ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення

Пояснювальна записка

до магістерської роботи
на ступінь вищої освіти магістр

на тему: «**ПОКРАЩЕННЯ СИСТЕМИ ВІЗУАЛЬНОГО ВІДОБРАЖЕННЯ ПУХЛИН ГОЛОВНОГО МОЗКУ НА ОСНОВІ МЕТОДІВ СЕГМЕНТАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ**»

Виконала: студентка 6 курсу, групи ПДМ–61
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Колесник Олена Сергіївна
(прізвище та ініціали)
Керівник Поперешняк С.В.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

Київ – 2022

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти «Магістр»
Спеціальність підготовки 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В. Негоденко
“ _____”_____ 2022 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Колесник Олена Сергіївна
(прізвище, ім‘я, по батькові)

1.Тема роботи: «Покращення системи візуального відображення пухлин головного мозку на основі методів сегментації цифрових зображень»

Керівник роботи Поперешняк Світлана Володимирівна, к.ф.-м.н., доц.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від«12» жовтня 2022 року №122

2. Строк подання студентом роботи 31.12.2022

3. Вихідні дані до роботи: Матеріали переддипломної практики, монографії, підручники, навчальні посібники, статті та тези конференцій вітчизняних і зарубіжних авторів, Інтернет–ресурси з питань сегментації цифрових зображень МРТ головного мозку

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити).

4.1. Аналіз існуючих програмних рішень та засобів сегментації цифрових зображень МРТ головного мозку

4.2. Проектування архітектури програмного забезпечення для сегментації цифрових зображень МРТ головного мозку.

4.3. Розробка програмного забезпечення.

5. Перелік графічного матеріалу(презентація)

5.1. Мета, об’єкт та предмет дослідження

5.2. Перетворення зображення у градації сірого

5.3. Метод Отцу

5.4. Метод зв’язних компонент

5.5. Фільтр Гаусса

5.6. Морфологічні операції

5.7. Результати моделювання

6. Дата видачі завдання «14» жовтня 2022 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання на магістерську роботу	14.10.2022	Виконано
2	Аналіз літератури	17.10.2022	Виконано
3	Аналіз існуючих методів, концепцій та алгоритмів вирішення завдання	19.10.2022	Виконано
4	Побудова алгоритмічної моделі основних процесів	26.10.2022	Виконано
5	Опис розробленого алгоритму	10.11.2022	Виконано
6	Розроблення програмного забезпечення	15.11.2022	Виконано
7	Тестування розробленого програмного забезпечення	18.11.2022	Виконано
8	Написання та оформлення пояснювальної записки	20.11.2022	Виконано
9	Розробка графічних та презентаційних матеріалів	11.12.2022	Виконано
10	Отримання рецензії		
11	Затвердження пояснювальної записки роботи завідувачем кафедри		
12	Захист магістерської роботи	17.01.2023	

Студент _____ Колесник О.С.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Поперешняк С.В.
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина магістерьскої роботи : 79 с., 30 рис., 1 табл., 9 додат., 12 джерел.

АЛГОРИТМ, ПОРОГ, ЗВ'ЯЗНІ КОМПОНЕНТИ, МОРФОЛОГІЧНІ ОПЕРАЦІЇ, ЕРОЗІЯ, ДИЛАТАЦІЯ, МАРКЕР, ВОДОДІЛ, СЕГМЕНТАЦІЯ

Об'єкт дослідження – процес сегментації цифрових зображень.

Предмет дослідження – метод ефективної сегментації МРТ головного мозку на основі водорозділу та порогового значення.

Мета роботи – поліпшення ручного або напівавтоматичного аналізу МРТ головного мозку за рахунок автоматичної обробки з використанням методів комп'ютерного зору.

Методи дослідження – пороговий метод сегментації Отцу, морфологічні операції (метод виділення зв'язних компонент), операції ерозії і дилатації, методи вододілу (з маркерами, шляхом дилатації). Для моделювання обробки проміжних даних сегментації було використано бібліотеку OpenCv. Для імплементації інтерфейсу користувача – PyQt5.

Досліджено методи сегментації зображення та алгоритм сегментації на основі водорозділу та порогового значення. Реалізовано систему візуального відображення пухлин головного мозку на основі методів сегментації цифрових зображень з користувацьким інтерфейсом. Запропоновано алгоритми удосконалення швидкості сегментації цифрових зображень та комбінування методів сегментації для покращення результатів сегментації.

В результаті досліджень було отримано алгоритм, що застосовується для виділення пухлин на цифрових даних (МРТ); а також алгоритм для обробки масиву зображень головного мозку, що дозволяє спростити подальшу обробку великої кількості цифрових ресурсів і їх сегментацію зі збереженням у документ Microsoft Word, шляхом створення результуючого шаблону.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ООП	–	об’єктно–орієнтоване програмування.
ОС	–	операційна система.
ПЗ	–	програмне забезпечення.
КТ	–	комп'ютерна томографія.
МРТ	–	магнітно–резонансна терапія.
МР	–	магнітно–резонансна.

ВСТУП

МРТ є одним з найбільш інформативних методів візуалізації, а для дослідження пухлин мозку має ряд переваг в порівнянні з іншими методами, зокрема, завдяки здатності отримувати тривимірні зображення з високою міжтканинною контрастністю.

Обсяг інформації про анатомічні структури головного мозку, представлений на МРТ-зображеннях, настільки великий, що візуальний аналіз, найбільш широко використовуваний в клінічній практиці, не дозволяє повністю використовувати отримані в ході дослідження дані, так як вони залишаються за межами можливостей людського ока і не враховуються.

Наявна кількість програмного забезпечення на ринку для сегментації цифрових зображень головного мозку обмежена, а саме ПЗ являє собою продукт, що важко налаштовується.

Актуальність теми: розпізнавання відхилень на зображеннях головного мозку комп'ютерною системою грає велике значення для полегшення роботи медичних закладів, адже спрощує процес виявлення непримітних ореолів пухлин на початковому етапі хвороби, коли людське око майже не здатне відрізнити невеликі відмінності у кольорах. Аналізуючи велику кількість даних (цифрових зображень) комп'ютер може з точністю до 99,9% визначити наявність пухлин головного мозку на цифрових зображення МРТ.

Метою роботи є поліпшення ручного або напівавтоматичного аналізу МРТ головного мозку за рахунок автоматичної обробки з використанням методів комп'ютерного зору.

Завдання:

1. Дослідити методи сегментації зображень МРТ головного мозку;
2. Розробити алгоритм сегментації цифрового зображення методом ефективної сегментації на основі водорозділу та порогового значення;
3. Створити систему візуального відображення пухлин головного мозку з користувацьким інтерфейсом на основі розробленого алгоритму;

4. Провести тестування системи на вибірці зображень;
5. Провести порівняльний аналіз з іншими методами сегментації.

Об’єкт дослідження – процес сегментації цифрових зображень.

Предметом дослідження є метод ефективної сегментації МРТ головного мозку на основі водорозділу та порогового значення.

Методи дослідження: Для розробки алгоритму були використані: пороговий метод сегментації Отцу, морфологічні операції (метод виділення зв'язних компонент), операції ерозії і дилатації, методи вододілу (з маркерами, шляхом дилатації). Для моделювання обробки проміжних даних сегментації було використано бібліотеку OpenCv. Для імплементації інтерфейсу користувача – PyQt5.

Новизна одержаних результатів. Удосконалено швидкість сегментації зображень (до декількох секунд), автоматизовано обробку масиву МРТ зображень, автоматизовано створення результуючої документації (у “.doc” файл).

Практичне значення одержаних результатів: Практична цінність отриманих результатів полягає в можливості виявлення пухлин головного мозку на перших етапах розвитку захворювання для полегшення та оптимізації роботи медичних працівників.

1 ОСНОВНІ ПОНЯТТЯ ТА МЕТОДИ СЕГМЕНТАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ

Сегментація зображення – це широко використовуваний метод обробки і аналізу цифрових зображень для поділу зображення на кілька частин або областей, часто на основі характеристик пікселів в зображенні. Сегментація зображення може включати відділення переднього плану від фону або кластеризацію областей пікселів на основі подібності кольору або форми. Наприклад, нормальним застосуванням сегментації зображення в медичній візуалізації є виявлення та маркування пікселів на зображенні або вокселей тривимірного об'єму, які представляють пухлину в головному мозку або інших органах пацієнта.

1.1 Огляд методів сегментації

Процес кластеризації зображень, тобто пошуку в них однорідних областей, називається сегментацією. Вона вважається першим етапом аналізу зображень, це – базова процедура практично у всіх завданнях обробки зображень за допомогою систем комп'ютерного зору. Як і інші завдання цієї області, сегментація не може бути повністю формалізована, вона включає в себе елементи фільтрації перешкод і виділення зображень [1].

Деякими практичними застосуваннями сегментації зображень є:

1. Виявлення пухлин та інших патологій;
2. Визначення обсягів тканин;
3. Виділення об'єктів на супутникових світлинах;
4. Розпізнавання об’єктів.

Сегментація та класифікація [13, 21] МРТ-зображень повинні бути дуже ефективними для відповідного аналізу пухлин головного мозку. Ідеальна сегментація пухлин головного мозку за зображеннями МРТ є дуже складним і важливим завданням при плануванні лікування та діагностиці, яка включає

вилучення зображень з однієї або декількох областей і створення області інтересу. Для виявлення пухлин головного мозку були розроблені різні алгоритми, такі як методи на основі регіонів, методи на основі порогів, підходи до класифікації, методи, що деформуються, і моделі глибокого навчання [40]. Деформовані моделі є одними з найпопулярніших підходів, які використовуються в МРТ-зображеннях для сегментації пухлин головного мозку. Проведено велику роботу у напрямі вивчення можливостей сегментації зображень МРТ, які можуть надати деяку змістову інформацію з медичних даних. Дослідження пухлин головного мозку проводилися вивчення різних типів пухлин і форми спинномозкової рідини, що ускладнює виявлення пухлини. Незалежно від переваги методу сегментації МРТ-зображень, якість методів сегментації істотно залежить від контрастності медичних зображень, неповних меж і ступеня шумів. Глибоке навчання відіграє важливу роль у виявленні пухлин головного мозку. Існує кілька алгоритмів сегментації зображень, таких як пороговий метод, сегментація на основі областей, нечітка кластеризація, кластеризація k-середніх, нейронні мережі, метод набору рівнів, метод Оцу, нейтральні мережі, алгоритм вододілу і т. д.

CNN [2, 58] широко використовується для класифікації та виявлення пухлин. CNN має багаторівневу архітектуру [36], а також масштабування моделі, що дозволяє збільшити розмір моделі підвищення точності. Це найбільша перевага CNN перед іншими методами сегментації та реалізацією спеціального обладнання для систем штучного інтелекту. Постановка завдання роботи полягає в тому, щоб проаналізувати ефективність різних методів сегментації зображень для виявлення пухлини головного мозку на основі різних параметрів, таких як час відгуку, точність, повнота та прецизійність. Основний внесок дослідження полягає у вивченні різних методів сегментації зображень, включаючи CNN, та застосуванні їх до набору даних BRATS для оцінки часу моделювання та продуктивності. Це дослідження допоможе дослідникам визначити найкращий підхід до сегментації пухлини та попередньо оцінити параметри моделювання та продуктивності.

Окейлі та ін [3] обговорили стратегію диференціації інтрааксіальних утворень головного мозку та визначення точності МРТ-зображень. Наглядова рада

установи схвалила стратегію для звичайної МРТ, перфузійної МРТ, протонної МР-спектроскопії, дифузійно-зваженої МРТ та класифікації інтрааксіальних утворень як первинних новоутворень низького ступеня злоякісності, метастатичних новоутворень та первинних новоутворень високого ступеня злоякісності. Анила та ін [5] використовували концепцію множини і видалення шуму для виявлення аномальної поведінки мозку. Множинний дозвіл був заснований на апроксимації на основі кривої та контрреле. Метод контрацепції показав найкращі результати при розпізнаванні аномалії у головному мозку. Балафар та ін [6] обговорили різні методи виявлення та сегментації пухлин головного мозку на МРТ-зображеннях. Точність виявлення пухлини залежить від методів сегментації. Вони обговорили випадкову модель Маркова, алгоритм вододілу, анатомічні відхилення, сегментацію на основі Атласу, метод на основі кількох регіонів, карти, що самоорганізуються, і методи квантування векторів навчання. Передбачається, що сегментацію мозку можна покращити, якщо об'єднати метод на основі атласу та розпаралелювання. Чаудхарі та ін [9] вивчали концепцію сегментації зображень з використанням методу кластеризації, а для виявлення пухлин головного мозку використовується машина опорних векторів (SVM). Класифікатори змогли виявити сім ознак і SVM довів точність 94,6%.

Голіпур та ін [14] зазначили, що автоматична функціональна локалізація та функціональна візуалізація головного мозку дуже важливі для тимчасового та вищого дозволу пухлин головного мозку. Функціональні карти корисні виявлення деменції, яка може виявити безліч відмінностей між здоровими і пухлинними пацієнтами. Це призведе до правильних інтерпретацій та висновків. Ратан та ін [43] розглянули різні методи сегментації пухлини за даними МРТ, так як цей процес вимагає багато часу, як це було встановлено медичними експертами. Вони вказали на різні методи, такі як інтенсивність, текстура, регіональний, кластеризація, класифікація, нечіткий, нейронна мережа, знання, край, імовірнісний, злиття, SVM, методи набору рівнів, водозбірний басейн, посібник з Атласу, морфологія, нечіткі середні значення C і k означає алгоритми на основі кластеризації. Вони припустили, що комбінований підхід граничної обробки з SVM або Basian може

забезпечити найкращі результати виявлення пухлини головного мозку. Ратан та ін [45] запропонували алгоритм для виявлення пухлин на МРТ-зображеннях за допомогою аналізу симетрії та розрахунку площі пухлини. Вони використовують серединні фільтри, граничне значення та морфологічну операцію для виявлення пухлини. Лі та ін [29] використовували метод уніфікованого набору рівнів для напівавтоматичної сегментації пухлини печінки, щоб інтегрувати попередню інформацію про пухлину, градієнт зображення та регіональної конкуренції. Він був застосований безпосередньо до зображень комп'ютерної томографії (КТ) з контрастним посиленням, а нечітка кластеризація без вчителя використовувалася для імовірнісного розподілу пухлин печінки.

Тапалія та ін [52] використовували метод набору рівнів для сегментації пухлини головного мозку з використанням автоматичного вибору локальної статистики. Порогові значення зображень були раціоналізовані та автоматично налаштовувалися для всіх МР-зображень. Гоель та ін [16] обговорили алгоритм вододілу та метод встановлення рівня для сегментації МРТ-зображень виявлення областей пухлини головного мозку. Порівняльне дослідження проводиться для оцінки продуктивності та часу відгуку в MATLAB. Метод встановлення рівня дав добрі результати в порівнянні з методом Оцу. Мустаким та ін [35] обговорили, що виявлення пухлини головного мозку можливе з використанням сегментації вододілу, порогової сегментації та морфологічних операторів. Вони успішно змоделювали зразки людського мозку, використовуючи відскановані зображення МРТ. Патил та ін [41] використовували алгоритм вододілу та морфологічні оператори для виявлення пухлин на МРТ-зображеннях головного мозку. Для відсканованих МРТ-зображень використовувалися функції шумоподавлення, сегментація областей та морфологічні оператори.

Ремія та ін [44] використовували метод Fuzzy-C-середніх для фільтрації шуму МРТ-зображень. Метод точно досліджує точну ідентифікацію пухлини головного мозку. Метод Оцу застосовувався для сегментації зображень. Автори стверджували, що їхній підхід нечітких С-засобів дав хороші результати, навіть якщо пацієнт думає про пухлину. Сайн та ін [46] пояснили будову головного мозку

людини та запропонували алгоритм виявлення пухлини в головному мозку з використанням методу сегментації Оцу. Амін та ін [4] використовували DWT для злиття зображень, що дало повну інформацію про області пухлини головного мозку на МРТ. Частковий дифузний фільтр використовується для усунення шуму, а для сегментації пухлини застосовується метод глобальної порогової обробки. Ході та ін [25] використовували DWT для виявлення пухлини головного мозку. МРТ є дуже важливим методом у багатьох випадках і здатний забезпечити детальний аналіз зображення людського тіла. Зображення МРТ використовувалися для тесту і пухлина була сегментована з цих зображень. Кумар та ін [27] запропонували використовувати ДВП для сегментації зображення та визначили вертикальну, похилу та кругову області. Вони використовують HAAR DWT, в якому зображення було розподілено на чотири піддіапазони "LL", "LH", "HL" і "HH" з його коефіцієнтами або + 1, або -1. Шрі та ін [48] відзначили, що ідентифікація, виявлення та сегментація точного положення пухлини головного мозку на МРТ-зображеннях є дуже трудомістким та стомлюючим процесом.

Система потребує швидкого перетворення та обчислення, оскільки МРТ-зображення мають безліч модальностей та шумів. Вони використовували DWT для сегментації зображення з морфологічними фільтрами для видалення шуму, та продуктивність системи була покращена. Сінгх та ін [49] запропонував використовувати DWT та дискретне косинусне перетворення (DCT) для додатків обробки зображень. Вони використовували DCT та DWT для стиснення зображень та нанесення водяних знаків. Процеси включають операції сегментації зображення і найкращі результати були отримані за допомогою DWT. Виджи та ін [53] припустили, що сегментація вододілу використовується для автоматичного виявлення пухлин головного мозку, а інструменти автоматизованого проектування (CAD) можуть використовуватися для ідентифікації пухлин у процесі ручної сегментації. Це допомагає для візуалізації 2D- та 3D-зображення пухлини для доступу до пухлини та планування хірургічного втручання. Ляо та ін [32] використовували зображення об'єднаної медичної експертної групи фотографів (JPEG) для додатків адаптивного приховування даних, щоб зберегти різницю між

значеннями коефіцієнта DCT і сусідніми блоками DCT в тому самому положенні шляхом впровадження міжблокових змін для безпеки пацієнта.

Концепція реверсивного приховування для шифрування та дешифрування даних зображення [31] використовувалася для оцінки складності кожного блоку з мінімальною частотою помилок по бітах. Дискретне перетворення Фур'є DFT [30] та стиснення використовувалися для поділу даних, що ховаються для шифрування зображення, та дешифрування для забезпечення безпеки. Джозеф та ін [22] пояснили сегментацію пухлини за допомогою кластеризації K-середніх, а морфологічні оператори використовувалися, щоб уникнути неправильної кластеризації областей. Чжан та ін [56] використовували метод адаптивної винеровской фільтрації для шумоподавлення. Різні морфологічні оператори та функції використовувалися для виключення немозкової тканини. Метод K-середніх+++кластеризації використовувався з комбінацією методу Fuzzy-C-середніх для сегментації зображень. Техніка кластеризації як підвищила стабільність алгоритму, а й знизила чутливість параметрів кластеризації. Моескопс та ін [33, 34] використовували CNN для автоматичної сегментації МР-зображень мозку на деякі класи тканин. Техніка сегментації була застосована до п'яти різних наборів даних з різними віковими групами і були отримані точні результати. Ольшевська та ін [37, 38] обговорювали продуктивність автономної інтелектуальної системи зору, яка оцінювалася на основі частоти хибнопозитивних результатів, частоти хибнонегативних результатів, точності та прецизійності.

Перейра та ін [42] використовували CNN для сегментації зображень МРТ головного мозку з невеликим ядром (3 × 3). Невелике ядро допомагає у розробці глибшої архітектури CNN. Результати перевірені за базами даних BRATS-2013 та BRATS 2015. Сита та ін [47] запропонували автоматичне виявлення пухлини головного мозку за допомогою більш глибокої архітектури та класифікації CNN. Моделювання виконується за допомогою мови програмування Python. Архітектура CNN складається з трьох шарів із невеликими ядрами. CNN досягла точності 97,5% у наборі даних тестування (BRATS) 2015 року. Ван та ін [54] використовували багатомасштабну CNN для сегментації дерматоскопічного зображення. Спочатку

зображення було попередньо оброблене з використанням підвищення контрастності та сегментовано для покращення набору даних з використанням функції втрати сегментації. Каур та ін [24] застосували різні методи машинного навчання, такі як випадковий ліс, SVM, дерева рішень та підтримка K-NN, для оцінки продуктивності даних в реальному часі для віддаленого моніторингу зупинок, в яких випадковий ліс передбачив точність 96,42% для раку молочної залози та інших захворювань. хвороби, яка була високою у співчутті до інших алгоритмів.

Дарган та ін [11, 12] представили стислий огляд необхідності глибокого навчання для медичних зображень та подальшого використання машинного навчання для оцінки продуктивності. Глибоке навчання представляє нові методи та інфраструктури обробки даних, що дозволяють комп'ютерам вивчати різноманітні уявлення та об'єкти. Обговорювалася структура біометричної системи розпізнавання, яка включає процес сервалу та послідовності для ідентифікації поведінкових модальностей. Кумар та ін [28] передбачив випадки захворювання та смерті від COVID-19 в Італії, Іспанії, Японії, Індії, США та Великобританії. За моделлю стежили з використанням методів авторегресійної інтегрованої ковзної середньої (ARIMA) та довготривалої короткочасної пам'яті (LSTM). Гош та ін [15] застосували алгоритми машинного навчання, такі як KNN, метод опорних векторів і багатошаровий персептрон (MLP) для даних мікрочіпа дезоксирибонуклеїнової кислоти (ДНК). Бансал та ін [7] використовували алгоритми машинного навчання для розпізнавання об'єктів. Характеристики об'єктів були вилучені з використанням різних класифікацій, таких як дерево рішень, KNN та випадковий ліс з точністю 80,8%, 74,8% та 85,9% відповідно. Кумар та ін [8] застосували машинне навчання для виявлення та розпізнавання осіб на довільному зображенні, щоб отримати розуміння та знання людини для розпізнавання та вивчення даних про осіб. Гупта та ін [19] використовували масштабно-інваріантне перетворення ознак (SIFT) та прискорення надійної ознаки (SURF) з машинним навчанням випадкового лісу та дерева рішень для вивчення різних ознак обличчя [26] і досягли точності 99,7%

1.1.1 Морфологічні методи

Двійкові зображення можуть містити безліч недоліків. Зокрема, двійкові області, створені простою установкою порогових значень, спотворюються шумом і текстурою. Обробка морфологічного зображення переслідує мету усунення цих недоліків шляхом урахування форми і структури зображення.

Морфологічна обробка зображень – це набір нелінійних операцій, пов'язаних з формою або морфологією елементів зображення. Згідно Вікіпедії, морфологічні операції покладаються тільки на відносний порядок значень пікселів, а не на їх числові значення, і тому особливо підходять для обробки двійкових зображень [6].

Морфологічні методи досліджують зображення за допомогою невеликої форми або шаблону, званого структурують елементом. Елемент структурування розташовується у всіх можливих місцях зображення і порівнюється з відповідною околицею пікселів. Деякі операції перевіряють, «вписується» чи елемент в околиці, в той час як інші перевіряють, «потрапляє» він в околиці або перетинає їх за допомогою таких логічних операцій, як: NOT (логічне «НЕ»); AND (логічне «І»); OR (логічне «АБО»); XOR (виключає «АБО»).

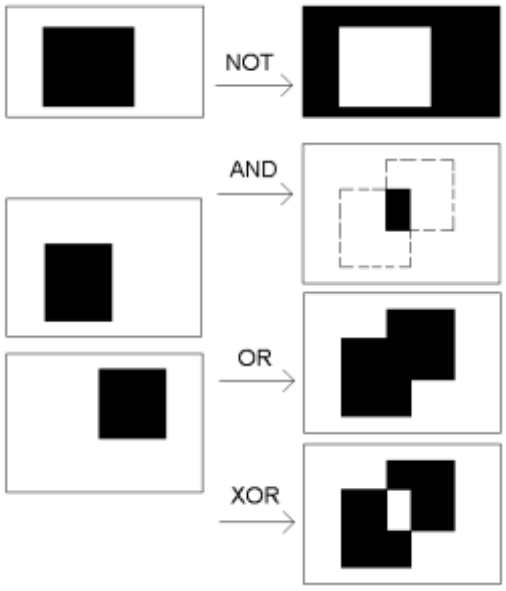


Рисунок 1.1 – Приклади найпростіших логічних операцій над зображенням

Приклади відповідних процедур представлені на рис. 1.1. Також в морфологічних методах використовуються ерозія і дилатація – операції, призначені в першу чергу для виявлення різних морфологічних особливостей зображень, з використанням різних структурних елементів.

<u>До морфологічних методів відносять:</u>

1. Метод виділення границь

Границя множини A, яку позначено через $\beta(A)$, може бути отримана шляхом виповнення операції ерозії A по B, а потім отримання різничної множини між множиною A і результатом її ерозії.

$$\beta(A)=A\setminus(A\ominus B), \tag{1.1}$$

де B – будь–який примітив операції.

На рис. 1.2. показаний механізм роботи процедури. Зліва показано вихідне зображення, праворуч – зображення, отримане відніманням з множини A результату ерозії множини по деякому примітиву B.

Рисунок 1.2 – Приклад отримання меж шляхом виділення границь

2. Метод виділення зв'язкових компонент

Ставлення зв'язності між елементами зображення є фундаментальним поняттям. Для того, щоб встановити, чи є два елементи зображення зв'язними,

необхідно, щоб вони були сусідами і рівні їх яскравості задовольняли заданому критерію подібності.

Нехай S – деяка підмножина елементів зображення. Два елементи p і q називаються зв'язними в S, якщо між ними існує шлях, що цілком складається з елементів множини S. Для кожного пікселя p з S множини усіх пікселів, пов'язаних з ним в S, називається зв'язковою компонентою (або компонентною пов'язаністю) S. Якщо множина S містить тільки одну компоненту зв'язності, вона називається зв'язковою множиною.

На практиці виділення компонент зв'язності на зображенні займає центральне місце в багатьох прикладних задачах аналізу зображень.

