об’ємного рендерингу - прямий і непрямий рендеринг. Вибір конкретного типу залежить від поставленої задачі та інколи від засобів представлення моделі.

3.3 Прямий об'ємний рендеринг

Головною ідеєю прямого рендерингу (англ. “direct volume rendering”) є отримання тривимірної моделі одразу з воксельної моделі. Вокселі проеціюються на екран (двовимірне зображення) у відповідний піксель. Вокселі можуть зафарбовуватися як напівпрозорі і тому є можливість заглянути в об’єкт безпосередньо і побачити, що в ньому знаходиться.

При прямому рендерингу можуть використовуватися два підходи: передовий (англ. “forward”) і зворотний (англ. “backward”). При передовому процесі ми визначаємо колір вихідного зображення для кожного пікселя – тому не всі вокселі можуть прийняти участь у результаті, а при зворотному – ми обробляємо всі вокселі, які потім проеціюють на вихідне зображення.

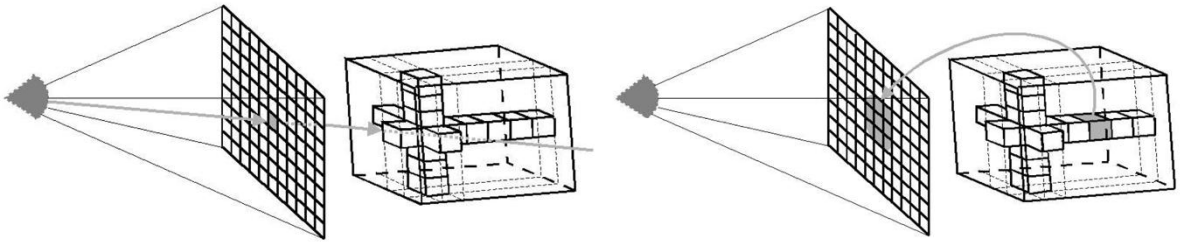


Рисунок 3.3.1 Передовий (англ. “forward”) і зворотний (англ. “backward”) підхід прямого рендерингу

Існують декілька алгоритмів прямого рендерингу, але найбільш використовуваний – “кидальник променів” (англ. “ray casting”).

3.3.1 Алгоритм “кидальник променів”

Алгоритм “кидальник променів” симулює те, як людина сприймає навколишній світ. Є певна точка зору, з якої випускаються промені у напрямку до моделі. Після того як промінь зіштовхнеться з першим непрозорим пікселем, ми відображаємо його у відповідне положення екрану. Оскільки піксель може бути напівпрозорим, промінь продовжує рухатися далі і відображати наступні вокселі. Звичайно промінь у більшості випадках не буде чітко проходити через воксель, тому значення вокселів інтерполюються – можуть братися найближчі сусіди, або бідь який інший спосіб. Найбільшим недоліком алгоритму є те, що він дуже затратний у часі виконання. Тому вводяться деякі ідеї для прискорення роботи.

Перша ідея – припиняти просунення променю далі, якщо наступні вокселі не будуть сильно впливати на саме зображення (тобто попередні вокселі настільки непрозорі, що наступні вокселі просто не видно).

Друга ідея – перестрибувати області, які мають однакову інтенсивність і рівномірний розподіл без втрати релевантної інформації. Ця ідея має багато різновидів реалізації. Найвідомішими є:

- “структура ієрархічного збереження” (англ. “hierarchical spatial data structure”) ділить воксельну модель на куби, в яких вокселі мають однакове значення, які легко перестрибувати;

- “обмеження об’єктів в куби” (англ. “bounding boxes around objects”), під час якого об’єкти обмежуються в куби, і таким чином ми перестрибуємо порожні вокселі (повітря);
- “адаптивний прохід променю” (англ. “adaptive ray traversal”) застосовує до порожніх вокселів швидкий прохід – може перестрибувати відразу декілька поки не зустріне непорожній воксель;
- “хмари близькості” (англ. “proximity clouds”) – розширення попередньої реалізації – вводить відстань до найближчого непорожнього вокселя – на цю відстань можна безпечно перестрибнути;
- “однорідне прискорення” (англ. “homogeneity-acceleration”) зменшує розміри моделі, з’єднуючи декілька сусідніх вокселів в один – таким чином швидко оминаються однорідні області.

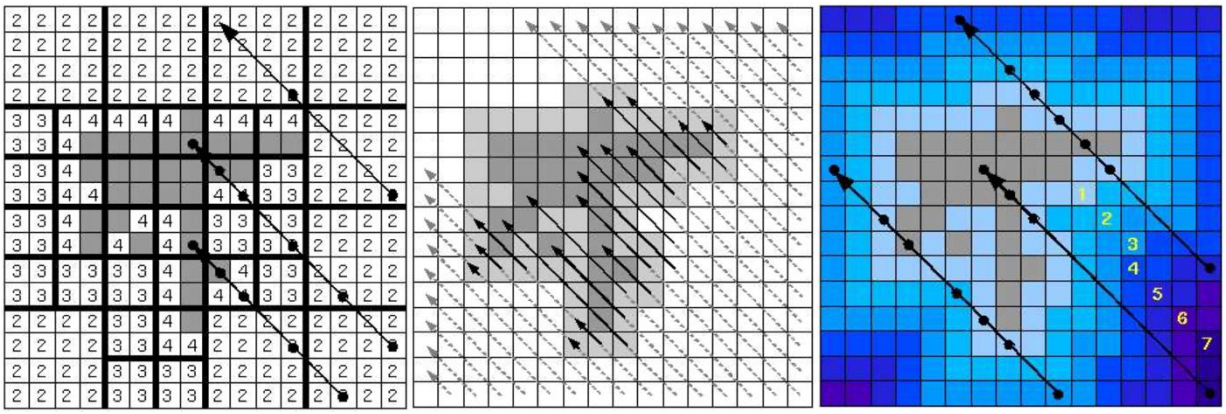


Рисунок 3.3.1.1 Приклади проходу променів в різних реалізаціях: (з права на ліво) “структура ієрархічного збереження”, “адаптивний прохід променю”, “хмари близькості”

Ці реалізації можна комбінувати (наприклад, використати “хмари близькості” з “однорідним прискоренням”), отримуючи більш швидкий рендеринг, а також можна застосувати ці підходи під час взаємодії користувача з моделлю (наприклад, коли користувач почав обертати модель, можна навмисно використати прискорення, але пожертвувати якістю моделі в деяких випадках).

3.3.2 Реалістичне зафарбовування тканин

В наступні таблиці наведено відповідність кольорів певним значення HU знімків КТ, які можна використати для реалістичного зафарбовування тканин людини. Проміжні значення мають інтерполюватися.

Таблиця 3.3.3.1 Реалістичне зафарбовування КТ знімків

Тип тканини	**HU значення**	**RGB колір**
Повітря	-1000	(0, 0, 0)
Легені	від -600 до -400	(194, 105, 82)
Жир	від -100 до -60	(194, 166, 115)
М’які тканини	від 40 до 80	(102 ↔ 153, 0, 0)
Кістки	від 400	(255, 255, 255)

3.4 Непрямий об'ємний рендеринг

Непрямий рендеринг (англ. “indirect volume rendering”), на відміну прямому, спочатку обробляє отримані на вхід дані, переводячи їх в новий домен, а саме воксельна модель перетворюється на ізоповерхню, яка представлена полігональною сіткою. Такий тип рендерингу дуже часто обирається через його швидкість, можливість прискорити за рахунок апаратного забезпечення, хоча сам результат візуалізації не є дуже точним.

Першою спробою виділення поверхні був алгоритм “контурне трасування”. Основна ідея цього підходу було знаходження контурів об’єкту нашого інтересу на кожному зрізі та створення між контурами граней. Однак цей алгоритм мав багато недоліків, головні з яких: як правильно з’єднувати контури; між якими контурами треба робити грані; що робити, якщо граней декілька.

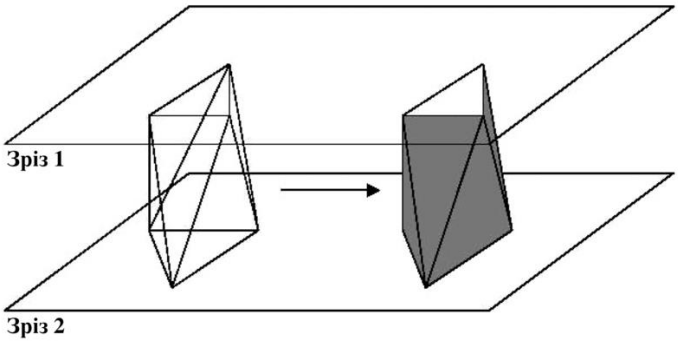


Рисунок 3.4.1 Приклад об'єднання об'єктів на сусідніх зрізах

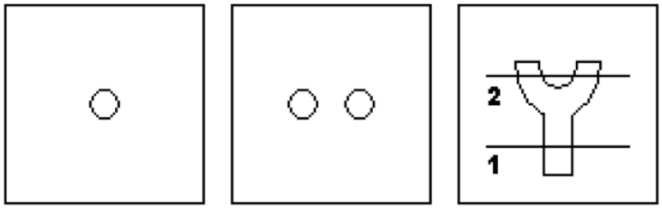


Рисунок 3.4.2 Приклад неоднозначності з'єднання контурів на сусідніх зрізах

Потім був винайдений інший підхід, який назвали "підгонкою поверхні". Цей підхід має багато різновидів, але основні принципи такі:

- обираємо ізозначення c;
- помічаємо вокселі '+', якщо $f(x, y, z) \geq c$, інакше '–';
- поміщаємо графічні примітиви (наприклад, трикутники), між вокселями з різними знаками.

Таким чином, ми наче огортаємо вокселі, які нас цікавлять і отримуємо шукану модель.

