Західноукраїнський національний університет
Фаультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Полагнюк Іван Васильович

АВТОМАТИЧНА СЕГМЕНТАЦІЯ ЗОБРАЖЕНЬ НА ОСНОВІ МЕТРИКИ ФРЕШЕ / AUTOMATIC IMAGE SEGMENTATION BASED ON FRECHET METRICS

спеціальність; 123 – Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія

Кваліфікаційна робота

Виконав студент групи Кім-21
І.В. Полагнюк
_____

Науковий керівник:
д.т.н., професор, О.М. Березький
_____

Кваліфікаційну роботу
допущено до захисту
«___» _____2021 р.

Завідувач кафедри КІ
О.М.Березький
_____

Тернопіль – 2021

ЗМІСТ

ВСТУП

Актуальність дослідження. Сегментація зображень – це одна із операцій середнього рівня. Призначена для розділення зображення на області та об’єкти, які його утворюють. Кінцевий успіх комп’ютерних процедур аналізу зображень залежить від точності сегментації [1]. Сегментація буває ручною, автоматизованою та автоматичною. При ручній сегментації, елементи виділяються експертом вручну. Для пришвидшення цього процесу використовують автоматизовані та автоматичні алгоритми. При цьому, виникає задача оцінки результатів сегментації, оскільки дані алгоритми показують не завжди точний результат [2-8].

Дана проблема є також актуальною при обробці біомедичних зображень. Оскільки, на даний момент, виділення клітин на зображеннях виконується експертом вручну за допомогою мікроскопу. Коректне виділення об’єктів на біомедичних зображеннях автоматичними алгоритмами сегментації призводить до зменшення впливу людини в цей процес та збільшення точності обчислення інформативних показників. Таким чином, дослідження алгоритмів сегментації та порівняння їх за кількісними показниками є актуальною темою дослідження.

Мета роботи – дослідження та розроблення алгоритмів порівняння контурів зображень в метриці Громова-Фреше і створення програмного засобу для оцінки похибок алгоритмів сегментації.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- дослідити сучасні алгоритми сегментації зображень;
- дослідити метрики та алгоритми реалізації метрики Фреше;
- розробити алгоритми пошуку відстані між контурами у метриці Громова-Фреше;
- розробити модуль для пошуку відстані між контурами у метриці Громова-Фреше;
- виконати тестування розробленого модуля.

Об’єкт дослідження. Процес аналізу сегментації.

Предмет дослідження. Кількісна оцінка результатів сегментації.

Методи досліджень. Якість сегментації зображень можна оцінити на суб'єктивному (якісному) так і на об'єктивному (кількісному) рівні [8]. Суб'єктивні критерії – це критерії візуального сприйняття, отримувані в процесі експертизи деякою групою експертів. Об'єктивні критерії – це критерії, отримані внаслідок порівняння (знаходження різниці) кількісних ознак сегментованого та еталонного (сегментованого експертом) зображень [8]. Одним із об'єктивних методів порівнянь є використання метрик.

Наукова новизна. Розроблено алгоритми автоматичної сегментації зображень на основі метрики Громова-Фреше.

Практичне значення. Розроблено модуль, який реалізує алгоритми автоматичної сегментації на основі метрики Громова-Фреше.

Публікації результатів досліджень. За результатами досліджень опубліковані двоє тез доповідей V науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі» (2 грудня 2021 р., м. Тернопіль, Західноукраїнський національний університет) [9, 10]:

1. Палагнюк І. В., Клімовський Д. Б., Вдодович О. В., Бучинський Т. Б. Сегментація зображень з використанням метричних мір. *Інтелектуальні комп'ютерні системи та мережі* : тези доп. V Наук.-практ. конф. молодих вчених і студентів (2 груд. 2021 р.). Тернопіль : ЗУНУ, 2021. С.
2. Вдодович О. В., Клімовський Д. Б., Палагнюк І. В., Бучинський Т. Б. Алгоритми порівняння зображень в метричних просторах. *Інтелектуальні комп'ютерні системи та мережі* : тези доп. V Наук.-практ. конф. молодих вчених і студентів (2 груд. 2021 р.). Тернопіль : ЗУНУ, 2021. С.