Нехай Y – деяка зв'язкова компонента з множини A. Припустимо, що нам відома точка p ∈ Y. Тоді всі елементи компоненти Y можуть бути отримані за допомогою рекурентного виразу:

$$Xk=(Xk-1 \oplus B) \cap Ak=1,2,3,\ldots, \qquad (1.2)$$

де $X_0=p$; B – будь–який примітив операції.

Використання примітиву A в якості обмежуючого обґрунтовано тим, що всі розшукувані пікселі мають значення 1. Взяття перетину з множиною A на кожному кроці ітерації виключає з результатів дилатації ті значення, які припадають на нульові елементи. Даний алгоритм застосовується в разі будь–якого кінцевого числа зв'язкових компонент, з яких складається множина A, за умови, що всередині кожної компоненти зв'язності відома хоча б одна точка.

На рис. 1.3. показана процедура обробки зображення, починаючи з деякої компоненти p. Процедура завершується тоді, коли на поточному кроці не було не виявлено жодної нової зв'язкової компоненти.

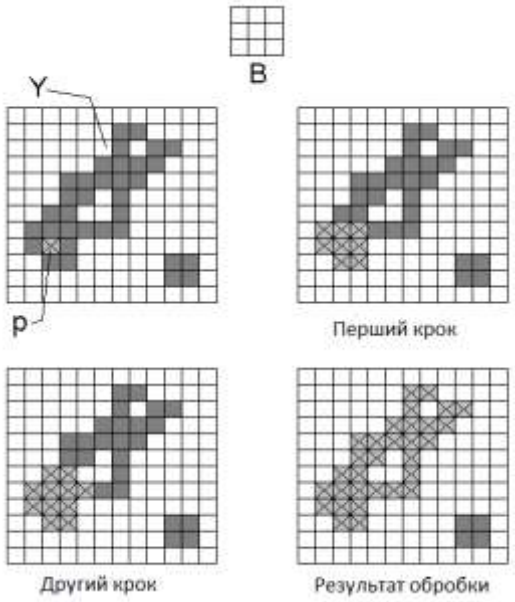


Рисунок 1.3 – Приклад виконання операції виділення зв'язкових компонент

1.1.2 Порогові методи

Встановлення граничних значень – це процес створення чорно–білого зображення з зображення в градаціях сірого шляхом установки білого кольору саме тих пікселів, значення яких перевищує заданий поріг, і установки інших пікселів на чорний. Зображення в градаціях сірого з встановленням граничних значень може використовуватися для створення двійкових зображень. Прості методи порогової обробки включають заміну кожного пікселя в зображенні чорним пикселем, якщо інтенсивність зображення src (x, y) менше деякої фіксованої константи T (тобто src (x, y) <T), або білого пікселя, якщо інтенсивність зображення більше цієї константи.

На рис. 1.4. показані гістограми деяких зображень. В даному випадку такі гістограми відповідають зображенню зі світлими об'єктами на темному фоні. Можна бачити, що пікселі групуються навколо основних центрів. Для виділення цих областей, потрібно вибрати деяке значення T і визначити всі точки, які мають f (x, y) ≤T, як ті, що належать об'єкту, а в іншому випадку – належать фону. Тоді отримане на виході зображення визначається виразом:

$$g(x,y) = \begin{cases} 1, if\ f(x,y) > T \\ 0, if\ f(x,y) \leq T, \end{cases} \quad (1.3)$$

де 1 – значення для пікселя, відповідного об'єкту;

0 – значення для пікселя, відповідного фону.

Якщо значення T однаково для всіх точок зображення, то такий поріг називають глобальним. Якщо значення T залежить від просторових координат x і y, то такий поріг називають динамічним. Якщо ж T залежить від значення f (x, y), то такий поріг називають адаптивним.

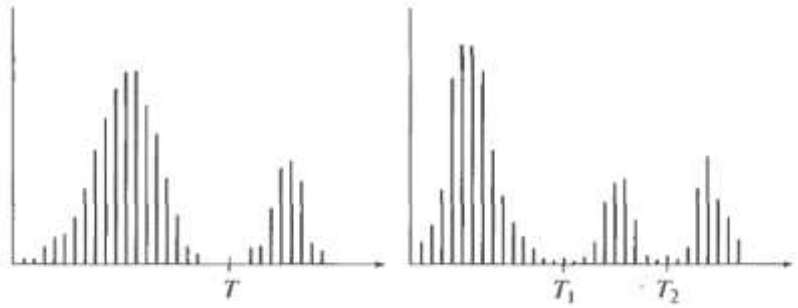


Рисунок 1.4 – Приклад гістограм з можливістю поділу одиночним (T) і множинним (T_1, T_2) порогами

Основні порогові методи:

1. Граничний метод з глобальним порогом

Даний метод є найпростішим. Після вибору глобального порога відбувається поелементно перевірка всього зображення. Процедура передбачає поділ зображення на дві області: перша відноситься до об'єкту, друга – до фону. В даному випадку успішність цілком залежить від того, наскільки добре діаграма піддається поділу. Успішного застосування даного методу можна очікувати в умовах контрольованого освітлення.

На рис. 1.5. зліва наведено вихідне зображення, праворуч – отримане після обробки. Для даного методу можливий автоматичний вибір порога. Для цього застосовується наступний алгоритм:

1. вибирається деяка початкова оцінка порогу T;

2. використовуючи поріг T, сегментуємо зображення на дві області G1 і G2;
3. обчислюємо значення μ1 і μ2 середніх значень яскравості областей G1 і G2;
4. обчислюємо нове значення порога за формулою: T = 12 (μ1 + μ2);
5. обчислюємо кроки 2–4 до тих пір, поки різниця значень T при сусідніх ітераціях не опиниться менше або дорівнює деякому △ T

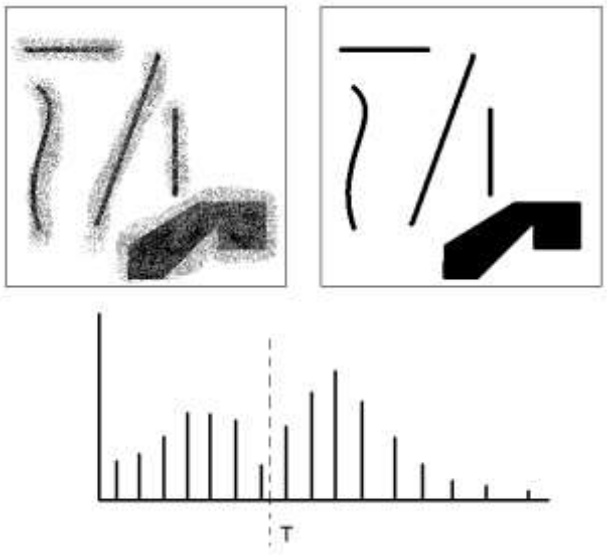


Рисунок 1.5 – Приклад поділу зображення з використанням глобального порога T

Обчислення середніх значень яскравості має сенс, якщо апріорі відомо, що фон і об'єкт мають площі, які можна порівняти на зображенні. Якщо ж площа об'єкта мала, то домінуючим буде фон і в цьому випадку поділ по середньої яскравості області буде не дуже хорошим. У цьому випадку в якості обчислюваних критеріїв μ1 і μ2 можна використовувати половину суми мінімального і максимального значень яскравості.

2. Граничний метод з адаптивним порогом

У попередньому пункті було зазначено, що метод з глобальним порогом хороший до тих пір, поки ми маємо контрольоване освітлення. Як тільки освітлення стає нерівномірним, гістограма, що добре розділялась може перетворитися в гістограму, що погано розділяється, і метод не спрацює. В цьому випадку вихідне зображення слід розділити на підобласті, в кожній з яких для сегментації шукається

і використовується свій поріг. Основною проблемою тут є завдання розбиття зображення на підобласті і вибір для кожної з них свого порога.

Оскільки поріг залежить від характеристик підобласті зображення, такий поріг називають адаптивним. На рис. 1.6. наведено приклад використання глобального і адаптивного порогів.

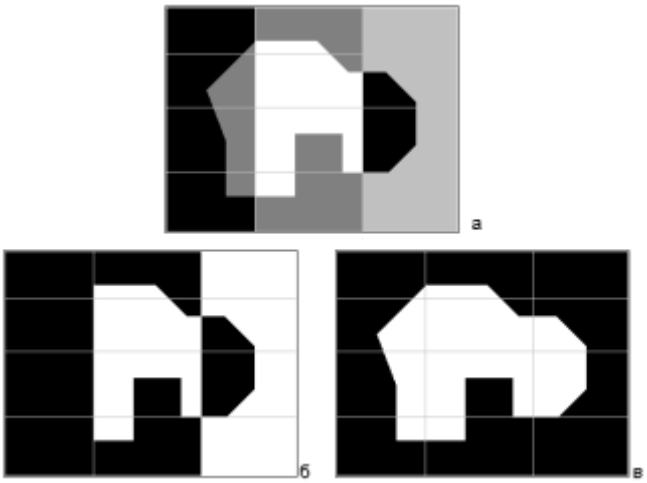


Рисунок 1.6 – Приклад обробки зображення глобальним (б) і адаптивним (в) порогами

На рис. 1.6. (а) показано початкове зображення деякої області. На рис. 1.6. (б) представлений результат використання глобального порога. Права частина області, що шукається загубилася, так як значення пікселів фону і пікселів об'єкта злилися і були відсічені як фон. На рис. 1.6. (в) показаний результат використання адаптивного порога.

Як критерій розбиття зручно використовувати поняття дисперсії освітлення. Тобто зображення розбивається на області, освітленість яких приблизно однакова.

Дисперсія обчислюється за формулою:

$$\sigma^2(z) = \sum_{i=0}^{L-1}(z_i - m)^2 p(z_i), \quad (1.4)$$

де z – величина, відповідна яскравості елементів зображення;

а p(zi), i = 0,1,2, ..., L–1 − гістограма z, де L позначає число різних рівнів яскравості.

1.1.3 Методи нарощування областей

Сегментація вододілів – це ще один регіональний метод, який бере свій початок в математичної морфології.

При сегментації вододілів зображення розглядається як топографічний ландшафт з гребенями і долинами. Значення висоти ландшафту зазвичай визначаються значеннями сірих відповідних пікселів або величиною їх градієнта. Ґрунтуючись на такому тривимірному поданні, перетворення вододілу розбиває зображення на водозбірні басейни. Для кожного локального мінімуму водозбірний басейн включає всі точки, шлях найбільш крутого спуску яких закінчується в мінімумі. Вододіли відокремлюють басейни один від одного. Перетворення вододілу повністю розкладає зображення і, таким чином, призначає кожен піксель або регіону, або вододілу. При наявності зашумленних даних медичних зображень виникає велика кількість невеликих ділянок. Це проблема відома як «надмірна сегментація».

Лінії, утворені крапками–гребенями, являють собою лінії вододілів, тому основним завданням даного методу є саме пошук ліній вододілів.

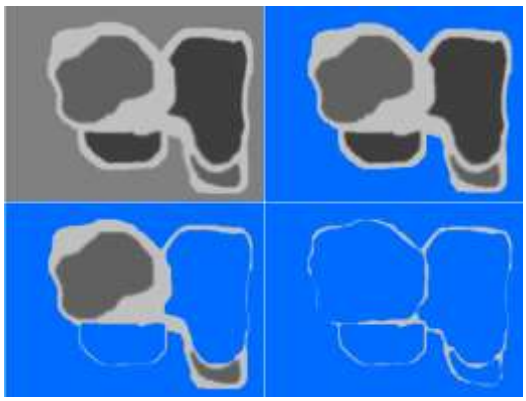


Рисунок1.7 – Приклад роботи алгоритму водорозділів

Алгоритм методу водорозділу (нарощування областей):

1. У місцях локального мінімуму утворюємо «дірки», через які вода почне заповнювати тривимірну поверхню.

2. Якщо вода з двох сторін гребеня готова об'єднатися в один басейн, встановлюємо перегородку.
3. Коли над водою залишаться тільки загородки, зупиняємо алгоритм.

Отримані таким чином перегородки і є необхідні лінії вододілів.

На рис. 1.7. наведено покрокове виконання алгоритму. На верхньому лівому малюнку показаний оригінальний «рельєф» деякого зображення, на нижньому правому – результат роботи алгоритму.

1. *Вододіл шляхом дилатації*

Інший найпростіший спосіб побудови перегородок – використання морфологічної дилатації. Приклад використання цього способу наведено на рис. 1.8. Як примітив в даному випадку використовується матриця 3 * 3 елементи.

Рисунок 1.8 – Побудова вододілів методом дилатації

Під час роботи алгоритму використовуються два правила:

1. застосування дилатації повинно обмежуватися тільки своєю областю;
2. дилатація не повинна застосовуватися в тих точках, де можливе злиття двох областей.

На рис. 1.8. (А) показані дві початкові області. При застосуванні дилатації до областей (рис. 1.8. (Б)) їх площа збільшується. На наступному кроці (рис. 1.8.(В))

нарощувані області готові злитися в одну – значить необхідно встановити перегородку. Після того, як точки розділу знайдені, їм присвоюється значення рівне максимальної яскравості + 1. Це запобігає злиття областей в ході роботи алгоритму. Важливо відзначити, що отримані лінії вододілу є зв'язковими компонентами, які не мають розривів в лініях сегментації.

Однак безпосереднє застосування методу вододілу може привести до надлишкової сегментації, викликаної локальними нерівностями і шумом в зображенні (рис. 1.8. (А)). Практичне розв'язання цієї проблеми полягає в тому, щоб обмежити допустиму кількість областей шляхом включення додаткового кроку попередньої обробки. Такий підхід заснований на ідеї маркерів.

2. *Вододіл з маркерами*

Маркер являє собою зв'язну компоненту, що належить зображенню. Розрізняють зовнішні (відповідають фону) і внутрішні (що відносяться до об'єкта) маркери.

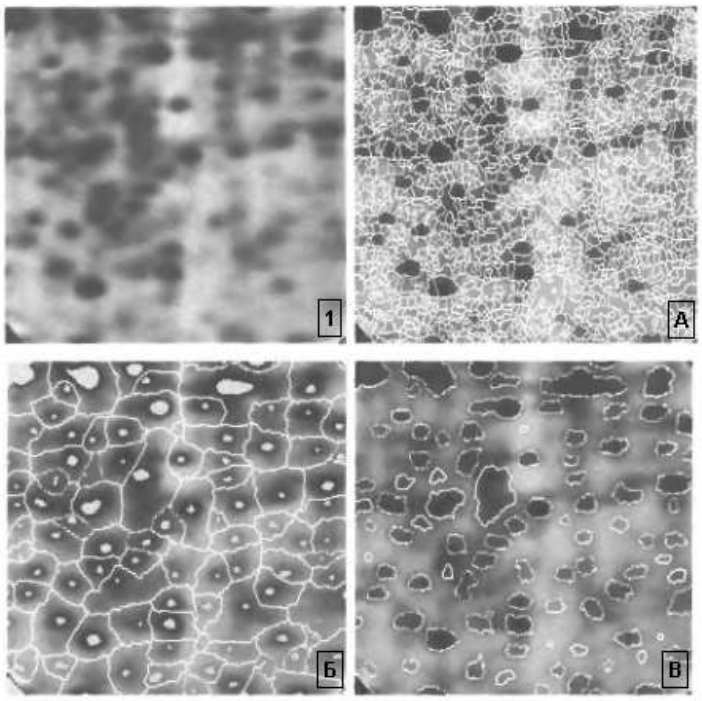


Рисунок 1.9 – Приклад вододілу з використанням маркерів

Процедура вибору маркера складається з двох основних етапів:

1. передобробка;
2. вироблення критеріїв, яким повинні задовольняти маркери.

Нехай внутрішній маркер визначено як область з наступними критеріями:

1. оточена точками з великим значенням яскравості;
2. її точки утворюють компоненту зв'язності;
3. всі точки мають однакове значення яскравості.

Внутрішні маркери, знайдені за цими 3 правилам, показані на рис. 1.9. (Б) світло–сірим кольором. Далі застосовуємо алгоритм сегментації по вододілах з тим обмеженням, що в якості локальних мінімумів розглядаються тільки локальні маркери. В результаті отримаємо зображення, що набагато краще читається в порівнянні з вихідним зображенням (рис. 1.9. (В)).

Наведений випадок – найпростіший. У загальному випадку маркери можуть мати більш складний опис, що включає розміри, форму, місце розташування, відстані, текстурні і інші ознаки. Найбільшою перевагою маркерів є те, що можна використовувати апріорні знання про завдання і ефективно використовувати їх при вирішенні задач сегментації.

1.1.4 Текстурні методи

Текстурні методи - спираються при аналізі на дифузні (колір, відбивна здатність) властивості поверхні аналізованого об'єкта. Представлені в цій категорії методи є наборами складних операторів, які здатні звести процес розпізнавання поверхонь до простого завдання розрізнення рівнів яскравості.

Метод текстурних дескрипторів

Зорова система людини здатна з легкістю розпізнавати і розрізняти текстури. Виявляється, набагато складніше охарактеризувати і виділити швидше «дифузні»

властивості текстури за допомогою точно певних параметрів, які дозволяють комп'ютеру виконувати це завдання.

Довільна структура, яка поширюється на більшу область в зображенні, безумовно, не розпізнається як текстура. Таким чином, основною властивістю текстури є мала елементарна структура, яка повторюється періодично або квазіперіодично в просторі, як малюнок на шпалерах. Таким чином, досить описати малу елементарну структуру і правила повторення. Останні задають характеристичний розмір текстури.

Аналіз текстури можна порівняти з аналізом структури твердих тіл - тематикою, що вивчається у фізиці твердих тіл, хімії і мінералогії. Фізики в області твердих тіл повинні визначити повторювану структуру і розподіл атомів в елементарній комірці. Аналіз структури ускладнюється тим, що і структури, і періодичне повторення можуть проявляти істотні випадкові флуктуації.

Рисунок 1.10 – Приклад різних текстур

Орієнтація поверхонь є ключовою ознакою для іншої задачі обробки зображень - реконструкції тривимірної сцени по двовимірному зображенню. Якщо

поверхня об'єкта показує однорідну структуру, можна аналізувати орієнтацію і розміри текстури для знаходження орієнтації поверхні в просторі. Для цього необхідні характеристичні розміри і орієнтації текстури.

Текстурний аналіз є однією з тих областей в обробці зображень, якій все ще бракує фундаментальних знань. Тому література містить багато різних емпіричних і напівемпіричних підходів до текстурних аналізу. Протилежно представляється досить простий підхід до текстурних аналізу, який будує складні текстурні оператори по елементарним операторам.

Для текстурного аналізу використовуються тільки чотири фундаментальних текстурних оператора:

1. середнє значення;
2. дисперсія;
3. орієнтація;
4. масштаб,

які застосовуються на різних рівнях ієрархічної структури послідовності обробки зображень. Після розрахунку локальної орієнтації і локального масштабу, оператор усереднення і оператор дисперсії можна застосовувати знову, але на цей раз не до середнього значення і дисперсії рівнів яскравості, а до локальної орієнтації та масштабу.

Ці чотири основних текстурних оператора можна згрупувати в два класи. Середнє значення і дисперсія є незалежними від повороту і масштабу, в той час як оператори орієнтації і масштабу тільки визначають орієнтацію і масштаб відповідно. Цей важливий поділ між параметрами, інваріантними або неінваріантними щодо масштабу і повороту, значно спрощує текстурний аналіз. Значення цього підходу полягає в ортогональності набору параметрів, простоті і можливості їх застосування.

Один з найпростіших підходів, що застосовуються для опису текстури, полягає у використанні статистичних характеристик, що визначаються за гістограмою освітленості цілого зображення або його області. Нехай z – випадкова

величина, відповідна яскравості елементів зображення, а $p(z_i)$, i = 0,1,2, ..., L-1 – її гістограма, де L позначає число різних рівнів яскравості. Тоді центральний момент порядку n випадкової величини z дорівнює:

$$\mu_n(z) = \sum_{i=0}^{L-1}(z_i - m)^n p(z_i), \quad (1.5)$$

де m - середнє значення z (середня яскравість зображення):

$$m = \sum_{i=0}^{L-1} z_i p(z_i), \quad (1.6)$$

З цих рівнянь видно, що $\mu_0 = 1$ і $\mu_1 = 0$. Для опису текстури особливо важливий другий момент, тобто дисперсія $\sigma^2(z) = \mu_2(z)$. Вона є мірою яскравості контрасту, що можна використовувати для побудови дескрипторів відносної гладкості.

1.1.5 Методи теорії графів

Загальна ідея методів цієї групи є такою: зображення представляється як зважений граф, з вершинами в точках зображення. Вага ребра графа відображає подібність точок у деякому сенсі (відстань між точками за деякою метрикою). Розбиття зображення моделюється розрізами графа.

Зазвичай у методах теорії графів вводиться функціонал «вартості» розрізу, який відображає якість отриманої сегментації. Так, завдання розбиття зображення на однорідні області зводиться до оптимізаційної задачі пошуку розрізу мінімальної вартості на графі. Такий підхід дозволяє окрім однорідності кольору та текстури сегментів керувати також формою сегментів, їх розміром, складністю кордонів тощо.

Для пошуку розрізу мінімальної вартості застосовуються різні методи: жадібні алгоритми (на кожному кроці вибирається таке ребро, щоб сумарна вартість розрізу була мінімальною), методи динамічного програмування (гарантується, що, вибираючи на кожному кроці оптимальне ребро, отримаємо в результаті оптимальний шлях), алгоритм Дейкстри тощо. Розглянемо деякі методи теорії графів докладніше.

Метод Normalized Cut запропонований J. Shi, J. Malik (1997). Вводиться нормалізований функціонал якості розрізу так, щоб одночасно максимізувати різницю точок між класами та мінімізувати різницю точок усередині класу. Оптимізація нормалізованого функціоналу зводиться до задачі пошуку власних значень матриці попарних відстаней між усіма точками зображення. Для сегментації зображення на дві частини досить знайти друге за величиною власне значення такої матриці.

Складність ефективного алгоритму пошуку власних значень розрідженої матриці лінійна за кількістю точок зображення. Однак метод вимагає зберігання матриці розміром n* n, де n - число точок зображення, і тому у вихідному вигляді не застосовується до великих зображень.

Метод Nested Cuts запропонований Olga Veksler (2000). Основний принцип цього методу полягає у відділенні кожної точки зображення від спеціальної точки поза зображенням розрізом мінімальної вартості. При такому підході зображення ділиться на сегменти, що не перетинаються. Показано, що величиною сегментів зображення можна керувати, накладаючи обмеження вартості розрізу. У статті описується ефективний алгоритм, що ґрунтується на властивостях розбиття, однак цей метод працює надто повільно.

M. Pavan та M. Pelillo (2003) було запропоновано новий підхід, заснований на розрізах графа. Автори вводять таке визначення сегмента, що дозволяє переформулювати завдання пошуку розрізу на графі як задачу квадратичного програмування. Запропоновано метод розв'язання отриманої задачі, заснований на методах еволюційної теорії ігор. Цей підхід також вимагає зберігання пам'яті матриці попарних відстаней, як і метод Normalized Cuts.

Метод сегментації SWA (Segmentation by Weighted Aggregation) заснований на групуванні подібних точок зображення. Основна ідея методу полягає у побудові піраміди зважених графів, кожен із яких отримано з попереднього шляхом об'єднання схожих вершин.

На кожному кроці ваги зв'язків перераховуються. У процесі побудови піраміди обчислюються різні статистики, що характеризують форму, колір,

текстуру регіонів, ці статистики використовуються для обчислення подібності регіонів. Потім, наслідуючи ідеологію методів теорії графів, для отриманого графа вводиться функціонал вартості розрізу і шукається розріз мінімальної вартості. При цьому на відміну більшості методів теорії графів, SWA має складність O(n), де n - число точок зображення, причому число операцій для кожної точки становить лише кілька десятків.

У модифікації алгоритму на кожному наступному кроці аналізується і коригується результат попереднього агрегування, і навіть використовується інформація про межі отриманих сегментів.

Структура, що використовується для цих операцій - дуже схожа на дерево (хоча і реалізується масивом індексів). У ній для кожного пікселя зазначений предок, тобто покажчик (індекс) на деякий схожий за кольором піксель, що знаходиться у тому самому сегменті. Основні операції:

1. Пошук сегмента деякого пікселя 'x': йдемо по предках аж до верху. Найвищий піксель – це корінь дерева, «представник» даного сегмента на даний момент.
2. Об'єднання сегментів. Якщо пікселі мають різні «представники» - отже, вони належать різним сегментам, інакше корінь був би один. Для їхнього об'єднання «представника» сегмента меншої висоти (від найдальшого пікселя до кореня) посилаємо (з нього вказуємо) на більш довгого «представника», щоб не збільшувати висоту дерева. Тепер маємо об'єднаний сегмент із загальним представником.
3. Для того, щоб наступного разу не бігати далеко від пікселя до кореня, після того, як «представник» буде успішно виявлений, встановлюємо пряме посилання з пікселя – відразу на нього. Це скорочує шлях наступних пошуків і називається "path compression".
4. Тепер ми можемо ефективно шукати сегменти за пікселями і об'єднувати їх, а так само вибудовувати MST (Minimum Spanning Tree - мінімальне остовне дерево даного графа) за алгоритмом Краскала.

Рішення про те, з'єднувати області чи ні, приймається на підставі введених користувачем коефіцієнтів.

Якість роботи методів теорії графів залежить від вибору метрики. Тому для вибору оптимальної метрики застосовують машинне навчання. Основні проблеми методів теорії графів – це низька швидкість роботи та великі витрати пам'яті. Більшість методів вимагає зберігання пам'яті матриці попарних відстаней між точками зображення, розмір якої дорівнює квадрату числа точок. Такі обмеження роблять графові методи практично непридатними для великих зображень.

1.2 Ефективний алгоритм виявлення пухлини мозку за допомогою сегментації на основі вододілу та порогів

Розробка методів автоматичної сегментації пухлин головного мозку залишається однією з найскладніших завдань обробки медичних даних. Точна сегментація може поліпшити діагностику, наприклад, оцінку обсягу пухлини в часі. Однак ручна сегментація даних магнітного резонансу – це трудомістке завдання.

