Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Прикладних інформаційних технологій та електроінженерії
(повна назва факультету)
Біотехнічних систем
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

магістр
(назва освітнього ступеня)

на тему: Метод аналізу медичних зображень у комп'ютерній томографії

Виконав(ла): студент(ка) 6 курсу, групи РБмз-61
спеціальності 163 Біомедична інженерія
(шифр і назва спеціальності)

		Патей Я.В.
	(підпис)	(прізвище та ініціали)
Керівник		Яворська Є.Б.
	(підпис)	(прізвище та ініціали)
Нормоконтроль		Дедів Л.Є.
	(підпис)	(прізвище та ініціали)
Завідувач кафедри		Яворська Є.Б.
	(підпис)	(прізвище та ініціали)
Рецензент		Дедів І.Ю.
	(підпис)	(прізвище та ініціали)

Тернопіль
2023

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Прикладних інформаційних технологій та електроінженерії
(повна назва факультету)

Кафедра Біотехнічних систем
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Яворська Є.Б.
(підпис) (прізвище та ініціали)
« » 2023 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня магістр
(назва освітнього ступеня)

за спеціальністю 163 Біомедична інженерія
(шифр і назва спеціальності)

студенту Патею Ярославу Валерійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Метод аналізу медичних зображень у комп'ютерній томографії

Керівник роботи Яворська Євгенія Богданівна, к.т.н., доц.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «*20*» *листопада* 2023 року № *4/7-1063*

2. Термін подання студентом завершеної роботи

3. Вихідні дані до роботи Вимоги замовника, технічні умови, технічне завдання

4. Зміст роботи (перелік питань, які потрібно розробити)
1. Аналітична частина
2. Основна частина
3. Науково-дослідна частина
4. Охорона праці та безпека в надзвичайних ситуаціях

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці та	Зелінський І.М., доц. каф. ПВ		
безпека в надзвичайних	Клепчик В.М., ст. викл. каф. ОХ		
ситуаціях			

7. Дата видачі завдання 10.11.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Отримання завдання на виконання роботи	06.09.2023 р.	Виконано
2	Аналіз отриманого завдання	11.09.2023 р.	Виконано
3	Виконання розділу 1	26.09.2023 р.	Виконано
4	Виконання розділу 2	15.10.2023 р.	Виконано
5	Виконання розділу 3	02.11.2023 р.	Виконано
6	Виконання розділу 4	19.11.2023 р.	Виконано
7	Оформлення пояснювальної записки	24.11.2023 р.	Виконано
8	Оформлення презентаційного матеріалу	29.11.2023 р.	Виконано
9	Перевірка кваліфікаційної роботи на антиплагіат	04.12.2023 р.	Виконано
10	Попередній захист кваліфікаційної роботи	19.12.2023 р.	Виконано

Студент _____ Патей Я.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Яворська Є.Б.
(підпис) (прізвище та ініціали)

РЕФЕРАТ/ABSTRACT

Тема кваліфікаційної роботи: «Метод аналізу медичних зображень у комп’ютерній томографії» // Кваліфікаційна робота // Патей Ярослав Валерійович // Тернопільський національний технічний університет імені Івана Пулюя, факультет прикладних інформаційних технологій та електроінженерії, група РБм-61 // Тернопіль, 2023 // с. – 56, рис. – 18, табл. – , додат. – 2, бібліогр. – 29.

Ключові слова: КОМП'ЮТЕРНА ТОМОГРАФІЯ, КТ, МЕДИЧНІ ЗОБРАЖЕННЯ, МОДЕЛЮВАННЯ, ОБРОБКА, 3D-ОБ'ЄКТ

У роботі проводилася розробка алгоритму, який оптимізує процес створення моделі серця виходячи з знімків комп'ютерної томографії. Докладно вивчено анатомію серця та гіпоплазію лівих відділів серця. Наведено підходи обробки тривимірного зображення та побудову оболонки/геометрії 3D-моделі: з використанням методів програмування. Здійснено порівняння результатів. Проведено оцінку на фізичну адекватність отриманої моделі.

Keywords: COMPUTED TOMOGRAPHY, CT, MEDICAL IMAGING, MODELING, PROCESSING, 3D-OBJECT

At this work, we created the algorithm to optimize the process of creating a heart model based on computed tomography scan. The anatomy of the heart and hypoplasia of the left heart are studied in detail. Approaches to 3D image processing and 3D model shell/geometry construction are presented: using programming methods. Comparison of the results is carried out: an assessment on the physical adequacy of the resulting model is performed.

ЗМІСТ

ВСТУП

За даними ВООЗ серцеві хвороби є в світі основною причиною смерті – щорічно від ССЗ помирає більше людей, ніж від інших хвороб. За оцінками, за 2019 рік від них померло близько сімнадцяти млн осіб, що становить біля 30% від всіх летальних випадків. З них, біля семи млн. випадків настали внаслідок ІХС, а 6,2 мільйона – від інсульту. Велика пропорція захворювань реєструється у країнах нижче середнього рівнем доходу: понад 80% випадків смерті від ССЗ, які розподіляються практично рівномірно серед жінок і чоловіків. І лише 2030 року очікується, що від ССЗ, в основному від хвороб серця та інсульту, помре близько 23,6 мільйона людей. За прогнозами ці хвороби залишаться основними окремими причинами смерті. [4]

Завчасне виявлення різних проблем із серцем та подальшої якісної оцінки всіх ризиків необхідне дослідження у тому числі проведення КТ, що дозволить записати та проаналізувати внутрішню будову органів людини.

Проте аналіз КТ буває утруднений: можна прогаяти ділі, недооцінити фізичні розміри органів, немає можливості відпрацювати порядок дій перед проведенням різних операцій. Ідеальний варіант аналізу знімків, інакше кажучи, аналіз серця та судин, це безпосередньо наживо, тобто на самій людині. Однак, таке неможливо без проведення додаткових втручань у тіло людини або дуже довгої обробки знімків вручну з подальшою печаткою серця.

Метою кваліфікаційної роботи є підвищення ефективності роботи лікаря шляхом розробки алгоритму, який дозволить на підставі знімків КТ змоделювати за прийнятний час (не більше години) методами програмування серце (3D-тіло), придатне для друку на 3D принтері.

Вирішення цього завдання дозволить збільшити якість медичної діагностики та зменшити додаткові ризики перед проведенням операцій.

Для досягнення заявленої мети необхідно виконати такі завдання:

• вивчити формати зберігання цифрових зображень комп'ютерної томографії;

• Розробити алгоритм повної обробки зображень;

• Розробити алгоритм створення 3D-моделі;

• Вивчити формати файлів 3D-моделей

• Протестувати алгоритм, а результат порівняти з ідеальною моделлю та визначити якість переданої патології алгоритмом.

Вхідними даними для дослідження є знімок комп'ютерної томографії грудної клітки пацієнта з патологією серця та підтвердженим діагнозом.

РОЗДІЛ 1
АНАЛІТИЧНА ЧАСТИНА

У контексті розв'язуваного завдання під зображенням розуміється впорядкована сукупність спостережень, що чисельно вимірюються, зафіксованих в деякому обмеженому обсязі. Кожне таке спостереження описується кубом-вокселем - це елемент обсягу з відповідними координатами x, y, z. Іншими словами, кожен воксел визначається своєю інтенсивністю та розташуванням у просторі.

Подібні об'єкти інформації формуються для дослідження завдань аналізу внутрішньої будови органів людини. Одним із методів аналізу, що формує воксельне зображення, є ренгенівська комп'ютерна томографія (КТ) і використовується для дослідження внутрішніх органів людини.

КТ-знімок є серією аксіальних “зрізів” – двовимірних зображень однакового дозволу, отриманих шляхом складної комп'ютерної обробки різниць послаблення Rg^{o}-випромінювання різними за щільністю тканинами.

У сучасному комп'ютерному томографі рентгенівська трубка здійснює спіральне обертання навколо тіла пацієнта в аксіальній площині, постійно генеруючи випромінювання. Якщо точніше, трубка обертається по колу, і одночасно безперервно зміщується вперед або назад стіл з пацієнтом. У такий спосіб формуються зрізи об'єкта. А вся серія двовимірних зображень становлять один тривимірний знімок [1, 2]. На рисунку представлено візуалізацію процесу створення знімка КТ.

1.1. Формати для представлення оцифрованих медичних зображень

У медичній галузі основним форматом збереження цифрових медичних зображень є формат DICOM (DigitalImagingCommunications in Medicine) [1, 11]. Виробники медичного обладнання використовують цей формат для виведення результату проведеного дослідження. Існує велика кількість програмного

забезпечення, як комерційного, так і вільно розповсюджуваного для роботи з DICOM файлами.

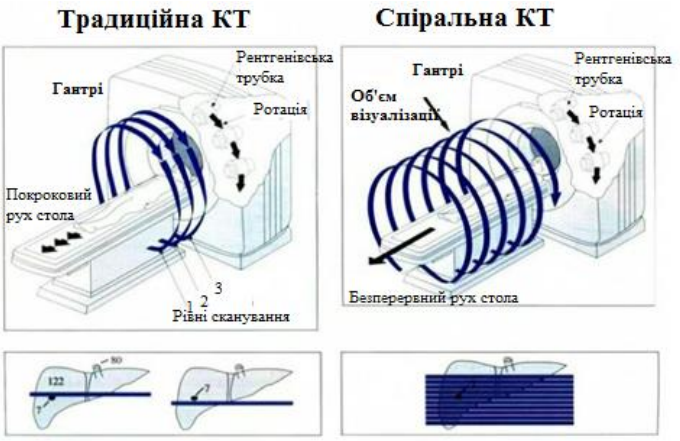

Рис. 1.1. Принцип роботи томографів

У науковій спільноті вагому популярність набув формату NITI (NeuroimagingInformaticsTechnologyInitiative) [13]. Він створювався для вирішення проблем у задачах візуалізації медичних зображень, таких як МРТ мозку.

В рамках даної роботи для вирішення задачі використовується програмний комплекс VidarDicomViwer3 [9]. Робочий простір програмного комплексу представлено рисунку.

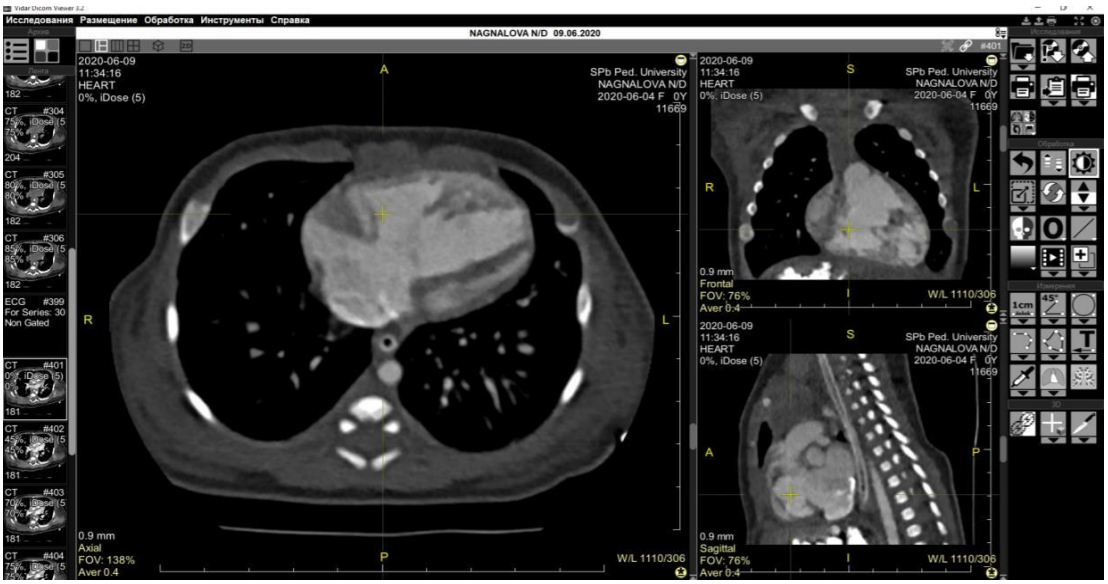

Рис. 1.2. Робочий простір VidarDicomViwer3

Програмний комплекс підтримує різні формати біомедичних зображень, зокрема їм підтримується розглянутий раніше формат DICOM. Також ця програма надає зручні інструменти візуалізації зображень у всіх проекціях.

1.2. Проблеми обробки біомедичних зображень

Перед збереженням знімка в DICOM форматі, оператор КТ-апарату самостійно вибирає область, яку потрібно дослідити, це опосередковано впливає на просторові характеристики обсягу одного вокселя, так як формоване зображення зберігається з фіксованою роздільною здатністю. На даний момент для КТ знімка, збереженого в DICOM-форматі, найчастіше зустрічається роздільна здатність 512 × 512. Але для різних пацієнтів вона відображає абсолютно різний об’єм. Також підсумкова кількість зрізів, що формуються під час дослідження, є непостійною.

У контексті алгоритмічного моделювання 3D-тіла гостро постає питання, як проводити первинну підготовку даних для однозначного визначення меж тіла. З одного боку, об'ємне зображення, отримане в результаті КТ дослідження, спочатку масштабовано таким чином, що від краю до краю зрізу розташовується область, яка нас цікавить і незалежно від віку і комплекції тіла пацієнта обробка повинна проводитися відносного того масштабу, який був заданий оператором КТ-апарату. Такий підхід змушує алгоритм працювати з фрагментами зображення без уявлення про фізичну відстань. З іншого боку, фрагмент об'ємного зображення, що обробляється алгоритмом, завжди повинен містити виміри, укладені у фіксований фізичний об'єм. Такий підхід дозволяє однозначно визначити та врахувати фізичні розміри серця.

РОЗДІЛ 2
ОСНОВНА ЧАСТИНА

2.1. Попередня обробка

Усі знімки КТ вимагають детальної та якісної обробки, оскільки необхідно досить точно передати орган, який згодом використовуватиметься лікарями для подальшого дослідження.

У попередній обробці необхідно навести деяку нормалізацію та збільшення контрастності знімка. Щоб на знімках можна було чітко побачити різні тканини організму.