Кваліфікаційна робота складається із вступу, трьох розділів, висновків, списку використаної літератури та додатків [11].

1 АНАЛІЗ МЕТОДІВ, МЕТРИК І АЛГОРИТМІВ ДЛЯ КІЛЬКІСНОЇ ОЦІНКИ РЕЗУЛЬТАТІВ СЕГМЕНТАЦІЇ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ

1.1 Аналіз біомедичних зображень

Цифрове зображення – це масив даних, отриманий шляхом дискретизації двовимірного зображення [12]. Цифрові зображення поділяються на растрові та векторні. Основним елементом растрового зображення є піксель. Кожен із пікселів зображення має такі характеристики:

– координати – місце пікселя у зображенні;
– яскравість – інтенсивність світла в даних координатах.

Координати пікселя представляють парою *(x, y)*. Яскравість пікселя у цих координатах записують функцією *f(x,y)*. В такому випадку зображення буде представлене масивом функцій яскравості [5].

Біомедичні зображення – це растрові зображення, отримані за допомогою будь-якої біомедичної техніки, що використовуються для візуального та автоматизованого аналізу в медицині та біології [1]. Дослідження у сфері біомедичних зображень об’єднує інженерів, фізиків, біологів та хіміків, які займаються розвитком методології експертизи біологічної структури і функціонування з допомогою зображень. Робота над даним видом зображень охоплює знання з магнітно-резонансної томографії, магнітно-резонансної спектроскопії, ядерної медицина, оптичної світлової мікроскопії та обробки і аналізу зображень.

Дослідження біомедичних зображень спрямовані на використанні математичних, інженерних, фізичних та хімічних методів для отримання кількісної інформації з біомедичних зображень [13]. Даний вид зображень використовується для діагностичних та терапевтичних цілей. Знімки фізіологічних процесів можуть бути отримані з допомогою сучасних давачів та комп’ютерної техніки. Біомедичні технології використовують рентгенівські промені (комп’ютерна томографія), звукові хвилі (ультразвукова діагностика), магнетизм (магнітно-резонансна томографія), радіоактивні фармацевтичні препарати, світло (ендоскопія) [14].

Внаслідок цього, стає можливою оцінка стану органів та тканин пацієнта, а також контроль його стану протягом часу лікування.

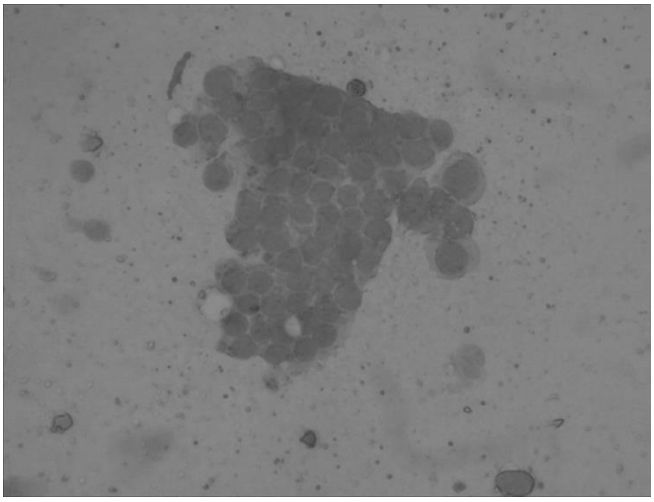
Гістологія – наука про тканини багатоклітинних тварин та людини. Дана наука вивчає не тільки тканини, але й клітини, з яких вони складаються, будову органів і систем організму. Згідно з цим існують такі розділи предмету: цитологія (наука про клітину), загальна гістологія (вивчає тканини), спеціальна гістологія (вивчає будову органів та їх систем) [15].

У сучасній гістології, цитології та ембріології використовуються різноманітні методи дослідження, що дозволяють всебічно вивчати процеси розвитку, будови і функції клітин, тканин і органів. Головними етапами цитологічного і гістологічного аналізу є вибір об'єкта дослідження, підготовка його для вивчення в мікроскопі, застосування методів мікроскопування, а також якісний і кількісний аналіз зображень [16].