3.4.1 Алгоритм "непрозорі кубики"

Алгоритм "непрозорі кубики" (англ. "cuberille (opaque cubes)") був запропонований у 1979 році Т. Херманом. Основні кроки алгоритму наступні:

- бінаризуємо значення вокселів відносно заданого значення;
- знаходимо граничні ("лицьові") вокселі та серед них шукаємо грані, нормалі котрих направлені назовні;

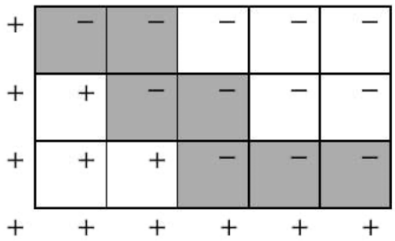


Рисунок 3.4.1.1 Приклад знаходження “лицьових” вокселів

Цей алгоритм поклав початок для наступних алгоритмів, які покращували пошук граней і збільшували точність самої моделі.

3.4.2 Алгоритм “крокуючі кубики”

Алгоритм “крокуючі кубики” (англ. “marching cubes”) – був винайдений ще у 1987 році Вільямом Е. Лоренсоном та Харви Е. Кляйном. Цей алгоритм був створений для виділення ізоповерхні з множини вокселів і є базовим.

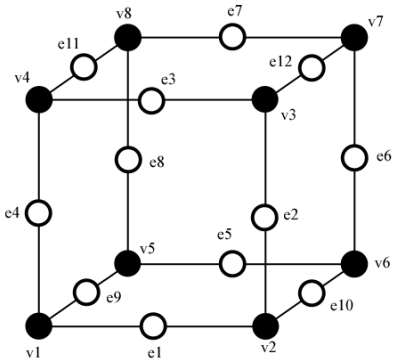


Рисунок 3.4.2.1 Запропонована нумерація вершин і ребер куба в алгоритмі “крокуючі кубики”

Основним елементом алгоритму є куб, вершини якого є суміжні вокселі нашої моделі. Відповідно до обраного ізозначення c, ми розділяємо вокселі на ті, що входять до нашої моделі (їх значення більше обраного c), і ті що знаходяться ззовні (значення менше c). Далі, для кожного такого кубу ми дивимося на його вершини та, спираючись на наш розподіл вокселів, розміщуємо трикутники всередині кубу.

Враховуючи те, що у нас всього 8 вершин, і кожна вершина може або входити, або не входити до об'єкту, то ми маємо всього $2^8 = 256$ різних випадків. Але більшість з них є однаковими з точністю до симетрії, тому, насправді, у нас всього 15 унікальних випадків. Тому щоб прискорити розміщення трикутників у кубі, ми можемо створити індекс, який за множиною вершин кубу буде казати як треба розмістити відповідні трикутники. Множину вершин будемо представляти бітовою маскою, довжина якої дорівнює кількості варіантів (15 або 256 в залежності від реалізації).

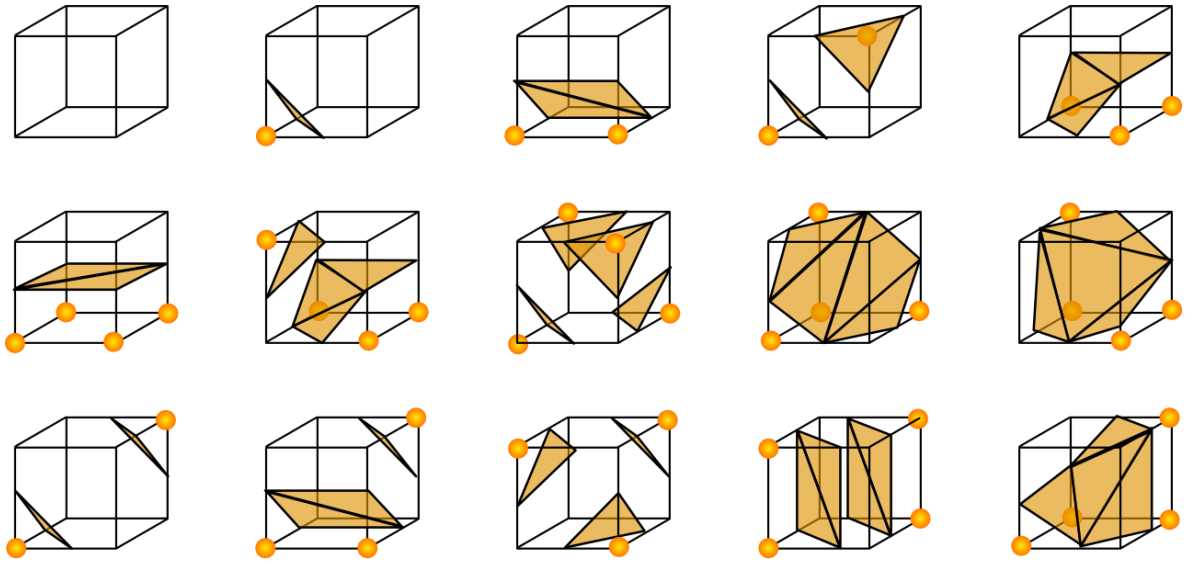


Рисунок 3.4.2.2 Унікальні способи розміщення трикутників в кубі для алгоритму "крокуючі кубики"

Отримана модель буде все ще не ідеальною, бо ми не враховуємо значення, яких набувають вокселі. Щоб покращити точність результуючої моделі, ми можемо скористатися лінійною інтерполяцією. Якщо між двома сусідніми вершинами кубу v_i і v_j, значення яких w_i і w_j відповідно, та відстань між якими d, має пройти поверхня, то, відповідно до обраного ізозначення c, поверхня пройде в точці:

$$p = v_i + \frac{c - v_i}{v_j - v_i} \times d \tag{3.1}$$

Таким чином, поверхні буду більш щільно огортати вокселі, і буду ближче до істинної моделі об’єкта.

Алгоритм “крокуючі кубики” має низку недоліків. Перший недолік це неоднозначність розміщення трикутників. В деяких випадках це може призвести до утворення дірок в моделі, або навпаки до з’єднання не бажаних елементів.

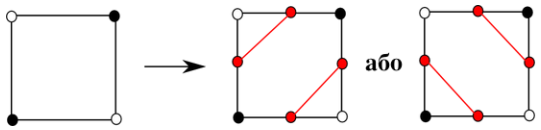


Рисунок 3.4.2.3 Приклад №1 неоднозначного розміщення граней в алгоритмі “крокуючі кубики”

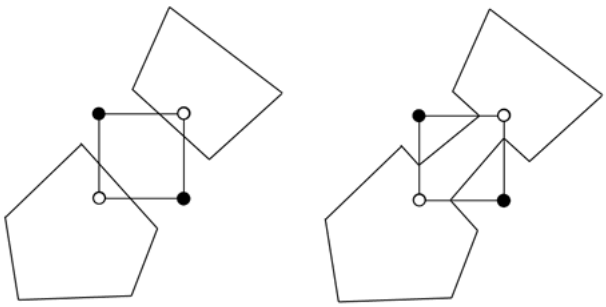


Рисунок 3.4.2.4 Приклад неоднозначного розміщення граней в алгоритмі “крокуючі кубики”

Також, алгоритм генерує дуже багато трикутників. В одному кубі може бути до 5 трикутників. Тому, розмір моделі може займати декілька десятків мегабайт для не дуже великих моделей.

Цікавим фактом є те, що цей алгоритм був запатентований до 2005 року, тому користуватися їм вільно було проблематично. Щоб обійти це обмеження, було створено алгоритм “крокуючі тетраедри” (англ. “marching tetrahedra”), який використовував ті самі принципи, що і цей алгоритм, але ділячи куб на тетраедри.

3.4.3 Алгоритм “крокуючі тетраедри”

Як було вже сказано, алгоритм “крокуючі тетраедри” майже ідентичний з алгоритмом “крокуючі кубики”, однак тепер кожен куб ділиться на 6 тетраедрів. Окрім цієї особливості, всі інші кроки алгоритму ідентичні. Цей алгоритм дає більш точну модель об’єкта за рахунок збільшення кількості ізоповерхневих трикутників, які генеруються під час алгоритму. Так само, як і для “крокуючих кубиків”, генерується індекс всіх можливих варіантів розміщення трикутників в тетраедрі, для більш швидкого їх знаходження, і використовується інтерполяція, для більш точної їх підгонки.

Перевагою тетраедрів над кубами є те, що тетраедри не мають неоднозначних випадків, їх топологія завжди однозначна. Також, всього існує $2^4 = 16$ випадків розміщення трикутників в тетраедрі, які можна скоротити до 3 унікальних з точністю до симетрії випадків. Але недоліком алгоритму може стати той факт, що генерується набагато більше трикутників, ніж під час алгоритму “крокуючі кубики”. Від 1 до 2 трикутників може бути розміщено в одному тетраедрі. Звичайно, це може бути і перевагою, однак моделі, які створюються під час “крокуючих тетраедрів”, будуть настільки тяжкими, що будуть потребувати більш потужне апаратне забезпечення для їх візуалізації. Результуючі моделі будуть займати приблизно в 3-4 рази більше пам’яті.

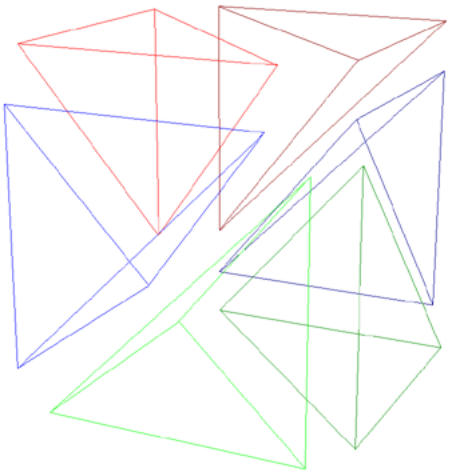


Рисунок 3.4.3.1 Спосіб розбиття куба на 6 тетраедрів

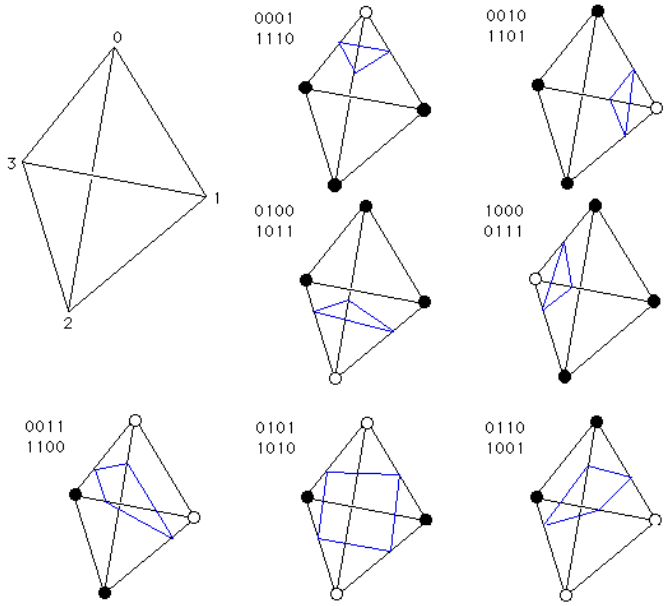


Рисунок 3.4.3.2 Унікальні способи розміщення трикутників в тетраедрах для алгоритму “крокуючі тетраедри”

3.4.4 Оптимізація візуалізації ізоповерхонь

Усі алгоритми підгонки ізоповерхні генерують дуже багато геометричних примітивів, що може стати завадою для інтерактивної візуалізації об’єктів. Проте існують способи вирішення цієї проблеми, такі як:

- Ієрархічна реконструкція поверхні;
- Реконструкція поверхні, що залежить від поля зору;
- Руйнування полігональної сітки.