Для обробки знімків КТ використовувалася бібліотека Python PyDicom [19]. Дана бібліотека дозволяє повністю прочитати файли формату .dcm і працює з інформацією, що зберігаються у знімках. Для швидкої роботи з інформацією знімків застосовувалася бібліотека NumPy [16].

2.1.1. Нормалізація

Для зручної та простої роботи зі знімками необхідно провести нормалізацію, тобто, щоб усі значення чисел знімка завжди знаходилися лише в тому самому діапазоні. Це здійснюється за рахунок того, що кожне значення буде помножено на певний коефіцієнт. Цей процес дозволить звести значення серії знімків до зручного вигляду, в даному випадку зводилося до 8 бітного числа, тобто від 0 до 255.

Значення чисел у знімку розтягуються/звужується (помножуються на певне число), а також зміщується до певної області нульового значення. Параметр коефіцієнта контрастності підбирається дослідним шляхом. Для реалізації збільшення контрастності та нормалізації знаходиться середнє та максимальне значення чисел у знімках.

На малюнку 2.1 представлено візуалізацію процесу збільшення контрастності.

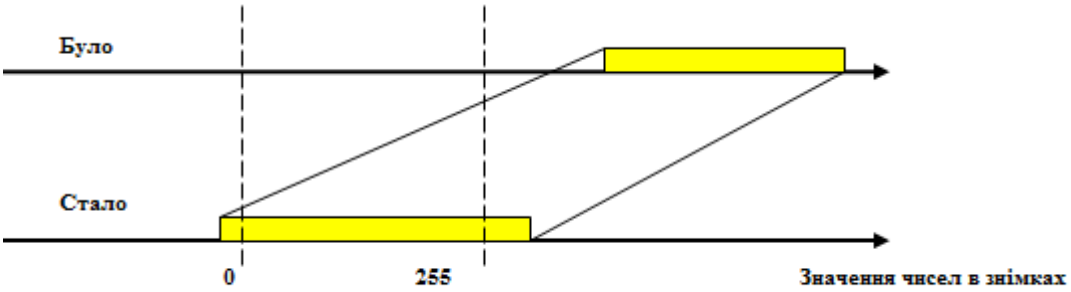

Рис. 2.1. Візуалізація процесу збільшення контрастності

Загальне значення чисел у знімках КТ в залежності від виду контрасту, який може використовуватися, варіюється в середньому від 900 до 5000, однак може відрізнятися даний діапазон від знімка до знімку.

Нижче наведена формула, за якою проводилася нормалізація та збільшення контрастності знімків.

$$A'_{i,j,k} = \frac{\sum_{i,j,k=1}^{U} A_{i,j,k}}{U} + C\left(A_{i,j,k} - \frac{\sum_{i,j,k=1}^{U} A_{i,j,k}}{U} \right)$$

$A_{i,j,k}$ – значеня вокселя з індексами;

$A'_{i,j,k}$ – результуюче занчення вокселя.

Якщо $A'_{i,j,k}$ від’ємне, то результуюче значення приймається рівним 0, якщо більше 255, то присвоюється також 255.

2.1.2. Виявлення меж

Для спрощення обробки знімків і подальше знаходження кордону об'єкта може бути зручним швидкість зміни значень або навіть швидкість зміни змін, тобто: $\Delta A_{i,j,k}$. Там де буде існувати межа тіло/тканина, може виникнути різкий перепад значень і їх набагато простіше відстежувати через слід градієнта знімка $tr\left(grad\left(A_{i,j,k}\right)\right)$: .

Нижче наведена формула, за якою можна зафіксувати швидкість змін:

$$tr\left(grad\left(A_{i,j,k}\right)\right)=\frac{A_{i+1,j,k}-A_{i-1,j,k}}{\Delta x}+\frac{A_{i,j+1,k}-A_{i,j-1,k}}{\Delta y}+\frac{A_{i,j,k+1}-A_{i,j,k-1}}{\Delta z},$$

де: $A_{i,j,k}$ – значення вокселя з відповідними індексами, Δx, Δy, Δz – віддаль між вокселями вздовж відповідних осей.

Для виявлення меж використано детектор Кенні (1986 р.). Детектор Кенні (оператор Кенні, алгоритм Кенні) – це алгоритм виявлення меж об'єктів на зображенні. Він має широке поширення і застосовується повсюдно. У бібліотеці OpenCV детектор Кенні реалізовано функції Canny.

Діяльність комбінувалися методи знаходження кордонів.

2.1.3. Діапазон та область

Для збільшення швидкості роботи програми необхідно на кожному етапі проводити оптимізацію, у тому числі за рахунок зменшення області обробки. Перед проведенням повної обробки знімків необхідно визначити в якій області 3D знімка знаходиться тіло, що цікавить, в даному випадку серце. Діапазон значень чисел необхідно визначити на цьому етапі, щоб з'явилася можливість однозначно відрізнити тканини серця від інших тканин, у тому числі кісток.

Щоб зменшити область обробки проводилося ручне кадрування знімків.

2.2. Повна обробка

Повна обробка є підготовленими знімками для подальшого створення 3D-тіла. Для цього необхідні початкові параметри обробки, отримані на попередньому етапі: область і діапазон. А також правильно підібраний алгоритм, який на підставі цих даних зробить повну обробку знімка: залишивши області, що належать серцю.

2.2.1. Виділення областей

У знімках обов'язково необхідно виділяти області, в яких присутній або відсутній матеріал, тобто тканина серця, щоб можна було надалі якісно побудувати 3D-об'єкт.

Для знаходження контурів об'єкта може допомогти бібліотека OpenCV [17] – база алгоритмів обробки зображень з відкритим кодом, яка реалізована на C/C++, розроблена для Python [7].

2.2.2. Обробка знімків для виділення внутрішньої області серця

Для знаходження областей внутрішньої стінки необхідно використовувати знімки з контрастом, завдяки контрасту видно всі стінки та межі внутрішньої оболонки серця. Оскільки контраст стосується лише внутрішньої стінки серця і не проникає всередину тканин – це дозволяє виділити внутрішню структуру серця.

Для коректного визначення діапазону значень чисел, щоб мінімізувати помилку побудови моделі, я звернувся до практикуючого серцево-судинного хірурга, він проконсультував мене, показав і розповів які ділянки знімка відносяться до серця, а які ні, а також вказав на патологічні зміни у наданих знімках. На рисунку 2.2 представлений приклад знімка.

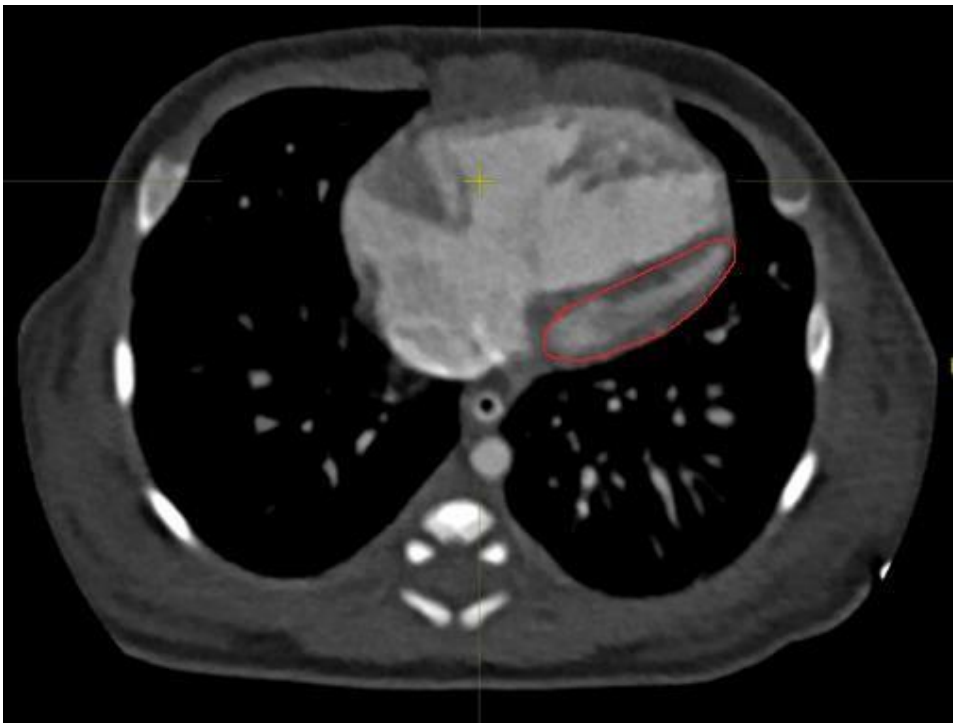

Рис. 2.2. Знімок серця (червоним обведено ділянку з патологією)

2.2.3. Обробка знімків для виділення зовнішньої стінки серця

Для зовнішньої стінки необхідно дуже точно визначити діапазон значень чисел, щоб максимальна кількість зовнішніх меж змогла виділитися на знімках. В

даному випадку краще використовувати знімки з контрастом. Оскільки навіть якщо не буде видно зовнішній кордон, то його можна буде набагато простіше відновити з таких знімків. Також при ситуації, коли у знімках можуть утворюватися порожнечі всередині цільного об'єкта, для чого необхідно додатково заповнювати матеріалом усередині об'єкта.

Для вирішення цього завдання застосовувалася бібліотека OpenCV, яка дозволяє працювати із зображеннями. Застосовувалася функція findContours [1]. Ця функція дозволяє знаходити контури у знімках, а також розділяти внутрішні та зовнішні контури, такі як контур отвору у бублика та зовнішній контур цього ж бублика. В основі роботи findContours лежить алгоритм Suzuki-Abe [10].

2.3. Marching cubes

Після обробки всіх знімків проводився запис в окрему 3D-матрицю, яка, у свою чергу, вже подавалася в окремий алгоритм, який створює 3D-модель. Для побудови 3D-об'єкта використовувався поширений алгоритм для вирішення подібних завдань, такий як Marching Cubes [17].

2.3.1.2D алгоритм

Для простоти розглянемо плаский варіант. По-перше, простір є площиною, а по-друге, простір розбивається на рівномірну сітку квадратів (осередків). Потім для кожного осередку визначаються значення чисел у вузлах цього осередку. Тим самим визначається знаходиться та чи інша вершина осередку всередині або зовні суцільної області.

На рисунку показано область, в якій функцією описано коло: чорними точками відзначені всі ті вершини, значення яких є додатніми, інші рівні або менше нуля.

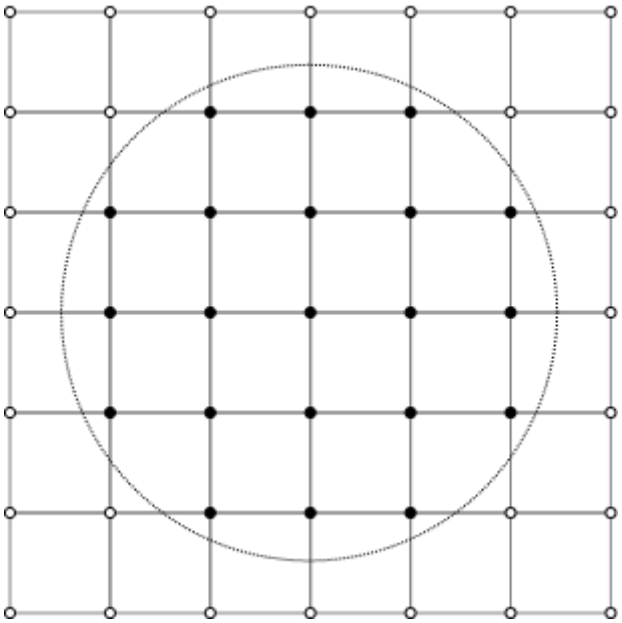

Рис. 2.3. Приклад дискретної області, яка містить коло

Потім обробляється кожна комірка окремо, заповнюючи її відповідною межею, як вказано за правилом на рисунку.

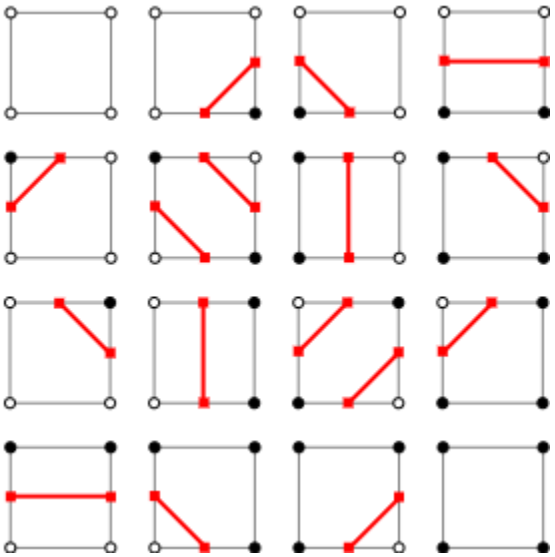

Рис. 2.4. Правило для 2D-алгоритму

Просте правило пошуку забезпечує 16 можливих комбінацій меж. У кожному випадку правило визначає, яка саме межа має бути відмальована.

Після повторення процесу для всіх осередків межі з'єднуються, створюючи готовий mesh-об'єкт.

При використанні даного правила, наведеного на рисунку 3.2 вийде наступний результат (рисунок 2.5.):

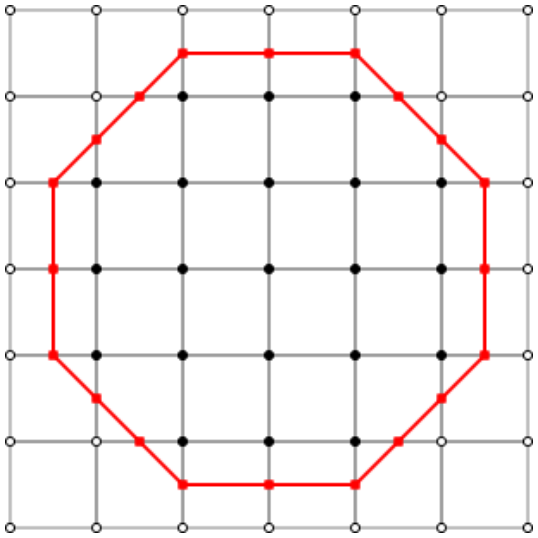

Рис. 2.5. Приклад роботи 2D-алгоритму

Отриманий результат схожий на вихідне коло в деякому наближенні, але в результаті присутні кути, яких не повинно бути. Ребра нахилені під кутами 45°. Таке відбувається через неповноцінність правила, яке не здатне задовольнити велику різноманітність меж. За правилом виходило, що точки всіх границь були розташовані на однаковій відстані від вузлів на одному ребрі осередку.