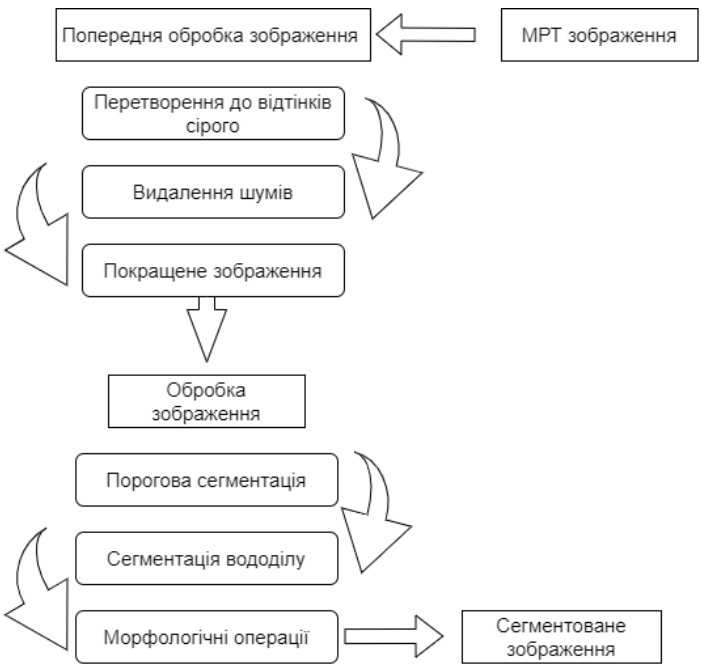


Рисунок 1.10 – Етапи виявлення пухлини

У методі для запобігання надмірної сегментації використовується сегментація вододілу, контрольована маркерами. Попередня обробка зображення МРТ є основним етапом, який усуває шум і згладжує зображення. Щоб запобігти неправильній класифікації тканин мозку та немозкових тканин, проводиться видалення черепа. Сегментація зображень здійснюється за допомогою маркерованої сегментації вододілу. Потім з сегментованого зображення за допомогою морфологічної операції визначають область пухлини. Після цього визначається місце розташування пухлинної області.

1.2.1 Фільтрація зашумленних зображень

Видалення шуму

Шум завжди присутній в цифрових зображеннях на етапах отримання, кодування, передачі і обробки зображень. Фільтрація даних зображення – стандартний процес, який використовується майже в кожній системі обробки зображень. Для цього використовуються фільтри. Вони видаляють шум з зображень, зберігаючи їх деталі. Вибір фільтра залежить від його поведінки і типу даних.

Різновиди фільтрів:

1. *Фільтр, заснований на обчисленні середнього арифметичного*

Такий фільтр є найпростішим. Він згладжує локальні варіації яскравості на зображенні, і видалення шуму відбувається за рахунок цього згладжування.

Нехай S_{xy} – деяка околиця розмірами m * n і з центром в точці (x, y). Необхідно обчислити середнє арифметичне по околиці S_{xy} . Таким чином, для довільної точки оброблюваної околиці маємо:

$$\hat{f}(x,y) = \frac{1}{mn}\sum_{(st)\in S_{xy}} g(s,t) \qquad (1.5)$$

Дану операцію можна представити у вигляді маски, все коефіцієнти якої рівні 1/mn.

2. Медіанний фільтр

Найбільш ходовим з фільтрів, заснованих на статистиках, є медіанний фільтр. Дія цього фільтра зводиться до заміни значення в точці зображення на медіану значень яскравості в околиці цієї точки.

$$\hat{f}(x,y) = med_{(st)\in S_{xy}}\{g(s,t)\} \tag{1.6}$$

При обчисленні медіани, значення в самій точці також враховується. Широка популярність цих фільтрів обґрунтована тим, що вони прекрасно пристосовані до придушення випадкових шумів, і при цьому приводять до найменшого розмивання в порівнянні з іншими фільтрами. Медіанні фільтри особливо успішно працюють у випадках імпульсного шуму.

3. Гаусів шум

Гаусів шум є найбільш простим з математичної точки зору. Спектральні складові цього типу шуму рівномірно розподілені по всьому діапазону задіяних частот. Прикладами білого шуму є шум водоспаду або ефірні перешкоди. У природі і техніці «чисто» білий шум (тобто білий шум, який має однакову спектральну потужність на всіх частотах) не зустрічається (з огляду на те, що такий сигнал мав би нескінченну потужність), проте під категорію білих шумів потрапляють будь–які шуми, спектральна щільність яких однакова (або слабо відрізняється) в розглянутому діапазоні частот. Функція щільності розподілу Гаусова шуму випадкової величини z має вигляд:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2} \tag{1.7}$$

де z є значенням яскравості,

μ – середнє значення випадкової величини z,

σ – її середньоквадратичне відхилення.

1.2.2 Видалення черепної коробки з цифрового зображення

Для методів сегментації пухлини головного мозку першим кроком є видалення черепа з вихідного МРТ–зображення. Це важливий крок, тому що на багатьох МРТ–зображеннях череп виглядає як одна з найяскравіших областей зображення і різко контрастує з іншими областями мозку, такими як сіра речовина. У типах МРТ–зображень, використовуваних в цій дипломній роботі, пухлини також виглядають як яскраві області, і, таким чином, щоб обмежити виникнення хибнопозитивних сегментів, череп повинен бути вилучений з зображення [7].

Техніки видалення черепа можна розділити на 3 основні категорії:

1. **Методи, засновані на інтенсивності**. Вони засновані на порогової класифікації. Основним недоліком цього підходу є його значна чутливість до коливань інтенсивності (в разі МРТ, викликаного, наприклад, неоднорідністю магнітного поля, зареєстрованим шумом або навіть властивостями пристрою).
2. **Методи, засновані на морфології**. Основна ідея полягає в тому, щоб об'єднати використання морфологічних операцій, порогових значень і методів виявлення країв, щоб найбільш точно відокремити область мозку від навколишньої тканини.
3. **Методи на основі деформованої моделі**, в якій застосовується деформація активної контури та підгонка для локалізації областей мозку та її ідентифікація за допомогою характеристик зображення.

Алгоритм, представлений у цій роботі, відноситься до груп методів, заснованих на морфології.

1. порогова обробка зображень,
2. заповнення прогалин в витягнутих об'єктах за допомогою морфологічних операторів,
3. виявлення країв і поліпшення країв, якщо це необхідно,
4. виділення найбільшої області зображення і створення бінарної маски,

5. об'єднання двійковій маски і вхідного зображення в якості вихідного зображення.

Після проведення порогової обробки зображення, інформація про яку зазначена у підрозділі 1.1.2., наступним кроком є створення маски, яка, помножена на вхідне зображення, дозволяє виділити тільки область тканини мозку. Маска розробляється з використанням морфологічних операцій, які засновані на застосуванні так званого «структурного елементу». Це набір пікселів, які можуть мати різні форми і розміри і містити будь–яку комбінацію значень 0 і 1. Якщо значення пікселя не має значення, його можна помітити в структурному елементі як z. У пропонованому методі застосовуються такі морфологічні оператори:

1. **Розширення (потовщення, dilatation).** Структурний елемент порівнюється з кожним пікселем зображення. Якщо хоча б один піксель околиці має значення, рівне «1», точка фокусування також отримує його (в іншому випадку присвоюється значення «0»). Типи структурних елементів сильно впливають на вихідне зображення.
2. **Ерозія (витончення, erosion).** Ця операція застосовує повернений структурний елемент до кожного пікселя зображення. Якщо хоча б один піксель в околиці має значення, рівне «0», точка фокусування також отримує це значення. В іншому випадку його значення не змінюється. Це операція, протилежна дилатації. На ерозію істотно впливає вибір конструктивного елементу.
3. **Відкриття (opening).** Накладення операції дилатації на результат ерозії вихідного зображення. Це викликає згладжування зображення (видалення деталей, чим більше використовується структурний елемент, тим сильніше згладжування зображення).
4. **Закриття (closing).** Накладення ерозійних операцій на результат дилатації вихідного зображення. Вона видаляє всі отвори на

зображенні і увігнутість нижче структурного елементу (чим більше структурний елемент, тим більше елементів заповнюється).

Виявлення країв дозволяє перевірити, чи є кореляція між краями маски (застосовується для вилучення мозку) і анатомічними краями вхідного зображення. Перекриття кордонів мозку на вхідному зображенні з кордонами, встановленими пропонованим методом видалення черепа, доводить, що алгоритм вірний. Для кожної площині зображення застосовується фільтр виявлення границь Кенні. Потім зображення, що представляє краї, об'єднується з відповідним вхідним зображенням.

Алгоритм виявлення границь Кенні:

1. *Згладжування*. Розмивання зображення для видалення шуму.
2. *Пошук градієнтів*. Межі відзначаються там, де градієнт зображення набуває максимальне значення.
3. *Подавлення НЕ–максимумів.* Тільки локальні максимуми відзначаються як кордони.
4. *Подвійна порогова фільтрація.* Потенційні межі визначаються порогами.
5. *Трасування області неоднозначності*. Підсумкові межі визначаються шляхом придушення всіх країв, незв'язаних з певними (сильними) межами.

1.2.3 Сегментація зображення методом Отцу та маркерованого водорозділу

У комп'ютерному зорі і обробці зображень метод Оцу використовується для автоматичного визначення порогових значень зображення. У простій формі алгоритм повертає єдиний поріг інтенсивності, який розділяє пікселі на два класи: передній план і фон. Цей поріг визначається шляхом мінімізації дисперсії інтенсивності всередині класу або, що еквівалентно, шляхом максимізації дисперсії між класами. Метод Оцу є одномірним дискретним аналогом дискримінантного аналізу Фішера, пов'язаний з методом оптимізації Дженкса і еквівалентний

глобальному оптимальному k–середньому, виконаному на гістограмі інтенсивності [5].

Метод Оцу шукає поріг, що зменшує дисперсію всередині класу, яка визначається як зважена сума дисперсій двох класів:

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t), \quad (1.8)$$

де ваги ω_i – це ймовірності двох класів, розділених порогом t,

σ_i^2 – дисперсія цих класів.

Алгоритм методу Отцу:

Нехай дано монохромне зображення $G(i,j), i = \overline{1, Height}, j = \overline{1, Width}$. Лічильник повторів k=0.

1. Обчислити гістограму ***p(l)*** зображення та частоту ***N(l)*** для кожного рівня інтенсивності зображення G.
2. Обчислити початкові значення для $\omega_1(0)$, $\omega_2(0)$ та $\mu_1(0)$, $\mu_2(0)$.
3. Для кожного значення $t = \overline{1, \max(G)}$ – полутону – горизонталь вісь гістограми:
    1. Оновлюємо ω_1, ω_2 та μ_1, μ_2.
    2. Обчислюємо $\sigma_b^2(t) = \omega_1(t)\omega_2(t)[\mu_1(t) - \mu_2(t)]^2$.
    3. Якщо $\sigma_b^2(t)$ більше, ніж наявне, то запам'ятовуємо σ_b^2 і значення порогу t.
4. Шуканий поріг відповідає максимуму $\sigma_b^2(t)$

$$N_T = \sum_{i=0}^{max(G)} p(i), \quad (1.9)$$

$$\omega_1(t) = \frac{\sum_{i=0}^{t-1} p(i)}{N_T} = \sum_{i=0}^{t-1} N(i), \ \omega_2(t) = 1 - \omega_1(t) \quad (1.10)$$

$$\mu_T = \frac{\sum_{i=0}^{max(G)} i*p(i)}{N_T} = \sum_{i=0}^{\max(G)} i * N(i), \quad (1.11)$$

$$\mu_1(t) = \frac{\sum_{i=0}^{t-1} i*p(i)}{N_T*\omega_1(t)} = \frac{\sum_{i=0}^{t-1} i*N(i)}{\omega_1(t)}\ ,\ \mu_2(t) = \frac{\mu_T - \mu_1(t)*\omega_1(t)}{\omega_2(t)} \quad (1.12)$$

Будь–яке зображення у відтінках сірого можна розглядати як топографічну поверхню, де висока інтенсивність означає піки і пагорби, а низька інтенсивність позначає западини. Починаємо заповнювати кожну ізольовану западину (локальні мінімуми) водою різного кольору (мітки). У міру підйому води, в залежності від найближчих піків (градієнтів), вода з різних западин, очевидно, різного кольору, почне зливатися. Щоб цього уникнути, будуємо перешкоди в місцях злиття води. Продовжуємо заповнювати водою і будувати перепони, поки всі вершини не опиняться під водою. Тоді створені бар'єри дають результат сегментації.

Але такий підхід дає сверхсегментований результат через шум або будь–які інші нерівності зображення. Отже, реалізувавши алгоритм вододілу на основі маркерів, в якому вказуємо, які точки западини повинні бути об'єднані, а які ні. Надаємо об'єкту різні ярлики. Позначаємо область, яка, є переднім планом або об'єктом, одним кольором (або інтенсивністю), позначаємо область, яка є фоном або не є об'єктом, іншим кольором і, нарешті, область, в якій ми не впевнені, позначаємо її нулем. Це маркер. Потім застосовуємо алгоритм вододілу. Потім маркер буде оновлено присвоєними мітками, а межі об'єктів матимуть значення – 1.

Алгоритм:

1. Починаємо з знаходження приблизної оцінки пухлини. Для цього використовуємо бінаризацію Отцу.
2. Тепер потрібно видалити невеликі білі шуми на зображенні. Для цього використовується морфологічне відкриття. Щоб видалити невеликі отвори в об'єкті, використовується морфологічне закриття. Ітак, визначаємо: область поруч із центром об'єкта є переднім планом, а область далеко від об'єкта – фоном. Єдина невідома область – це пограничная область пухлини.

3. Потрібно витягти область, в якій знаходитися пухлина за допомогою ерозії, яка видаляє граничні пікселі.

1.2.4 Сегментації на основі ефективного розрізу графа

Кожен піксель цифрового зображення можна представити вершиною у графі. А вага (довжина) ребра, що з'єднує сусідні вершини, виражається формулою:

$$W(v_i, v_j) = |I(p_i) - I(p_j)|, \tag{1.13}$$

де $I(p_i)$ – інтенсивність (яскравість) пікселя p_i.

У ході виконання алгоритму Краскала, на проміжному етапі ми матимемо кілька розрізнених сегментів (підмножин пікселів), з мінімальною сумарною вагою ребер усередині: сегменти будуть об'єднані ребрами мінімальної довжини, тобто з мінімальними «різницями інтенсивностей» між сусідніми пікселями. Тому сусідні пікселі всередині одного сегмента схожі за кольором. Але лише до деякого значення максимального ребра (перепаду інтенсивності).

Далі потрібно вибрати мінімальне на поточному етапі ребро. Для двох пікселів, які є суміжними вершинами цього ребра, визначаємо чи вони з одного сегмента:

1. Так, з одного сегмента: просто продовжуємо виконання алгоритму.
2. Ні. Необхідно визначити, чи представляють сегменти частини одного й того ж об'єкта на зображенні та чи слід його об'єднати, чи їх інтенсивності «суттєво» різняться.

Для визначення різниці між сегментами, з кожним вже побудованим сегментом ми асоціюємо деяку величину – максимальний перепад інтенсивностей усередині нього, тобто найдовше ребро в MST всередині сегмента:

$$Int(C) = \max_{e \in MST(C)} w(e), \tag{1.14}$$

Окремо його шукати не доведеться - досить просто зберігати довжину ребра, що додається при об'єднанні складових «підсегментів». Адже на момент об'єднання довжина ребра, що додається, була більшою, ніж у вже побудованих MST кожного «підсегменту», так як ребра обробляються у порядку, що зростає.

Отримуємо зразкове вирішальне правило для сегментів C_1, C_2:

$$D(C_1, C_2) = \begin{cases} true, Diff(C_1, C_2) > MInt(C_1, C_2) \\ false, merge\ segments \end{cases} \quad (1.15)$$

Виходить що, для того щоб сегменти «об'єдналися», перепад інтенсивностей на їхньому кордоні повинен бути меншим за максимальний перепад усередині кожного з сегментів, що об'єднуються:

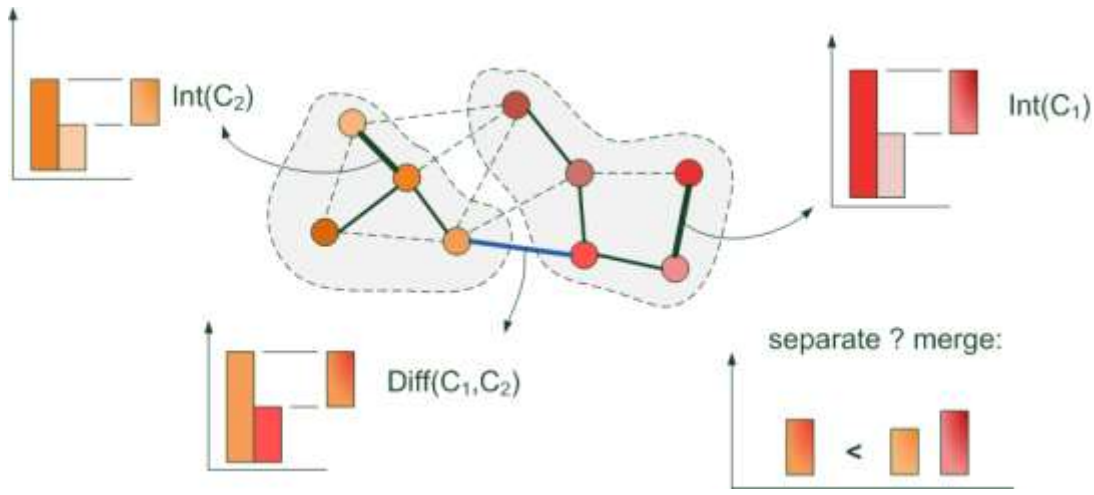


Рисунок 1.11 – Алгоритм об’єднання сегментів

На поверхні побудови графа за зображенням лежать два основні підходи (рис.1.12):

1. 4-зв'язані: кожен піксель з'єднуємо з сусідніми зверху/знизу та /ліворуч/праворуч. Така побудова приваблива тим, що кількість ребер у графі мінімальна.
2. 8-зв'язані: додатково до попереднього варіанта кожен піксель з'єднуємо із сусідніми, що знаходяться по діагоналі. Ребер, таким чином, у графі виходить більше, алгоритм виконується дещо повільніше. Але

сегментація, що отримується, повинна бути якіснішою – адже враховується більше зв'язків між пікселями.

Алгоритм повинен давати якісніші результати на 8-зв'язаному побудованому графі, проте 4-зв'язаний дає дуже прийнятні результати і при цьому виконується значно швидше. А як «відстань» для обробки кольорових зображень можемо взяти таку різницю кольору пікселів:

$$dist(p_i, p_j) = \sqrt{(r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}), \quad (1.16)$$

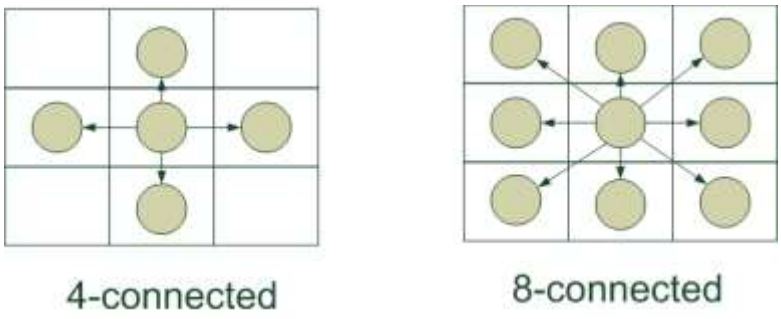


Рисунок 1.12 – Підходи для побудови графа з зображення

Однак, таке формулювання відстані має недолік. Тому в альтернативному варіанті, пропонується розглядати не тільки прилеглі пікселі, а відштовхнувшись від локальних властивостей зображення (пікселів, що розташовуються в безпосередній близькості), спробувати «перестрибнути через ближній об’єкт» і дотягнутися до продовження сегмента, що знаходиться на деякому віддаленні. Для цього як довжина ребра пропонується вибрати Евклідову відстань, що залежить як від положення пікселів (x, y), так і від їх кольору (r, g, b):

$$dist(p_i, p_j) = \sqrt{(x_i - x_j)^2 + (y_i - y_j)^2 + (r_i - r_j)^2 + (g_i - g_j)^2 + (b_i - b_j)^2}), \quad (1.17)$$

Тепер пікселі будуть вважатися «сусідами», якщо вони розташовані поруч, або мають схожий відтінок, хоча фізично можуть знаходитися на деякому віддаленні. Для побудови графа пропонується кожен піксель з'єднувати із 10 (20,

30) «найближчими». Це дає якіснішу сегментацію, але потребує більше обчислювальних ресурсів.

Повертаючись у початок, коли всі пікселі були розрізнені – між сусідніми міг спостерігатися істотний перепад інтенсивностей, який не дозволяє їм об'єднуватися (merge). Навряд чи нам надали мікроскопічну фотографію, де кожен піксель треба відокремити — швидше за все вони є частиною якогось більшого об'єкта. Щоб вони піддавалися «злиттю» (merge) більшою мірою, ніж вже побудовані великі сегменти, котрим важливіша коректність розбиття, у вирішальному правилі додають величину, залежить від обсягу побудованого сегмента:

$$T(C) = \frac{k}{|C|}, \tag{1.18}$$

де |C| - потужність аналізованого сегмента (кількість пікселів в ньому на поточний момент), а k - це вже параметр сегментації, що задається вручну. З такою поправкою у вирішальному правилі використовуватиметься формула:

$$MInt(C_1, C_2) = \min(Int(C_1) + T(C_1), Int(C_2) + T(C_2)), \tag{1.19}$$

"Поправка" поступово нівелюється зі зростанням сегмента (збільшенням |C|).

Насправді, замість подібного завдання T можна вибрати й іншу функцію, що враховує специфіку зображень, що обробляються: форму сегментів, положення на фотографії, певні колірні відтінки

Висновки до розділу 1

У даному розділі презентовано найпоширеніші методи сегментації зображень, які використовуються для формування якісної сегментації головного мозку. Розглянуто та виділено основні етапи сегментації зображень МРТ головногу мозку, такі як: фільтрація зашумлених зображень за допомогою Гаусового шуму, видалення черепної коробки морфологічними операціями, та саме сегментування за допомогою комбінації порогового методу Отцу та маркерованого водорозділу.

2 СИСТЕМА ВІЗУАЛЬНОГО ВІДОБРАЖЕННЯ ПУХЛИН ГОЛОВНОГО МОЗКУ НА ОСНОВІ МЕТОДІВ СЕГМЕНТАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ

2.1 Аналіз сучасних засобів програмної інженерії для вирішення задач виявлення пухлини головного мозку на МРТ зображеннях

Бібліотеки, що можуть бути використані для аналізу та порівняння результатів процесу сегментації:

Scikit-Image — одна з кращих бібліотек Python для обробки зображень з відкритим вихідним кодом, що представляє собою набір алгоритмів для обробки зображень. Вона вільна від обмежень з високою якістю і складається з рецензованого коду. Вона охоплює алгоритми сегментації, геометричні перетворення, аналіз, виявлення ознак і багато іншого. Популярна для роботи з масивами NumPy у якості зображень об’єктів.

SciPy — це відома бібліотека Python для обробки зображень, також відома як scipy.ndimage. Вона пропонує широкий спектр загальних функцій обробки та аналізу зображень, призначених для роботи з кількома масивами довільної розмірності. Існує кілька функцій, які мають загальні властивості з більш ефективним вихідним аргументом. Ndimage — це короткий термін для n-мірного зображення із загальними завданнями, такими як базові маніпуляції, фільтрація зображень, класифікація, відкриття та запис у файлах зображень і багато іншого.

Mahotas — популярна бібліотека Python для комп’ютерного перегляду та обробки зображень за кількома алгоритмами, реалізованими на C++ для ефективної роботи. Цей інструмент обробки зображень відомий тим, що працює з масивами Numpy з чистим інтерфейсом Python. Він містить понад 100 функцій для обробки зображень, таких як водорозділ, співпадіння та промах, звертання, морфологічна обробка та багато іншого.

Pillow — одна з відомих бібліотек Python для обробки зображень, популярна для архівації програм і пакетної обробки зображень. Це допомагає додати функції обробки зображень в інтерпретатори Python, пропонуючи розширену підтримку

форматів файлів і ефективне внутрішнє відображення. Цей інструмент обробки зображень призначений для більш швидкого доступу до даних, що зберігаються в базовому форматі пікселів. Можна створювати ескізи, перетворювати формати файлів, друкувати зображення і багато іншого.

OpenCV — один із популярних інструментів обробки зображень для задач комп'ютерного зору та обробки зображень для різних програм. Це бібліотека Python із відкритим вихідним кодом для обробки зображень і відео для розпізнавання осіб, ідентифікації об'єктів, а також людського почерка. Вона складається з кількох числових операцій із комбінацією Numpy. Бібліотека обробки зображень допомагає перетворювати зображення в багатомірні масиви для спрощення маніпуляцій.

SimpleITK — одна з кращих крос-платформених систем з відкритим вихідним кодом, що є бібліотекою Python для обробки зображень. Вона відома підтримкою 2D і 3D зображень з вибраним для них набором типів пікселів. Існує система реєстрації для швидкого вирівнювання 2D- і 3D-інтра- та інтермодальних зображень з безліччю фільтрів для робочих процесів сегментації зображень. Існує широкий спектр інструментів для оцінки результатів сегментації для підтримки 20 форматів файлів зображень із простим перетворенням між форматами.

Matplotlib дуже корисний як модуль зображення для роботи з зображеннями в Python. Він також включає два корисні методи, які використовуються для читання зображень, а також для відображення зображень. Він спеціалізується на 2D-графічних масивах як багатоплатформена бібліотека візуалізації даних на масивах Numpy.