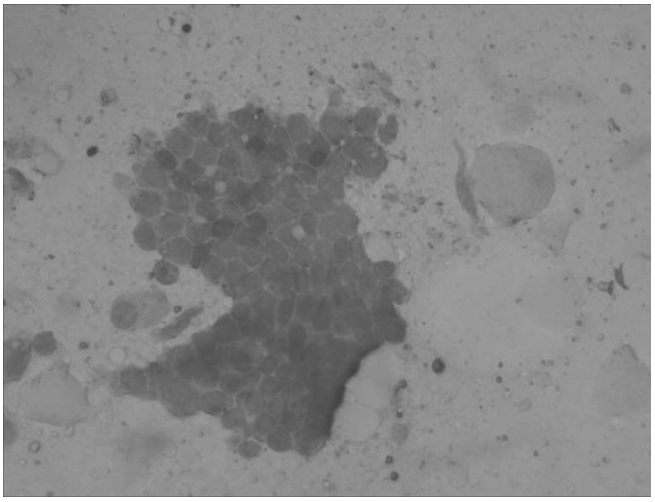
Сучасна гістологія має у своєму розпорядженні великий арсенал різних методів дослідження. Всі ці методи поєднує необхідність використання спеціального приладу – мікроскопа, тому всі вони є мікроскопічними методами. Залежно стану досліджуваного об'єкта ці методи діляться на вітальні (або суправітальні), коли досліджуються живі клітини, тканини, органи і навіть цілі організми, і поствітальні, коли досліджуються мертві об'єкти, особливо загиблі шляхом фіксації [17].

Для гістологічного дослідження використовують світлові або електронні мікроскопи. В даній роботі надалі розглядатимемо лише зображення, отримані за допомогою електронного мікроскопа. Приклади таких гістологічних зображень представлені на рисунку 1.1.

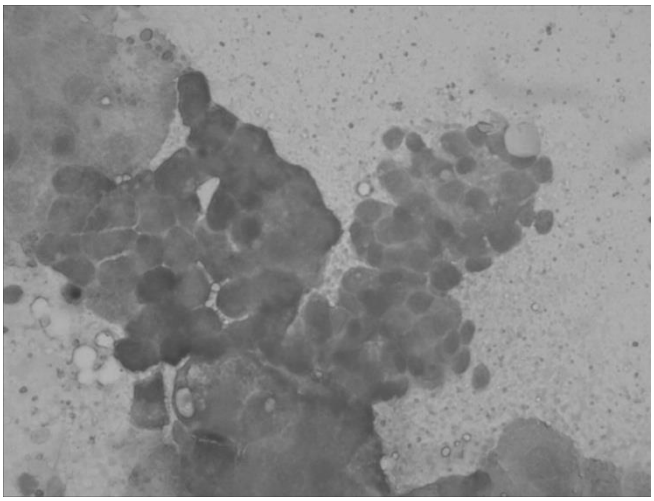
Як можна побачити із рисунку 1.1, гістологічні зображення мають нечіткі контури, розмитий фон та невелику кількість кольорів. Для їхнього аналізу необхідний фахівець, який виокремлює на зображенні клітини та проводить необхідні виміри. Це є складна робота, тому, проблема знаходження оптимального алгоритму сегментації для гістологічних зображень є актуальною на даний момент.



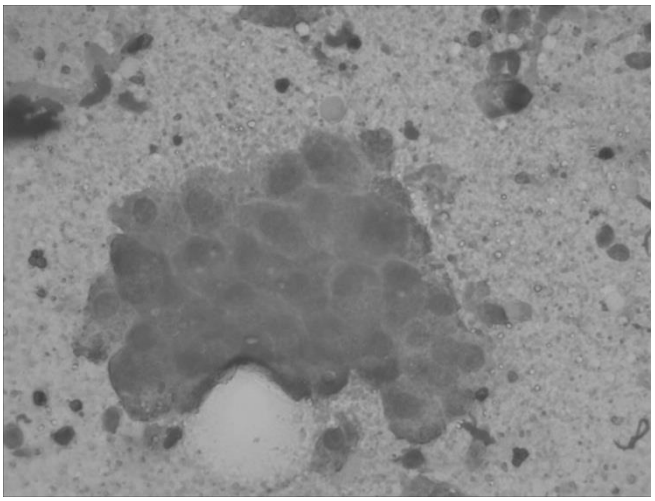
а)