2.3.2. Адаптивність

Найпростіший спосіб позбутися кутів буде адаптивний алгоритм. Замість простого завдання всіх вершин, відповідно до правила (рисунок 3.2), вершини можна розташувати так, щоб вони найкраще відповідали суцільній ділянці. Для цього необхідно не тільки знати, чи знаходиться точка всередині чи зовні, але також, наскільки вона знаходиться далеко від межі.

Якщо описується будується за формулою, то з'являється можливість застосовувати ідеальний адаптивний режим. Проте, таке неможливо, оскільки у роботі необхідно працювати лише з КТ знімками, які надають цю інформацію.

Це означає, що потрібна деяка функція, яка дозволить дати відповідь на питання, наскільки далеко точка знаходиться від справжньої межі об'єкта, тобто

наскільки глибоко точка знаходиться всередині/зовні. Розташування точки не обов'язково має бути точним, тому що воно використовується тільки для апроксимацій.

На рисунку 2.6. представлений приклад істинної межі (суцільна лінія) та апроксимуючої межі (штрих-пунктирна лінія).

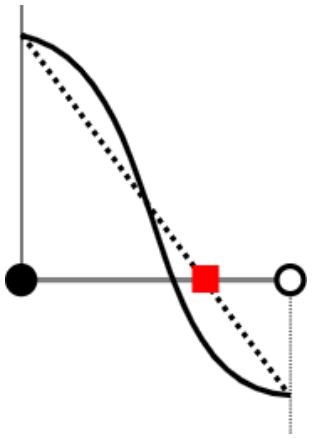

Рис. 2.6. Приклад істинної та апроксимуючої межі

2.3.3. 3D-алгоритм

У двовимірному просторі область розбивається на рівномірну сітку (приватний випадок), а потім для кожної вершини комірки визначається, де знаходиться точка – усередині або зовні суцільної області. Визначається за значенням числа: позитивне число – знаходиться всередині суцільної області, негативне або рівне нулю – виходить із межі ділянки тіла.

У 2D-алгоритмі осередок є квадрат, тобто осередок має всього 4 кута, і для кожного з них є два варіанти. Тим самим з'являється лише 16 можливих комбінацій станів кутів. У 3D-алгоритмі у комірки 8 кутів, що призводить до збільшення різних комбінацій. Усього виходить 256 аналізованих можливих варіантів і при цьому ці варіанти мають набагато складніші структури, на відміну від плоского.

Проте, всі різні варіанти не мають унікальності, тобто одні й ті самі межі можуть виявитися простим відображенням, поворотом або протилежним кордоном один одного. На рисунку представлено приклад межі, яка повертається по різних осях.

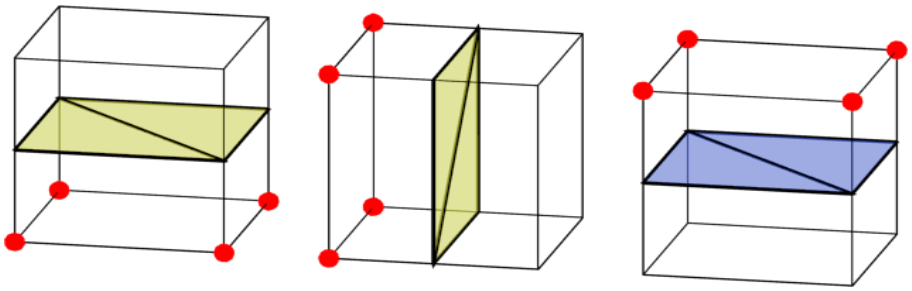

Рис. 2.7. Приклад

Червоні кути суцільні, тобто точки знаходяться всередині області. У першому випадку нижні кути суцільні, а верхні – порожні, тому для правильного відмальовування межі, що розділяє, необхідно розділити комірку вертикально. Для зручності зовнішня сторона забарвлена жовтою, а внутрішня сторона площини – синім. Інші два випадки можна знайти простим поворотом першого випадку.

Також є ще одне спрощення, на рисунку 2.8. представлений приклад межі, тільки випадки є протилежними: суцільні кути одного є порожніми для іншого, і навпаки.

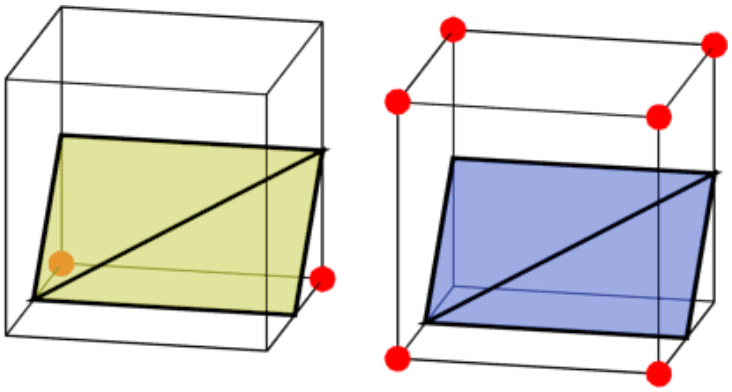

Рис. 2.8. Приклад протилежних випадків

Завдяки цим спрощенням можна отримати один випадок з іншого: оскільки кордон однаковий, достатньо лише перевернути нормаль, розгорнути кордон чи відобразити.

З урахуванням симетрії знадобиться розглянути лише 15 випадків, у тому числі генеруються й інші варіанти кордонів [15]. На рисунку 2.9. представлені всі унікальні межі плюс 3 додаткові.

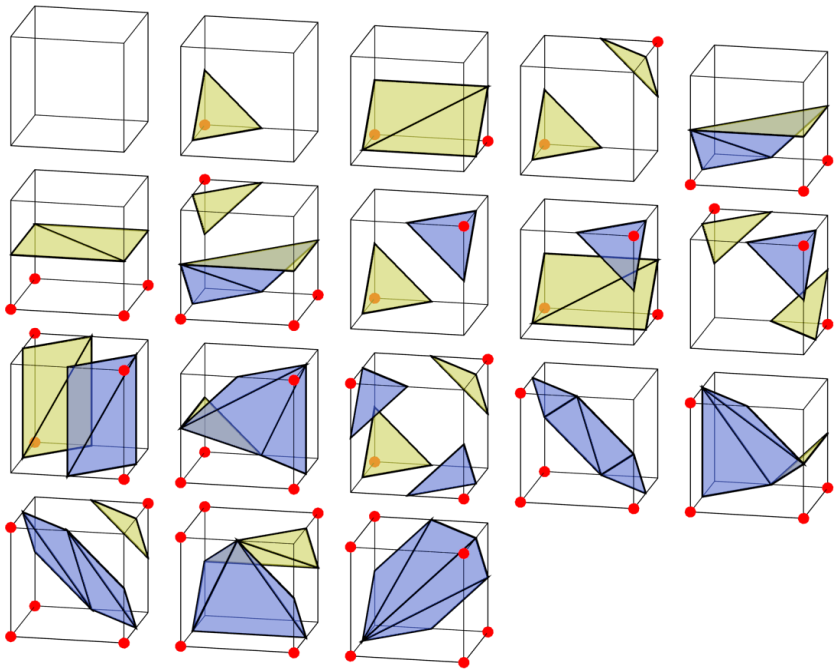

Рис. 2.9. Унікальні і три додаткових межі

На рисунку показані ті ж додаткові три варіанти в порівнянні з протилежними випадками, які дають подібну поверхню.

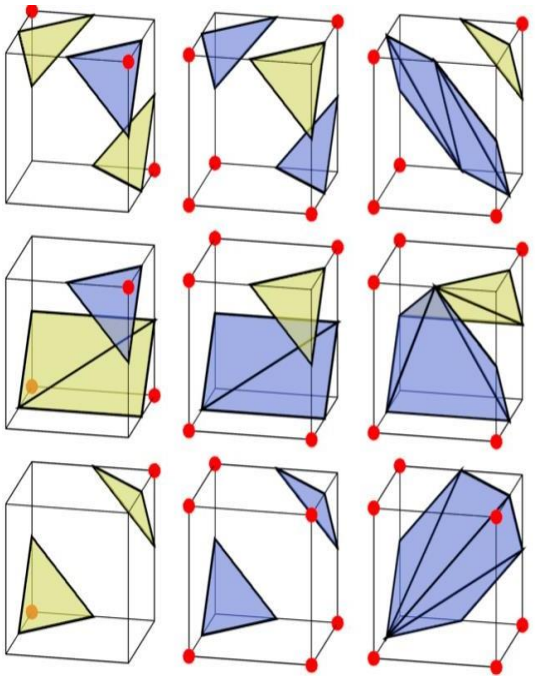

Рис. 2.10. Три додаткові межі в порівнянні

Всі стовпці правильно відокремлюють суцільну ділянку від порожньої, але це працює тільки в тому випадку, коли розглядається осередок окремо від інших і є єдиним.

Якщо розглядати ребра кожного полігону комірки, можна помітити, що вони різняться для другого і третього стовпців. Інвертовані випадки не правильно об'єднуватимуться з сусідніми осередками, залишаючи за собою отвори в поверхні. І тільки при використанні 3-х додаткових варіантів на поверхні тіла перестають утворюватися порожнечі, тим самим забезпечуючи правильне з'єднання осередків.

2.4 Запис у файл

У комп'ютерній графіці існує багато форматів файлів для запису 3D-об'єкта та безліч різних пакетів для їх зчитування та подальшої обробки.

Одні з найпопулярніших форматів для 3D графіки є: STL, OBJ, PLY, 3DS, FBX, DAE, STP, PRT та багато інших. Усі формати дозволяють передати, як мінімум, форму об'єкта, що задовольняє це завдання.

2.4.1. Формат STL (stereolithography)

За допомогою цього формату зберігають 3-мірні моделі / об'єкти. Збережувана інформація про об'єкти має вигляд списку граней та їх нормалей, у вигляді трикутників, які описують поверхню об’єкту. STL-файл є текстовим (ASCII), або двійковим [19].

Текстовий формат:

Початок файлу виду ASCII STL рядком: "solid name", де "name" – не обов’язковий рядок, хоча, якщо ім'я не написане, потрібно залишити пробіл після "solid".

Продовжується файл довільним числом трикутників, описаних наступним чином:

```
facet normal ni nj nk
outer loop
     vertex   v1x
     v1y      v1z
     vertex   v2x
     v2y      v2z
     vertex   v3x
     v3y v3z
endlo
op
endfacet
```

де кожне n і v – числа з плаваючою комою у форматі: "-2.648000e-002". Нормаль можна не записувати (записати лише "0,0,0"), але тоді обов'язково перелічити у необхідному порядку вершини, щоб утворилась права трійка, у цьому випадку нормаль поверхні автоматично обчислиться. Закінчення файлу за допомогою "endsolid name".

Двійковий формат.

Використовується у випадку, коли файл ASCII STL є великим. Початок файлу – заголовок (вісімдесят символів), який зазвичай не використовують. Оскільки із “solid” відбувається початок текстової версії, то не бажано з цього рядка його починати.

Після заголовку йде чотирьохбайтове ціле число, яким зазначається кількість граней у файлі. Потім йдуть дані, за допомогою яких маємо характеристику кожного трикутника, описаного дванадцятьма 32-бітними числами з плаваючою комою: три чидвобайтсла (нормаль) та по 3 для кожної з трьох вершин (координати XYZ). Потім йде двобайтний беззнаковий "short".

Структура файлу представлена у лістингу 2.1.

```
UINT8[80] - Header
UINT32 - Number of triangles

foreach triangle
REAL32[3] - Normal
vector REAL32[3] -
Vertex 1 REAL32[3] -
Vertex 2 REAL32[3] -
Vertex 3
UINT16 - Attribute
byte count end
```

Також, за допомогою формату можна додати інформацію щодо кольору, але в даній роботі немає необхідності його описувати.

2.4.2.. Формат OBJ

Розроблений WavefrontTechnologies. Є відкритим та використовується при розробленні 3D-графічних додатків. Експортується або імпортується у пакетах моделювання MAYA, BLENDER , 3DStudioMax та ін. Є загальноприйнятим, простим, має лише 3D-геометрію, а саме: розташування вершин, взаємопов'язаність координат текстури із вершиною, нормаль, параметри для створення полігонів [21]. Структура представлена в лістингу 2.2.

Лістинг 2.2 – структура OBJ

```
# Список вершин, з координатами (x, y, z, [w]), w є не
обов'язковим і по замовчуванню = 1.0.
 v x y
 z w v
 ...
 ...
 # Текстурні координати (u, v, [w]), w є не обов'язковими і по
 замовчуванню = 0. vt u v w
 vt ...
 ...
 # Нормалі (x, y ,z); можуть бути не нормовані.
 Vn x y
 z vn …
 …
 # Визначення поверхні
 f v1/vt1/vn1 v2/vt2/vn2
 v3/vt3/vn3 … f …
```

При описі поверхні потрібно перерахувати індекси її вершин враховуючи, що вона може мати три і більше вершини.