NumPy — це популярна бібліотека Python для обробки зображень з ndarray для встановлення та зміни кількості пікселів, обрізки зображень, об'єднання зображень і багато іншого. Можна виконати обробку кількох зображень без використання інших бібліотек Python. Це сприяє зменшенню кольору, бінарізації, вставці зі зрізом, позитивній або негативній інверсії та багатьох інших функціях для ефективного читання та збереження зображень.

SimpleCV — це відома платформа з відкритим вихідним кодом для створення додатків комп'ютерного зору з обробкою зображень. Це інтерфейс для бібліотек машинного перегляду з відкритим вихідним кодом на Python з інтерфейсом для камери, обробки зображень, перетворення форматів, вилучення функцій і багато іншого. Це допомагає виконувати завдання комп'ютерного зору дуже простими способами, що переважають над складним кодом.

Pgmagick — одна з кращих бібліотек Python для обробки зображень для бібліотеки GraphicMagick. Вона також відома як швейцарський армійський нож обробки зображень у спільноті мов програмування. Інструмент обробки зображень складається з набору інструментів і бібліотек, які допомагають редагувати зображення, а також маніпулювати ними.

2.2 Технічні засоби та структури

Мова програмування Python

Python – високорівнева мова програмування загального призначення з динамічною строгою типізацією і автоматичним управлінням пам'яттю, орієнтована на підвищення продуктивності розробника, читання коду і його якості, а також на забезпечення переносимості написаних на ньому програм. Мова є повністю об'єктно–орієнтованою – все є об'єктами. Незвичайною особливістю мови є виділення блоків коду пробільними відступами. Синтаксис ядра мови мінімалістичний, за рахунок чого на практиці рідко виникає необхідність звертатися до документації. Сама же мова відома як інтерпретована і використовується в тому числі для написання скриптів. Недоліками мови є часто більш низька швидкість роботи і більш високе споживання пам'яті написаних на ній програм в порівнянні з аналогічним кодом, написаним на компільованих мовах, таких як C або C ++.

Python є мультипарадигмальною мовою програмування, що підтримує імперативне, процедурне, структурне, об'єктно–орієнтоване програмування, метапрограмування і функціональне програмування. Завдання узагальненого

програмування вирішуються за рахунок динамічної типізації. Аспектно–орієнтоване програмування частково підтримується через декоратори, більш повноцінна підтримка забезпечується додатковими фреймворками. Такі методики як контрактне і логічне програмування можна реалізувати за допомогою бібліотек або розширень. Основні архітектурні риси – динамічна типізація, автоматичне керування пам'яттю, повна інтроспекція, механізм обробки виключень, підтримка багатопоточних обчислень з глобальним блокуванням інтерпретатора (GIL), високорівневі структури даних. Підтримується розбиття програм на модулі, які, в свою чергу, можуть об'єднуватися в пакети.

Середовище розробки PyCharm

PyCharm – інтегроване середовище розробки для мови програмування Python. Надає засоби для аналізу коду, графічний відладчик, інструмент для запуску юніт–тестів і підтримує веб–розробку на Django.

PyCharm пропонує великий набір інструментів з коробки: вбудований відладчик і інструмент запуску тестів, профілювальник Python, повнофункціональний вбудований термінал, інструменти для роботи з базами даних. IDE інтегрована з популярними системами контролю версій, містить вбудований SSH–термінал, підтримує можливості віддаленої розробки та віддалені інтерпретатори, а також інтеграцію з Docker і Vagrant.

Розумний редактор PyCharm призначений для максимально продуктивної розробки на Python, JavaScript, CoffeeScript, TypeScript, CSS і популярних мовах шаблонів. Функції автодоповнення, виявлення помилок і швидкі виправлення враховують особливості кожної з підтримуваних мов.

PyCharm надає широкі можливості реорганізації коду за допомогою рефакторингов Rename і Delete, Extract Method, Introduce Variable, Inline Variable, Inline Method і багатьох інших. Рефакторинги враховують особливості конкретної мови або фреймворка, допомагаючи вносити зміни по всьому проекту.

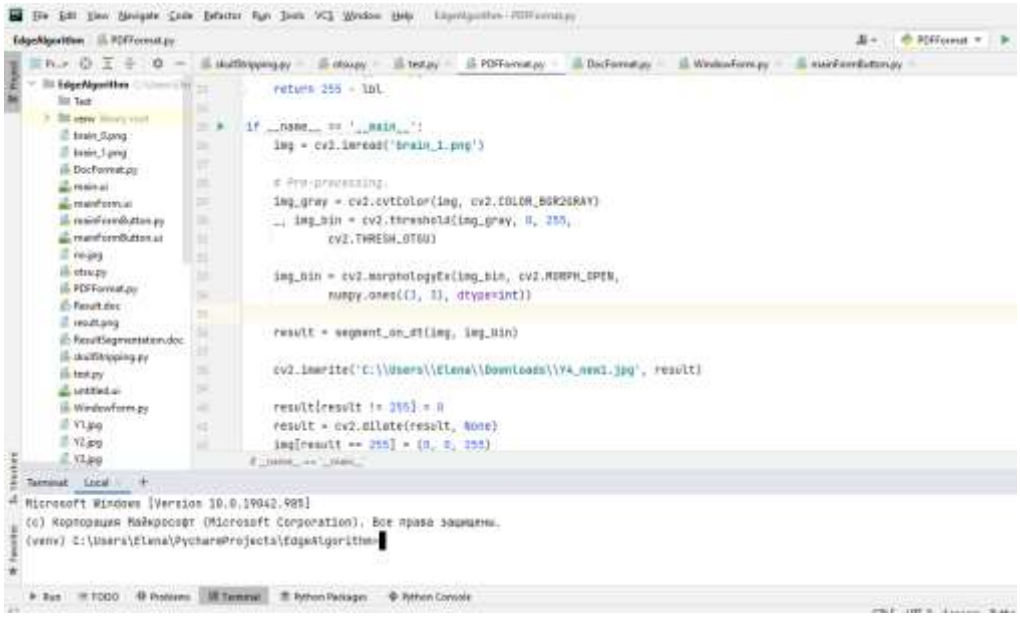


Рисунок 2.1 – Редактор файлу коду PyCharm

Фреймворк Qt

Qt – це кроссплатформна середа розробки додатків для настільних, вбудованих і мобільних пристроїв. Підтримувані платформи включають Linux, OS X, Windows, VxWorks, QNX, Android, iOS, BlackBerry, Sailfish OS і інші.

Qt сам по собі не є мовою програмування. Це фреймворк, написаний на C ++. Препроцесор, MOC (компілятор метаоб'єктів), використовується для розширення мови Python такими функціями, як сигнали і слоти. Перед етапом компіляції MOC аналізує вихідні файли, написані на розширеному Qt Python, і генерує з них вихідні коди Python, що відповідають стандарту. Таким чином, сам фреймворк і додатки / бібліотеки, що його використовують, можуть бути скомпільовані будь–яким стандартним компілятором C ++/Python.

Компоненти:

1. QtCore – класи ядра бібліотеки, які використовуються іншими модулями;
2. QtGui – компоненти графічного інтерфейсу;
3. QtWidgets – містить класи для класичних додатків на основі віджетів, модуль виділений з QtGui в Qt 5;
4. Qt QML – модуль для підтримки QML;

5. QtNetwork – набір класів для мережевого програмування. Підтримка різних високорівневих протоколів може змінюватися від версії до версії. У версії 4.2.x присутні класи для роботи з протоколами FTP і HTTP. Для роботи з протоколами TCP / IP призначені такі класи, як QTcpServer, QTcpSocket для TCP і QUdpSocket для UDP;
6. QtOpenGL – набір класів для роботи з OpenGL;
7. QtSql – набір класів для роботи з базами даних з використанням SQL. Основні класи даного модуля в версії 4.2.x: QSqlDatabase – клас для надання з'єднання з базою, для роботи з якою–небудь конкретною базою даних вимагає об'єкт, успадкований від класу QSqlDriver – абстрактного класу, який реалізується для конкретної бази даних і може вимагати для компіляції SDK бази даних. Наприклад, для складання драйвера під СУБД Firebird або InterBase потрібні .h–файли і бібліотеки статичного компонування, що входять в комплект поставки даної СУБД;
8. QtScript – класи для роботи з Qt Scripts;
9. QtSvg – класи для відображення і роботи з даними Scalable Vector Graphics (SVG);
10. QtXml – модуль для роботи з XML, підтримуються моделі SAX і DOM;
11. QtDesigner – класи створення розширень для своїх власних віджетів;

Середовище Qt Designer

Qt Designer – це інструмент програмного забезпечення Qt для проектування та побудови графічних інтерфейсів користувача (GUI) з компонентів Qt. Дозволяє складати та налаштовувати свої віджети чи діалоги візуально–інтерпретовано, і тестувати їх, використовуючи різні стилі та роздільну здатність.

Віджети та форми, створені за допомогою Qt Designer, легко інтегруються із запрограмованим кодом, використовуючи механізми сигналів і слотів Qt, що дозволяє легко призначати поведінку графічним елементам. Усі властивості, встановлені в Qt Designer, можна динамічно змінювати в коді. Крім того, такі

функції, як просування віджетів та спеціальні плагіни, дозволяють використовувати власні компоненти з Qt Designer.

Кожен віджет має свій набір властивостей, що визначається відповідним йому класом бібліотеки Qt. Властивості віджета можуть бути змінені за допомогою «Редактора властивостей». Характерною особливістю Qt Designer є підтримка візуального редагування сигналів і слотів. Так, наприклад, можна зв'язати сигнал, що генерується з переключення стану віджета CheckBox зі слотом, що відповідає за доступність іншого віджета.

Розроблений інтерфейс зберігається в файл з розширенням ui, який підключається до створюваної програмі за допомогою спеціальних методів бібліотеки Qt. Цей файл має xml–формат, і може, в разі необхідності, редагуватися в будь–якому текстовому редакторі.

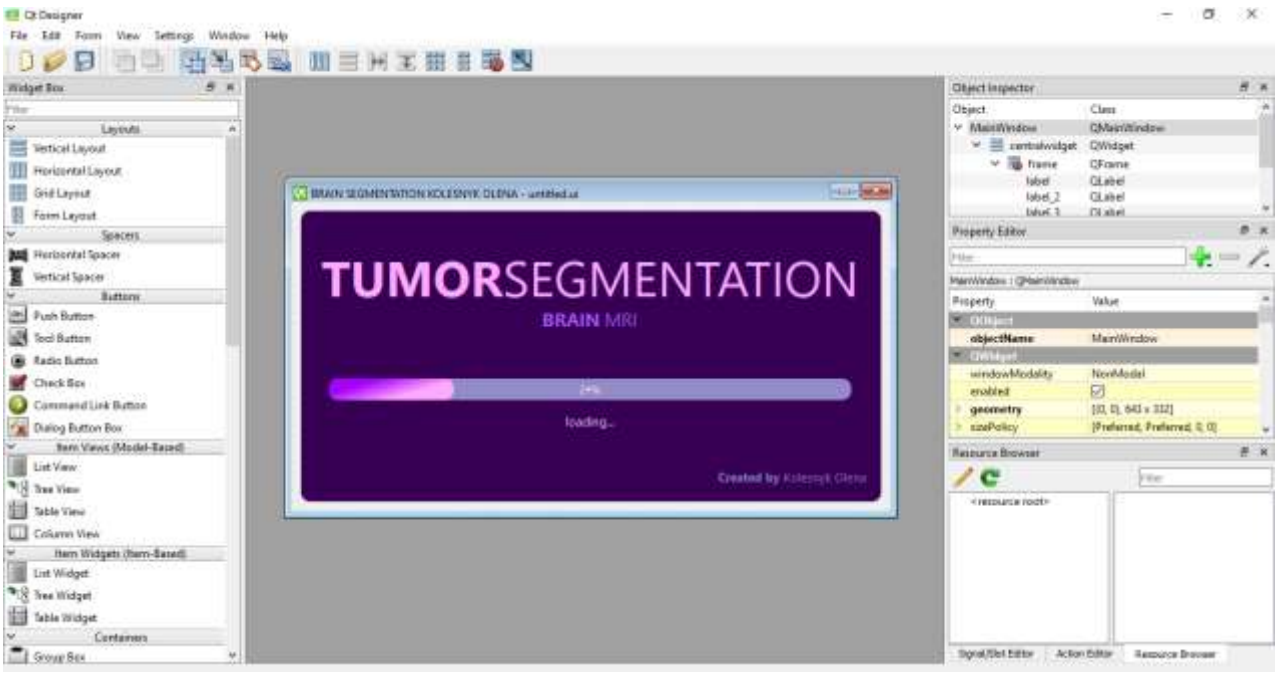


Рисунок 2.2 – Головне вікно Qt Designer

2.3 Опис програмного забезпечення

Для виконання поставленої задачі було розроблено програмне забезпечення для обробки цифрових зображень МРТ мозку.

Основні характеристики додатку:

Мова програмування: Python

Середовище розробки: PyCharm 2019

Мінімальні системні вимоги:

1. 64–бітна операційна система Widows;
2. Python 3.5 або вище;
3. 36 КБ доступного простору на жорсткому диску.

Вхідні дані:

1. цифрові зображення форматів JPEG (.jpg) та PNG (.png);
2. параметри: шляхи збереження файла/файлів;

Вихідні дані: сегментовані цифрові зображення МРТ головного мозку.

Після зчитування зображення з файлу, програма проводить перетворення від цифрового зображення до тривимірного масиву. На основі інтенсивності трьох каналів R, G, B зображення розмивається за допомогою формули визначення інтенсивності пікселя. Результатом є зображення в градаціях сірого.

Далі програма визначає поріг та перетворює зображення на бінарне за допомогою порогового методу Отцу. Після проведення порогової обробки зображення наступним кроком є створення маски, яка, помножена на вхідне зображення, дозволяє виділити тільки область тканини мозку. Використовуючи фільтр Кенні для виявлення границь, програма за допомогою методу зв'язних компонент виділяє найбільшу область зображення і робить маску, яку поєднує з початковим зображенням для отримання мозку без черепної коробки.

Зображення тканини мозку проходить фільтрацію шуму Гаусса, і за допомогою морфологічних операцій розширення та відкриття виділяє області фону(тканини мозку) та об'єкту (пухлини) для проведення сегментації методом маркерованого водорозділу. Області, що не належать ні до множини фону, ні до множини об'єкту визначаються, як маркери. Вони використовуються для проведення остаточної морфологічної операції – ерозії, що дозволяє виділити пухлину як окремий об'єкт.

Результатом роботи є виведений на екран масив зображень, що відображають поетапну обробку зображення, з фінальним результатом – сегментованим зображенням, де область пухлини позначена іншим кольором. Програма дозволяє обрати користувачу необмежну кількість зображень для проведення сегментації, а також записати результати сегментації у документ Microsoft Word.

2.3.1 Архітектура програмного забезпечення

Діаграма класів визначає типи класів системи та різного роду статичні зв'язки, які існують між ними. На діаграмах класів зображуються також атрибути класів, операції класів та обмеження, що накладаються на зв'язки між класами. Вигляд та інтерпретація діаграми класів суттєво залежить від точки зору (рівня абстракції): класи можуть представляти сутність предметної галузі (у процесі аналізу) або елементи програмної системи (у процесах проектування та реалізації).Широко застосовується не тільки для документування та візуалізації, але також для конструювання за допомогою прямого або зворотного проектування

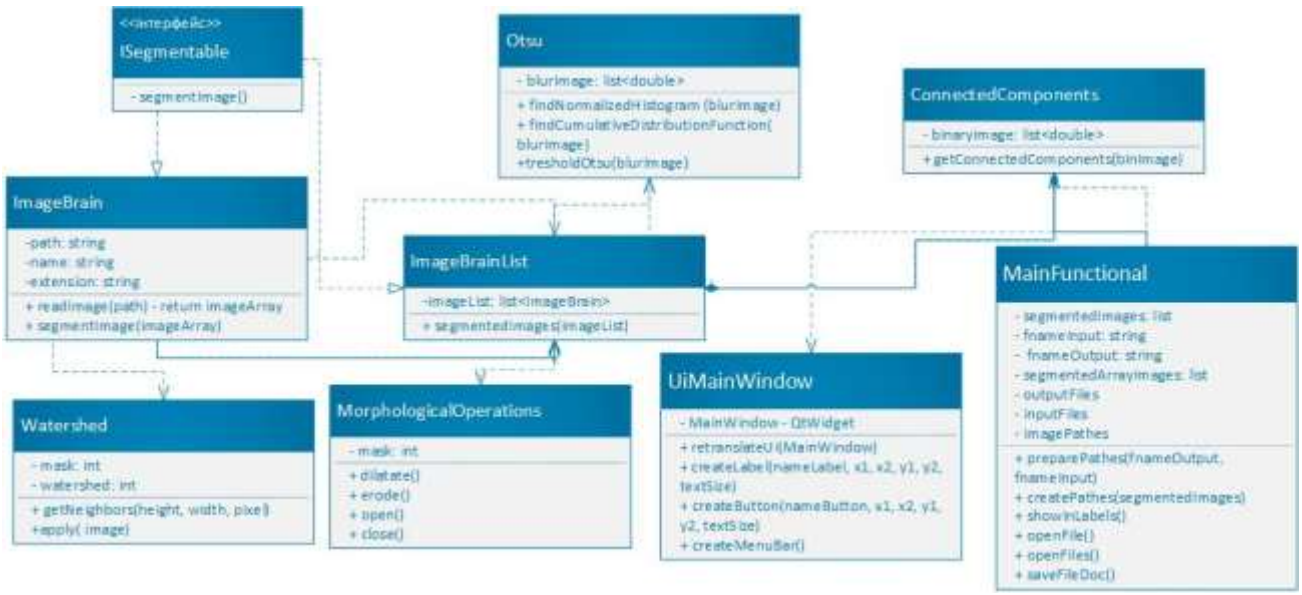


Рисунок 2.13 – Діаграма класів

Діаграма варіантів використання є найзагальнішим уявленням функціональних вимог до системи. Для подальшого проектування системи потрібні більш конкретні деталі, які описуються в документі, який називається сценарієм

варіанта використання або потоком подій . Сценарій докладно документує процес взаємодії дійової особи із системою, що реалізується в рамках варіанта використання. Основний потік подій визначає нормальний перебіг подій (за відсутності помилок). Альтернативні потоки описують відхилення від нормального перебігу подій (помилкові ситуації) та їхню обробку.

В даній системі актором є будь–який користувач, інших ролей не передбачено. А прецедентами є ті функції, що можливо виконати над системою.

Діаграми послідовностей моделюють взаємодії між об'єктами у єдиному сценарії використання. Вони ілюструють, як різні частини системи взаємодіють один з одним для виконання функції, а також порядок, у якому відбувається взаємодія у разі конкретного випадку використання.

Схема послідовності побудована таким чином, що вона є тимчасовою шкалою, яка починається зверху і поступово опускається, щоб відзначити послідовність взаємодій. Кожен об'єкт має стовпчик, а повідомлення, якими обмінюються між собою, представлені стрілками.

Рисунок 2.14 – Діаграма прецедентів

Рисунок 2.15 – Діаграма послідовностей

На данній діаграмі зображена послідовність процесів системи візуального відображення пухлин головного мозку на основі методів сегментації цифрових зображень. Вертикальними лініями позначені модулі обробки інформації та об’єкт «Коритсувач». Модулі відповідають за:

1. «Інтерфес користувача» – відповідає за обробку подій викликаних користувачем (дозволяє обрати зображення через діалогові вікна, а також відображає результати сегментації);
2. «Модуль передобробки зображення» – відповідає за попередню оброюку зображення перед початком сегментації, тобто переведення зображення у тривимірний масив, у градації сірого, та видалення шумів;
3. «Модуль сегментації» – відповідає за виділення пухлини на МРТ зображенні та повертає оброблений масив пікселів;
4. «Модуль результуючої документації» – відповідає за створення таблиці з відсегментованими зображеннями у документі Microsoft Word.

2.3.2 Реалізація програмного забезпечення

На першому етапі розробки було спроектовано інтерфейс користувача. Для реалізації інтерфейсу було обрано фреймворк Qt та середу QtDesigner, інформація

про них зазначена у підрозділі 2.1. QtDesigner був використаний для створення певного макету форми з віджетами у форматі файлу .ui. Щоб відредагувати код та динамічно створювати об'єкти, файл форми з розширенням ui був переформатований за допомогою влаштованого термінала PyCharm та команди: *pyuic5 –x example.ui –o example.py.*

Після отримання доступу до коду форми, було створено деякі функції для оптимізації та більшої читабельності коду. На рис. 2.3. наведено приклад реалізації функції динамічного створення віджетів QLabel.

Для використання функції createLabel() потрібно прописати код, в якому ініціалізувати масив основних параметрів функції, як показано на рис. 2.4.

Таким чином було створено інтерфейсну частину автоматизованої системи. Для того, щоб реалізувати функціонал було створено класи ImageBrain, ImageBrainArray, Otsu, Watershed, ConnectedComponents, Ui_MainWindow, MainFunctional та інтерфейс ISegmentable (див. Додаток Е).

Клас MainFunctional відповідає за функцкіональну частину інтерфейсу, тобто має подієво–орієнтовані функції. Для обробки подій панелі меню та подій при створенні головної форми було створено такі функції: openFile(), openFiles(), setOriginalImage(), viewLabels(), saveFileDoc(), preparePathes(), createPathes(), ShowInLabels().

```
def createLabel(self, nameLabel, x1, x2, y1, y2, textSize ):
    self.label = QtWidgets.QLabel(self.centralwidget)
    self.label.setGeometry(QtCore.QRect(x1, x2, y1, y2))
    font = QtGui.QFont()
    font.setPointSize(textSize)
    font.setUnderline(False)
    self.label.setFont(font)
    self.label.setFrameShape(QtWidgets.QFrame.WinPanel)
    self.label.setTextFormat(QtCore.Qt.AutoText)
    self.label.setScaledContents(True)
    self.label.setAlignment(QtCore.Qt.AlignCenter)
    self.label.setObjectName(nameLabel)
    self.widgets.append(self.label)
```

Рисунок 2.3 – Функція створення екземплярів віджету QLabel

```
self.widgets = []
self.nameLabels = [["img_1", "60", "20", "211", "191","20"],
                   ["img_2", "60", "300", "211", "191","20"],
                   ["img_3", "390", "20", "211", "191","20"]]
for n in range(len(self.nameLabels)):
        self.createLabel(str(self.nameLabels[n][0]), int(self.nameLabels[n][1]), int(self.nameLabels[n][2]),
                         int(self.nameLabels[n][3]), int(self.nameLabels[n][4]), int(self.nameLabels[n][5]))
```

Рисунок 2.4 – Ініціалізація масиву параметрів та виклик функції

```
def preparePathes(self):
    splitPath = self.fnameOutput.split('/')
    nameFileInput = (self.fnameInput.split('/'))
    nameFileArray = nameFileInput[-1].split('.')
    nameFile = nameFileArray[0]
    expFile = nameFileArray[1]
    del splitPath[-1]
    outputPath = "\\".join(splitPath) + "\\"
    return outputPath, nameFile, expFile
```

Рисунок 2.5 – Функція обробки та підготовки шляху місцерозташування файлу

З функції вказаної на рис. 2.5. ми отримуємо шлях для зберігання відсегментованого зображення, ім’я файлу та розширення. Далі ці дані використовуються у функції створення нових шляхів для збереження всіх проміжних зображень сегментації (Рис .2.6.).

```
def createPathes(self):
    outputPath, nameFile, expFile = self.preparePathes()
    imagePathes = list()
    for i in range(len(self.segmentedImages)):
        path =outputPath+nameFile+"_"+str(i)+"."+expFile
        cvToImage(self.segmentedImages[i], path)
        imagePathes.append(path)
    return imagePathes
```

Рисунок 2.6 – Функція створення нових шляхів для збереження файлів

До шляху директорії додається назва оригінального зображення, порядковий номер та розширення. Наприклад оригінальне зображення має назву “brain1.png”, тоді вихідним масивом буде [“brain1_0.png”, “brain1_1.png”, “brain1_2.png”, “brain1_3.png”,“brain1_4.png”], де перший елемент масиву буде містити оригінальне зображення, другий елемент – зображення в градаціях сірого, третій – зображення оброблене пороговим методом Отцу, четвертий – зображення мозку без черепної коробки, п’ятий – результат сегментації.

Розглянемо клас Otsu, що відповідає за знаходження порогів методом Отцу. Основними функціями якого є знаходження нормалізованої гістограми, функції кумулятивного розподілу та саме порогу.

Функція tresholdOtsu(self, blurImage) на рис. 2.8. демонструє реалізацію методу Отцу, формули якого можна побачити у підрозділі 1.2.3. (1.8–1.12).