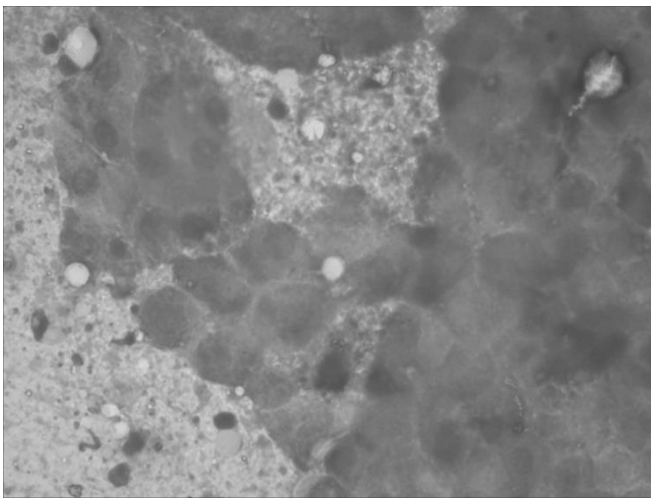
б)



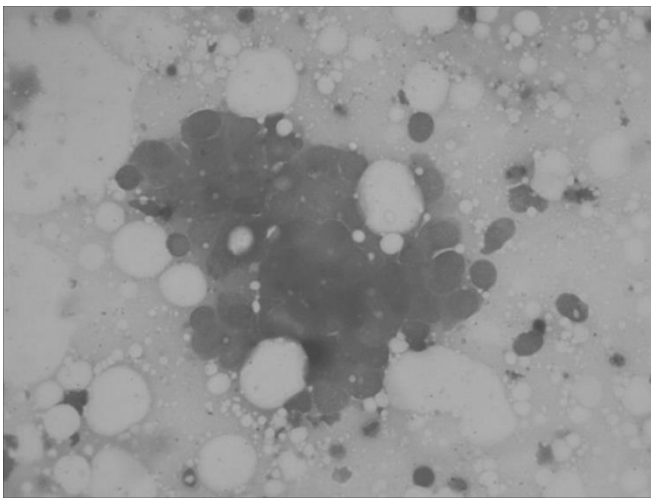
в)



г)



д)



е)

Рисунок 1.1 – Гістологічні зображення, що використані для досліджень

Цитологія – наука, яка досліджує та вивчає клітини, їхню будову та функції [17-19]. Клітина є елементарною живою системою багатоклітинних організмів на рівні якої зберігається сукупність усіх проявів життєдіяльності. Цитологічні зображення отримуються від світлової мікроскопії. Цитологічний метод є одним із методів дослідження та діагностування ракових клітин. Мікрооб'єктами на цитологічному зображенні є окремі, випадково розміщені клітини [18]. Приклади цитологічних зображень наведені на рисунках 1.2-1.4.