2.4.3.Формат PLY

Використовується для зберігання 3-мірних даних, таких як 3D сканери. Підтримує опис об'єкту як список плоских полігонів, зберігає різні властивості об'єкту, такі як колір, прозорість, нормаль, текстура тощо. Є дві версії: ASCII та бінарний файл [18]. Заголовок текстовий, починається зі слова "ply". Наступний рядок вказує, який PLY формат використано:

```
format ascii 1.0
format binary_little_endian 1.0
format binary_big_endian 1.0
```

Який елемент збережено у файлі та кількість таких елементів описано за допомогою "element". Нижче наведено запис для файлу з 12 вершинами, кожна з них записана як три числа із плаваючою комою:

```
element vertex 12
property float x property float y property float z
```

Початок рядка "property" – інформація про вершини. Розрізняють два варіанти, в залежності від джерела PLY -файлу: один з char, один з int. Опис граней:

```
element face 10
property list uchar int vertex_indices
```

Слово list вказує на вигляд в якому представлено дані.

Закінчення заголовку: "end_header".

2.4.4. Висновки за форматами

Усі представлені формати можна прочитати у відповідних пакетах з моделювання mesh-об'єктів, і тривимірну модель можна записати у будь-який із представлених форматів. Однак тільки формат obj підходив для кінцевої реалізації.

Формат STL не оптимізовано:

• Для кожного полігону необхідно прописати координати всіх 3-х вершин незалежно від типу запису

• Цей формат не підтримує n-гони, тобто полігони, що складаються більш ніж з 3-х вершин

• Необхідно вказати правильний порядок вершин або знаходити нормаль у полігоні.

У зв'язку з вищезгаданим може виникнути проблема зі швидкістю запису у файл, тобто може знадобитися досить багато часу тільки для запису.

Формат PLY простий у реалізації, оптимізований, але необхідно вказати обов'язкову кількість вершин у полігонах у файлі, також не у всіх пакетах підтримується цей формат, у зв'язку з чим можуть виникнути проблеми з подальшою постобробкою.

Формат OBJ, як і PLY, є простим, але не вимагає обов'язкового вказування кількості вершин і полігонів у файлі, він відкритий, тому читається практично у всіх пакетах з моделювання.

З вищенаведених причин як базовий формат збереження 3D-моделей було обрано OBJ.

РОЗДІЛ 3
НАУКОВО-ДОСЛІДНА ЧАСТИНА

3.1. Пакети для 3D моделювання

Для відповідної постобробки необхідно скористатися пакетами моделювання, які працюють із mesh-об'єктами та дозволять зробити постобробку.

3.1.1. 3ds Max

Autodesk3dsMax – ПЗ для 3D-моделювання [5]. Має великий набір інструментів для створення, різноманітних по формі та складності, тримірних моделей зокрема:

• моделювання полігонів, такі як EditableMesh (редагована поверхня) та EditablePoly (редагований полігон);

• моделювання на базі неоднорідних сплайнів (NURBS) та ін.

Методи можуть поєднуватись, що спрощує роботу з mesh-об'єктами. Також пакет моделювання має відкритий код на python всіх модифікаторів, які використовуються.

3.1.2. Blender

ПЗ для побудови 3D моделей, містить засоби моделювання, скульптинг, анімацію, симуляцію, рендеринг, створення 2D-анімації. [6]. Даний пакет використовує методи моделювання та інструменти, що і у 3dsMax, і багато інших.

3.2. Інструменти постобробки

3.2.1. Інструмент Edit poly

Основний інструмент, завдяки якому проводилася постопрацювання, є модифікатор edit poly [3], який дозволяє точково редагувати будь-які мінімальні елементи в моделях: точки, ребра, полігони, отвори, об'єкти.

З використанням цього модифікатора були використані такі інструменти:

• Weld – зливає найближчі вершини, що знаходяться на відстані менше, ніж встановлено.

• Cap – заповнює порожнечі на поверхні об'єкта.

• Connect – додає додаткові вузли між сусідніми ребрами, які вибрано заздалегідь.

3.2.2. Інструмент Relax

Для зменшення шуму поверхні тіла використовувався модифікатор Relax. Цей модифікатор дозволяє згладити поверхню з різною інтенсивністю, зменшуючи кількість шуму у моделях [8].

На рисунку представлено параметри даного модифікатора.

Рис. 3.1. Модифікатор Relax

3.3. Реалізація постобробки

Як основний пакет моделювання, що використовується в постобробці, був обраний 3ds Max, тому що цей пакет дозволяє досить точно і в спрощеному варіанті працювати з mesh-об'єктами. А blender був використаний як пакет, в якому проводилася візуальна оцінка отриманої моделі, оскільки цей пакет працює швидше і дозволяє добре візуалізувати точки в просторі.

На рисунку представлений інтерфейс 3ds Max’а, в сцені якого знаходиться внутрішні оболонка серця.

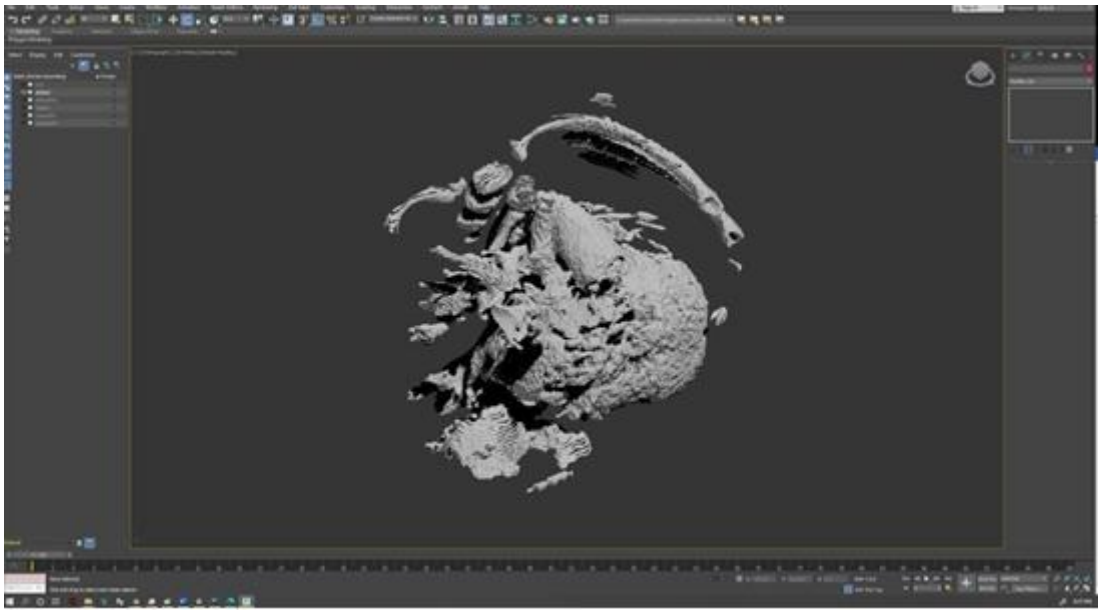

Рис. 3.2. Інтерфейс 3ds Max’а з внутрішньою оболонкою серця

Нижче представлено блок-схему алгоритму постобробки.

Рис. 3.3. Блок-схема

На рисунку показана кінцева модель серця, отримана в процесі роботи алгоритму та подальшої постобробки

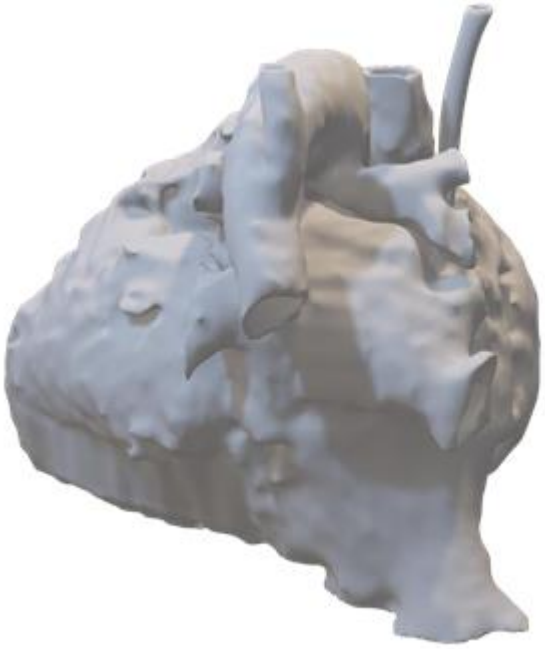

Рис. 3.4. Кінцева модель серця

3.4. Візуальна оцінка

На знімках КТ, які були надані для розробки алгоритму, зображено серце немовляти з гіпоплазією лівих відділів серця.

3.4.1 Будова серця

Серце безперечно складний і важливий інструмент у тілі людини. Тільки завдяки цьому органу надходить необхідні поживні речовини та кисень для харчування та роботи всього організму. І для розуміння як саме працює серце необхідно знати його структуру, а для моделювання це дозволить порівняти та коректно його побудувати.

Однак у людини у процесі розвитку внутрішня структура серця змінюється. Так, у ще ненародженої дитини та деякий період, після її народження зовсім інша структура серця, оскільки, перебуваючи в утробі матері, плід дихає за допомогою плаценти, а його серцю зовсім не потрібно перекачувати кров у легені (мале коло кровообігу), щоб насичувати організм киснем.

Відповідно, його легенева артерія і аорта до народження з'єднані протокою, що забезпечує надходження крові безпосередньо у велике коло кровообігу, минаючи мале, а також отвір між верхніми камерами (правим і лівим передсердями). Але незабаром, після народження, протока закривається і формуються окремо ліва легенева артерія та аорта.

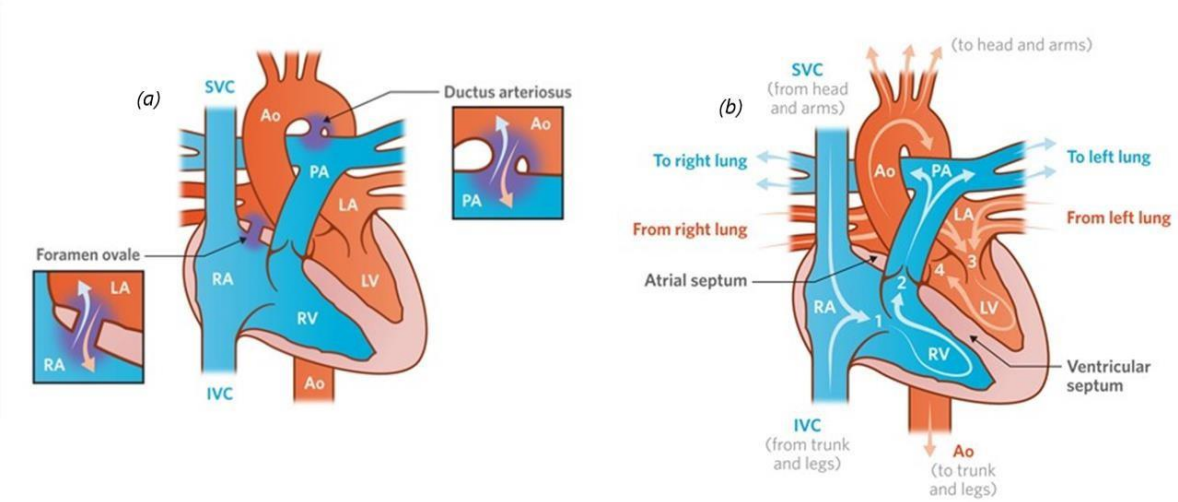

Рис. 3.5. Схематичне зображення серця немовляти (a) та дорослої людини (b)

3.4.2. Гіпоплазія

Гіпоплазія – це вада розвитку, при якій весь орган або його частина зменшена в розмірах, а функції органу знижені [12]. Гіпоплазія лівого шлуночка характеризується недорозвиненням та функціональною слабкістю лівого шлуночка. У новонароджених дітей ця аномалія розвитку є найчастішою причиною смерті.

На рисунку зображено серце з гіпоплазією лівих відділів серця (a) та здорове серце (b). В даному випадку на рисунку лівого відділу практично немає, але такий випадок не завжди проявляється.

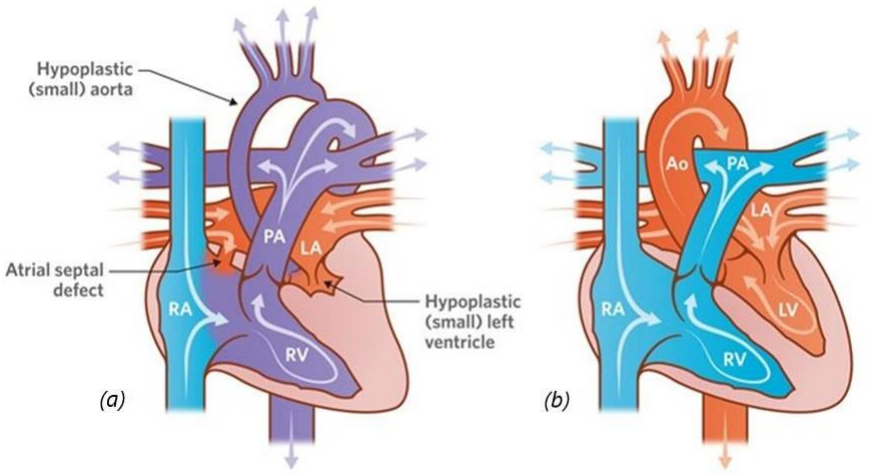

Рис. 3.6. Серце з гіпоплазією лівих відділів (a) та здорове серце (b)

На рисунку представлено серце з невеликим лівим шлуночком (точка 5), який не може постачати організм кров’ю.

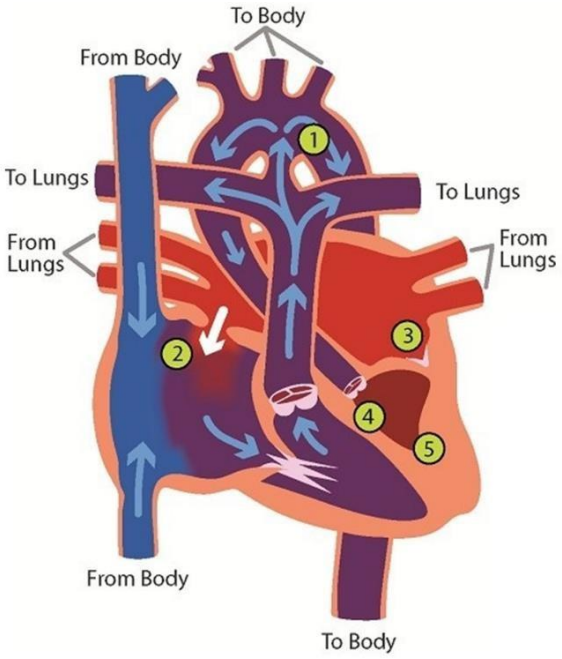

Рис. 3.7. Схематичне зображення серця з малим лівим шлуночком

3.4.3. Порівняння

На рисунках (див. після тексту) представлена модель, отримана в ході роботи, та фотографії для порівняння, на яких відображено серце, отримане вручну.