Наступним, не менш важливим класом, є клас ConnectedComponents, що реалізує маркерування зв’язних компонент. Алгоритм роботи основного методу класу представлений на рис. 2.9.

```
class Otsu:
    def findNormalizedHistogram(self, blurImage):
        hist = cv2.calcHist([blurImage], [0], None, [256], [0, 256])
        hist_norm = hist.ravel() / hist.sum()
        return hist_norm
    def findCumulativeDistributionFunction(self, blurImage):
        hist_norm = self.findNormalizedHistogram(blurImage)
        Q = hist_norm.cumsum()
        return Q
```

Рисунок 2.7 – Шаблон класу Otsu

```
def tresholdOtsu(self, blurImage):
    bins = np.arange(256)
    fn_min = np.inf
    thresh = -1
    hist_norm = self.findNormalizedHistogram(blurImage)
    Q = self.findCumulativeDistributionFunction(blurImage)
    for i in range(1, 256):
        p1, p2 = np.hsplit(hist_norm, [i])  # probabilities
        q1, q2 = Q[i], Q[255] - Q[i]  # cum sum of classes
        if q1 < 1.e-6 or q2 < 1.e-6:
            continue
        b1, b2 = np.hsplit(bins, [i])  # weights
        # finding means and variances
        m1, m2 = np.sum(p1 * b1) / q1, np.sum(p2 * b2) / q2
        v1, v2 = np.sum(((b1 - m1) ** 2) * p1) / q1, np.sum(((b2 - m2) ** 2) * p2) / q2
        # calculates the minimization function
        fn = v1 * q1 + v2 * q2
        if fn < fn_min:
            fn_min = fn
            thresh = i
    return thresh
```

Рисунок 2.8 – Функція знаходження порогу методом Отцу

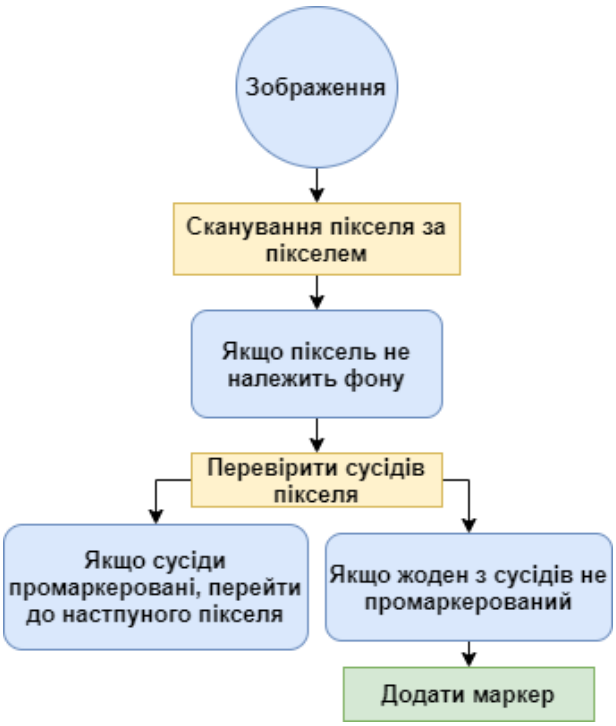


Рисунок 2.9 – Алгоритм роботи методу getConnectedComponents()

```
def getConnectedComponents(bin_image):
    im = Image.open(bin_image)
    (h, w) = im.size
    yc, xc = np.where(bin_image != 0)
    queue = []
    connected_array = np.zeros((h, w))  # позначення масиву
    counter = 1
    for elem in range(len(xc)):
        # перебирати всі ненульові елементи
        i = yc[elem]
        j = xc[elem]
        if connected_array[i, j] == 0:
            # ще не позначені продовжувати
            connected_array[i, j] = counter
            queue.append((i, j))
            while len(queue) != 0:
                # робота через чергу
                current = queue.pop(0)
                i, j = current
```

Рисунок 2.10 – Частина реалізації методу getConnectedComponents()

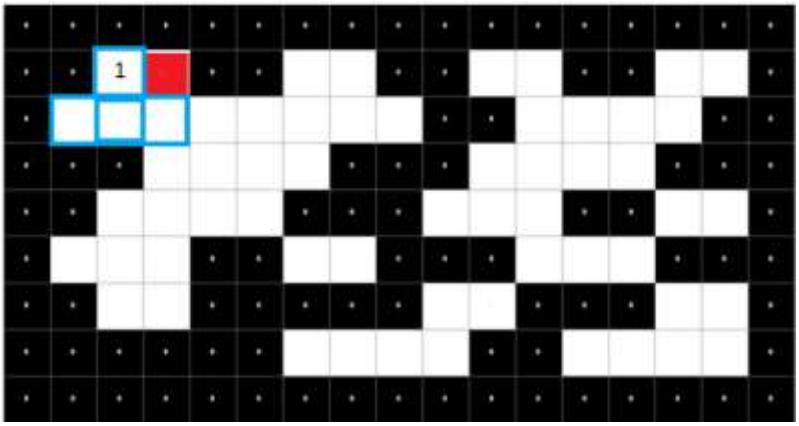


Рисунок 2.11 – Перевірка сусідів пікселя на наявність маркера

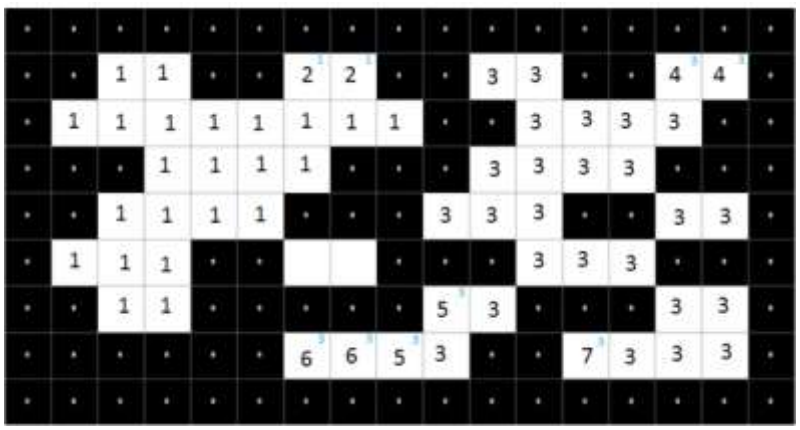


Рисунок 2.12 – Результат пошуку зв’язних компонент

2.3.3 Інструкція користувача

Коротка інструкція:

1. Запустити програмне забезпечення;
2. В меню обрати пункт «Відкрити» та натиснути на підпункт меню «Відкрити файл» або «Відкрити файли»
3. З діалогового вікна обрати файл формату jpg або png;
4. В наступному діалоговому вікні обрати місце для збереження результатів сегментації;
5. Натиснути на кнопку «Відкрити»;
6. Зачекати поки зображення з'являться на екрані.
7. У випадку обраного підпункту меню «Відкрити файли» натиснути на кнопку «Наступне зображення» для відображення результатів сегментації іншого зображення з обраних;
8. При потребі створення результуючої документації обрати в меню пункт «Зберегти»;
9. В діалоговому вікні обрати місце та ім'я файлу для збереження документу;
10. Після цього відкриється файл Microsoft Word, за бажанням його можна відредагувати та зберегти. Нередагований файл автоматично зберігається за обраним користувачем в попередньому пункті шляхом.
11. Користувач має можливість скористатися довідкою обравши у меню пункт «Довідка»;
12. По закінченню роботи в програмі натиснути кнопку «Завершити програму».

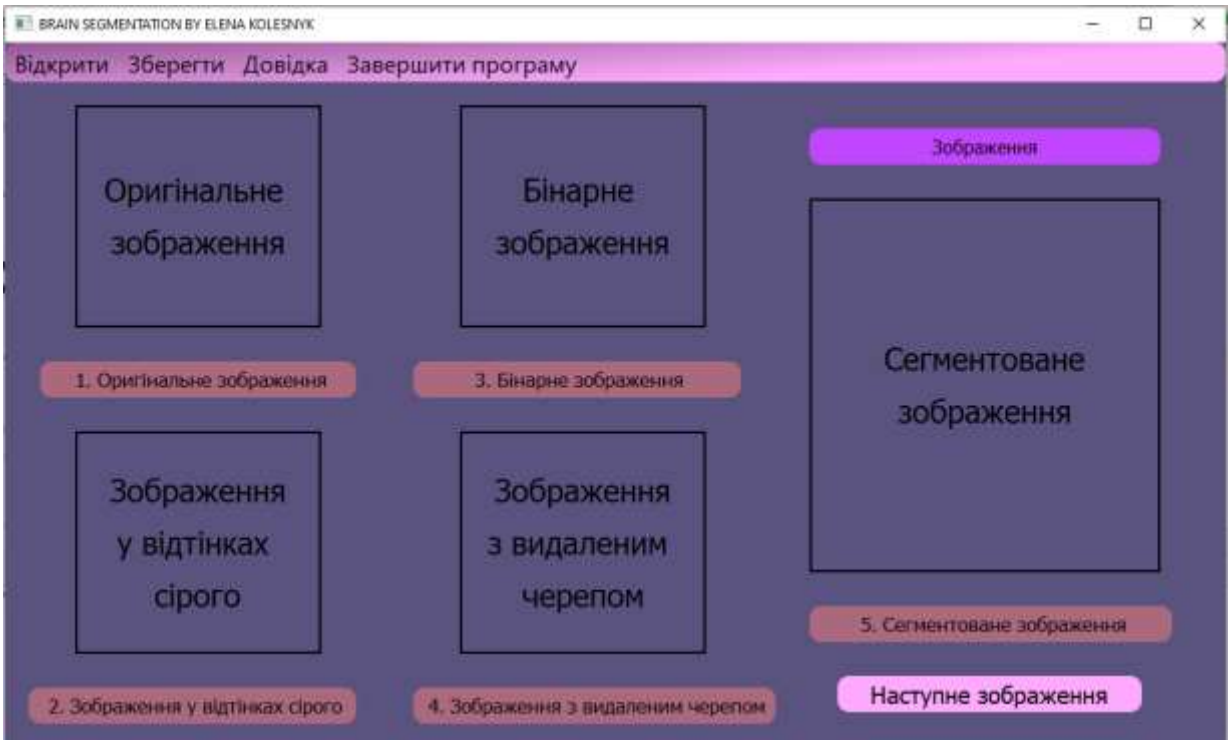


Рисунок 2.16 – Головне вікно системи візуального відображення пухлин головного мозку

Вікно складається з 2 основних областей:

1. Область меню;
2. Область відображення результатів.

Область відображення складається з 5 віджетів для демонстрації зображень, кнопки–перемикача, та віджету для відслідковування порядкового номеру зображення в обраній послідовності. Для початку роботи потрібно в меню обрати пункт «Відкрити файл» для зчитування файлу зображення формату «JPG» або «PNG». У випадку коректного зчитування у області відображення, кожний віджет буде заповнений відповідними даними (перший – оригінальне зображення, другий – зображення у градаціях сірого, третій – бінарне зображення, четвертий – зображення з видаленим черепом, п’ятий віджет – відсегментоване зображення). Зображення у градаціях сірого, бінарне зображення відображаються для показання проміжних етапів сегментації.

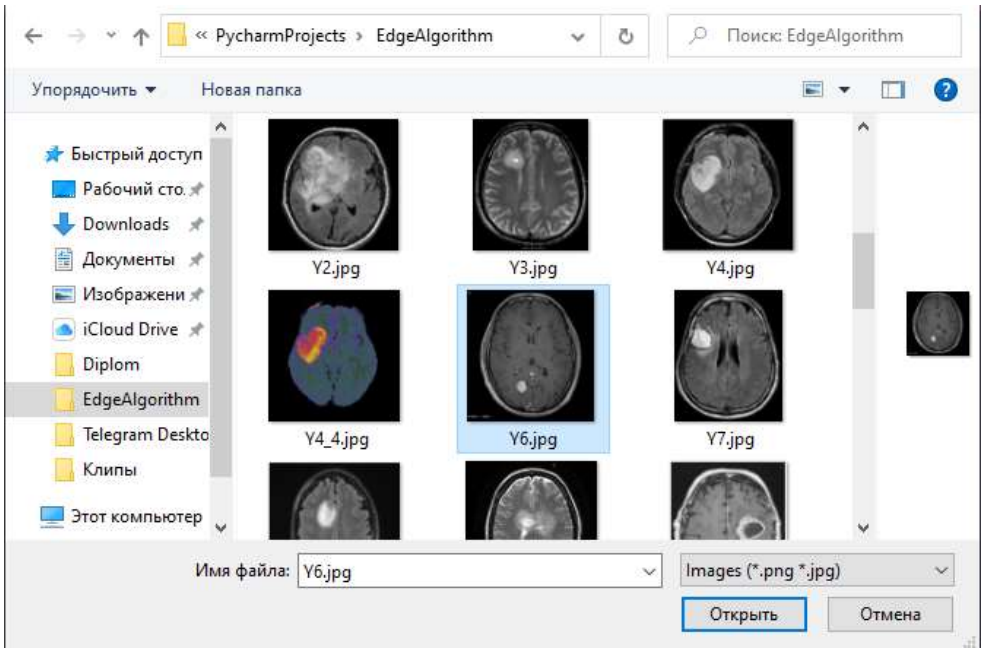


Рисунок 2.17 – Діалогове вікно вибору файлу для сегментації

На рис. 2.18. показані результати обробки вхідного зображення поетапно. Спочатку на основі інтенсивності трьох каналів R, G, B зображення розмивається за допомогою формули визначення інтенсивності пікселя. Зображення під номером 3 показує результат порогової сегментації методом Отцу, наступним кроком є результат поєднання морфологічних операції і методу знаходження зв'язних компонент у вигляді зображення з видаленим черепом. На зображені під номером 5 відображений результат сегментації, пухлина мозку виділена іншим кольором.

Користувач має можливість обрати масив файлі для сегментації. Для цього в меню потрібно обрати «Відкрити файли» та лівою клавішою миші виділити одразу декілька зображень.

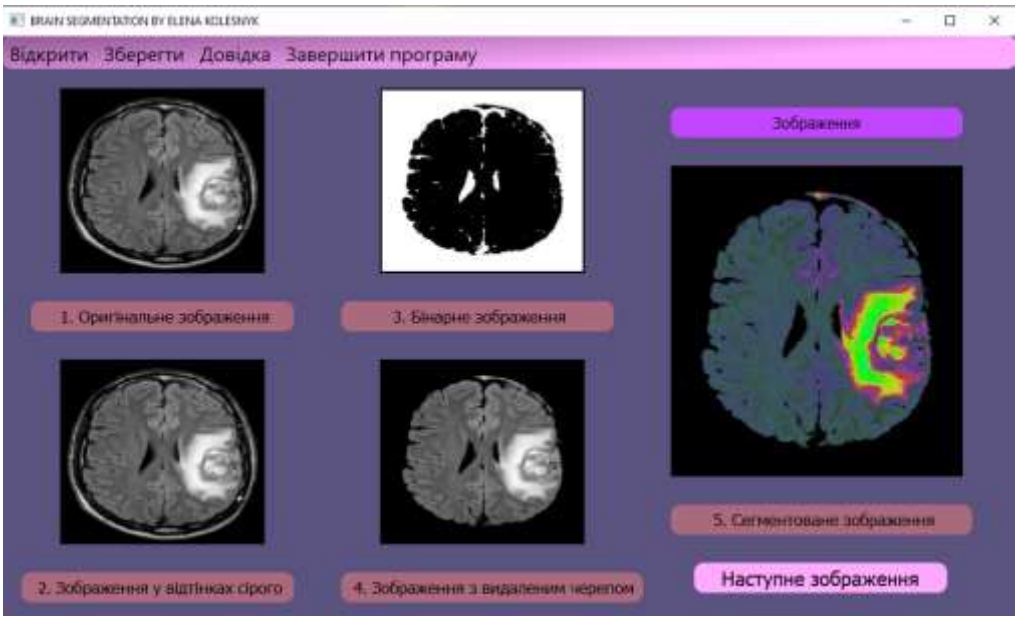


Рисунок 2.18 – Результат обробки МРТ зображення

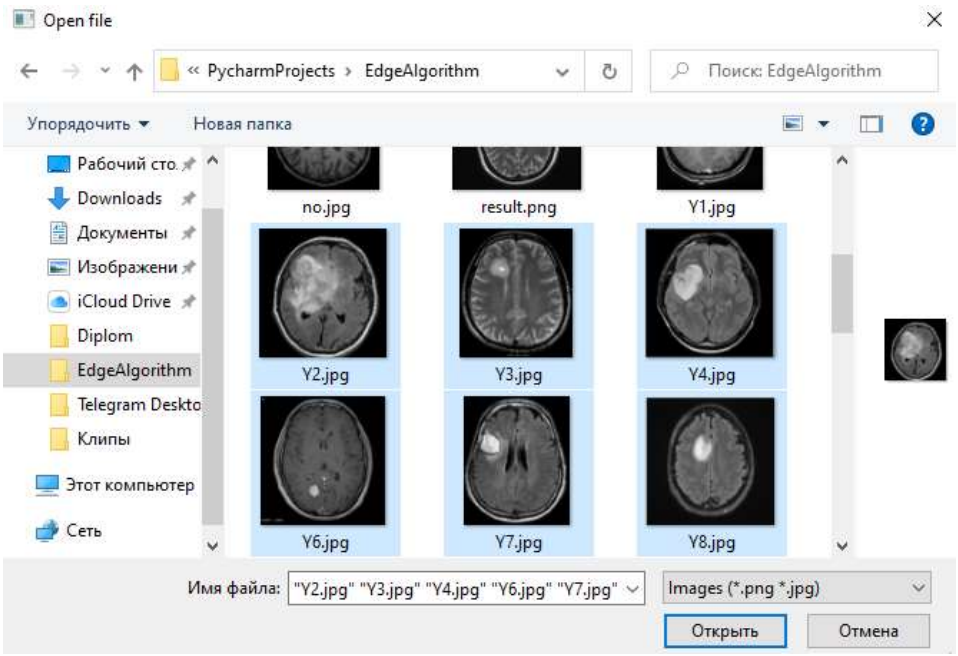


Рисунок 2.19 – Діалогове вікно для вибору декількох файлів

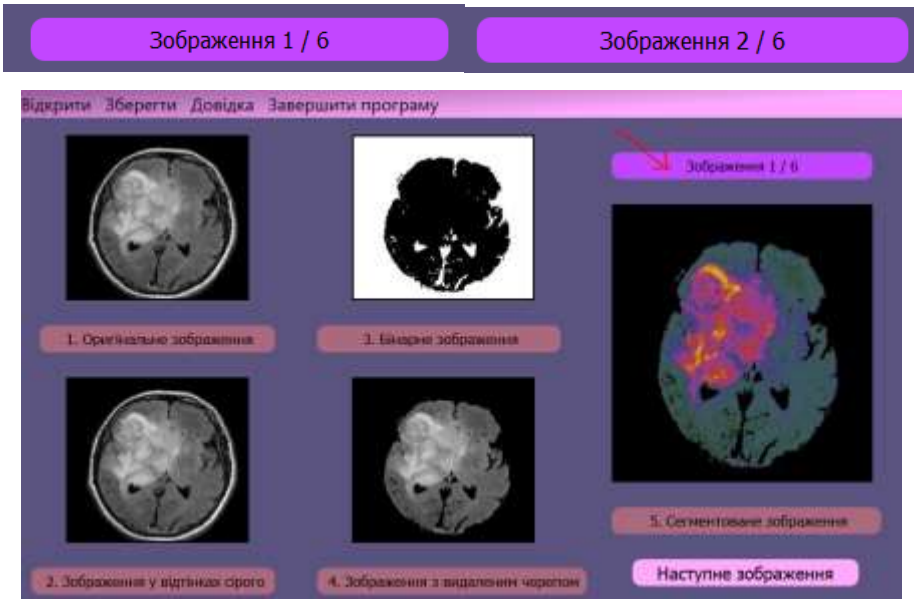


Рисунок 2.20 – Вікно обробки масиву зображень

Таким чином користувач натиснувши кнопку «Наступне зображення» може переключатися від одного зображення до наступного. На віджеті «Зображення» відображається порядковий номер зображення, що переглядається, та кількість завантажених користувачем файлів. Результуючою дією є створення документації у форматі Word документу з розширенням .doc. Для цього користувач має натиснути на панелі меню вкладку «Зберегти» та обрати підпункт «Зберегти у форматі doc». Після відкриття діалогового вікна та вибору місцерозташування майбутнього результуючого документу у системі користувача відкриється документ Microsoft Word (Рис. 2.21.). Після відкриття користувач має змогу відредагувати файл та зберегти нову версію документа.

Рисунок 2.21 – Приклад результуючого документу сегментації декількох зображень

2.4 Порівняння з методом сегментації бібліотеки OpenCv

```
def segment_on_dt(a, img):
    border = cv2.dilate(img, None, iterations=5)
    border = border - cv2.erode(border, None)

    dt = cv2.distanceTransform(img, 2, 3)
    dt = ((dt - dt.min()) / (dt.max() - dt.min()) * 255).astype(numpy.uint8)
    _, dt = cv2.threshold(dt, 180, 255, cv2.THRESH_BINARY)
    lbl, ncc = label(dt)
    lbl = lbl * (255 / (ncc + 1))
    # Заповнення маркерів
    lbl[border == 255] = 255

    lbl = lbl.astype(numpy.int32)
    cv2.watershed(a, lbl)

    lbl[lbl == -1] = 0
    lbl = lbl.astype(numpy.uint8)
    return 255 - lbl
```

Рисунок 2.22 – Приклад реалізації сегментації методами бібліотеки OpenCv

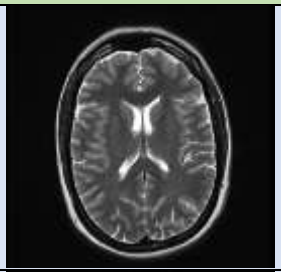
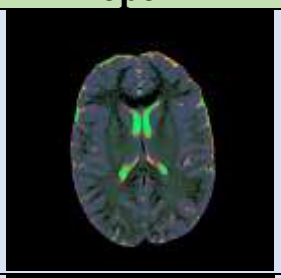
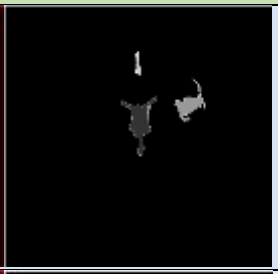
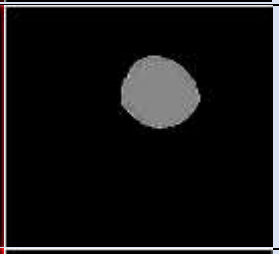
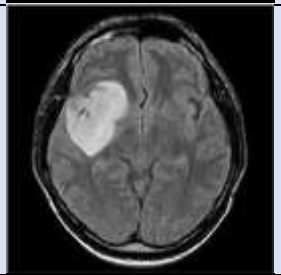
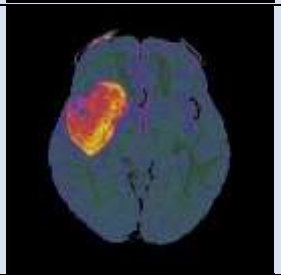
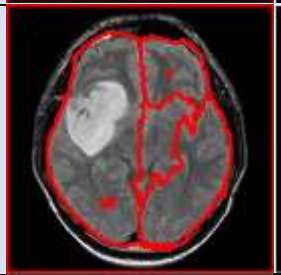
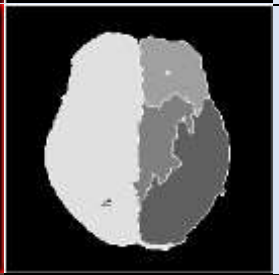
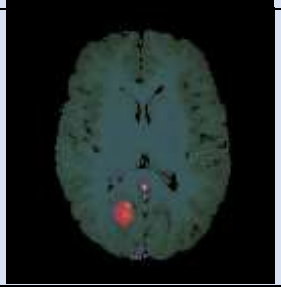
```
if __name__ == '__main__':
    img = cv2.imread('Y6_3.jpg')

    # передобробка
    img_gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    _, img_bin = cv2.threshold(img_gray, 0, 255,
            cv2.THRESH_OTSU)

    img_bin = cv2.morphologyEx(img_bin, cv2.MORPH_OPEN,
            numpy.ones((3, 3), dtype=int))

    result = segment_on_dt(img, img_bin)

    cv2.imwrite('C:\\Users\\Elena\\Downloads\\Y2_new1.jpg', result)

    result[result != 255] = 0
    result = cv2.dilate(result, None)
    img[result == 255] = (0, 0, 255)
    cv2.imwrite('C:\\Users\\Elena\\Downloads\\Y2_new2.jpg', img)
```

Рисунок 2.23 – Приклад передобробки зображення та виклику функції сегментації

1. Cv2.morphologyEx(cv2.MORPH_OPEN) – відкриття, ще одна назва ерозії, за якою слідує розширення. Це корисно для видалення шуму.

2. cv2.dilate() – протилежність ерозії. Елемент пікселя дорівнює «1», якщо хоча б один піксель під ядром дорівнює «1». Таким чином, збільшується біла область на зображенні або збільшується розмір об'єкта переднього плану. Зазвичай в таких випадках, як видалення шуму, за ерозією слідує розширення. Тому що ерозія прибирає білі шуми, але вона також стискає і об'єкт.
3. Cv2.threshold(cv.THRESH_OTSU) – алгоритм знаходить оптимальне порогове значення, яке повертається в якості першого виводу.

Таблиця 2.1 – Результати сегментації різними методами

№	Оригінальне зображення	Сегментоване зображення на основі методу вододілу та порогів	Сегментовані зображення методами з бібліотеки OpenCv	
1				
2				
3				
4				

Аналізуючи результати сегментації обох програм, можна зроботи такий висновок: не дивлячись на обширність бібліотеки OpenCv та легкість використання, сегментація деяких типів зображень видає некоректні результати. У випадку методу, реалізованому у дипломній роботі, комбінація методів порогової сегментації та вододілу в певній послідовності та повторюваності (з різними вхідними масивами) дає кращий результат. Методи бібліотеки OpenCv хоча і створені для обробки зображень, але не повністю пристосовані до сегментації саме МРТ зображень мозку.

Висновки до розділу 2

У даному розділі була представлена реалізація системи візуального відображення пухлин головного мозку. Використавши фреймворк Qt було створено інтерфейс програми, що полегшує та покращує роботу користувача із системою. На даний момент система являє собою цінний продукт, яким може скористуватися будь–яка людина не маючи спеціальної медичної освіти або спеціальної підготовки. Система розроблена таким чином, щоб уникнути максимум помилок при обробці та сегментації зображення, тому всі методи та функції приймають автоматичну кількість ітерацій, а поріг знаходиться без втручання користувача (як це буває з глобальним порогом).

ВИСНОВКИ

1. Проведено аналіз зарубіжної літератури та розглянуто сучасні підходи до сегментації зображення та його попередньої обробки.

2. Розглянуто метод сегментації МРТ зображень головного мозку на основі вододілу та порогів. Визначено основні складові алгоритму методу та комбінації для кращої роботи системи.

3. Розроблено систему візуального відображення пухлин головного мозку.

4. Проведено дослідження залежності результатів сегментації від типу зображення (див. Додаток Д). В результаті аналізу отриманих даних було виявлено, що в залежності від форми черепа, або кута нахилу черепної коробки на МРТ зображенні може змінюватися якість сегментації. Також, чим контрастніше оригінальне зображення (залежить від МРТ апарату) тим більша ймовірність правильної сегментації пухлини. В залежності від кількості об'єктів у масиві зображень змінюється і час сегментації, але методи, використані у роботі, дозволяють сегментувати 50 зображень менше ніж за хвилину, що є одним з найкращих результатів цієї предметної області.