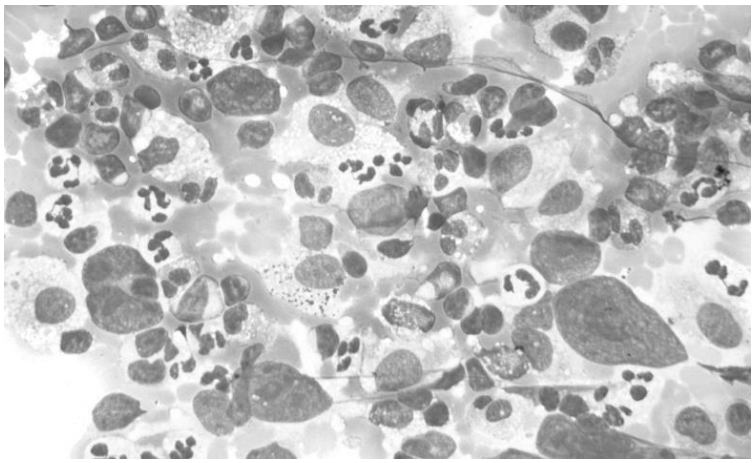


Рисунок 1.2 – Цитологічне зображення тканини із лімфогранулематозом

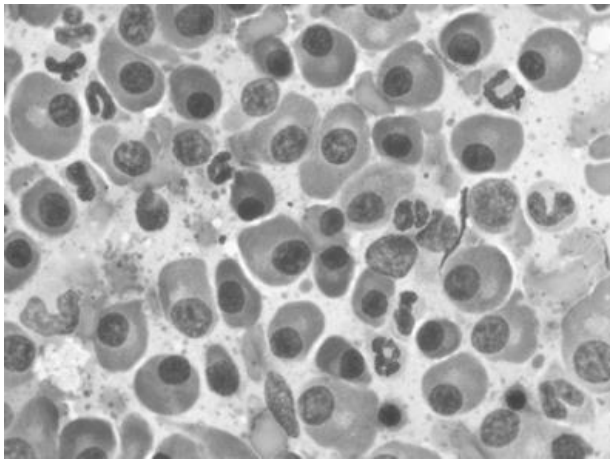


Рисунок 1.3 – Цитологічне зображення клітин із множинними пухлинами

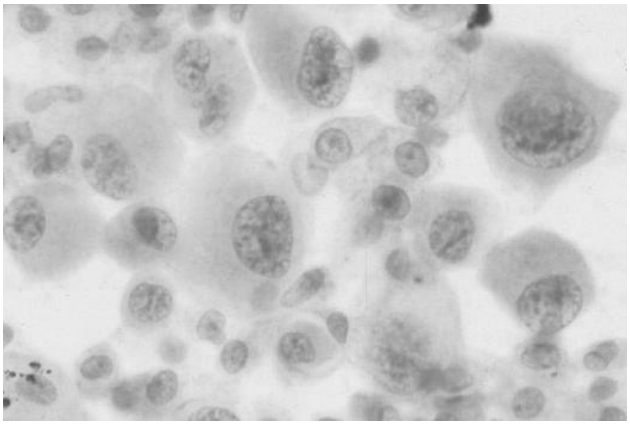


Рисунок 1.4 – Цитологічне зображення клітин із мезотеліомою

При аналізі гістологічних та цитологічних зображень необхідно враховувати наступні особливості: низька чіткість; містять мікрооб'єкти, оточені складним з погляду геометричних та оптичних характеристик фоном; відмінності у рівнях яскравості об'єктів такі самі, як і в навколишнього фону; залежно від ступеня оптичного збільшення зображення одні мікрооб'єкти виділяються краще, інші втрачаються; містять області з структурою, що повторюється; стабільність колірної палітри зображень зразків із відомих барвників [14].

Для якісного опису мікрооб'єктів на цитологічних зображеннях використовуються такі характеристики [20]: межі цитоплазми (чіткі, нечіткі), розташування вакуолей (по периферії, біля ядра), форма ядра (кругла, овальна).), обриси ядра (рівні, нерівні), ядерний поліморфізм (слабкий, сильний), стратифікація ядер клітин, наявність «голих» ядер (ядра клітин, що повністю втратили свою цитоплазму), структуру хроматину (однорідну, нерівномірну, дрібнозернисту, крупнозернисту), наявність внутрішньоядерних включень, вакуолей в ядрі, кількість ядерців (поодинокі, множинні), їхнє місцерозташування (центральне, ексцентричне) [21-22].

На основі цього, можна зробити висновок, що основними труднощами автоматизованого аналізу гістологічних та цитологічних зображень є наявність шумів та артефактів, неоднорідність фону та складна геометрична форма елементів для виділення.

1.2 Аналіз алгоритмів сегментації

В обробці зображень існують три рівні: низький, середній та високий. Процеси низького рівня мають на меті застосування операцій попередньої обробки (зменшення шуму, підвищення контрасту, покращення різкості). До методів середнього рівня належать сегментація, виділення контуру, опис контуру, стиснення, виділення інформативних ознак (контурна функція, її характер). До операцій високого рівня належать розпізнавання зображень, рукописного тексту.