При ієрархічній реконструкції поверхні генерується декілька версій однієї і тієї самої моделі з різною роздільною здатністю. Щоб згенерувати модель з меншою точністю, 8 суміжних вокселя з’єднуються в один, таким чином зменшуючи розміри самої воксельної моделі. Різні версії моделі відображаються відповідно до рівня деталізації (англ. “LOD – level of detail”), який може залежати від відстані до об’єкту. Зазвичай вводять 3 рівня деталізації, яких достатньо для відображення будь-якої сцени (див. додаток Б).

При реконструкції поверхні, ми можемо відображати модель в залежності від поля зору, а саме місце на яке ми дивимося буде мати більшу деталізацію, ніж ті, що знаходяться далі від точки фокусу або взагалі не входять в поле

зору. Таки чином сцени з великою кількістю різних об’єктів будуть генеруватися швидше, якщо в поле зору будуть потрапляти лише декілька з них.

При руйнуванні полігональної сітки, ми намагаємося спростити утворено ізоповерхню прибираючи зайві трикутники. Основний принцип руйнівних алгоритмів – це з’єднання декількох трикутників в один полігон, а потім триангулювати його на меншу кількість трикутників. Цей підхід може поєднуватися з ієрархічною реконструкцією, як альтернатива інтерполяції вокселів.

РОЗДІЛ 4. РЕАЛІЗАЦІЯ ПРОГРАМНОГО ПРОДУКТУ

4.1 Колекція даних

Щоб протестувати програму на реальних знімках було використано “The Cancer Imaging Archive” (TCIA) - це база даних з відкритим доступом медичних зображень для дослідження раку. Вона містить у собі безліч КТ і МРТ, що зберігаються у форматі DICOM. Дані організовані у колекції, які зазвичай мають загальний тип раку та/або представляють одну і ту саму частину тіла. Більшість колекцій у архіві зображень про рак можна отримати без облікового запису, однак деякі з них обмежуються лише певними користувачами, тому для доступу до них потрібен обліковий запис. Щоб завантажити колекцію знімків з TCIA треба скористатися додатком NBIA.

4.2 Мова програмування та програмне забезпечення

Програмний застосунок був цілком написаний на мові програмування Python версії 3.6.

Для тестування та дослідження роботи алгоритмів використовувалося середовище Jupyter Notebook версії 5.7.8. Всі бібліотеки завантажувалися за допомогою дистрибутива Anaconda версії 2019.10.

Створення програмного застосунку відбувалося за допомогою середовищ PyCharm версії 2019.3 та XCode версії 11.3.1.

4.3 Використані бібліотеки

Далі наведений список використаних під час розробки бібліотек.

- cv2 – бібліотека комп’ютерного зору OpenCV, включає готові алгоритми, функції для обробки зображень та взаємодію з ними. Використовувалася для попередньої обробки зображень (морфологічні перетворення, застосування масок, пошук границь та градієнтів, фільтрацію шумів) та сегментації (порогування).

- SimpleITK - це спрощений інтерфейс з відкритим кодом до бібліотеки “Insight Segmentation and Registration Toolkit” (ITK). Доступна на декількох мовах програмування, включаючи C ++, Python. Використовувалася для сегментації зображень – “ріст регіонів” та інших алгоритмів.
- vtk – обгортка для бібліотеки VTK з відкритим кодом для маніпулювання та відображення наукових даних. Використовувалася для 3D-рендерингу та тривимірної взаємодії з об’єктами.
- matplotlib - бібліотека для побудови графіків Python. Використовувалася для відображення зображень та побудови гістограм.
- numpy - бібліотека для наукових обчислень з Python. Використовувалася для представлення зображень у формі матриць та виконання операцій над ними (множення векторів та матриць, обертання, масштабування та транслювання об’єктів).
- scipy - призначена для виконання наукових і інженерних розрахунків. Інколи використовувалася для інтерполяції воксельної моделі.
- skimage - бібліотека, яка має набір алгоритмів обробки зображень. Включає в собі алгоритми виділення ізоповерхонь.
- stl – бібліотека, яка надає можливість створення тривимірних моделей у форматі STL.
- pydicom – надає можливість перегляду та завантаження файлів формату DICOM.
- PyQt5 – обгортка для бібліотеки Qt. Використовувалася для побудови графічного інтерфейсу.

4.4 Реалізація алгоритму на мові Python

Першим і найдовшим етапом розробки програмного застосунку було тестування алгоритмів сегментації та візуалізації у середовищі Jupyter Notebook та PyCharm. Були розглянуті усі наведені в теоретичній частині алгоритми, проведено їх порівняння та обрані найкращі з них. Під час цього етапу були

підібрані оптимальні методи, які покращували результати роботи, та проведено рефакторинг коду.

Під час другого етапу було створено програмні застосунки, які надавали зручний графічний інтерфейс. Перший застосунок використовує принципи прямого і непрямого рендерингу, генерує тривимірні моделі, отримуючи на вхід КТ або МРТ знімки, і написаний цілком на мові Python. Другий – додаток під iOS для перегляду моделей непрямого рендерингу.

4.4.1 Зчитування даних

Як було вже згадано, найпоширенішим форматом збереження медичних знімків є DICOM. Тому зчитуємо дані в цьому форматі. Щоб дізнатися товщину зрізів можемо скористатися полем ImagePositionPatient або SliceLocation – відняти два сусідніх зрізи. Товщина нам знадобиться для пропорційного відображення нашої моделі. У разі якщо це КТ знімки віднімаємо значення 1000, щоб значення відповідали таблиці 1.2.1.1. Всі дані зберігаємо у тривимірному масиві бібліотеки numpy – це і є наша воксельна модель.

4.4.2 Сегментація

На цьому кроці можливо декілька варіантів. Перший варіант застосувати або не застосувати алгоритм “ріст регіонів” (а саме використовується “зв’язний поріг” див. пункт 1.3), щоб виділити певну структуру. Наприклад ми можемо виділити шкіру і в результаті отримати маску (див. пункт 1.1.5). Ця маска дає нам можливість або виокремити тільки шкіру або навпаки видалити її з результату. Так само ми можемо виділити і інші структури і послідовно віднімати або об’єднувати від воксельної моделі, залишаючи тільки об’єкти, що нас цікавлять. До отриманих масок додатково застосовуємо морфологічні перетворення – “закривання” (в залежності від розміру знімків, але застосовувалося вікно розміром 11 на 11), щоб закрити дірки, за необхідності можна застосувати й інші.

Наступним кроком може стати порогування, оскільки в результаті можуть потрапити тканини різного значення HU, тому можемо виділити наприклад тільки м'язи. Приклади роботи можна побачити на додатках В, Г, Д, Е, К.

4.4.3 Візуалізація

Останнім кроком є візуалізація, яку поділимо на два варіанти: прямий або непрямий рендеринг.

Якщо обираємо прямий рендеринг, то у першу чергу можемо зменшити розмір нашої воксельної моделі. Використовуємо або зменшення у два або в чотири рази. Далі підготуємо “кидальника променів” - задаємо кольори зафарбовування пікселів та їх прозорість. Обов’язковим кроком є задання реального розміру одного вокселя – товщину ми визначили під час зчитування даних, ширина і довжина також задається у самих даних. Додатково задаємо параметри сцени – освітлення та тіні, якщо потрібні, та властивості матеріалу (ambient, diffuse, specular). Також задаємо початкове положення камери і її напрямок. Після застосування кидальника променів бачимо результат. Дуже зручно, що алгоритм написаний за допомогою OpenGL і оптимізований під GPU, що дає значне прискорення візуалізації. Також плюсом є те, що під час взаємодії з моделлю – її обертання – візуалізація навмисно погіршується, щоб покращити користувацьку взаємодію. Додатковим функціоналом є можливість зробити зрізи моделі, що краще розуміти внутрішню структуру моделі (див. додатки Ж і Л). Приклад прямого рендерингу можна побачити на додатках В, Г, Д, Е, Ж, К, Л.

У випадку якщо обирається непрямий рендеринг, то відразу застосовуємо алгоритм “крокуючі кубики”. На виході ми отримаємо модель – список вершин та граней. Відразу застосовуємо матричні перетворення для того щоб розтягнути модель у відповідності до розмірів одного вокселя (приклади наведені в додатках М і Н). Модель зберігаємо у форматі stl та можемо відкрити для перегляду або в стандартних переглядачах Windows та MacOS, або

відкриваємо у застосунку iOS для перегляду у доповненій реальності (див. додаток Р). Інтерфейс загальної програми наведений в додатку П.

4.5 Огляд готових рішень

Найбільш відомою і популярною програмою є 3D Slicer -це безкоштовна платформа з відкритим кодом для дослідження медичних зображень, обробки зображень та тривимірної візуалізації, постійно покращується і підтримується завдяки Національного інституту охорони здоров'я та всесвітньої спільноти розробників. Головним функціоналом є:

- читання / запис DICOM та різних інших форматів;
- інтерактивна візуалізація об'ємних воксельних зображень, полігональних сіток та тощо;
- ручне редагування;
- автоматична сегментація зображень.