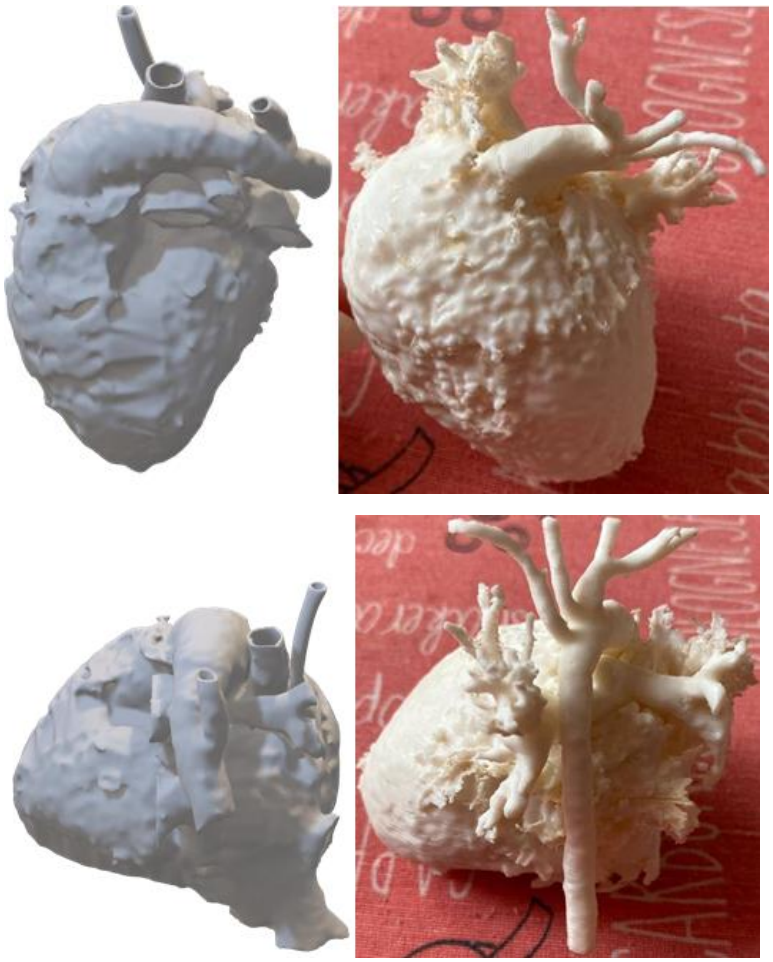

Рис. 3.8. Порівняння отриманої моделі (2) з ідеальною

На рисунку представлені зрізи серця в яких чітко видно гіпоплазію лівого відділення серця (обведено червоним область патології

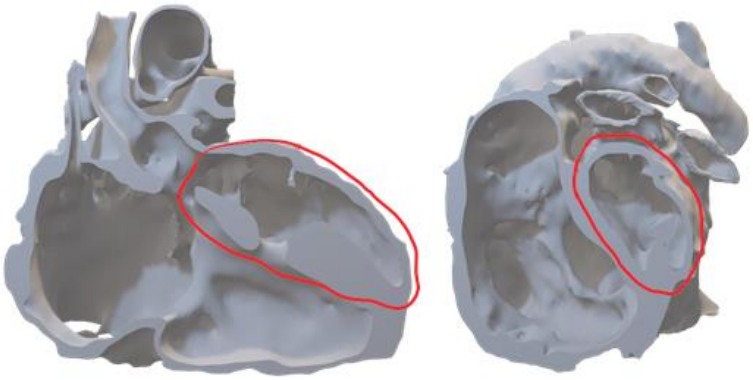

Рис. 3.9. Зрізи серця із внутрішньою структурою

Під час консультації з хірургами було встановлено, що на знімках КТ не були відображені клапани, то на зрізі видно лише місця, де вони повинні бути. Клапани

серця КТ не передає, оскільки їх стінки є дуже тонкими, а також із-за постійного руху під час процедури КТ

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

4.1. Охорона праці

Оскільки, у кваліфікаційнійроботі магістра розглядається питання, яке стосується радіаційного впливу при рентгенівському випромінюванні, то доцільно у підрозділі з охорони праці розглянути питання радіаційного захисту персоналу рентгенкабінетів.

Як відомо, робота в рентгенівських кабінетах пов'язана з шкідливими виробничими чинниками. Найнебезпечніше них рентгенівське випромінювання, тому радіаційний захист персоналу кабінетів є однією з головних умов техніки безпеки охорони здоров'я при проведенні рентгенологічних досліджень. Рентгенівські промені, як і інші види іонізуючого випромінювання, мають виражену біологічну дію. Першим ефектом при взаємодії гамма-квантів тканинами організму людини являється виникнення збудження, тобто іонізація атомів молекул наступними біохімічними реакціями, що швидко розвиваються, в соматичному генетичному напрямі. При високих разових і сумарних дозах можуть наступити і зміни в окремих органах в організмі в цілому.

Особам, які працюють в рентгенівських кабінетах, необхідно правильно оцінювати радіаційну обстановку в кабінеті і передусім знати якісні, а іноді кількісні характеристики випромінювання. В діють "Норми радіаційної безпеки України" (НРБУ-97/Д-2000) з доповненням: ДГН 6.6.1- 6.5.061-2000 Радіаційний захист від джерел потенційного опромінення (НРБУ-97/Д-2000), регламентуючі умови безпечної роботи персоналу кабінетів дозволяючі здійснювати дієвий контроль за радіаційною обстановкою в медичних установах. Відповідно до цих норм визначено три категорії осіб, які працюють з іонізуючим випромінюванням, для яких встановлені різні гранично допустимі дози випромінювання :

1. Категорія А – персонал рентгенівського кабінету, що постійно працює з рентгенівською апаратурою (лікар-рентгенолог, рентгенолаборант, санітарка).

2. Категорія Б – персонал медичної установи, який працює в приміщеннях, суміжних з рентгенівським кабінетом і не зайнятий безпосередньо роботою рентгенівською апаратурою, а також персонал, що бере іноді участь в проведенні рентгенологічних досліджень (анестезіолог, хірург та ін.), і особи, супроводжуючі хворого.

3. Категорія В – населення краю, республіки, країни.

Визначено також три групи органів, що мають різну чутливість до випромінювання:

1. Гонади, червоний кістковий мозок, а також усе тіло при його загальному .

2. М'язи, щитовидна залоза, печінка, , ШКТ та ін. органи, які не відносяться до груп 1 3.

3. Кісткова тканина, шкірний покрив, кисті, передпліччя, гомілковостопні суглоби стопи.

По фактичних рівнях індивідуальних доз, обумовлених зовнішнім внутрішнім опроміненням персонал медичної установи підрозділяють на дві групи:

- Особи, умови праці яких такі, що доза може перевищувати 0,3 річний ГДД; для осіб цієї групи обов'язковий індивідуальний дозиметричний контроль.

- Особи, умови праці яких такі, що доза не може перевищувати 0,3 річний ГДД; для осіб цієї групи індивідуальний дозиметричний контроль не обов'язковий. Потрібний тільки контроль потужності дози рентгенівського опромінення на робочому місці, за даними якого оцінюють дози опромінення персоналу.

Для забезпечення безпечних умов роботи в кабінеті мають бути прийняті заходи по захисту персоналу від дій не лише рентгенівського випромінювання, але і ін. чинників : електричного струму, пилу і пари шкідливих , шуму, що виникає при роботі апаратури і так далі.

При устаткуванні рентгенівського кабінету має бути повністю неможливлений дотик персоналу з струмоведучими частинами електричних кіл в ході проведення рентгенологічних досліджень.

Конструкція рентгенівського апарату, як правило, забезпечує захистперсоналу від доступу до струмоведучих частин. Усі високовольтні елементи забезпечені ізоляцією, захищені металевими оболонками і заземлені.

Також заземлені усі металеві доступні для дотику частини. Електричну міцність ізоляції перевіряють при випуску апаратів із заводу, а якість заземлення – при здачі рентгенівського кабінету в експлуатацію.

Заземлення рентгенівської апаратури повинне здійснюватися спеціальними провідниками. Використання як заземляючі провідники елементів металевих конструкцій будівель, сталевих труб, електропроводок, алюмінієвих оболонок кабелів і тому подібне допускається як додатковий захід. Не дозволяється використовувати як заземляючих провідників водопровідні труби, що проходять в будівлі, мережі центрального опалювання, каналізації, а також трубопроводи для паливних вибухонебезпечних сумішей.

Електричні кабелі, що сполучають елементи рентгенівського комплексу один з одним і електричною мережею мають бути прокладені в поглибленнях підлоги захищені металевими кожухами від механічних ушкоджень і хімічних дій.

В процесі навантаження рентгенівської трубки, особливо при просвічуваннях, випромінювач нагрівається інтенсивно. Допустима температура нагрівання випромінювача 85ºС. Температура усіх інших частин апарату, доступних для дотику, як правило, не повинна перевищувати 50ºС.

Концентрація свинцю і його неорганічних з'єднань на поверхні стін підлоги рентгенівських кабінетів не повинні перевищувати гранично допустимої величини 0,5 мг/см^2.

Для послаблення шкідливої дії свинцю на організм людини поверхня захисних пристроїв і пристосувань, виготовлених зі свинцю, має бути покрита подвійним шаром масляної або емалевої фарби. Захисні фартухи і козирки з просвинцьованної гуми поміщають в пластикові або церату футляри.

Під рукавички з просвинцьованної гуми слід тонкі бавовняні рукавички, щоб зменшити поверхню зіткнення шкіри рук зі свинцевмісним матеріалом рукавичок.

Після закінчення роботи із засобами індивідуального захисту з просвинцьованної гуми, працівники кабінету повинні ретельно вимити руки теплою водою з милом.

При роботі з рентгенографічними апаратами в повітрі робочих приміщень утворюються шкідливі домішки стиролу, озону, оксидів азоту, пари ацетону і

толуолу. ГДК домішок в повітрі приміщення складають: стирол – 5мг/м^3, озон і оксиди азоту – 0,1 мг/м^3, пари ацетону – 200 мг/м^3, пари толуолу – 50 мг/м^3. Для зниження концентрації шкідливих домішок в повітрі обов'язково використовують примусову вентиляцію, що забезпечує кратність повітрообміну, рівну 3. У комплект оснащення ксеролабораторії повинні входити індивідуальні протипилові респіратори по кількості працюючих.

Рівень шумових навантажень (звукового тиску) на робочих місцях персоналу не повинен перевищувати 60 дБ, в приміщеннях періодичного перебування персоналу – 70 дБ.

4.2. Безпека в надзвичайних ситуаціях

У підрозділі розглянуто питання природних та штучних джерел опромінення, їх параметри та вплив електромагнітного випромінювання на людину.

Природні та штучні джерела електромагнітних полів (ЕМП). Параметри полів і випромінювань. Діапазони електромагнітних хвиль. Інтенсивний розвиток електроніки, радіо- та комп'ютерної техніки викликав забруднення природного середовища електромагнітними випромінюваннями. Джерела електромагнітних полів (ЕМП) можуть бути природного та антропогенного характеру.

Штучними джерелами випромінювань є потужні радіотелевізійні, радіолокаційні станції, станції мобільного зв'язку, недосконалі комп'ютери, високовольтні лінії електрозв'язку, електротранспорт, електростанції й підстанції, промислові установки високочастотного нагріву, вимірювальні прилади, мікрохвильові печі, телевізори, електроплити, праски, холодильники, а також будь-які елементи, що підключені до мережі.

До *природних джерел* належать: Земля, Сонце, Космос. Електричне поле Землі має середню напруженість Е = 130 н/м . Менша напруженість у полюсів, більша - у екватора. Ці величини змінюються під впливом сонячної активності, енергії космічних випромінювань. До цих вічно існуючих полів і випромінювань адаптувалося усе живе.

Електромагнітні випромінювання антропогенного походження розглядають як один з різновидів енергетичних забруднювачів, тому що вони негативно впливають на організм людини, на інші живі організми та здійснюють шкідливий вплив на екологічні системи ЕМП мають енергію і поширюються у вигляді електромагнітних хвиль. Основними параметрами електромагнітних хвиль є довжина хвилі, частота коливань, швидкість поширення. Мірою вимірювання забруднення електромагнітними полями є напруженість (В/м).

Частота коливань визначається в герцах (Гц) . Класифікація електромагнітних випромінювань за частотою:

- низькочастотні випромінювання (НЧ): 0,003 Гц-30 кГц;
- радіохвилі високочастотного (ВЧ) діапазону: 30 кГц-300 МГц;
- радіохвилі ультрависокочастотного діапазону (УВЧ): 30300 МГц;
- надвисокочастотні СВЧ: 300 МГц-300 ГГц

Чинники, від яких залежать наслідки дії ЕМП на біологічні об'єкти. Наслідки впливу ЕМП на людину. Заходи захисту від ЕМП. Рівень інтенсивності ЕМП в зв'язку з зростанням кількості їх джерел та потужності наразі різко виріс. В деяких районах він в сотні раз перевищує значення середнього натурального "природного фону". Електромагнітні поля негативно впливають на людей, які безпосередньо працюють із джерелами випромінювань, а також на населення, яке проживає поблизу джерел випромінювання. Ступінь впливу електромагнітних випромінювань на організм людини залежить від діапазону частот, інтенсивності впливу відповідних чинників, тривалості опромінення, характеру випромінювання, режиму опромінення, розмірів поверхні тіла, яка опромінюється та індивідуальних особливостей організму.