Сегментація зображень – це одна із операцій середнього рівня. Призначена для розділення зображення на області та об’єкти, які його утворюють. Степінь деталізації розділення зображення залежить від вирішуваної задачі. Сегментацію завершують, коли об’єкти, які нас цікавлять, ізольовані. Сегментація – одна із найскладніших завдань обробки зображень. Кінцевий успіх комп’ютерних процедур аналізу зображень залежить від точності сегментації [1].

Сегментація активно використовується при обробці медичних зображень, наприклад, для виявлення пухлин та інших патологій, визначення обсягів тканин, різних діагностик, планування лікування. Основним етапом процесу аналізу біомедичних препаратів є виокремлення мікрооб’єктів, тобто, сегментація біомедичного зображення.

Алгоритми сегментації засновані на властивостях розривності та однорідності яскравості. Розглянемо алгоритми, які засновані на розривності яскравості. Існують три основних види розривів яскравості в цифрових зображеннях:

1) точка;
2) лінія;
3) перепад.

Для знаходження таких розривів яскравості найчастіше використовують пороговий алгоритм. При цьому, задається деяке порогове обмеження. Після квантування, функція зображення відображає елементи зображення з рівнем яскравості більшим, ніж порогове значення, у значення 1, а меншим за порогове – в 0. Порогову сегментацію можна представити співвідношенням (1.1).

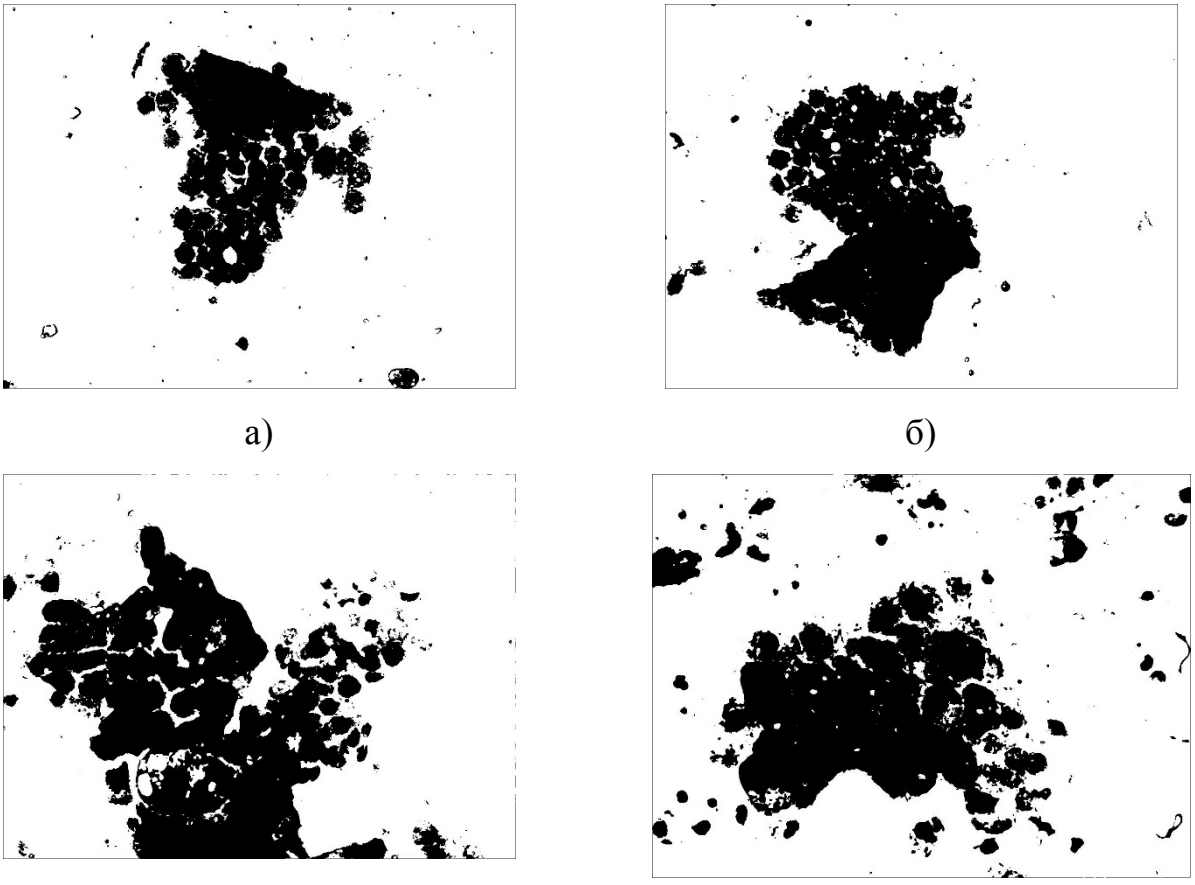
$$p(x,y) = \begin{cases} 1, \text{якщо } f(x,y) > T \\ 0, \text{якщо } f(x,y) \leq T \end{cases} \quad (1.1)$$