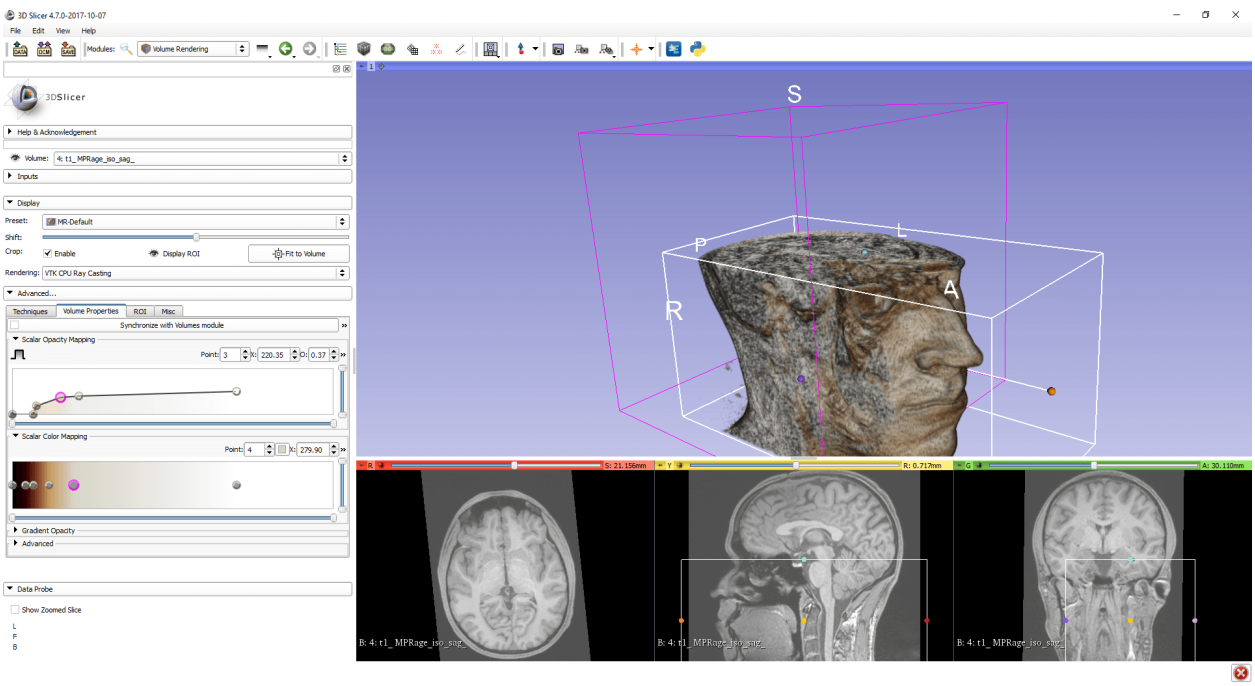


Рисунок 4.5.1 Графічний інтерфейс програми 3D Slicer

Ще відомою є програма 3D-Doctor. Функціонал програми дуже схожий на 3D Slicer. В даний час використовується в провідних лікарнях, медичних навчальних закладах та наукових організаціях по всьому світу.

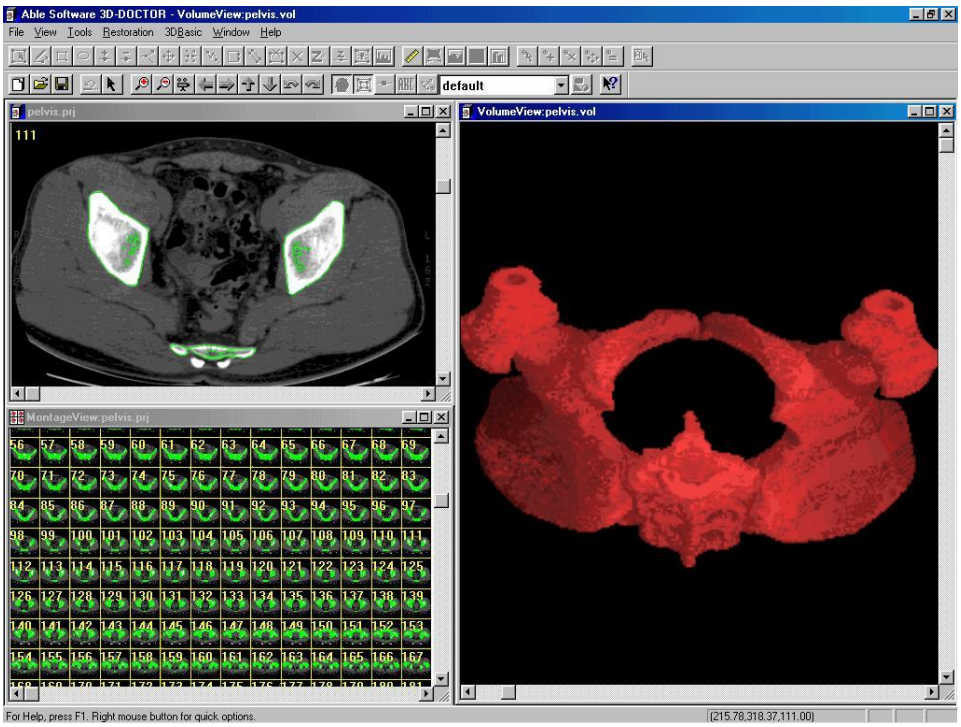


Рисунок 4.5.2 Графічний інтерфейс програми 3D-Doctor

Іншими відомим програмами є: Horos, Osiri-LXIV, InVesalius 3, Materialise Mimics, OsiriX, AMIDE, MedSeg, Vesalius3D.

Висновки

У цій роботі було досліджено та проаналізовано алгоритми тривимірної сегментації та візуалізації. Було досліджено предметну область та сфери використання запропонованої теми, оглянуто етапи сегментації та візуалізації медичних знімків. Було досліджено етапи обробки зображень та моделей – фільтрація шумів, пошук контурів, порогування та застосовування масок, перетворення морфологічні та матричні, зменшення розміру моделей та ізоповерхонь. Також велику увагу отримали типи візуалізації – прямий і непрямий рендеринг.

Також, під час роботи було створено програмний застосунок на мові програмування Python для обробки медичних знімків, сегментацію та перетворення їх у тривимірні моделі. Додатково було створено мобільний застосунок під iOS для перегляду моделей непрямого рендерингу в доповненій реальності. Були оглянуті та протестовані готові рішення.

Роботу можна продовжити дослідженням покращених методів сегментації за допомогою нейронних мереж та штучного інтелекту.

Список літератури

4D Volume Rendering [Онлайновий] // Nvidia. - Nvidia. - https://www.nvidia.com/content/GTC/documents/1102_GTC09.pdf.

Bankman Isaac Handbook of Medical Image Processing and Analysis [Книга]. - 2011. - cc. 4-17,77.

Bourke Paul Polygonising a scalar field [Онлайновий]. - May 1994 p.. - http://paulbourke.net/geometry/polygonise/.

Introduction to ITK Segmentation in SimpleITK Notebooks [Онлайновий] // SimpleITK. - http://insightsoftwareconsortium.github.io/SimpleITK-Notebooks/.

Oppelt Arnulf Imaging Systems for Medical Diagnostics: Fundamentals, Technical Solutions and Applications for Systems Applying Ionizing Radiation, Nuclear Magnetic Resonance and Ultrasound [Книга]. - 2011. - стр. 47-51.

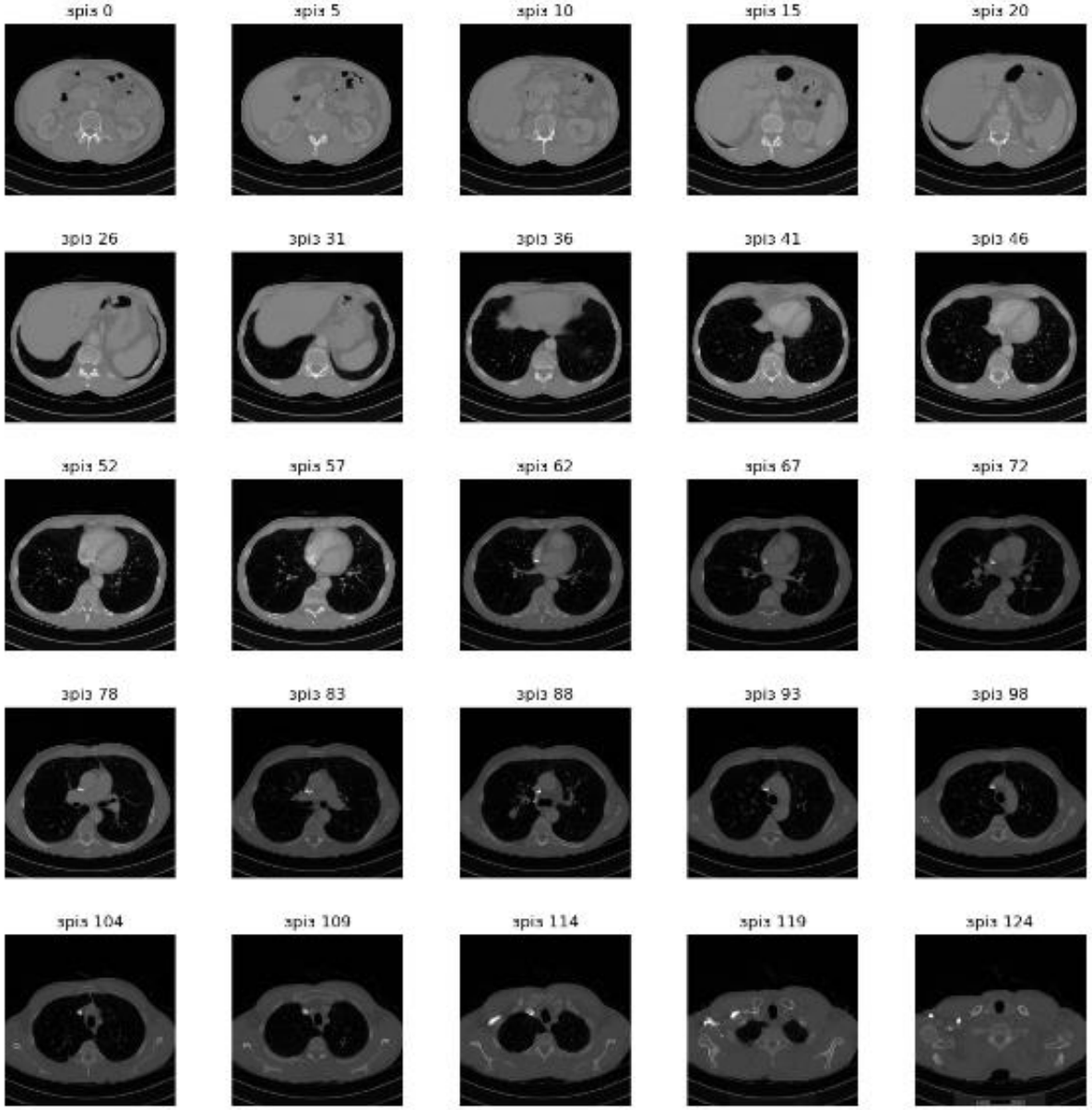
Prof. Dr. Markus Gross Dr. Marios Papas [Онлайновий] // CGL. - 2019 p.. - https://cgl.ethz.ch/teaching/cg19/home.php.