5. Найкращі результати по сегментації пухлин дали групи зображень, черепна коробка яких не перевищує певну ширину та знаходиться на деякій відстані від тканини самого мозку. Метод, що було розглянуто у дипломній роботі не є універсальним, але однозначно може використовуватися для початкового аналізу пухлин головного мозку .

6. Перспективою даної роботи є удосконалення алгоритму видалення черепа, проведення автоматичного аналізу (виду, розміру та стадії) пухлини за допомогою нейронних мереж та автоматизація оцінювання якості сегментації.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Akhtar, N.; Agarwal, N.; Burjwal, A. K-mean algorithm for Image Segmentation using Neutrosophy. In Proceedings of the ICACCI International Conference on Advances in Computing, Communications and Informatics, New Delhi, India, 24–27 September 2014; pp. 2417–2421.

2. Al-Galal SAY, Alshaikhli IFT, Abdulrazzaq MM. MRI brain tumor medical images analysis using deep learning techniques: a systematic review. Health Technol. 2021;11(2):267–282. doi: 10.1007/s12553-020-00514-6.

3. Al-Okaili RN, Krejza J, Woo JH, Wolf RL, O Rourke DM, Judy KD, Melhem ER. Intraaxial brain masses: MR imaging-based diagnostic strategy-initial experience. Radiology. 2007;243(2):539–550. doi: 10.1148/radiol.2432060493.

4. Amin J, Sharif M, Gul N, Yasmin M, Shad SA. Brain tumor classification based on DWT fusion of MRI sequences using convolutional neural network. Pattern Recognit Lett. 2020;129:115–122. doi: 10.1016/j.patrec.2019.11.016.

5. Anila S, Sivaraju SS, Devarajan N. A new contour let based multiresolution approximation for MRI image noise removal. Natl Acad Sci Lett. 2017;40(1):39–41. doi: 10.1007/s40009-016-0498-1.

6. Balafar MA, Ramli AR, Saripan MI, Mashohor S. Review of brain MRI image segmentation methods. Artif Intell Rev. 2010;33(3):261–274.

7. Bansal M, Kumar M, Kumar M, Kumar K. An efficient technique for object recognition using Shi-Tomasi corner detection algorithm. Soft Comput. 2021;25(6):4423–4432. doi: 10.1007/s00500-020-05453-y.

8. Bisht A, Kumar A (2019) DWT chip design and FPGA synthesis for image processing. Int J Recent Technol Eng (IJRTE) 8:1–10

9. Chaudhary A, Bhattacharjee V. An efficient method for brain tumor detection and categorization using MRI images by K-means clustering & DWT. Int J Inform Technol. 2020;12(1):141–148. doi: 10.1007/s41870-018-0255-4.

10. Cheng, H.; Guo,Y.; Zhang, Y. A novel image segmentation approach based on neutrosophic set and improved fuzzy c-means algorithm. New Math. Nat. Comput. 2011, 7, 155–171.

11. Dargan S, Kumar M, Ayyagari MR, Kumar G. A survey of deep learning and its applications: a new paradigm to machine learning. Arch Comput Methods Eng. 2019;27(4):1071–1092. doi: 10.1007/s11831-019-09344-w.

12. Dargan S, Kumar M. A comprehensive survey on the biometric recognition systems based on physiological and behavioral modalities. Expert Syst Appl. 2020;143:113114. doi: 10.1016/j.eswa.2019.113114.

13. Davis FG, Malmer BS, Aldape K, Barnholtz-Sloan JS, Bondy ML, Brännström T, Kruchko C (2008) Issues of diagnostic review in brain tumor studies: from the Brain tumor epidemiology consortium. Cancer Epidemiol Prev Biomarkers 17(3):484–489

14. Gholipour A, Kehtarnavaz N, Briggs R, Devous M, Gopinath K. Brain functional localization: a survey of image registration techniques. IEEE Trans Med Imaging. 2007;26(4):427–451. doi: 10.1109/TMI.2007.892508.

15. Ghosh KK, Begum S, Sardar A, Adhikary S, Ghosh M, Kumar M, Sarkar R (2021) Theoretical and empirical analysis of filter ranking methods: experimental study on benchmark DNA microarray data. Expert Syst Appl 169:114485

16. Goel A, Chikara D, Srivastava AK, Kumar A (2016) Medical imaging with brain tumor detection and analysis. Int J Comput Sci Inform Secur 14(9):228

17. Gonzalez, R.C. Digital Image Processing, 2nd ed.; Prentice Hall: Upper Saddle River, NJ, USA, 2002.

18. Guo,Y.; Cheng, H.-D. New neutrosophic approach to image segmentation. PatternRecognit. 2009,42,587–595.

19. Gupta S, Thakur K, Kumar M. 2D-human face recognition using SIFT and SURF descriptors of face's feature regions. Visual Comput. 2020;37(3):447–456. doi: 10.1007/s00371-020-01814-8.

20. Hanbay, K.; Talu, M.F. Segmentation of SAR images using improved artificial bee colony algorithm and neutrosophic set. Appl. Soft Comput. 2014, 21, 433–443.

21. Iqbal S, Ghani MU, Saba T, Rehman A. Brain tumor segmentation in multi-spectral MRI using convolutional neural networks (CNN) Microsc Res Tech. 2018;81(4):419–427. doi: 10.1002/jemt.22994.

22. Joseph RP, Singh CS, Manikandan M. Brain tumor MRI image segmentation and detection in image processing. Int J Res Eng Technol. 2014;3(1):1–5.

23. Karabatak, E.; Guo, Y.; Sengur, A. Modified neutrosophic approach to color image segmentation. J. Electron. Imaging 2013, 22, 013005.

24. Kaur P, Kumar R, Kumar M. A healthcare monitoring system using random forest and internet of things (IoT) Multimedia Tools Appl. 2019;78(14):19905–19916. doi: 10.1007/s11042-019-7327-8.

25. Khode KMR, Salwe SR, Bagade AP, Raut RD. Efficient brain tumor detection using wavelet transform. Int J Eng Res Appl. 2017;7:55–60.

26. Kumar A, Kumar M, Kaur A. Face detection in still images under occlusion and non-uniform illumination. Multimedia Tools Appl. 2021;80(10):14565–14590. doi: 10.1007/s11042-020-10457-9.

27. Kumar A, Rastogi P, Srivastava P. Design and FPGA implementation of DWT, image text extraction technique. Procedia Comput Sci. 2015;57:1015–1025. doi: 10.1016/j.procs.2015.07.512.

28. Kumar M, Gupta S, Kumar K, Sachdeva M. Spreading of COVID-19 in India, Italy, Japan, Spain, UK, US: a prediction using ARIMA and LSTM model. Digit Government: Res Pract. 2020;1(4):1–9. doi: 10.1145/3411760.

29. Li BN, Chui CK, Chang S, Ong SH. A new unified level set method for semi-automatic liver tumor segmentation on contrast-enhanced CT images. Expert Syst Appl. 2012;39(10):9661–9668. doi: 10.1016/j.eswa.2012.02.095.

30. Liao X, Li K, Yin J. Separable data hiding in encrypted image based on compressive sensing and discrete Fourier transform. Multimedia Tools Appl. 2017;76(20):20739–20753. doi: 10.1007/s11042-016-3971-4.

31. Liao X, Shu C. Reversible data hiding in encrypted images based on absolute mean difference of multiple neighboring pixels. J Vis Commun Image Represent. 2015;28:21–27. doi: 10.1016/j.jvcir.2014.12.007.

32. Liao X, Yin J, Guo S, Li X, Sangaiah AK. Medical JPEG image steganography based on preserving inter-block dependencies. Comput Electr Eng. 2018;67:320–329. doi: 10.1016/j.compeleceng.2017.08.020.

33. Moeskops P, Benders MJ, Chiţă SM, Kersbergen KJ, Groenendaal F, de Vries LS, Išgum I. Automatic segmentation of MR brain images of preterm infants using supervised classification. NeuroImage. 2015;118:628–641.

34. Moeskops P, Viergever MA, Mendrik AM, De Vries LS, Benders MJ, Išgum I. Automatic segmentation of MR brain images with a convolutional neural network. IEEE Trans Med Imaging. 2016;35(5):1252–1261..

35. Mustaqeem A, Javed A, Fatima T. An efficient brain tumor detection algorithm using watershed & thresholding based segmentation. Int J Image Graphics Signal Process. 2012;4(10):34. doi: 10.5815/ijigsp.2012.10.05.

36. Nazir M, Shakil S, Khurshid K (2021) Role of deep learning in brain tumor detection and classification (2015 to 2020): a review. Comput Med Imaging Graph 91:101940

37. Olszewska JI (2019) Designing transparent and autonomous intelligent vision systems. In: ICAART 2, pp 850–856

38. Olszewska JI, Houghtaling M, Goncalves PJ, Fabiano N, Haidegger T, Carbonera JL, Prestes E. Robotic standard development life cycle in action. J Intell Robotic Syst. 2020;98(1):119–131. doi: 10.1007/s10846-019-01107-w.

39. Pal, N.R.; Pal, S.K. A review on image segmentation techniques. Pattern Recognit. 1993, 26, 1277–1294.

40. Pardakhti N, Sajedi H. Brain age estimation based on 3D MRI images using 3D convolutional neural network. Multimedia Tools Appl. 2020;79(33):25051–25065. doi: 10.1007/s11042-020-09121-z.

41. Patil RC, Bhalchandra AS. Brain tumour extraction from MRI images using MATLAB. Int J Electron Communication Soft Comput Sci Eng. 2012;2(1):1–4.

42. Pereira S, Pinto A, Alves V, Silva CA. Brain tumor segmentation using convolutional neural networks in MRI images. IEEE Trans Med Imaging. 2016;35(5):1240–1251. doi: 10.1109/TMI.2016.2538465.

43. Ratan R, Sharma S, Sharma SK. Brain tumor detection based on multi-parameter MRI image analysis. ICGST-GVIP J. 2009;9(3):9–17.

44. Remya R, Parimala GK, Sundaravadivelu S (2019) Enhanced DWT Filtering technique for brain tumor detection. IETE J Res: 1–10

45. Roy S, Bandyopadhyay SK (2012) Detection and quantification of brain tumor from MRI of brain and it's symmetric analysis. Int J Inform Communication Technol Res 2(6):477–483

46. Sain PK, Singh M. Brain tumor detection in medical imaging using MATLAB. Int Res J Eng Technol. 2015;2(2):191–196.

47. Seetha J, Raja SS. Brain tumor classification using convolutional neural networks. Biomed Pharmacol J. 2018;11(3):1457. doi: 10.13005/bpj/1511.

48. Shree NV, Kumar TNR. Identification and classification of brain tumor MRI images with feature extraction using DWT and probabilistic neural network. Brain Inf. 2018;5(1):23–30. doi: 10.1007/s40708-017-0075-5.

49. Singh AK, Dave M, Mohan A. Hybrid technique for robust and imperceptible image watermarking in DWT–DCT–SVD domain. Natl Acad Sci Lett. 2014;37(4):351–358. doi: 10.1007/s40009-014-0241-8.

50. Smarandache, F. A Unifying Field in Logics Neutrosophic Logic. In Neutrosophy, Neutrosophic Set, Neutrosophic Probability; American Research Press: Rehoboth, DE, USA, 2005.

51. Smarandache, F. Neutrosophy. Neutrosophic Probability, Set, and Logic, ProQuest Information & Learning; Infolearnquest: Ann Arbor, MI, USA, 1998.

52. Thapaliya K, Pyun JY, Park CS, Kwon GR. Level set method with automatic selective local statistics for brain tumor segmentation in MR images. Comput Med Imaging Graph. 2013;37(7–8):522–537.

53. Viji KA, Jayakumari J (2011) Automatic detection of brain tumor based on magnetic resonance image using CAD system with watershed segmentation. In: 2011

international conference on signal processing, communication, computing and networking technologies, pp 145–150

54. Wang H, Roa AC, Basavanhally AN, Gilmore HL, Shih N, Feldman M, Madabhushi A. Mitosis detection in breast cancer pathology images by combining handcrafted and convolutional neural network features. J Med Imaging. 2014;1(3):03400. doi: 10.1117/1.JMI.1.3.034003.

55. Wu,Z.; Leahy, R. An optimal graph theoretic approach to data clustering: Theory and its application to image segmentation. IEEE. Trans. Pattern Anal. Mach. Intell. 1993, 15, 1101–1113.

56. Zhang C, Shen X, Cheng H, Qian Q. Brain tumor segmentation based on hybrid clustering and morphological operations. Int J Biomed Imaging. 2019;2019:1–10. doi: 10.1155/2019/7305832.

57. Zhang, M.; Zhang, L.; Cheng, H. A neutrosophic approach to image segmentation based on watershed method. Signal Process. 2010, 90, 1510–1517.

58. Zikic D, Ioannou Y, Brown M, Criminisi A (2014) Segmentation of brain tumor tissues with convolutional neural networks. Proceedings MICCAI-BRATS, pp 36–39.

Додаток А Приклад реалізації функціональної частини інтерфейсу користувача

```
class MainFunctional(object):

# sgmIm – масив cv2 результатів, fIn – шлях до обраного файла, fOut – шлях для збереження
global segmentedImages, fnameInput, fnameOutput

# sgmArrImg – масив cv2 результатів, OutF – шлях для збереження,
# InF – масив шляхів до обраних зображень, imgPth – масив шляхів відсегментованих зображень
global segmentedArrayImages, outputFiles, inputFiles, imagePathes

     def createLabel(self, nameLabel, x1, x2, y1, y2, textSize ):
         self.label = QtWidgets.QLabel(self.centralwidget)
         self.label.setGeometry(QtCore.QRect(x1, x2, y1, y2))
         font = QtGui.QFont()
         font.setPointSize(textSize)
         font.setUnderline(False)
         self.label.setFont(font)
         self.label.setFrameShape(QtWidgets.QFrame.WinPanel)
         self.label.setTextFormat(QtCore.Qt.AutoText)
         self.label.setScaledContents(True)
         self.label.setAlignment(QtCore.Qt.AlignCenter)
         self.label.setObjectName(nameLabel)
         self.widgets.append(self.label)

     def setup(self, MainWindow):
         MainWindow.setObjectName("MainWindow")
         MainWindow.resize(1047, 600)
         MainWindow.setStyleSheet("background-color: rgb(90, 83, 127);")
         self.centralwidget = QtWidgets.QWidget(MainWindow)
         self.centralwidget.setObjectName("centralwidget")

         self.widgets = []
         self.nameLabels = [["img_1", "60", "20", "211", "191","20"],
                            ["img_2", "60", "300", "211", "191","20"],
                            ["img_3", "390", "20", "211", "191","20"]]
         for n in range(len(self.nameLabels)):
                 self.createLabel(str(self.nameLabels[n][0]),
                  int(self.nameLabels[n][1]), int(self.nameLabels[n][2]),
                   int(self.nameLabels[n][3]), int(self.nameLabels[n][4]),
                   int(self.nameLabels[n][5]))

          self.retranslateUi(MainWindow)
          QtCore.QMetaObject.connectSlotsByName(MainWindow)

     def preparePathes(self):
         splitPath = self.fnameOutput.split('/')
         nameFileInput = (self.fnameInput.split('/'))
         nameFileArray = nameFileInput[-1].split('.')
         nameFile = nameFileArray[0]
         expFile = nameFileArray[1]
         del splitPath[-1]
         outputPath = "\\".join(splitPath) + "\\"
         return outputPath, nameFile, expFile

     def createPathes(self):
         outputPath, nameFile, expFile = self.preparePathes()
         imagePathes = list()
         for i in range(len(self.segmentedImages)):
```

```
            path =outputPath+nameFile+"_"+str(i)+"."+expFile
            cvToImage(self.segmentedImages[i], path)
            imagePathes.append(path)
        return imagePathes

    def createPathesForArray(self):
        pathFileNames = []
        imagePathes = [[0] * 5 for i in range((len(self.segmentedArrayImages)))]
        pathes = []

        for j in range(len(self.inputFiles)):
            outputPath, nameFile, expFile =
    self.preparePathesToArray(r""+self.inputFiles[j])
            pathFileNames.append([outputPath, nameFile, expFile])
        for i in range(len(self.segmentedArrayImages)):
            for y in range(len(self.segmentedArrayImages[i])):
                path = pathFileNames[i][0] + pathFileNames[i][1] + "_" +
    str(y) + "." + pathFileNames[i][2]
                cvToImage(self.segmentedArrayImages[i][y], path)
                imagePathes[i][y] = path

        return imagePathes

    def ShowInLabels(self):
        imagePathes = self.createPathes()
        for i in range(len(imagePathes)):
            self.setOriginalImage(imagePathes[i], i)

    def saveFileDoc(self):
        resultDoc = ResultDoc()

        options = QFileDialog.Options()
        fnameOutputDoc, filt = QtWidgets.QFileDialog.getSaveFileName(self, 'Save
    file', '/desktop', "*.doc", options=options)
        resultDoc.createDoc(self.imagePathes,  r""+fnameOutputDoc)
        os.startfile(r""+fnameOutputDoc)
    def setOriginalImage(self, image, numLabel):
        if numLabel == 0:
            self.label.setPixmap(QtGui.QPixmap(image))
        elif numLabel == 1:
            self.label_2.setPixmap(QtGui.QPixmap(image))
        elif numLabel == 2:
            self.label_3.setPixmap(QtGui.QPixmap(image))
        elif numLabel == 3:
            self.label_4.setPixmap(QtGui.QPixmap(image))
        else:
            self.label_9.setPixmap(QtGui.QPixmap(image))

    def viewLabels(self):

        if self.counter<len(self.imagePathes):

            for y in range(5):
                self.label_11.setText("Зображення " + str(self.counter+1) + "  /
    " + str(len(self.imagePathes)))
                self.setOriginalImage(self.imagePathes[self.counter][y], y)
            self.counter += 1
```

Додаток Б Приклад реалізації класу для знаходження зв'язних компонент

```
class ConnectedComponents:
    def getConnectedComponents(bin_image):
        im = Image.open(bin_image)
        (h, w) = im.size
        yc, xc = np.where(bin_image != 0)
        queue = []
        connected_array = np.zeros((h, w))  # позначення масиву
        counter = 1
        for elem in range(len(xc)):
            # перебирати всі ненульові елементи
            i = yc[elem]
            j = xc[elem]
            if connected_array[i, j] == 0:
                # ще не позначені продовжувати
                connected_array[i, j] = counter
                queue.append((i, j))
                while len(queue) != 0:
                    # робота через чергу
                    current = queue.pop(0)
                    i, j = current
                    if i == 0 and j == 0:
                        coords = np.array([[i, i + 1], [j + 1, j]])
                    elif i == h - 1 and j == w - 1:
                        coords = np.array([[i, i - 1], [j - 1, j]])
                    elif i == 0 and j == w - 1:
                        coords = np.array([[i, i + 1], [j - 1, j]])
                    elif i == h - 1 and j == 0:
                        coords = np.array([[i, i - 1], [j + 1, j]])
                    elif i == 0:
                        coords = np.array([[i, i, i + 1], [j - 1, j + 1, j]])
                    elif i == h - 1:
                        coords = np.array([[i, i, i - 1], [j - 1, j + 1, j]])
                    elif j == 0:
                        coords = np.array([[i, i + 1, i - 1], [j + 1, j, j]])
                    elif j == w - 1:
                        coords = np.array([[i, i + 1, i - 1], [j - 1, j, j]])
                    else:
                     coords = np.array([[i, i, i + 1, i - 1], [j - 1, j + 1, j, j]])

                    for k in range(len(coords[0])):
                        # перебирати пікселі сусідів,
                        # якщо вони не позначені та не дорівнюють нулю,
                        # тоді присвоюється поточна мітка
                        if connected_array[coords[0, k], coords[1, k]] == 0 and
                              bin_image
                              coords[0, k], coords[1, k]] != 0:
                            connected_array[coords[0, k], coords[1, k]] =
                              counter
                              queue.append((coords[0, k], coords[1, k]))
                              counter += 1

        return connected_array, counter - 1
```

Додаток В Приклад реалізації класу знаходження порога методом Отцу

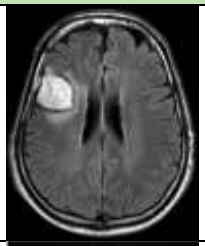
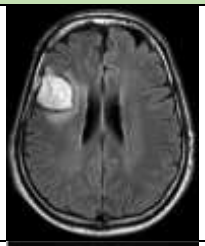
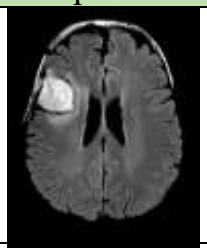
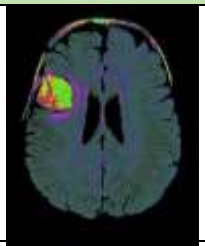
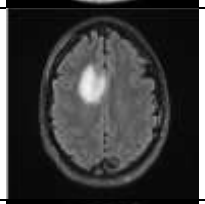
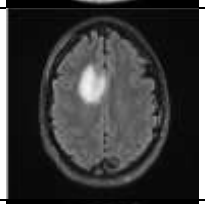
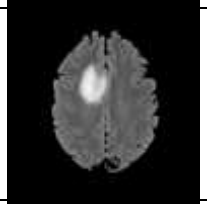
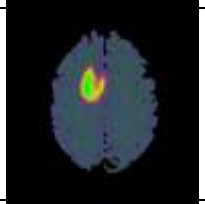
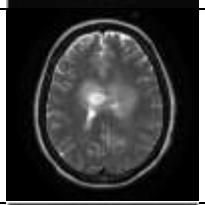
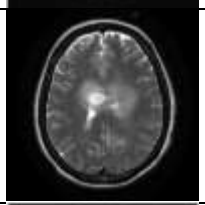
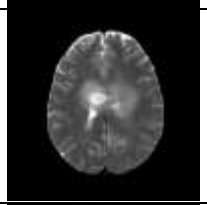
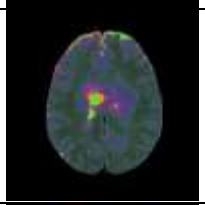
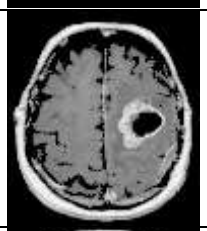
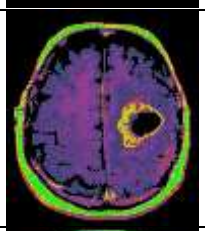
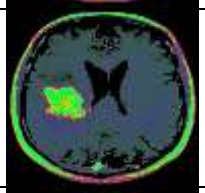
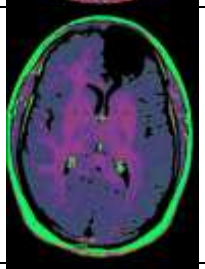
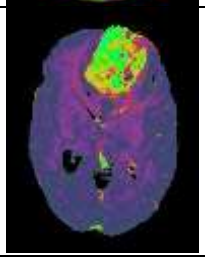
```
class Otsu:
    def findNormalizedHistogram(self, blurImage):
        hist = cv2.calcHist([blurImage], [0], None, [256], [0, 256])
        hist_norm = hist.ravel() / hist.sum()
        return hist_norm
    def findCumulativeDistributionFunction(self, blurImage):
        hist_norm = self.findNormalizedHistogram(blurImage)
        Q = hist_norm.cumsum()
        return Q
    def tresholdOtsu(self, blurImage):
        bins = np.arange(256)
        fn_min = np.inf
        thresh = -1
        hist_norm = self.findNormalizedHistogram(blurImage)
        Q = self.findCumulativeDistributionFunction(blurImage)
        for i in range(1, 256):
            p1, p2 = np.hsplit(hist_norm, [i])  # ймовірності
            q1, q2 = Q[i], Q[255] - Q[i]  # совокупна сумма класів
            if q1 < 1.e-6 or q2 < 1.e-6:
                continue
            b1, b2 = np.hsplit(bins, [i])  # ваги
            # пошук середніх та відхилень
            m1, m2 = np.sum(p1 * b1) / q1, np.sum(p2 * b2) / q2
            v1, v2 = np.sum(((b1 - m1) ** 2) * p1) / q1, np.sum(((b2 - m2) ** 2)
                * p2) / q2
            # обчислення функцію мінімізації
            fn = v1 * q1 + v2 * q2
            if fn < fn_min:
                fn_min = fn
                thresh = i
        return thresh
```

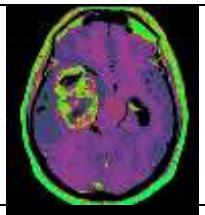
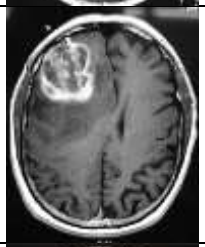
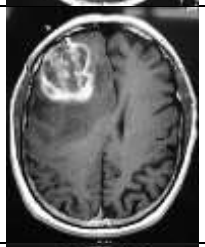
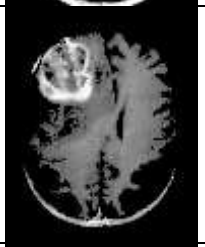
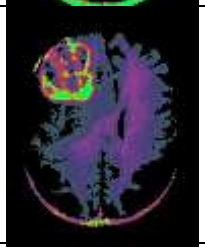
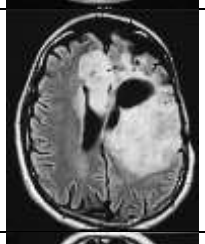
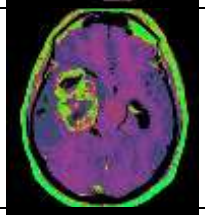
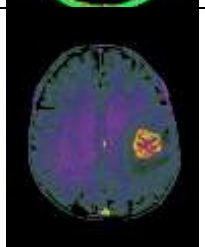
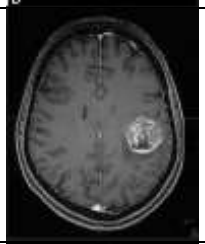
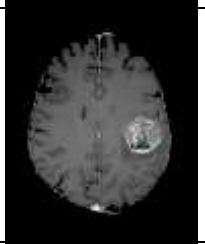
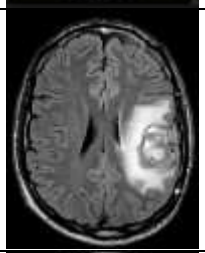
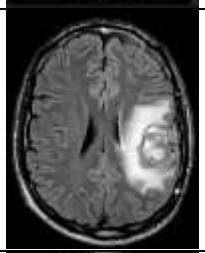
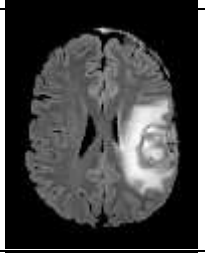
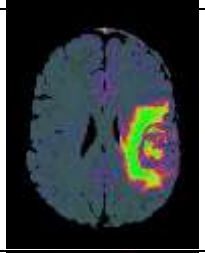
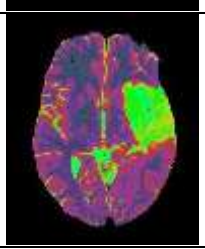
Додаток Г Приклад реалізації створення результуючого документа