Рівень електромагнітних випромінювань у районах, де розташовані потужні радіопередавальні та локаційні станції, часто перевищує допустимі санітарні норми, що дуже шкодить здоров'ю людей, які мешкають поруч таких станцій. Вплив ЕМП характеризується *біологічною дією*. Вони завдають шкоди нервовій системі, спричинюють головний біль і сильну втому, зумовлюють розвиток неврозів, безсоння, зниження точності робочих рухів, млявість, порушення в системах і органах (шлунку, печінки, селезінки, підшлункової залоз),

функціональні зсуви в діяльності нервово-психічної, серцево-судинної, ендокринної, кровотворної систем, фіксуються зміни показників білкового та вуглеводного обміну, змінюється склад крові, зафіксовані порушення на клітинному рівні Вплив ЕМП на біологічні об'єкти залежить від інтенсивності опромінення .

Теплова дія характеризується загальним підвищенням температури тіла, подібним до пропасного стану або локалізованого нагріву тканини. Впливаючи на живу тканину організму, ЕМП викликає змінну поляризацію молекул і атомів, які складають клітини, внаслідок чого відбувається небезпечний нагрів. Надмірне тепло може нанести шкоду окремим органам і всьому організму людини. Особливо шкідливий перегрів таких органів, як очі, мозок, нирки тощо. З ростом інтенсивності проявляється вплив на нервову систему, умовно-рефлекторну діяльність, клітини печінки, підвищення тиску, викликає зміни у корі головного мозку, втрату зору.

Для запобігання професійних захворювань, які виникають під впливом ЕМП, розроблені на основі медикобіологічних досліджень санітарні норми та правила щодо радіотехнічних і електротехнічних об'єктів . Вони регламентують також умови експлуатації з метою охорони населення від шкідливого впливу випромінювань.

Для захисту людини від дії електромагнітних опромінювань застосовуються різні засоби і заходи захисту: захист часом, відстанню, екранування джерел випромінювання, зменшення випромінювання безпосередньо в самому джерелі випромінювання, встановлення санітарних кордонів навколо джерела ЕМП, екранування робочих місць, виділення зон випромінювання, дистанційний контроль і керування в екранованому приміщенні, медичні огляди, додаткова відпустка, скорочені робочі дні, застосування засобів індивідуального захисту.

4.3. Висновки до розділу 5

Даний розділ роботи висвітлює питання радіаційного захисту персоналу рентгенкабінетів та проведено аналіз природних та штучних джерел опромінення, їх параметри та вплив електромагнітного випромінювання на людину.

ЗАГАЛЬНІ ВИСНОВКИ

З використанням інструментів програмування було розроблено метод первинної підготовки біомедичних зображень, що дозволяє забезпечити однакове представлення вхідних даних, що підлягають алгоритмічній обробці. При цьому збільшується контрастність знімків, для більш точного та якісного відокремлення серця від інших органів.

Реалізований алгоритм обробки об'ємних зображень дозволять прискорити процес обробки знімків та подальшу побудову 3D-моделі серця, придатної для друку на 3D-принтері. При цьому даний алгоритм передає внутрішню структуру серця, а також його патологію. Це дозволяє лікарям якісно оцінити проблеми, оцінити фізичні розміри органу та завчасно розробляти порядок дій перед проведенням оперативних втручань.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Яворський Б.І. Методи та засоби комп'ютерної реконструктивної томографії: навчальний посібник / Т.М. Рафа, Б.І. Яворський. –Тернопіль: Крок, 2010. – 107 с.

2. Rea Paul (2022). Medical visualization and applications of technology. Cham: Springer. ISBN 978-3-031-06735-8.

3. Ooijen Peter M. A. van (2021). Basic knowledge of medical imaging informatics: undergraduate Level and Level I. Cham: Springer. ISBN 978-3-030-71885-5

4. Rea Paul (2020). Biomedical visualisation. Volume 7. Cham: Springer. ISBN 978-3-030-43961-3

5. Zhou S. Kevin; Rueckert Daniel; Fichtinger Gabor (2020). Handbook of medical image computing and computer assisted intervention. London, United Kingdom. ISBN 0-12-816176-0

6. Medical Image Computing and Computer Assisted Intervention — MICCAI 2019 (англ.). doi:10.1007/978-3-030-32239-7

7. Badoer Emilio, ред. (2012). Visualization Techniques: From Immunohistochemistry to Magnetic Resonance Imaging. Neuromethods. Totowa, NJ: Humana Press. ISBN 978-1-61779-896-2

8. Dhawan Atam P. (2011). Medical image analysis (2nd edition). Piscataway, NJ. ISBN 978-0-470-91854-8

9. Серія книг Evidence-Based Imaging.

10. Di Liu,Y Tao та ін. (2023-03). Rationalized deep learning super-resolution microscopy for sustained live imaging of rapid subcellular proces Nature Biotechnology (англ.) 41 (3). С.367–377. ISSN 1546-1696. DOI:10.1038/s41587-022-01471-3.Autodesk

11. 3ds max, Wikipedia, URL: https://wikipedia.org/wiki/Autodesk_3ds_Max

12. Blender [Електронний ресурс], Wikipedia, URL: https://wikipedia.org/wiki/Blender.

13. OpenCV [Електронний ресурс], Wikipedia, URL: https://wikipedia.org/wiki/OpenCV#cite_note-Learning_OpenCV-2.

14. Relax модифікатор [Електронний ресурс], 3DLANCER, URL: https://3dlancer.net/ru/lessons/3d-max/relax-modifikator-54.

15. Vidar Dicom Viewer [Електронний ресурс], ПЗ ВІДАР, URL: https://povidar/dicom-viewer/v3/.

16. Suzuki S, Abe K. – Topological Structural Analysis of Digitized Binary Images by Border Following, учебник, CVGIP. 1985.

17. International standard to transmit, store, retrieve, print, process, and display medical imaging information [Електронний ресурс], Digital Imaging and Communications in Medicine, URL: https://www.dicomstandard.org.

18. Hypoplastic Left Heart Syndrome HD [Електронний ресурс], The Royal Children's Hospital Melbourne, URL: https://www.rch.org.au/cardiology/heart_defects/Hypoplastic_Left_Heart_Syndrome

19. NIfTI-1 format description [Електронний ресурс], Neuroimaging Informatics Technology Initiative, URL: https://nifti.nimh.nih.gov.

20. Marching Cubes [Електронний ресурс], Wikipedia, URL: https://en.wikipedia.org/wiki/Marching_cubes.

21. The Marching Cubes [Електронний ресурс], The Marching Cubes, URL: http://users.polytech.unice.fr/~lingrand/MarchingCubes/algo.html,.

22. NumPy [Електронний ресурс], NumPy, URL: https://numpy.org/.

23. OpenCv [Електронний ресурс], OpenCV, URL: https://opencv.org/.

24. PLY (file format) [Електронний ресурс], Wikipedia, URL: https://en.wikipedia.org/wiki/PLY_(file_format).

25. PyDicom [Електронний ресурс], PYDICOM, URL: http://pydicom.org/.

26. STL (file format) [Електронний ресурс], Wikipedia, URL: https://en.wikipedia.org/wiki/STL_(file_format).

27. Wavefront .obj file [Електронний ресурс], Wikipedia, URL: https://en.wikipedia.org/wiki/Wavefront_.obj_file.

28. Стручок В.С. Безпека в надзвичайних ситуаціях. Методичний посібник для здобувачів освітнього ступеня «магістр» всіх спеціальностей денної бо та заочної (дистанційної) форм навчання / В.С.Стручок. — Тернопіль: ФОП Паляниця В. А., 2022. — 156 с.

29. Методичні рекомендації до виконання, оформлення та захисту кваліфікаційних робіт для здобувачів другого (магістерського) рівня вищої освіти за спеціальністю 163 «Біомедична інженерія» галузі знань 16 «Хімічна інженерія та біоінженерія» / уклад.: Хвостівський М.О., Яворська Є.Б. Тернопіль: ТНТУ, 2023. 57 с.

ДОДАТКИ

ДОДАТОК А
Апробація результатів дослідження

УДК 681.326
Богатирчук І.П., Дичик І.О., Патей Я.В., Яблонський Д.С.
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ПОРІВНЯЛЬНИЙ АНАЛІЗ МЕТОДІВ ТА ЗАСОБІВ ПРЕДСТАВЛЕННЯ МЕДИЧНИХ ДАНИХ

I. Bohatyrchuk, I. Dychyk, Patey Ya., D. Yablonskiy,

COMPARATIVE ANALYSIS OF METHODS AND MEANS OF MEDICAL DATA PRESENTATION

При проведенні медико-біологічних досліджень часто постає задача зберігання великих об'ємів медичних даних. Відповідно, і методи зберігання та ефективної обробки таких даних постійно розвиваються та удосконалюються. Тому, на перших порах важко вибрати оптимальний варіант форми зберігання даних.

У роботі представлено порівняльну характеристику найбільш уживаних методів. Найпростіший метод зберігання та представлення даних - зберігання в текстовому файлі. При цьому дані, що стосуються одного об'єкту, зберігаються в одній стрічці і закінчуються символом переводу стрічки. Різні дані в одній стрічці (тобто, наприклад, прізвище, ім'я, по батькові та електронна адреса) розділюються комами або символами табуляції (\t). Подальшим розвитком методів зберігання даних є технологія електронних таблиць (наприклад, Microsoft Excel). Як виявилось, при цьому дані редагувати зручніше, адже вже готова таблична структура. Для великих об'ємів даних (більше 1000 записів) та даних, які складно або неефективно представляти у вигляді двовимірної таблиці, створені бази даних (БД) та системи управління БД (СУБД).

База даних - це об'єднання таблиць, що стосуються однієї теми (наприклад, база даних пацієнтів медичного закладу). Для ефективної обробки БД існує стандарт мови структурованих запитів SQL (Structured Query Language). Окрім того, більшість СУБД побудовані за технологією "клієнт-сервер", що дозволяє розділяти сервер БД і сервер обробки даних на різні комп'ютери в мережі. Новим напрямком у технологіях представлення даних є побудована на стандарті SGML (Standard Generalized Markup Language — Стандартна Узагальнена Мова Розмітки) та орієнтована на Web розширена мова розмітки XML (eXtended Markup Language).

Використовуючи HTML-подібні теги можна виділяти певні частини тексту як такі, що відносяться до певної теми. Основна перевага XML - це можливість задання власних тегів, також дані представляються у деревовидній формі. XML-документ можна використовувати для створення баз знань.

В сервер MS SQL версії 2000 і пізніше вбудована можливість видачі даних в форматі XML. Результати аналізу показують, що в задачах зберігання великих об'ємів даних доцільно використовувати СУБД з використанням технології „клієнт-сервер", для задач зберігання даних складної структури – мову XML, а для зберігання даних невеликого об'єму доцільно обмежитись електронними таблицями чи текстовими файлами.

ДОДАТОК Б

Програмний код

Файл settings.py

```
ADAPTIVE = True
XMIN = 104
XMAX = 294
YMIN = 126
YMAX = 403
ZMIN = 50
ZMAX = 170
Dc = 10
DWN_LIM_1 = 171
UP_LIM_1 = 255
PATH_IN = 'C:/VKR/pictures/S4010'
PATH_OUT = 'C:/VKR/Script_1/V4/Output_files'
```

Файл common.py

```
import settings
def adapt(v0, v1):
    assert (v1 > 0) != (v0 > 0),
    if settings.ADAPTIVE:
        return (0 - v0) / (v1 - v0)
    else:
        return 0.5
```

Файл utils.py

```
import math
class V3:
    def __init__(self, x, y, z):
        self.x = x
        self.y = y
        self.z = z
    def normalize(self):
        d = math.sqrt(self.x*self.x+self.y*self.y+self.z*self.z)
        return V3(self.x / d, self.y / d, self.z / d)
class Tri:
    def __init__(self, v1, v2, v3):
        self.v1 = v1
        self.v2 = v2
        self.v3 = v3
    def map(self, f):
        return Tri(f(self.v1), f(self.v2), f(self.v3))
class Mesh:
```

```
    def __init__(self, verts=None, faces=None):
        self.verts = verts or []
        self.faces = faces or []
    def extend(self, other):
        l = len(self.verts)
        f = lambda v: v + l
        self.verts.extend(other.verts)
        self.faces.extend(face.map(f) for face in other.faces)
    def __add__(self, other):
        r = Mesh()
        r.extend(self)
        r.extend(other)
        return r
    def translate(self, offset):
        new_verts = [V3(v.x + offset.x, v.y + offset.y, v.z +
offset.z) for v in self.verts]
        return Mesh(new_verts, self.faces)
def make_obj(f, mesh):
    for v in mesh.verts:
        f.write("v {} {} {}\n".format(v.x, v.y, v.z))
    for face in mesh.faces:
        if isinstance(face, Tri):
            f.write("f {} {} {}\n".format(face.v1, face.v2,
face.v3))
```