Vince John Geometry for Computer Graphics: Formulae, Examples and Proofs [Книга]. - 2006. - cc. 11-12,35-41.

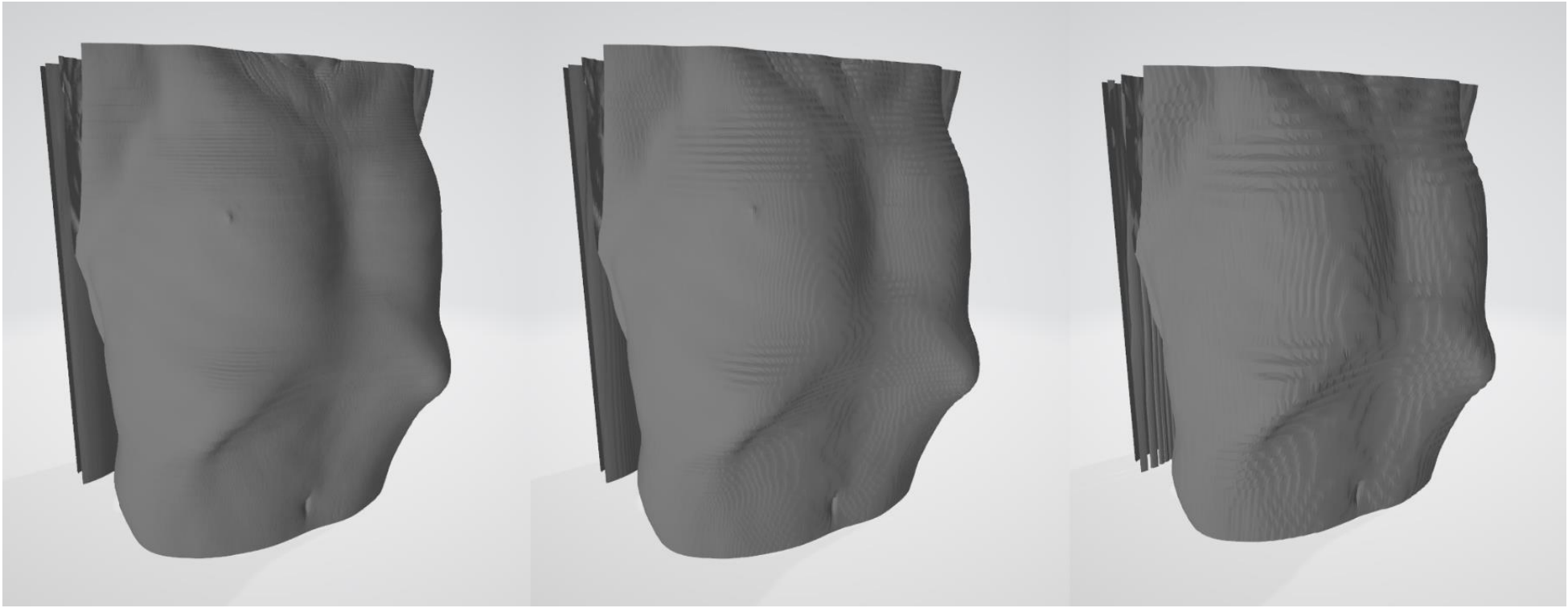
vtk [Онлайновий] // vtk. - https://vtk.org/.

Weiskopf Daniel GPU-Based Interactive Visualization Techniques [Книга]. - 2006. - cc. 11-18,21.

Додаток А (обов’язковий) Приклад комп’ютерної томографії

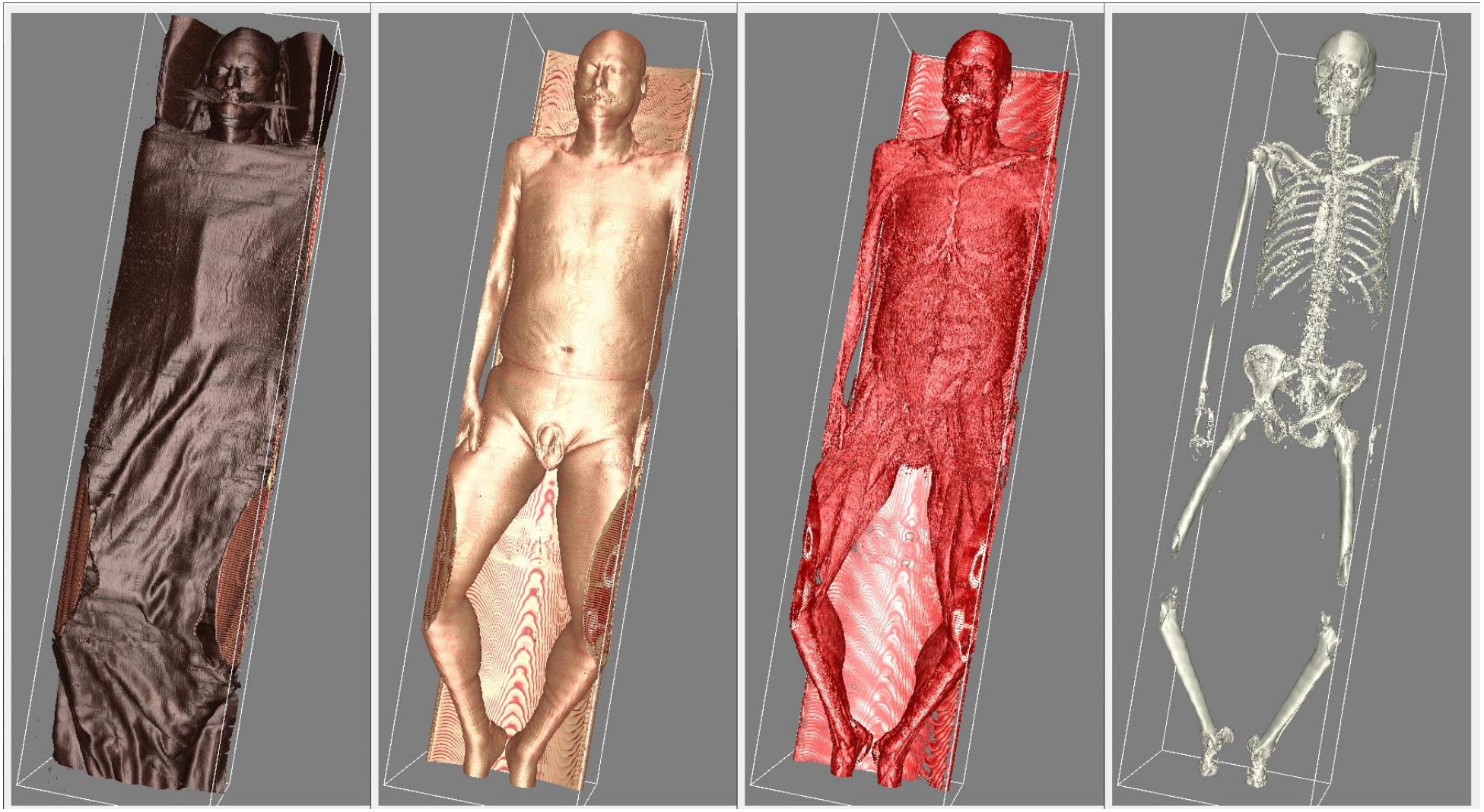


Додаток Б (обов’язковий) Вплив зменшення розміру воксельної моделі на результуючу ізоповерхню