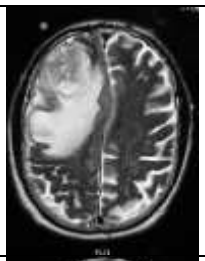
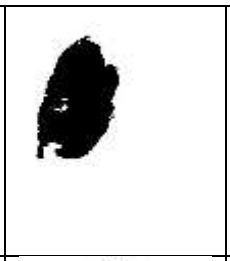
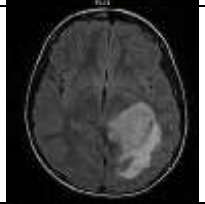
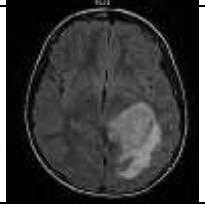
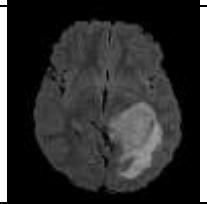
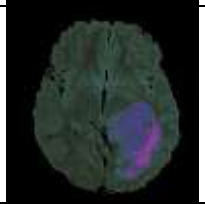
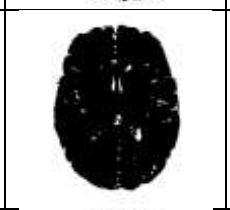
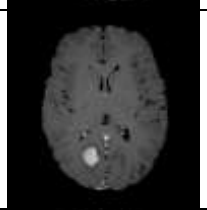
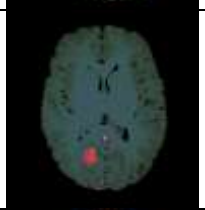
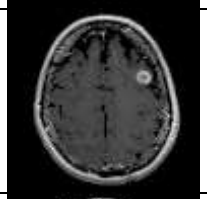
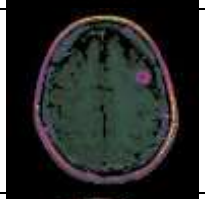
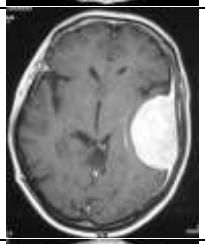
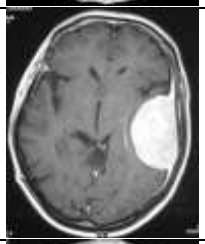
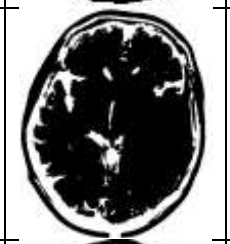
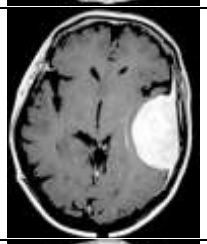
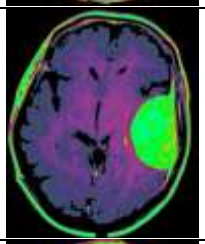
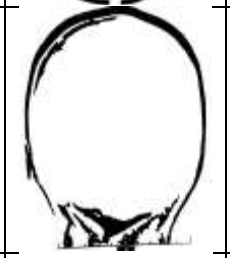
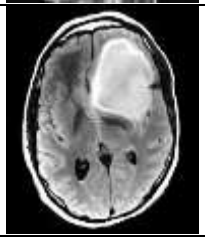
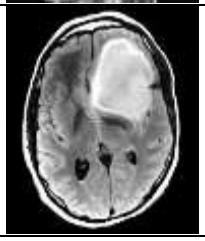
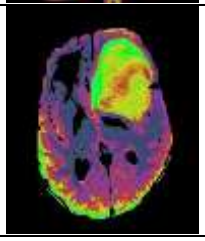
```
class ResultDoc:

    def cvToImage(self, cvArray):
        img = Image.fromarray(cvArray)
        return img
    def createDoc(self, images,  path):
        doc = Document()
        path1 = r'C:/Users/Elena/Downloads'
        os.chdir(path1)
        p = os.getcwd()
        head = doc.add_heading()
        head.paragraph_format.alignment = WD_PARAGRAPH_ALIGNMENT.CENTER
        run1 = head.add_run('Результати сегментації', 0)
        run1.font.size = Pt(30)

        numberImages = len(images)
        table = doc.add_table(rows=numberImages+1, cols=6,style="Table Grid")

        for row in range(numberImages+1):
            cells = table.rows[row].cells
            for col in range(6):
                if row == 0 and col==0:
                    cells[col].text = '№'
                if row == 0 and col == 1:
                    cells[col].text = 'Оригінальне зображення'
                if row == 0 and col == 2:
                    cells[col].text = 'Зображення у відтінках сірого'
                if row == 0 and col == 3:
                    cells[col].text = 'Бінарне зображення'
                if row == 0 and col == 4:
                    cells[col].text = 'Зображення з видаленим черепом'
                if row == 0 and col == 5:
                    cells[col].text = 'Сегментоване зображення'
                if row!=0 and col==0:
                    p = cells[col].paragraphs[0]
                    r = p.add_run()
                    r.text = str(row)
                if row!=0 and col!=0:
                    p = cells[col].paragraphs[0]
                    r = p.add_run()

                    count = r""+images[row-1][col-1]
                    r.add_picture(r""+images[row-1][col-1], width=Inches(1))

        self.changeFont(table)
        doc.save(path)


    def changeFont(self, table):
        for row in table.rows:
            for cell in row.cells:
                paragraphs = cell.paragraphs
                for paragraph in paragraphs:
                    for run in paragraph.runs:
                        font = run.font
                        font.size = Pt(10)
```

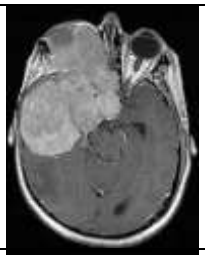
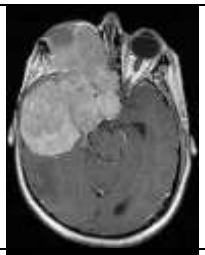
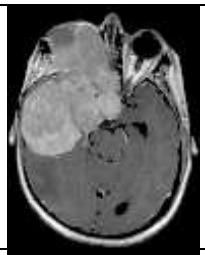
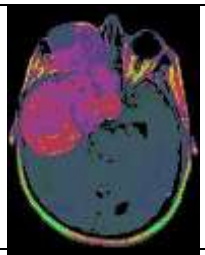
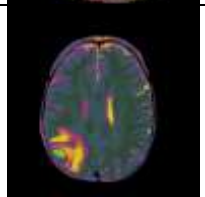
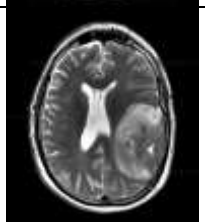
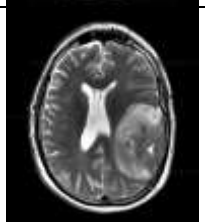
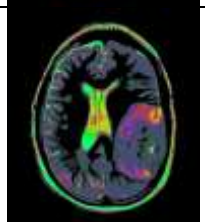
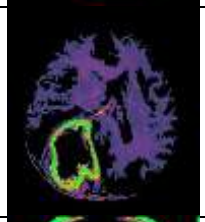
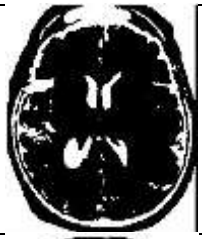
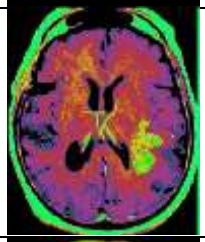
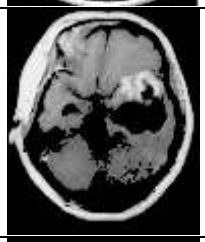
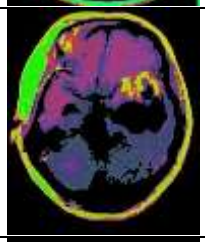
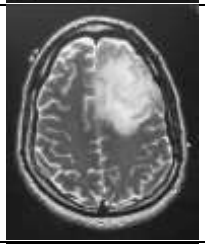
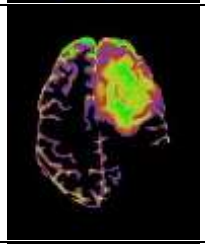
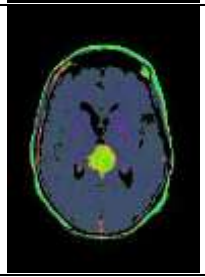
Додаток Д Результати сегментації

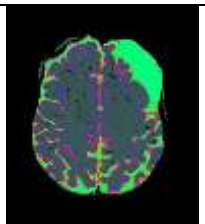
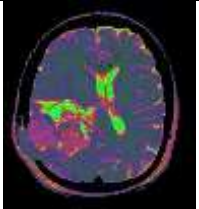
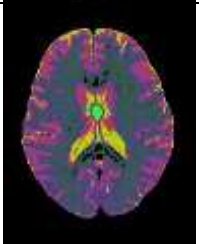
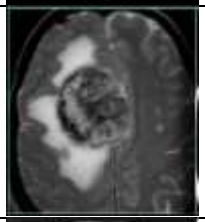
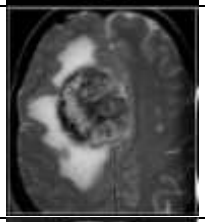
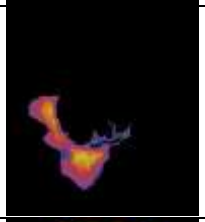
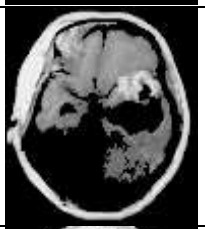
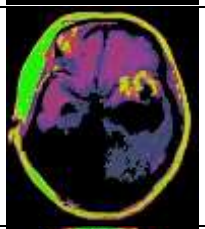
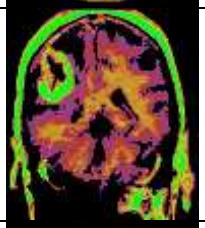
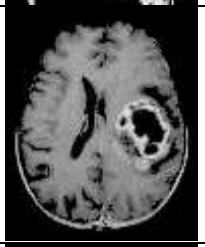
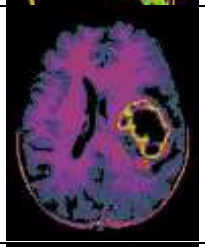
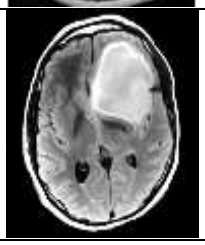
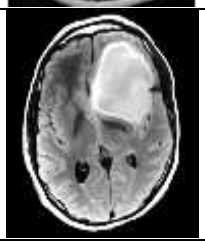
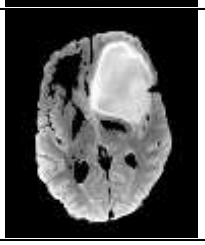
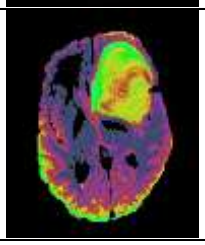
Таблиця Д.1 – Результуючий документ

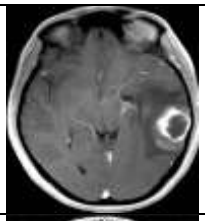
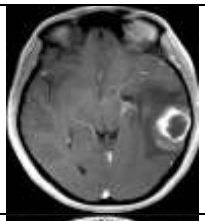
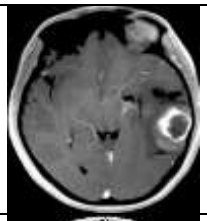
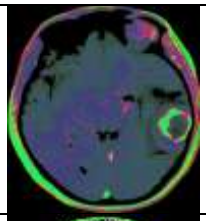
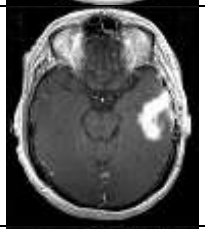
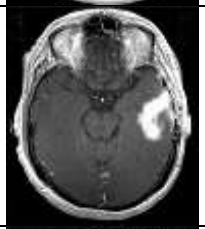
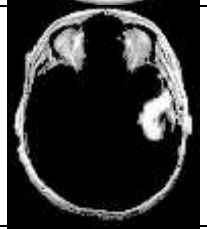
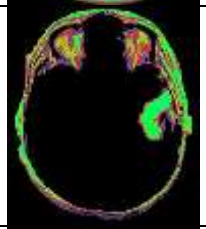
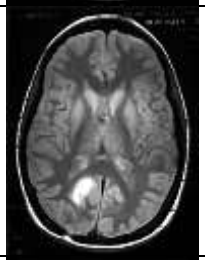
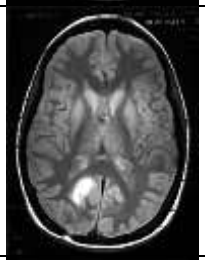
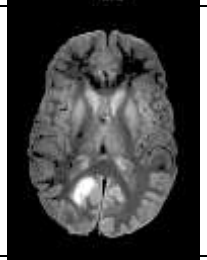
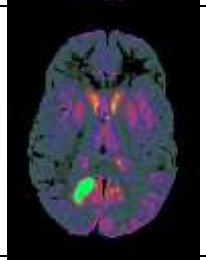
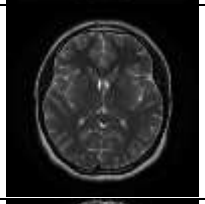
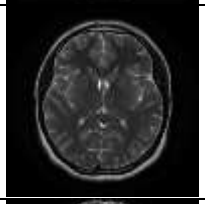
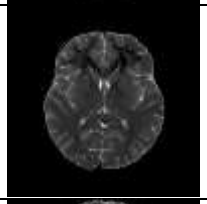
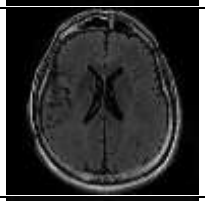
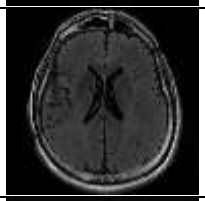
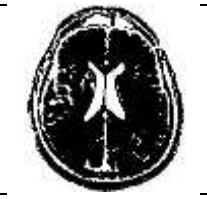
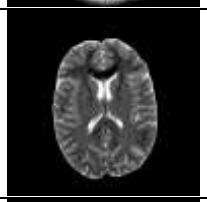
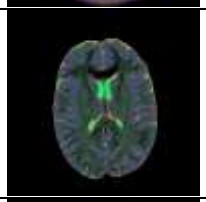
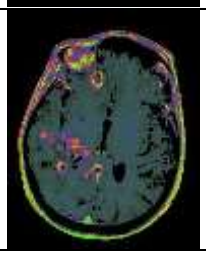
№	Оригінальне зображення	Зображення у відтінках сірого	Бінарне зображення	Зображення з видаленим черепом	Сегментоване зображення
1					
2					
3					
4					
5					
6					
7					

8					
9					
10					
11					
12					
13					
14					
15					

16					
17					
18					
19					
20					
21					
22					
23					

24					
25					
26					
27					
28					
29					
30					
31					

32					
33					
34					
35					
36					
37					
38					
39					

40					
41					
42					
43					
44					
45					
46					
47					

Додаток Е Software Architecture Document

Software Architecture Document

Brain Segmentation Tool (System of visual display of brain tumors based on digital image segmentation methods)

Kolesnyk Olena

Version 1.2

December 2022

Revision History

NOTE: *The revision history cycle begins once changes or enhancements are requested after the initial version of the Software Architecture Document has been completed.*

Date	Version	Description	Author
1/12/2022	1.0	Initial version of SAD	Kolesnyk Olena
10/12/2022	1.1	Some changes after first review	Kolesnyk Olena
17/12/2022	1.2	Ready for the next review	Kolesnyk Olena

Table of Contents

Software Architecture Document

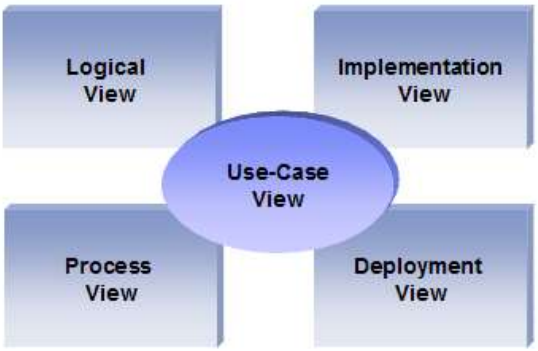
1. Introduction

This document provides a high level overview and explains the whole architecture of Brain Segmentation Tool (BS tool). It is explains how an online user will be able to create and maintain software development process definitions and includes the underlying architecture of the tool.
The document provides a high–level description of the goals of the architecture, the use cases support by the system and architectural styles and components that have been selected to best achieve the use cases. This framework then allows for the development of the design criteria and documents that define the technical and domain standards in detail.

1.1. Purpose

The Software Architecture Document (SAD) provides a comprehensive architectural overview of Brain Segmentation Tool (BS tool). It presents a number of different architectural views to depict different aspects of the system. It is intended to capture and convey the significant architectural decisions which have been made on the system.

In order to depict the software as accurately as possible, the structure of this document is based on the "4+1" model view of architecture [KRU41].



The "4+1" View Model allows various stakeholders to find what they need in the software architecture.

1.2. Scope

The scope of this SAD is to depict the architecture of Brain Segmentation Tool (BS tool) online application created by the student of Taras Shevchenko National University of Kyiv.
This document describes the aspects of Brain Segmentation Tool (BS tool) design that are considered to be architecturally significant; that is, those elements and behaviors that are most fundamental for guiding the construction Brain Segmentation Tool and for understanding this project as a whole. Stakeholders who require a technical understanding of Brain Segmentation Tool are encouraged to start by reading this document, then reviewing the Brain Segmentation Tool UML model, and then by reviewing the source code.

1.3. Definitions, Acronyms, and Abbreviations

- Windows – Operation System
- SAD – Software Architecture Document
- RUP – Rational Unified Process
- UML – Unified Modeling Language
- User – This is any user who use Brain Segmentation Tool

1.4. References

[SRS]: Software Requirements Specification

[MedBiquitous]: Sample SAD,
http://medbiq.org/std_specs/techguidelines/softwarearchitecture.pdf

[KRU41]: The "4+1" view model of software architecture, Philippe Kruchten, November 1995,
http://www3.software.ibm.com/ibmdl/pub/software/rational/web/whitepapers/2003/Pbk4p1.pdf

1.5. Overview

In order to fully document all the aspects of the architecture, the Software Architecture Document contains the following subsections.

Section 2: describes the use of each view

Section 3: describes the architectural constraints of the system

Section 4: describes the functional requirements with a significant impact on the architecture

Section 5: describes the most important use–case realization.

Section 6: describes design's concurrency aspects

Section 7: describes how the system will be deployed.

Section 8: describes any significant persistent element.

Section 9: describes any performance issues and constraints

Section 10: describes any aspects related to the quality of service (QoS) attributes

2. Architectural Representation

This document details the architecture using the views defined in the "4+1" model [KRU41], but using the RUP naming convention. The views used to document Brain Segmentation Tool application are:

Use Case view

Audience: all the stakeholders of the system, including the end–users.
Area: describes the set of scenarios and/or use cases that represent some significant, central functionality of the system. Describes the actors and use cases for the system, this view presents the needs of the user and is elaborated further at the design level to describe discrete flows and constraints in more detail. This domain vocabulary is independent of any processing model or representational syntax (i.e. XML).
Related Artifacts : Use–Case Model, Use–Case documents

Logical view

Audience: Designers.
Area: Functional Requirements: describes the design's object model. Also describes the most important use–case realizations and business requirements of the system.
Related Artifacts: Design model

Process view

Audience: Integrators.
Area: Non–functional requirements: describes the design's concurrency and synchronization aspects.
Related Artifacts: (no specific artifact).

Module Decomposition view

Audience: Programmers.
Area: Software components: describes the modules and subsystems of the application.
Related Artifacts: Implementation model, components

Deployment view

Audience: Deployment managers.
Area: Topology: describes the mapping of the software onto the hardware and shows the system's distributed aspects. Describes potential deployment structures, by including known

and anticipated deployment scenarios in the architecture we allow the implementers to make certain assumptions on network performance, system interaction and so forth.
Related Artifacts: Deployment model.

3. Architectural Goals and Constraints

Server side
No server side.

Client Side
Users will be use Windows devices to access Brain Segmentation Tool. Users are requiring using modern Windows devices with Windows 7+ operation system version.

3.1. Security

User right will be grated to scan area and recognize objects. No other roles provided. User data won't be shared in any way.

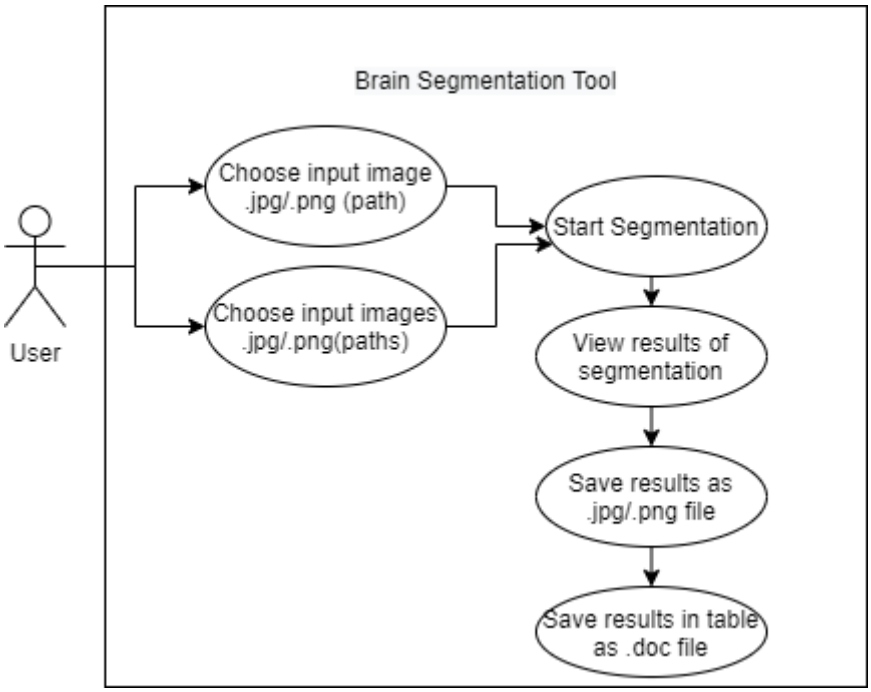
3.2. Performance

There is no particular constrains related to system performance.
It is anticipated that the system should respond to any request well under standard timeouts (20 seconds), also system performance can depend on available hardware. Therefore, actual performance can be determined only after system deployment and testing.

4. Use–Case View

This is a list of use–cases that represent major functionality of the final system [SRS]:

- Choose input image (.png/.jpg format, path)
- Choose input images (.png/.jpg format, paths)
- Start segmentation
- View results of segmentation
- Save results as .jpg/.png file
- Save results in table as .doc file



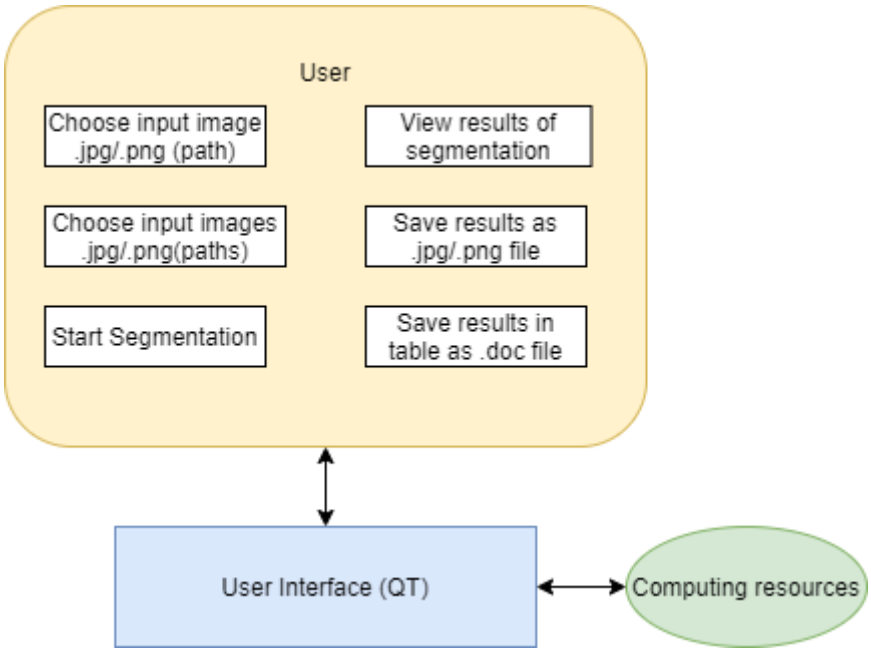
4.1. Actors

As described in the actors' correspondence diagram below, user could be one type:

1. User – default actor. He can choose, segment images and create result document.

4.2. Use–Case Realizations

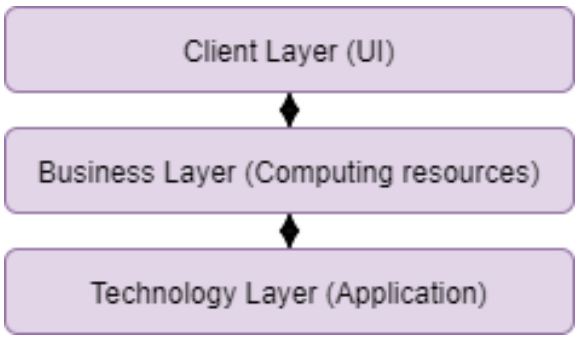
Use case functionality diagram below describes how design elements provide the functionalities identified in the significant use–cases. Use cases are displayed as functionalities for the system. Functionality may enclose more than one use–case.