де *p(x,y)* – значення порогової сегментації;

f(x,y) – яскравість пікселя у точці *(x,y)*;

T – значення порогу.

У разі, коли *p(x,y)* дорівнює 1, таку точку називають точкою об'єкту. В іншому випадку – точкою фону. Таким чином зображення розділяється на об'єкт та фон. На рисунку 1.5 (а, б, в, г) представлені зображення з рисунку 1.1 (а, б, в, г), сегментовані за допомогою порогового алгоритму.



а) б)

в) г)

Рисунок 1.5 – Зображення, сегментовані алгоритмом порогової сегментації

Порівнюючи рисунок 1.1 із рисунком 1.5 можна побачити, що мікрооб'єкти, розташовані поряд, об'єднані в одну групу. Таким чином, між клітинами відсутні границі. Мікрооб'єкти, які розташовані окремо від інших, після сегментації чітко

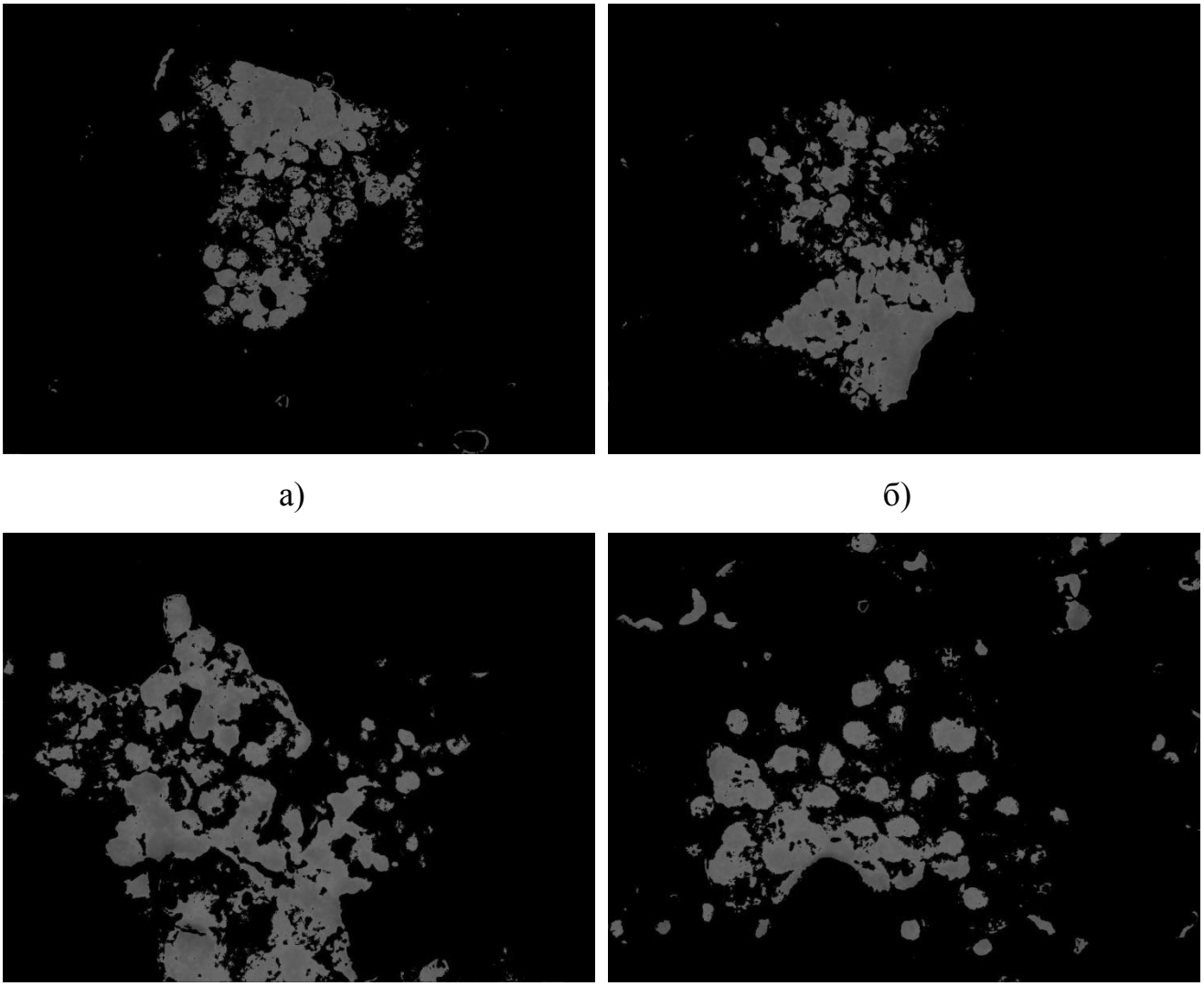
видно. Таким чином, можна зробити висновок про непридатність використання порогової сегментації для такого типу гістологічних зображень.