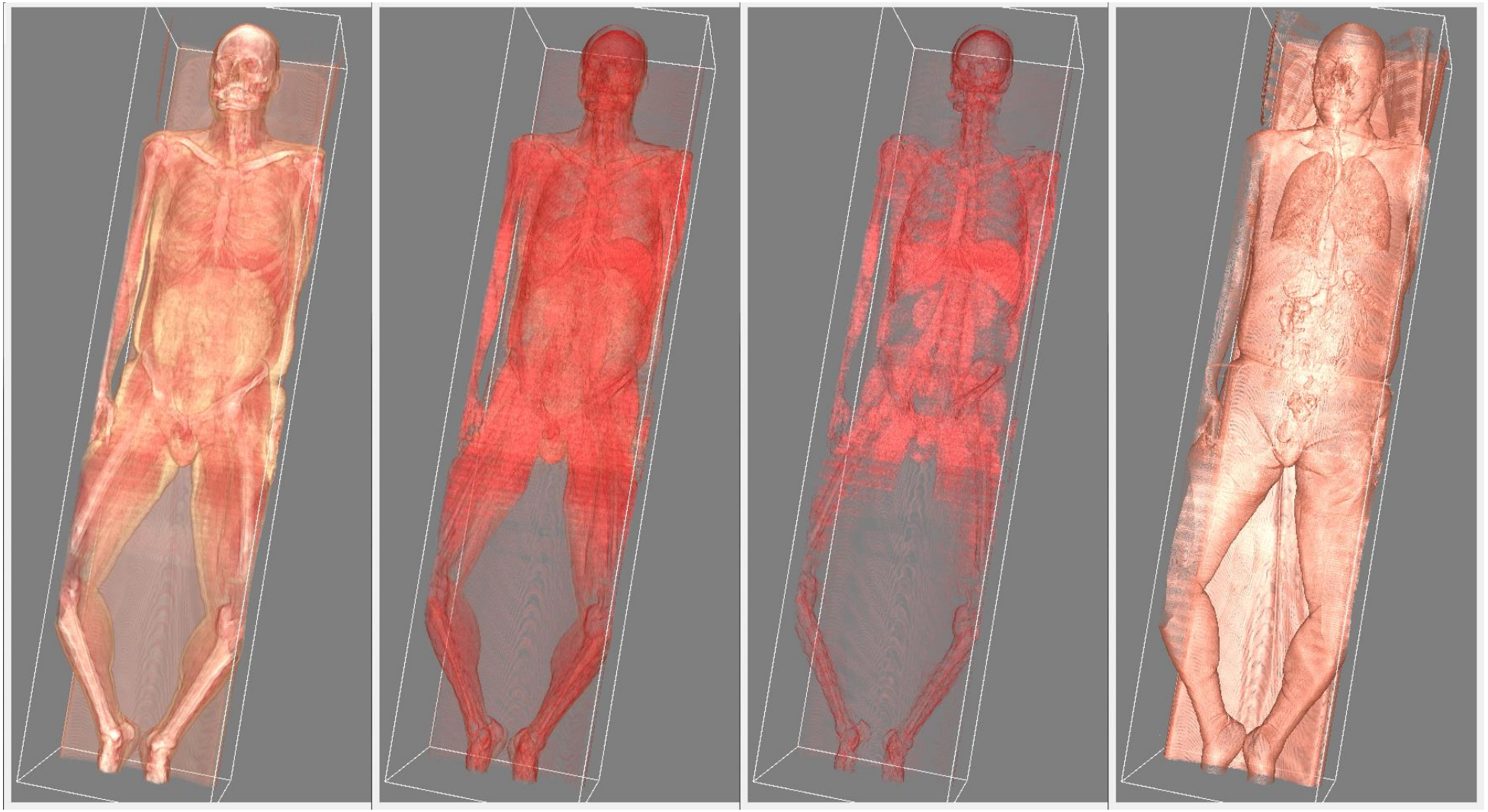
Оригінальна модель		*Зменшена у 2 рази*		*Зменшена у 4 рази*	
Вершини:	1 424 565	Вершини:	330 226	Вершини:	73 982
Грані:	2 834 012	Грані:	654 140	Грані:	145 518
Розмір:	143 МБ	Розмір:	39,9 МБ	Розмір:	6,93 МБ

Додаток В (обов’язковий) Приклад прямого рендерингу всього тіла з різним мінімальним порогом для виокремлення певних тканин



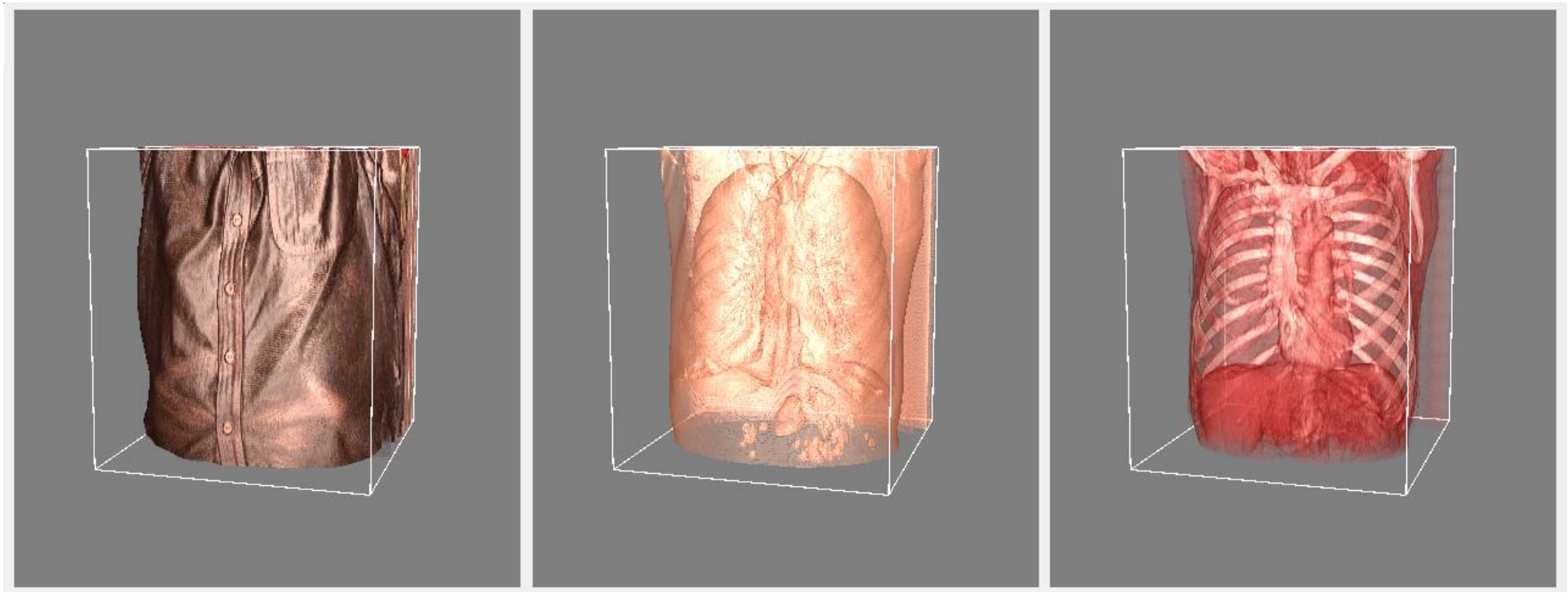
~ −1000 ~ −500 ~ 50 ~ 300

Додаток Г (обов’язковий) Приклад прямого рендерингу всього тіла з різним пороговим діапазоном та напівпрозорими вокселями



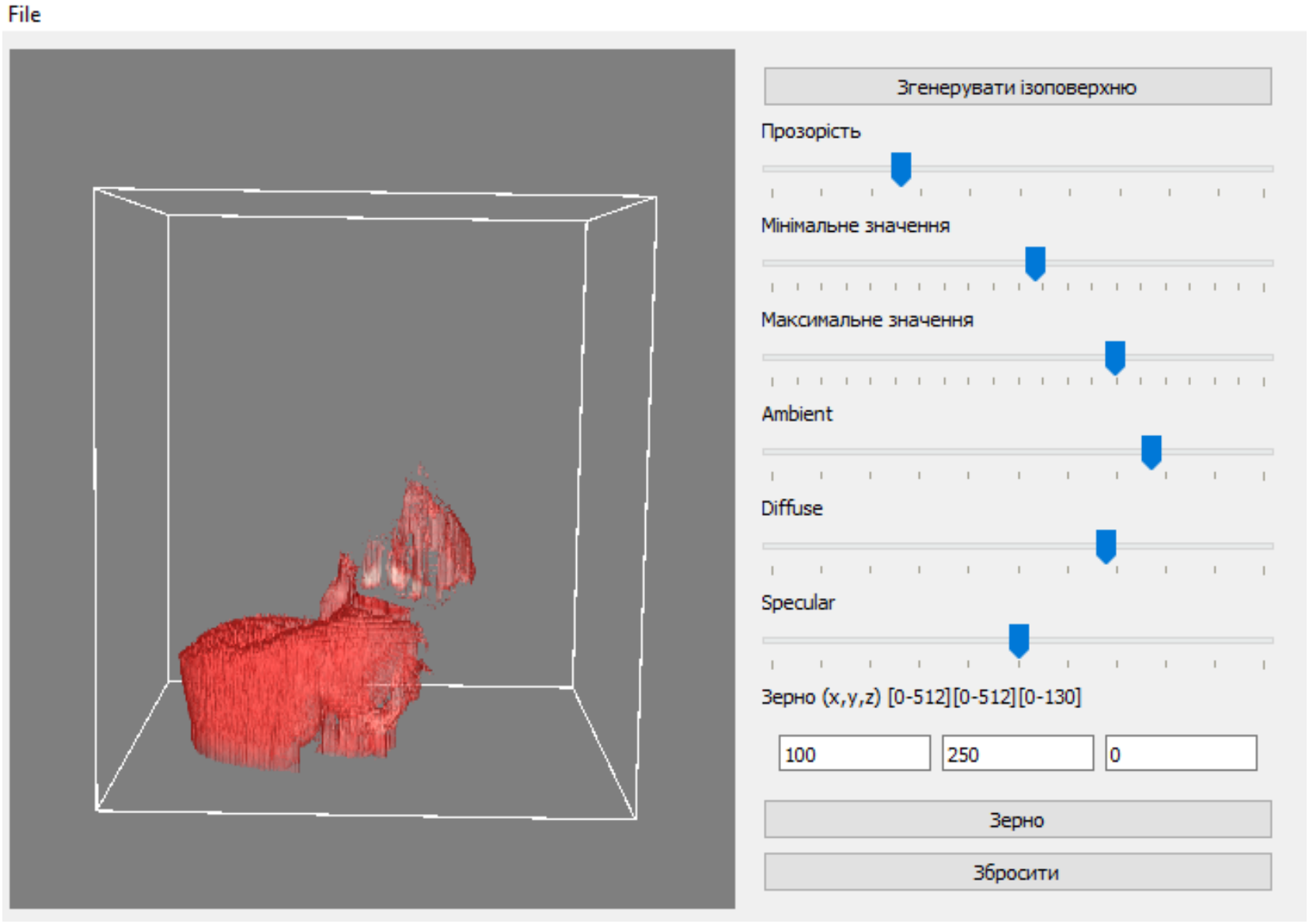
від −500 до 1000 | від 30 до 300 | від 100 до 300 | від −500 до 30