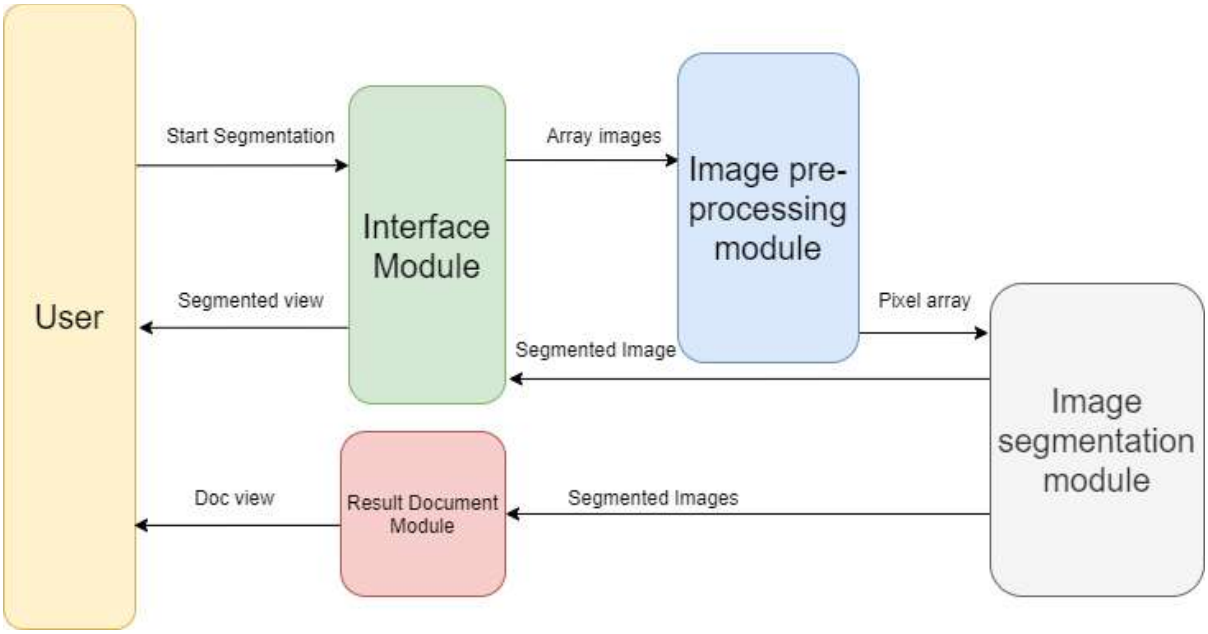
5. Logical View

5.1. Overview

Brain Segmentation Tool is divided into layers based on the N–tier architecture [KRU41].



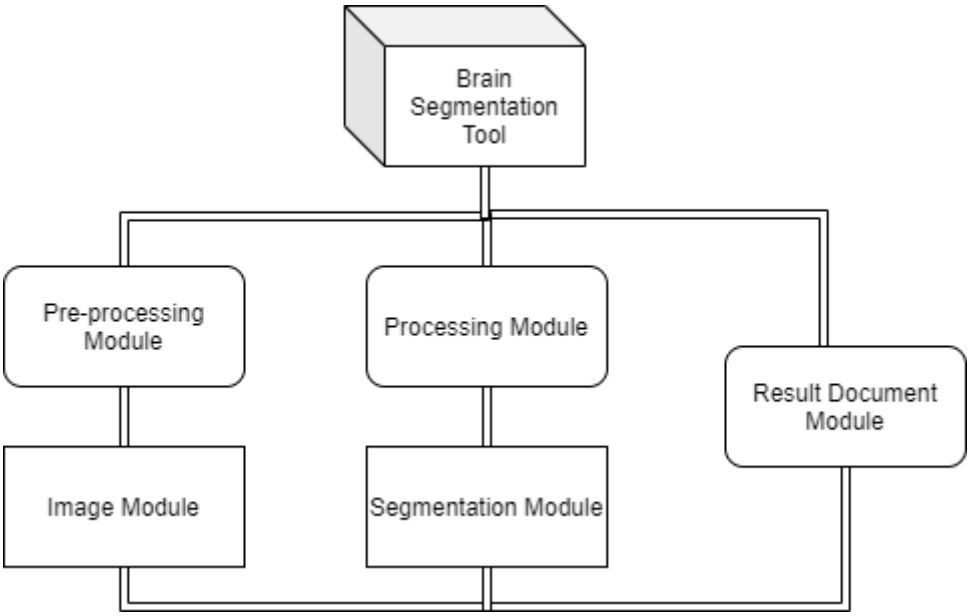
6. Process View



- User – default actor
- Image pre–processing module – preparation of the image for segmentation (in grayscale, noise)
- Image segmentation module – the classes that are responsible for the segmentation methods
- Result document module –

7. Module Decomposition View

Module decomposition view based on principles separation of concerns and abstraction and supports goals of modifiability and usability.



8. Deployment View

Brain Segmentation tool deployment has not been considered yet. All future implementation details will be included in this section.

9. Size and Performance

Volumes

- One (1) user per session
- App will occupy ~35–40 MB on device

Performance

- With maximum load all transactions well under standard timeout – 20 seconds.

10. Issues and concerns

- More UI customization.
- Improving the skull removal algorithm.
- Automatic analysis (type, size and stage) of the tumor using neural networks.
- Automation of segmentation quality assessment.

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ (Презентація)

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

МАГІСТЕРСЬКА РОБОТА

«СИСТЕМА ВІЗУАЛЬНОГО ВІДОБРАЖЕННЯ ПУХЛИН ГОЛОВНОГО МОЗКУ НА ОСНОВІ МЕТОДІВ СЕГМЕНТАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ»

Виконала: студентка групи ПДМ-61 Колесник Олена Сергіївна

Керівник: к.ф.-м.н., доц., зав. кафедри КІ Поперешняк Світлана Володимирівна

Київ - 2022

МЕТА, ОБ'ЄКТ, ПРЕДМЕТ ДОСЛІДЖЕННЯ

Мета роботи: поліпшення ручного або напівавтоматичного аналізу МРТ головного мозку за рахунок автоматичної обробки з використанням методів комп'ютерного зору.

Об'єкт дослідження: процес сегментації цифрових зображень

Предмет дослідження: є метод ефективної сегментації МРТ головного мозку на основі водорозділу та порогового значення

АКТУАЛЬНІСТЬ СЕГМЕНТАЦІЇ ЦИФРОВИХ ЗОБРАЖЕНЬ МРТ ГОЛОВНОГО МОЗКУ

- спрощення процесу виявлення непримітних ореолів пухлин на початковому етапі хвороби, коли людське око майже не здатне відрізнити невеликі відмінності у кольорах;
- якісний і кількісний аналіз зображення, точний вимір розміру і об'єму виявленої пухлини;
- виявлення з точністю до 99,9% наявності пухлин головного мозку на цифрових зображення МРТ;
- полегшення та вдосконалення роботи медичних працівників (онкологів, рентгенологів);

3

ПОСЛІДОВНІСТЬ ДІЙ АЛГОРИТМУ

1. Перетворення цифрового зображення на тривимірний масив;
2. Перетворення зображення у градації сірого;
3. Перетворення зображення на бінарне методом Отцу;
4. Виділення найбільшої області методом зв'язних компонент;
5. Об'єднання створеної маски з вхідним зображенням для видалення черепа;
6. Застосування фільтра Гаусса для зниження рівня шуму;
7. Застосування морфологічних операцій для виділення тканини мозку та пухлини;
8. Сегментація методом маркерованого водорозділу;
9. Обробка вихідного масиву до вигляду зображення.

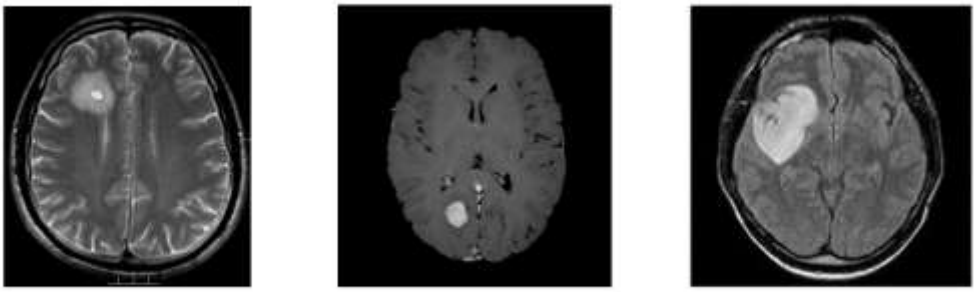
4

ПЕРЕТВОРЕННЯ ЗОБРАЖЕННЯ У ГРАДАЦІЇ СІРОГО

$$Y' = 0.299R + 0.587G + 0.114B$$

де **Y'** - яскравість
R, G, B - елементи тривимірного масиву зображення

Результати:



5

МЕТОД ОТЦУ

$$\sigma_\omega^2(t) = \omega_1(t)\sigma_1^2(t) + \omega_2(t)\sigma_2^2(t),$$

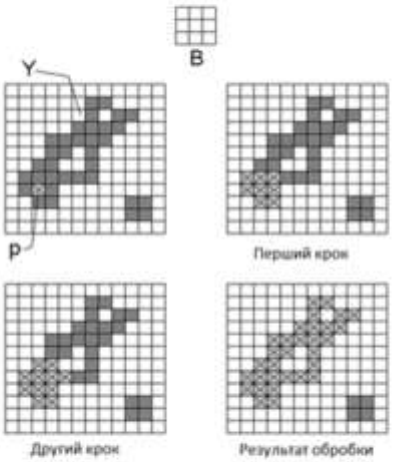
ω_i – це ймовірності двох класів (об'єкт, фон), розділених порогом

σ_i^2 – дисперсія цих класів

Результати:

6

МЕТОД ЗВ'ЯЗНИХ КОМПОНЕНТ



Нехай Y - деяка зв'язкова компонента з множини A. Припустимо, що нам відома точка p ∈ Y. Тоді всі елементи компоненти Y можуть бути отримані за допомогою рекурентного виразу:

$$Xk=(Xk-1\oplus B)\cap Ak=1,2,3,\ldots,$$

де **Xo=p;**
B - будь-який примітив операції.

Для того, щоб встановити, чи є два елементи зображення зв'язними, необхідно, щоб вони були сусідами і рівні їх яскравості задовольняли заданому критерію подібності.

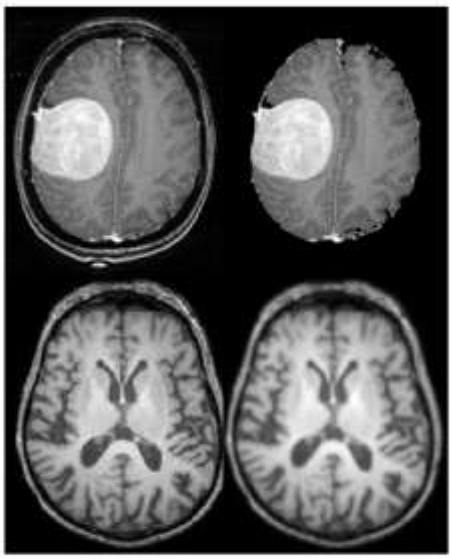
7

ФІЛЬТР ГАУССА

$$p(z) = \frac{1}{\sqrt{2\pi}\sigma} e^{-(z-\mu)^2/2\sigma^2}$$

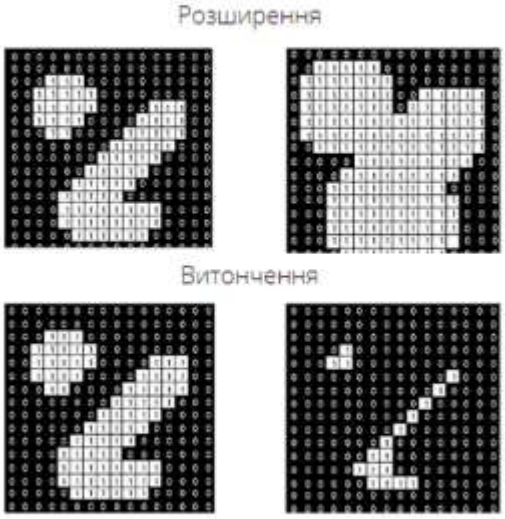
де **z** є значенням яскравості,
μ - середнє значення випадкової величини z,
σ- її середньоквадратичне відхилення.

Завданням обробки зображення може бути як поліпшення (відновлення, реставрація) зображення по якомусь певному критерію, так і спеціальне перетворення, що кардинально міняє зображення. В останньому випадку обробка зображень може бути проміжним етапом для подальшого розпізнавання зображення (наприклад, для виділення контуру об'єкта).



8

МОРФОЛОГІЧНІ ОПЕРАЦІЇ

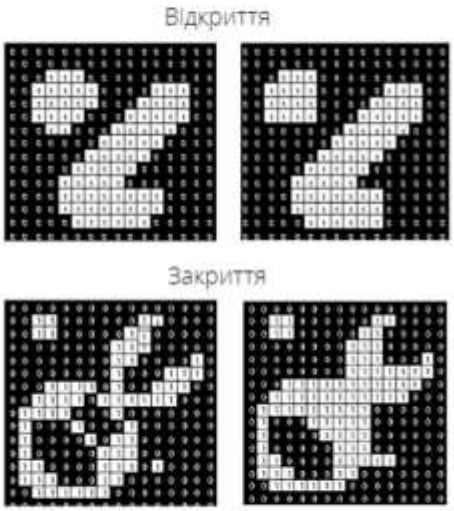


Розширення (потовщення, dilatation). Структурний елемент порівнюється з кожним пікселем зображення. Якщо хоча б один піксель околиці має значення, рівне «1», точка фокусування також отримує його (в іншому випадку присвоюється значення «0»).

Ерозія (витончення, erosion). Ця операція застосовує повернений структурний елемент до кожного пікселя зображення. Якщо хоча б один піксель в околиці має значення, рівне «0», точка фокусування також отримує це значення. В іншому випадку його значення не змінюється.

9

МОРФОЛОГІЧНІ ОПЕРАЦІЇ

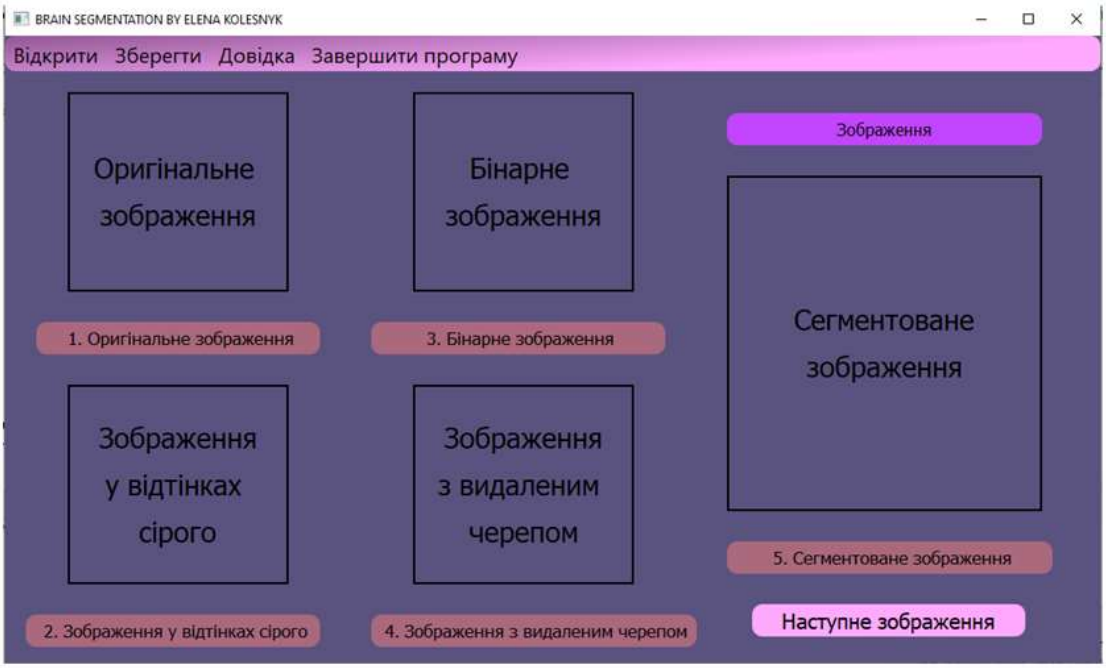


Відкриття (opening). Накладення операції дилатації на результат ерозії вихідного зображення. Це викликає згладжування зображення (видалення деталей, чим більше використовується структурний елемент, тим сильніше згладжування зображення).

Закриття (closing). Накладення ерозійних операцій на результат дилатації вихідного зображення. Вона видаляє всі отвори на зображенні і увігнутість нижче структурного елементу (чим більше структурний елемент, тим більше елементів заповнюється).

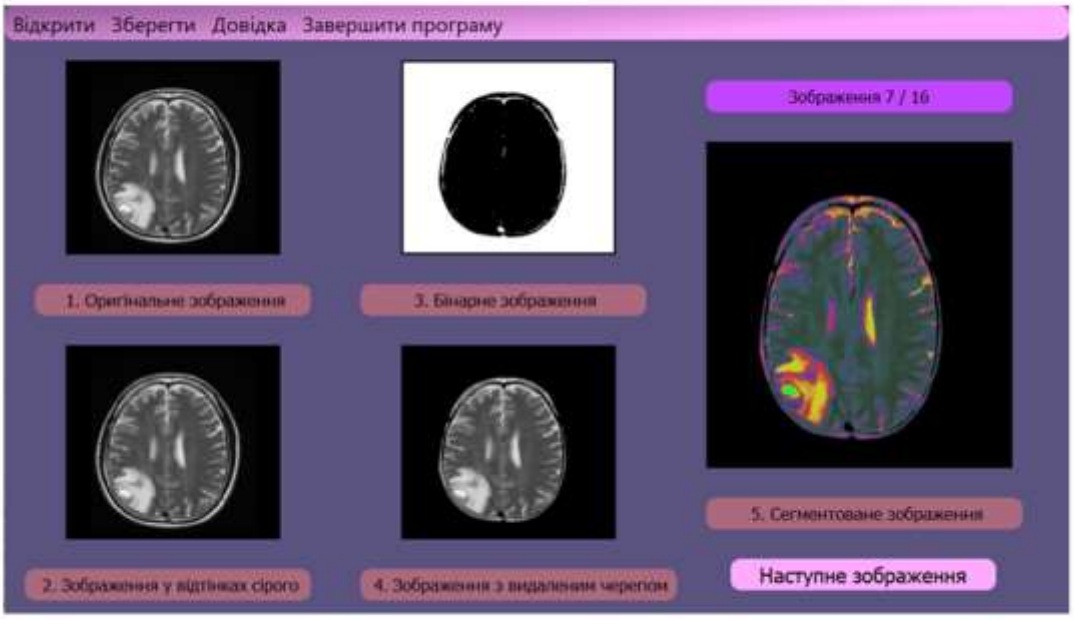
10

ПРАКТИЧНИЙ РЕЗУЛЬТАТ. ІНТЕРФЕЙС КОРИСТУВАЧА



11

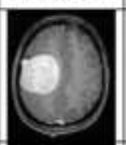
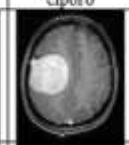
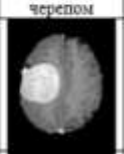
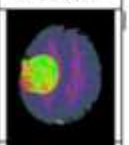
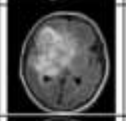
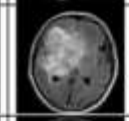
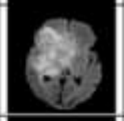
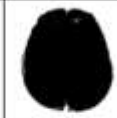
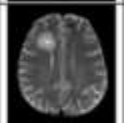
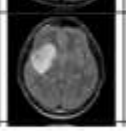
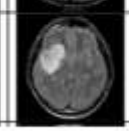
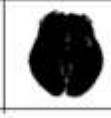
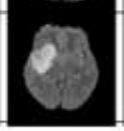
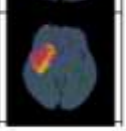
ПРАКТИЧНИЙ РЕЗУЛЬТАТ. ІНТЕРФЕЙС КОРИСТУВАЧА



12

РЕЗУЛЬТУЮЧИЙ ДОКУМЕНТ

Результати сегментації

№	Оригінальне зображення	Зображення у відтінках сірого	Бінарне зображення	Зображення з видаленим черепом	Сегментоване зображення
1					
2					
3					
4					

13

РЕЗУЛЬТАТИ МОДЕЛЮВАННЯ. ОЦІНКА ЧАСУ ВИКОНАННЯ

```
--- 0.2002720832824707 секунд --- C:/Users/Elena/Downloads/archive/yes/Y245.jpg
--- 0.2568333148956299 секунд --- C:/Users/Elena/Downloads/archive/yes/Y246.JPG
--- 0.20336246490478516 секунд --- C:/Users/Elena/Downloads/archive/yes/Y247.JPG
--- 0.22903919219970703 секунд --- C:/Users/Elena/Downloads/archive/yes/Y248.JPG
--- 0.24486136436462402 секунд --- C:/Users/Elena/Downloads/archive/yes/Y249.JPG
--- 0.2976531982421875 секунд --- C:/Users/Elena/Downloads/archive/yes/Y250.jpg
--- 0.2053225040435791 секунд --- C:/Users/Elena/Downloads/archive/yes/Y251.JPG
--- 0.22376680374145508 секунд --- C:/Users/Elena/Downloads/archive/yes/Y252.jpg
--- 0.20130610466003418 секунд --- C:/Users/Elena/Downloads/archive/yes/Y253.JPG
--- 0.18222475051879883 секунд --- C:/Users/Elena/Downloads/archive/yes/Y254.jpg
--- 0.2669532299041748 секунд --- C:/Users/Elena/Downloads/archive/yes/Y255.JPG
--- 0.19345569610595703 секунд --- C:/Users/Elena/Downloads/archive/yes/Y256.JPG
--- 0.36107802391052246 секунд --- C:/Users/Elena/Downloads/archive/yes/Y257.jpg
--- 0.3302750587463379 секунд --- C:/Users/Elena/Downloads/archive/yes/Y258.JPG
--- 0.30683445930480957 секунд --- C:/Users/Elena/Downloads/archive/yes/Y259.JPG
---Кількість зображень:155 за 42.868818283081055 секунд ---
```

Середній час обробки одного зображення: **0,28 секунди**

14

ВИСНОВКИ

1. Проведено аналіз зарубіжної літератури та розглянуто сучасні підходи до сегментації зображення та його попередньої обробки.
2. Розглянуто метод сегментації МРТ зображень головного мозку на основі вододілу та порогів. Визначено основні складові алгоритму методу та комбінації для кращої роботи системи.
3. Розроблено систему візуального відображення пухлин головного мозку на основі методів сегментації цифрових зображень.
4. Проведено дослідження залежності результатів сегментації від типу зображення. В результаті аналізу отриманих даних було виявлено, що в залежності від форми черепа, або кута нахилу черепної коробки на МРТ зображенні може змінюватися якість сегментації. Також, чим контрастніше оригінальне зображення (залежить від МРТ апарату) тим більша ймовірність правильної сегментації пухлини. В залежності від кількості об'єктів у масиві зображень змінюється і час сегментації, але методи, використані у роботі, дозволяють сегментувати 50 зображень менше ніж за хвилину, що є одним з найкращих результатів цієї предметної області.
5. Найкращі результати по сегментації пухлин дали групи зображень, черепна коробка яких не перевищує певну ширину та знаходиться на деякій відстані від тканини самого мозку. Метод, що було розглянуто у дипломній роботі не є універсальним, але однозначно може використовуватися для початкового аналізу пухлин головного мозку.
6. Перспективою даної роботи є удосконалення алгоритму видалення черепа, проведення автоматичного аналізу (виду, розміру та стадії) пухлини за допомогою нейронних мереж та автоматизація оцінювання якості сегментації

ПУБЛІКАЦІЇ ТА АПРОБАЦІЯ РОБОТИ

Статті:

1. Поперешняк О. М., Колесник О.С. Система візуального відображення пухлин головного мозку на основі методів сегментації цифрових зображень// Зв'язок. №3 (139), 2021. *(робота ще не опублікована, прийнято до друку)*

Тези доповідей:

1. Поперешняк О. М., Колесник О.С. Сегментація пухлин головного мозку з магнітно-резонансних зображень за допомогою адаптивного порогового визначення та алгоритму розрізу графа. // III науково-практична конференція «Проблеми комп'ютерної інженерії». – Київ: ДУТ, 2022. (*ще не опублікована, прийнято до друку)*
2. Поперешняк О. М., Колесник О.С. Відображення пухлин головного мозку на основі методів сегментації цифрових зображень. // II Міжнародна науково-технічна конференція «Системи і технології зв'язку, інформатизації та кібербезпеки: актуальні питання і тенденції розвитку». – Київ: ВІТІ, 2022. (*ще не опублікована, але подана до друку)*

ДЯКУЮ ЗА УВАГУ!