Файл Test.py

```
import pydicom as PDCM
import cv2 as cv
import os
import os.path
import numpy as np
from settings import PATH_IN, PATH_OUT, DWN_LIM_1, UP_LIM_1, Dc,
XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX
from marching_cubes import make_circle_obj
import matplotlib.pyplot as plt
import time
import sys
def cleaner(BD, rows, cols, h, RW):
    New_Img = np.zeros((rows, cols, h), np.uint8)
    for i in range(XMIN, XMAX):
        for j in range(YMIN, YMAX):
            for k in range(ZMIN, ZMAX):
                if (BD[i, j, k] < DWN_LIM_1) or (BD[i, j, k] >
UP_LIM_1):
                    New_Img[i, j, k] = 0
                else:
                    New_Img[i, j, k] = BD[i, j, k]
    def border(i, j, k, xy, BB):
        S = 0
        for ii in range(i, i + xy):
            for jj in range(j, j + xy):
```

```
                if BB[ii, jj, k] == 0:
                    S += 1
                if BB[ii, jj, k + xy] == 0:
                    S += 1
        for ii in range(i, i + xy):
            for kk in range(k+1, k + xy-1):
                if BB[ii, jj, kk] == 0:
                    S += 1
                if BB[ii, jj + xy, kk] == 0:
                    S += 1
        for kk in range(k+1, k + xy-1):
            for jj in range(j+1, j + xy-1):
                if BB[ii, jj, kk] == 0:
                    S += 1
                if BB[ii + xy, jj, kk] == 0:
                    S += 1
        return S
    if RW:
        for kk in range(ZMIN, ZMAX):
            contours, hierarchy = cv.findContours(New_Img[:, :,
kk], cv.RETR_EXTERNAL, cv.CHAIN_APPROX_NONE)
            for i in range(1, len(contours)):
                c = contours[i]
                cv.drawContours(New_Img[:, :, kk], [c], -1, (50),
-1)
    else:
        DC = 2 * (3*Dc*Dc - 6*Dc + 4)
        for i in range(XMIN - Dc + 1, XMAX):
            for j in range(YMIN - Dc + 1, YMAX):
                for k in range(ZMIN - Dc + 1, ZMAX):
                    if (border(i, j, k, Dc, New_Img) == DC):
                        for ii in range(i + 1, i + Dc - 1):
                            for jj in range(j + 1, j + Dc - 1):
                                for kk in range(k + 1, k + Dc -
1):
                                    New_Img[ii, jj, kk] = 0
        return New_Img
def Dicom_to_Image(Path, rows, cols, CL):
    DCM_Img = PDCM.read_file(Path)
    Instance_Number = int(DCM_Img.get(0x00200013).value
    y1 = ''.join(map(str, DCM_Img.get(0x00281050).value))
    y2 = ''.join(map(str, DCM_Img.get(0x00281051).value))
    if (DCM_Img.get(0x00281052) is None):
        Rescale_Intercept = 0
    else:
        Rescale_Intercept = int(DCM_Img.get(0x00281052).value)
    if (DCM_Img.get(0x00281053) is None):
        Rescale_Slope = 1
    else:
        Rescale_Slope = int(DCM_Img.get(0x00281053).value)
    NM = np.zeros((rows, cols), np.uint8)
    Pixels = DCM_Img.pixel_array
    if CL:
```

```
        Xmin = XMIN
        Xmax = XMAX
        Ymin = YMIN
        Ymax = YMAX
    else:
        Xmin = 1
        Xmax = rows
        Ymin = 1
        Ymax = cols
    coefficient = 3.5
    avg = 0
    NN = np.zeros(Pixels.shape)
    for i in range(Pixels.shape[0]):
        for j in range(Pixels.shape[1]):
            NN[i, j] = int(Pixels[i, j] * 255 / 4500)
    for i in range(Pixels.shape[0]):
        for j in range(Pixels.shape[1]):
            r = NN[i, j]
            avg += r
    avg /= Pixels.shape[0] * Pixels.shape[1]
    palette = []
    for i in range(256):
        temp = int(avg + coefficient * (i - avg))
        if temp < 0:
            temp = 0
        elif temp > 255:
            temp = 255
        palette.append(temp)
    for i in range(Xmin, Xmax):
        for j in range(Ymin, Ymax):
            r = int(NN[i, j])
            NM[i, j] = palette[r]
        return NM, Instance_Number
def main(Input_Folder, Output_Folder, CL, RW):
    Input_Image_List = os.listdir(Input_Folder)
    if os.path.isdir(Output_Folder) is False:
        os.mkdir(Output_Folder)
    DCM_Img = PDCM.read_file(Input_Folder + '/I10')
    rows = DCM_Img.get(0x00280010).value
    cols = DCM_Img.get(0x00280011).value
    Fll = np.zeros((rows, cols, len(Input_Image_List)))
    Fll_g = np.zeros((rows, cols, len(Input_Image_List)))
    for i in range(1, len(Input_Image_List)):
        Output_Image, Instance_Number =
Dicom_to_Image(Input_Folder + '/' + Input_Image_List[i], rows,
cols, CL)
        Fll_g[:, :, Instance_Number] = Output_Image
    if CL:
        Fll = cleaner(Fll_g, rows, cols, len(Input_Image_List),
RW)
    else:
        Fll = Fll_g
    for i in range(1, len(Input_Image_List)):
```

```
        cv.imwrite(Output_Folder + '/XX/' + str(i).zfill(4) +
'.jpg', Fll[:, :, i])
    for j in range(1, rows - 2):
        cv.imwrite(Output_Folder + '/YY/' + str(j).zfill(4) +
'.jpg', Fll[j, :, :])
    for k in range(1, cols - 2):
        cv.imwrite(Output_Folder + '/ZZ/' + str(k).zfill(4) +
'.jpg', Fll[:, k, :])
    return Fll
if __name__== "__main__":
    Input_Folder = PATH_IN
    Output_Folder = PATH_OUT
    t_b1 = time.time()
    DCM_Img = PDCM.read_file(Input_Folder + '/I10')
    rows = DCM_Img.get(0x00280010).value
    k_z = DCM_Img.get(0x00180088).value
    k_xy = DCM_Img.get(0x00181100).value / (rows - 1)
    FLL_M = main(Input_Folder, Output_Folder, True, False)
    t_e1 = time.time()
    t_b2 = time.time()
    make_circle_obj("output.obj", FLL_M, k_z, k_xy)
    t_e2 = time.time()
    print("Время на создание 3D модели = ", (t_e2 - t_b2) / 60)
    print("Общее потраченное время = ", (t_e2 - t_b1) / 60)
```

Файл marching_cubes.py

```
import pydicom as dicom
import os
import time
import numpy as np
import sys
from common import adapt
from settings import XMIN, XMAX, YMIN, YMAX, ZMIN, ZMAX,
DWN_LIM_1, UP_LIM_1
import math
from utils import V3, Tri, Mesh, make_obj
VERTICES = [
    (0,   0,   0),
    (1,   0,   0),
    (1,   1,   0),
    (0,   1,   0),
    (0,   0,   1),
    (1,   0,   1),
    (1,   1,   1),
    (0,   1,   1),
]
cases = [[],
 [[8, 0, 3]],
 [[1, 0, 9]],
 [[8, 1, 3], [8, 9, 1]],
 [[10, 2, 1]],
```

[[8, 0, 3], [1, 10, 2]],
[[9, 2, 0], [9, 10, 2]],
[[3, 8, 2], [2, 8, 10], [10, 8, 9]],
[[3, 2, 11]],
[[0, 2, 8], [2, 11, 8]],
[[1, 0, 9], [2, 11, 3]],
[[2, 9, 1], [11, 9, 2], [8, 9, 11]],
[[3, 10, 11], [3, 1, 10]],
[[1, 10, 0], [0, 10, 8], [8, 10, 11]],
[[0, 11, 3], [9, 11, 0], [10, 11, 9]],
[[8, 9, 11], [11, 9, 10]],
[[7, 4, 8]],
[[3, 7, 0], [7, 4, 0]],
[[7, 4, 8], [9, 1, 0]],
[[9, 1, 4], [4, 1, 7], [7, 1, 3]],
[[7, 4, 8], [2, 1, 10]],
[[4, 3, 7], [4, 0, 3], [2, 1, 10]],
[[2, 0, 10], [0, 9, 10], [7, 4, 8]],
[[9, 10, 4], [4, 10, 3], [3, 10, 2], [4, 3, 7]],
[[4, 8, 7], [3, 2, 11]],
[[7, 4, 11], [11, 4, 2], [2, 4, 0]],
[[1, 0, 9], [2, 11, 3], [8, 7, 4]],
[[2, 11, 1], [1, 11, 9], [9, 11, 7], [9, 7, 4]],
[[10, 11, 1], [11, 3, 1], [4, 8, 7]],
[[4, 0, 7], [7, 0, 10], [0, 1, 10], [7, 10, 11]],
[[7, 4, 8], [0, 11, 3], [9, 11, 0], [10, 11, 9]],
[[4, 11, 7], [9, 11, 4], [10, 11, 9]],
[[9, 4, 5]],
[[9, 4, 5], [0, 3, 8]],
[[0, 5, 1], [0, 4, 5]],
[[4, 3, 8], [5, 3, 4], [1, 3, 5]],
[[5, 9, 4], [10, 2, 1]],
[[8, 0, 3], [1, 10, 2], [4, 5, 9]],
[[10, 4, 5], [2, 4, 10], [0, 4, 2]],
[[3, 10, 2], [8, 10, 3], [5, 10, 8], [4, 5, 8]],
[[9, 4, 5], [11, 3, 2]],
[[11, 0, 2], [11, 8, 0], [9, 4, 5]],
[[5, 1, 4], [1, 0, 4], [11, 3, 2]],
[[5, 1, 4], [4, 1, 11], [1, 2, 11], [4, 11, 8]],
[[3, 10, 11], [3, 1, 10], [5, 9, 4]],
[[9, 4, 5], [1, 10, 0], [0, 10, 8], [8, 10, 11]],
[[5, 0, 4], [11, 0, 5], [11, 3, 0], [10, 11, 5]],
[[5, 10, 4], [4, 10, 8], [8, 10, 11]],
[[9, 7, 5], [9, 8, 7]],
[[0, 5, 9], [3, 5, 0], [7, 5, 3]],
[[8, 7, 0], [0, 7, 1], [1, 7, 5]],
[[10, 2, 1], [0, 5, 9], [3, 5, 0], [7, 5, 3]],
[[8, 2, 0], [5, 2, 8], [10, 2, 5], [7, 5, 8]],
[[2, 3, 10], [10, 3, 5], [5, 3, 7]],
[[9, 7, 5], [9, 8, 7], [11, 3, 2]],
[[0, 2, 9], [9, 2, 7], [7, 2, 11], [9, 7, 5]],

```
[[3,       2, 11],        [8, 7, 0], [0, 7, 1], [1, 7, 5]],
 [[11, 1, 2], [7, 1, 11], [5, 1, 7]],
 [[3, 1, 11], [11, 1, 10], [8, 7, 9], [9, 7, 5]],
 [[11, 7, 0], [7, 5, 0], [5, 9, 0], [10, 11, 0], [1,                                  10,     0]],
 [[0, 5, 10], [0, 7, 5], [0, 8, 7], [0, 10, 11], [0,                                  11,     3]],
 [[10, 11, 5], [11, 7, 5]],
 [[5, 6, 10]],
 [[8, 0, 3], [10, 5, 6]],
[[0,      9,    1],      [5, 6,        10]],
[[8,      1,    3],      [8, 9,        1], [10, 5, 6]],
[[1,      6,    2],      [1, 5,        6]],
[[6,      2,    5],      [2, 1,        5], [8, 0, 3]],
[[5,      6,    9],      [9, 6,        0], [0, 6, 2]],
[[5,      8,    9],      [2, 8,        5], [3, 8, 2], [6, 2, 5]],
[[3,      2,     11], [10, 5, 6]],
[[0,      2,     8], [2, 11, 8], [5, 6, 10]],
[[3,      2,     11], [0, 9, 1], [10, 5, 6]],
[[5,      6,     10], [2, 9, 1], [11, 9, 2], [8, 9, 11]],
 [[11, 3, 6], [6, 3, 5], [5, 3, 1]],
 [[11, 8, 6], [6, 8, 1], [1, 8, 0], [6, 1, 5]],
 [[5, 0, 9], [6, 0, 5], [3, 0, 6], [11, 3, 6]],
[[6,      9,    5],      [11, 9, 6], [8, 9, 11]],
[[7,      4,    8],      [6, 10, 5]],
[[3,      7,    0],      [7, 4, 0], [10, 5, 6]],
[[7,      4,    8],      [6, 10, 5], [9, 1, 0]],
[[5,      6,     10], [9, 1, 4], [4, 1, 7], [7, 1, 3]],
[[1,      6,    2],     [1,     5,    6], [7,        4,     8]],
[[6,      1,    5],     [2,     1,    6], [0,        7,     4], [3, 7,          0]],
[[4,      8,    7],     [5,     6,    9], [9,        6,     0], [0, 6,          2]],
[[2,      3,    9],     [3,     7,    9], [7,        4,     9], [6, 2,          9], [5, 6,          9]],
[[2,       11, 3], [7, 4, 8], [10, 5, 6]],
[[6,       10, 5], [7, 4, 11], [11, 4, 2], [2,                                  4, 0]],
[[1,      0,    9],     [8,     7,     4], [3, 2, 11], [5, 6, 10]],
[[1,      2,    9],     [9,     2,     11], [9, 11, 4], [4, 11, 7], [5,                    6, 10]],
[[7,      4,    8],      [11, 3, 6], [6, 3, 5], [5, 3, 1]],
 [[11, 0, 1], [11, 4, 0], [11, 7, 4], [11, 1, 5], [11, 5, 6]],
 [[6, 9, 5], [0, 9, 6], [11, 0, 6], [3, 0, 11], [4, 8, 7]],
[[5,      6,    9],      [9, 6, 11], [9, 11, 7], [9, 7, 4]],
 [[4, 10, 9], [4, 6, 10]],
 [[10, 4, 6], [10, 9, 4], [8, 0, 3]],
 [[1, 0, 10], [10, 0, 6], [6, 0, 4]],
[[8,      1,    3],      [6, 1, 8], [6, 10, 1], [4, 6, 8]],
[[9,      2,    1],      [4, 2, 9], [6, 2, 4]],
[[3,      8,    0],      [9, 2, 1], [4, 2, 9], [6, 2, 4]],
[[0,      4,    2],      [2, 4, 6]],
[[8,      2,    3],      [4, 2, 8], [6, 2, 4]],
 [[4, 10, 9], [4, 6, 10], [2, 11, 3]],
 [[11, 8, 2], [2, 8, 0], [6, 10, 4], [4, 10, 9]],
[[2,       11, 3], [1, 0, 10], [10, 0, 6], [6, 0, 4]],
[[8,       4, 1], [4, 6, 1], [6, 10, 1], [11, 8, 1], [2,                        11, 1]],
[[3,       1, 11], [11, 1, 4], [1, 9, 4], [11, 4, 6]],
[[6,       11, 1], [11, 8, 1], [8, 0, 1], [4, 6, 1], [9,                        4, 1]],
[[3,       0, 11], [11, 0, 6], [6, 0, 4]],
```

```
[[4, 11, 8], [4, 6, 11]],
[[6, 8, 7], [10, 8, 6], [9, 8, 10]],
[[3, 7, 0], [0, 7, 10], [7, 6, 10], [0, 10, 9]],
[[1, 6, 10], [0, 6, 1], [7, 6, 0], [8, 7, 0]],
```