Додаток Д (обов’язковий) Приклад прямого рендерингу легень з різним пороговим діапазоном та напівпрозорими вокселями



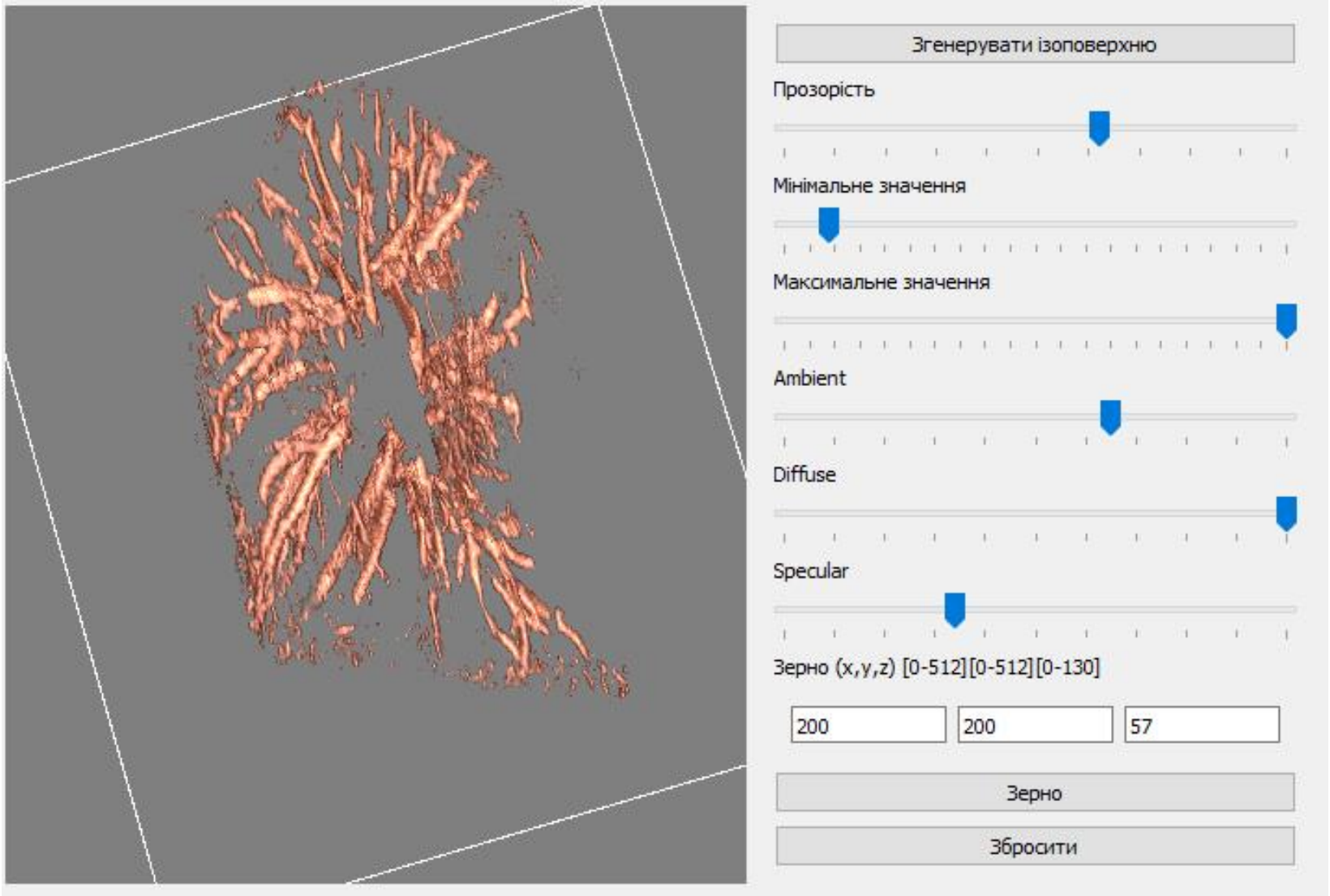
від −1000 до 1000 | від -500 до 30 | від 30 до 300

Додаток Е (обов’язковий) Приклад сегментації печінки людини за допомогою алгоритму «ріст регіонів»

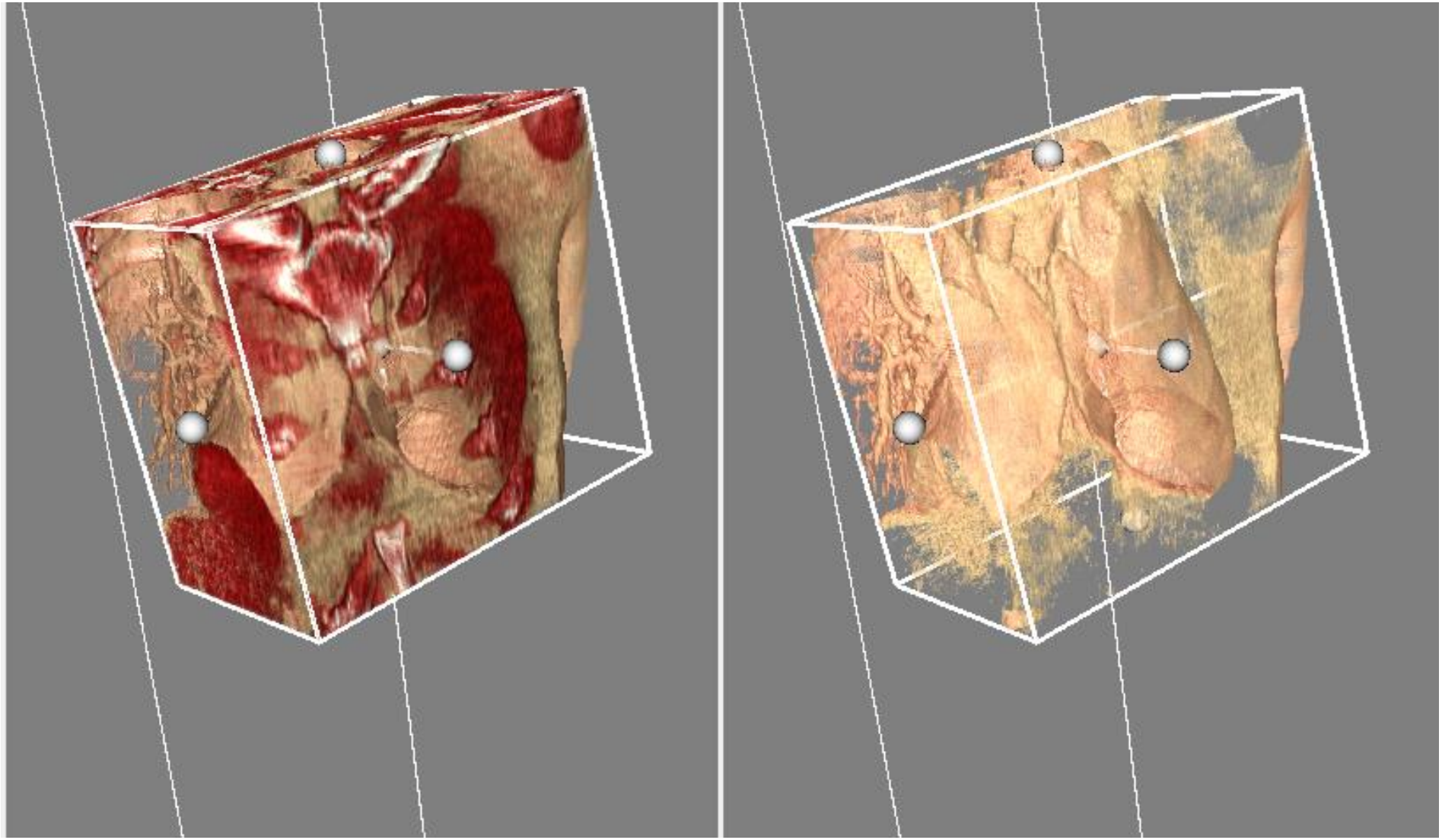


Додаток Ж (обов’язковий) Приклад динамічного зрізування всього тіла під час прямого рендерингу

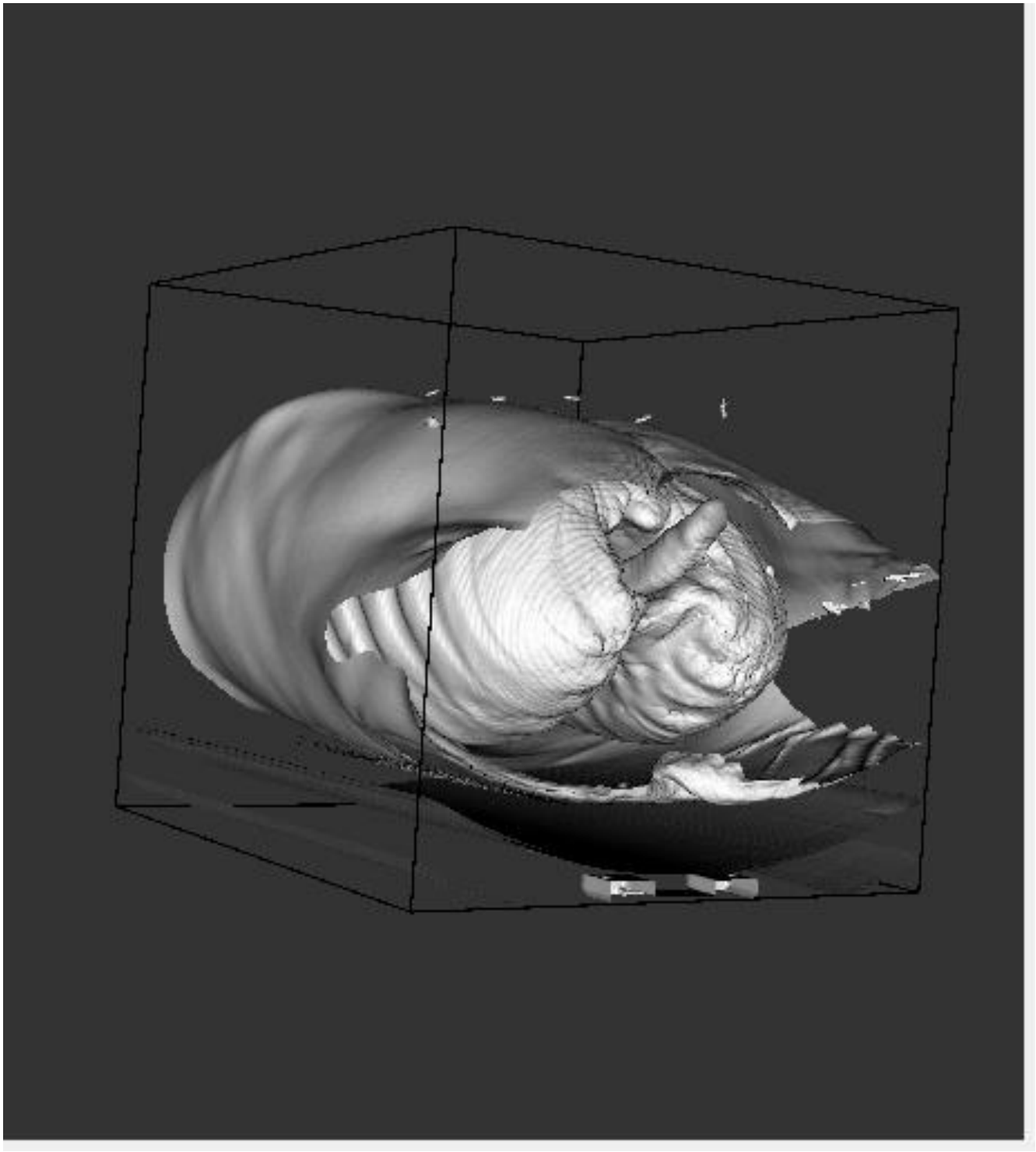
Додаток К (обов’язковий) Приклад роботи алгоритму «ріст регіонів» для сегментації легенів



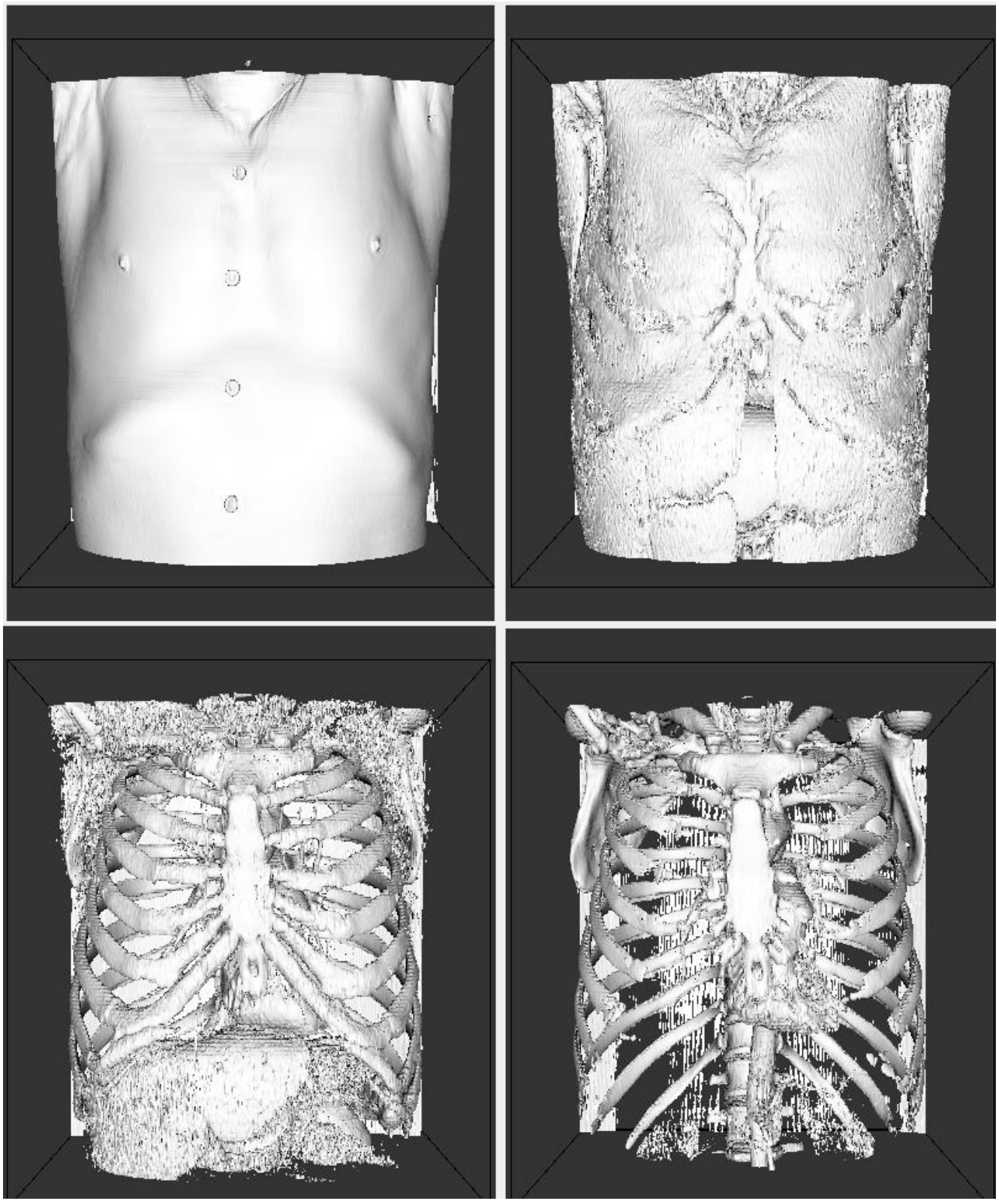
Додаток Л (обов’язковий) Приклад динамічного зрізування під час прямого рендерингу для виділення певної області тіла людини



Додаток М (обов’язковий) Приклад згенерованої ізоповерхні грудного відділу людини

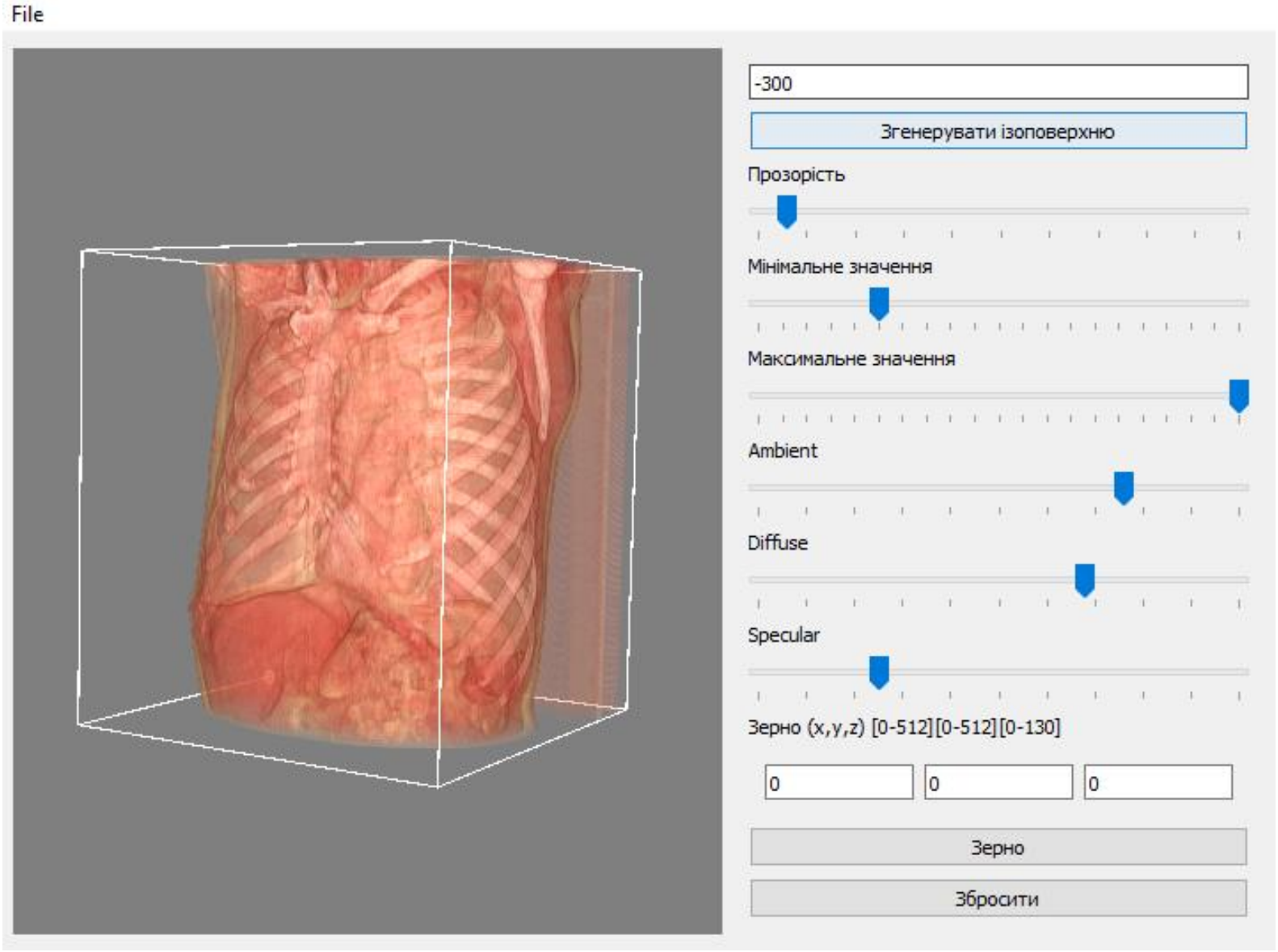


Додаток Н (обов’язковий) Приклад згенерованої ізоповерхні грудного відділу людини при різному пороговому ізозначені



−300	30
110	240

Додаток П (обов’язковий) Графічний інтерфейс і приклад роботи створеного програмного продукту



Додаток Р (обов’язковий) Графічний інтерфейс і приклад роботи створеного програмного продукту під iOS з технологією доповненої реальності