[[10, 1, 6], [6, 1, 7], [7, 1, 3]],
[[9, 8, 1], [1, 8, 6], [6, 8, 7], [1, 6, 2]],
[[9, 7, 6], [9, 3, 7], [9, 0, 3], [9, 6, 2], [9, 2, 1]],
[[7, 6, 8], [8, 6, 0], [0, 6, 2]],
[[3, 6, 2], [3, 7, 6]],
[[3, 2, 11], [6, 8, 7], [10, 8, 6], [9, 8, 10]],
[[7, 9, 0], [7, 10, 9], [7, 6, 10], [7, 0, 2], [7, 2, 11]],
[[0, 10, 1], [6, 10, 0], [8, 6, 0], [7, 6, 8], [2, 11, 3]],
[[1, 6, 10], [7, 6, 1], [11, 7, 1], [2, 11, 1]],
[[1, 9, 6], [9, 8, 6], [8, 7, 6], [3, 1, 6], [11, 3, 6]],
[[9, 0, 1], [11, 7, 6]],
[[0, 11, 3], [6, 11, 0], [7, 6, 0], [8, 7, 0]],
[[7, 6, 11]],
[[11, 6, 7]],
[[3, 8, 0], [11, 6, 7]],
[[1, 0, 9], [6, 7, 11]],
[[1, 3, 9], [3, 8, 9], [6, 7, 11]],

```
[[10, 2, 1], [6, 7, 11]],
[[10, 2, 1], [3, 8, 0], [6, 7, 11]],
[[9, 2, 0], [9, 10, 2], [11, 6, 7]],
[[11, 6, 7], [3, 8, 2], [2, 8, 10], [10, 8, 9]],
```

[[2, 6, 3], [6, 7, 3]],
[[8, 6, 7], [0, 6, 8], [2, 6, 0]],
[[7, 2, 6], [7, 3, 2], [1, 0, 9]],
[[8, 9, 7], [7, 9, 2], [2, 9, 1], [7, 2, 6]],
[[6, 1, 10], [7, 1, 6], [3, 1, 7]],
[[8, 0, 7], [7, 0, 6], [6, 0, 1], [6, 1, 10]],
[[7, 3, 6], [6, 3, 9], [3, 0, 9], [6, 9, 10]],

```
[[7, 8, 6], [6, 8, 10], [10, 8, 9]],
[[8, 11, 4], [11, 6, 4]],
[[11, 0, 3], [6, 0, 11], [4, 0, 6]],
```

[[6, 4, 11], [4, 8, 11], [1, 0, 9]],
[[1, 3, 9], [9, 3, 6], [3, 11, 6], [9, 6, 4]],
[[8, 11, 4], [11, 6, 4], [1, 10, 2]],
[[1, 10, 2], [11, 0, 3], [6, 0, 11], [4, 0, 6]],
[[2, 9, 10], [0, 9, 2], [4, 11, 6], [8, 11, 4]],
[[3, 4, 9], [3, 6, 4], [3, 11, 6], [3, 9, 10], [3, 10, 2]],
[[3, 2, 8], [8, 2, 4], [4, 2, 6]],
[[2, 4, 0], [6, 4, 2]],
[[0, 9, 1], [3, 2, 8], [8, 2, 4], [4, 2, 6]],
[[1, 2, 9], [9, 2, 4], [4, 2, 6]],

```
[[10, 3, 1], [4, 3, 10], [4, 8, 3], [6, 4, 10]],
[[10, 0, 1], [6, 0, 10], [4, 0, 6]],
[[3, 10, 6], [3, 9, 10], [3, 0, 9], [3, 6, 4], [3, 4, 8]],
[[9, 10, 4], [10, 6, 4]],
```

[[9, 4, 5], [7, 11, 6]],
[[9, 4, 5], [7, 11, 6], [0, 3, 8]],
[[0, 5, 1], [0, 4, 5], [6, 7, 11]],

```
[[11, 6, 7], [4, 3, 8], [5, 3, 4], [1, 3, 5]],
```

[[1, 10, 2], [9, 4, 5], [6, 7, 11]],
[[8, 0, 3], [4, 5, 9], [10, 2, 1], [11, 6, 7]],
[[7, 11, 6], [10, 4, 5], [2, 4, 10], [0, 4, 2]],
[[8, 2, 3], [10, 2, 8], [4, 10, 8], [5, 10, 4], [11, 6, 7]],
[[2, 6, 3], [6, 7, 3], [9, 4, 5]],
[[5, 9, 4], [8, 6, 7], [0, 6, 8], [2, 6, 0]],
[[7, 3, 6], [6, 3, 2], [4, 5, 0], [0, 5, 1]],
[[8, 1, 2], [8, 5, 1], [8, 4, 5], [8, 2, 6], [8, 6, 7]],
[[9, 4, 5], [6, 1, 10], [7, 1, 6], [3, 1, 7]],
[[7, 8, 6], [6, 8, 0], [6, 0, 10], [10, 0, 1], [5, 9, 4]],
[[3, 0, 10], [0, 4, 10], [4, 5, 10], [7, 3, 10], [6, 7, 10]],
[[8, 6, 7], [10, 6, 8], [5, 10, 8], [4, 5, 8]],
[[5, 9, 6], [6, 9, 11], [11, 9, 8]],
[[11, 6, 3], [3, 6, 0], [0, 6, 5], [0, 5, 9]],
[[8, 11, 0], [0, 11, 5], [5, 11, 6], [0, 5, 1]],
[[6, 3, 11], [5, 3, 6], [1, 3, 5]],
[[10, 2, 1], [5, 9, 6], [6, 9, 11], [11, 9, 8]],
[[3, 11, 0], [0, 11, 6], [0, 6, 9], [9, 6, 5], [1, 10, 2]],
[[0, 8, 5], [8, 11, 5], [11, 6, 5], [2, 0, 5], [10, 2, 5]],
[[11, 6, 3], [3, 6, 5], [3, 5, 10], [3, 10, 2]],
[[3, 9, 8], [6, 9, 3], [5, 9, 6], [2, 6, 3]],
[[9, 6, 5], [0, 6, 9], [2, 6, 0]],
[[6, 5, 8], [5, 1, 8], [1, 0, 8], [2, 6, 8], [3, 2, 8]],
[[2, 6, 1], [6, 5, 1]],
[[6, 8, 3], [6, 9, 8], [6, 5, 9], [6, 3, 1], [6, 1, 10]],
[[1, 10, 0], [0, 10, 6], [0, 6, 5], [0, 5, 9]],
[[3, 0, 8], [6, 5, 10]],
[[10, 6, 5]],
[[5, 11, 10], [5, 7, 11]],
[[5, 11, 10], [5, 7, 11], [3, 8, 0]],
[[11, 10, 7], [10, 5, 7], [0, 9, 1]],
[[5, 7, 10], [10, 7, 11], [9, 1, 8], [8, 1, 3]],
[[2, 1, 11], [11, 1, 7], [7, 1, 5]],
[[3, 8, 0], [2, 1, 11], [11, 1, 7], [7, 1, 5]],
[[2, 0, 11], [11, 0, 5], [5, 0, 9], [11, 5, 7]],
[[2, 9, 5], [2, 8, 9], [2, 3, 8], [2, 5, 7], [2, 7, 11]],
[[10, 3, 2], [5, 3, 10], [7, 3, 5]],
[[10, 0, 2], [7, 0, 10], [8, 0, 7], [5, 7, 10]],
[[0, 9, 1], [10, 3, 2], [5, 3, 10], [7, 3, 5]],
[[7, 8, 2], [8, 9, 2], [9, 1, 2], [5, 7, 2], [10, 5, 2]],
[[3, 1, 7], [7, 1, 5]],
[[0, 7, 8], [1, 7, 0], [5, 7, 1]],
[[9, 5, 0], [0, 5, 3], [3, 5, 7]],
[[5, 7, 9], [7, 8, 9]],
[[4, 10, 5], [8, 10, 4], [11, 10, 8]],
[[3, 4, 0], [10, 4, 3], [10, 5, 4], [11, 10, 3]],
[[1, 0, 9], [4, 10, 5], [8, 10, 4], [11, 10, 8]],
[[4, 3, 11], [4, 1, 3], [4, 9, 1], [4, 11, 10], [4, 10, 5]],
[[1, 5, 2], [2, 5, 8], [5, 4, 8], [2, 8, 11]],
[[5, 4, 11], [4, 0, 11], [0, 3, 11], [1, 5, 11], [2, 1, 11]],
[[5, 11, 2], [5, 8, 11], [5, 4, 8], [5, 2, 0], [5, 0, 9]],
[[5, 4, 9], [2, 3, 11]],
[[3, 4, 8], [2, 4, 3], [5, 4, 2], [10, 5, 2]],

```
 [[5,    4,    10], [10, 4, 2], [2, 4, 0]],
 [[2,    8,    3], [4, 8, 2], [10, 4, 2], [5, 4, 10], [0, 9, 1]],
 [[4,     10, 5], [2, 10, 4], [1, 2, 4], [9, 1, 4]],
 [[8,    3,    4],    [4,    3, 5], [5, 3, 1]],
 [[1,    5,    0],    [5,    4, 0]],
 [[5,    0,    9],    [3,    0, 5], [8, 3, 5], [4, 8, 5]],
 [[5,    4,    9]],
  [[7, 11, 4], [4, 11, 9], [9, 11, 10]],
 [[8,    0,    3],    [7,    11, 4], [4, 11, 9], [9, 11, 10]],
 [[0,    4,    1],    [1,    4, 11], [4, 7, 11], [1, 11, 10]],
  [[10, 1, 4],          [1, 3, 4], [3, 8, 4], [11, 10, 4], [7, 11, 4]]                                    ,
  [[9, 4, 1],           [1, 4, 2], [2, 4, 7], [2, 7, 11]],
 [[1,    9,    2],    [2,    9, 4], [2, 4, 11], [11, 4, 7], [3, 8,                                    0]],
  [[11, 4, 7], [2, 4, 11], [0, 4, 2]],
  [[7, 11, 4], [4, 11, 2], [4, 2, 3], [4, 3, 8]],
  [[10, 9, 2], [2, 9, 7], [7, 9, 4], [2, 7, 3]],
  [[2, 10, 7], [10, 9, 7], [9, 4, 7], [0, 2, 7], [8, 0, 7]],
  [[10, 4, 7], [10, 0, 4], [10, 1, 0], [10, 7, 3], [10, 3, 2]],
 [[8,    4,   7],    [10, 1, 2]],
 [[4,    1,   9],    [7, 1, 4], [3, 1, 7]],
 [[8,    0,   7],    [7, 0, 1], [7, 1, 9], [7, 9, 4]],
 [[0,    7,   3],    [0, 4, 7]],
 [[8,    4,    7]],
 [[9,    8,    10], [10, 8, 11]],
  [[3, 11, 0], [0, 11, 9], [9, 11, 10]],
  [[0, 10, 1], [8, 10, 0], [11, 10, 8]],
  [[11, 10, 3], [10, 1, 3]],
 [[1,    9,    2], [2, 9, 11], [11, 9, 8]],
 [[9,    2,    1], [11, 2, 9], [3, 11, 9], [0, 3, 9]],
 [[8,    2,    0], [8, 11, 2]],
  [[11, 2, 3]],
 [[2,    8,    3], [10, 8, 2], [9, 8, 10]],
 [[0,    2,    9], [2, 10, 9]],
 [[3,    2,    8], [8, 2, 10], [8, 10, 1], [8, 1, 0]],
 [[1,    2,    10]],
 [[3,    1,    8], [1, 9, 8]],
 [[9,    0,    1]],
 [[3,    0,    8]],
  []
def marching_cubes_single_cell(f, x, y, z, k_z, k_xy):
    f_eval = [None] * 8
    for v in range(8):
        v_pos = VERTICES[v]
        f_eval[v] = f[x + v_pos[0]][y + v_pos[1]][z + v_pos[2]]
    case = sum(2**v for v in range(8) if f_eval[v] > 0)
    faces = cases[case]
    def edge_to_boundary_vertex(edge):
        v0, v1 = EDGES[edge]
        f0 = f_eval[v0]
        f1 = f_eval[v1]
        t0 = 1 - adapt(f0, f1)
        t1 = 1 - t0
        vert_pos0 = VERTICES[v0]
```

```
        vert_pos1 = VERTICES[v1]
        return V3((x + vert_pos0[0] * t0 + vert_pos1[0] *
t1)*k_xy,
                  (y + vert_pos0[1] * t0 + vert_pos1[1] *
t1)*k_xy,
                  (z + vert_pos0[2] * t0 + vert_pos1[2] *
t1)*k_z)


    output_verts = []
    output_tris = []
    for face in faces:
        edges = face
        verts = list(map(edge_to_boundary_vertex, edges))
        next_vert_index = len(output_verts) + 1
        tri = Tri(
            next_vert_index,
            next_vert_index+1,
            next_vert_index+2,
        )
        output_verts.extend(verts)
        output_tris.append(tri)
    return Mesh(output_verts, output_tris)
def marching_cubes(f, k_z, k_xy, xmin=XMIN, xmax=XMAX, ymin=YMIN,
ymax=YMAX, zmin=ZMIN, zmax=ZMAX):
    mesh = Mesh()
    for x in range(xmin, xmax):
        for y in range(ymin, ymax):
            for z in range(zmin, zmax):
                cell_mesh = marching_cubes_single_cell(f, x, y,
z, k_z, k_xy)
                mesh.extend(cell_mesh)
    return mesh

def make_circle_obj(filename, fll_matrix, k_z, k_xy):
    mesh = marching_cubes(fll_matrix, k_z, k_xy)
    with open(filename, "w") as f:
        make_obj(f, mesh)
```