Наступним класом алгоритмів сегментації є алгоритми, засновані на однорідності яскравості. До цього класу належать алгоритми нарощування областей, кластеризації та водорозділу.

Метод нарощування областей являє собою процедуру, яка групує пікселі чи підобласті в більші області по наперед заданим критеріям. Спочатку береться множина точок, які відіграють роль «центрів кристалізації», а потім на них нарощують області шляхом приєднання до кожного центру тих пікселів з числа сусідів, які за своїми властивостями близькі до центру кристалізації (наприклад, мають яскравість чи колір в заданому діапазоні) [1]. Перевагою методу нарощування областей є простота розуміння та реалізації алгоритму. Недоліком є можливість хибного результату сегментації внаслідок об’єднання схожих пікселів без звертання уваги на їхню зв’язність.

Наступним типом алгоритмів сегментації є алгоритми кластеризації. Кластеризація – це поділ певного набору об'єктів на підмножини (кластери), що не перекриваються, таким чином, що кожен кластер містить схожі об'єкти, а об'єкти різних кластерів відрізняються один від одного. Серед усіх методів кластеризації метод кластеризації k-середніх є найрозповсюдженішим і найдослідженішим. Він мінімізує спотворення вхідного зображення, поділ даних між регіонами, що не перетинаються, що визначаються їх центрами. Переважання методу k-середніх пояснюється його основними перевагами: простотою, гнучкістю та швидкою збіжністю. Проте практичне застосування методу сильно обмежене його недоліками, зокрема: результати кластеризації методом k-середніх багато в чому залежать від вибору початкової конфігурації центроїдів (ініціалізації); Робота способу істотно уповільнюється при групуванні великих зображень [20-21].

На рисунку 1.6 (а, б, в, г) представлено зображення із рисунку 1.1 (а, б, в, г), сегментовані алгоритмом k-середніх.



а) б)

в) г)

Рисунок 1.6 – Зображення, сегментовані алгоритмом k-середніх

Наступний алгоритм є сегментація методом водоподілу. Поняття водоподілу засновано на представленні зображення у вигляді тривимірної поверхні, заданої двома просторовими координатами та рівнем яскравості в якості висоти поверхні. В такій інтерпретації розглядають точки трьох видів:

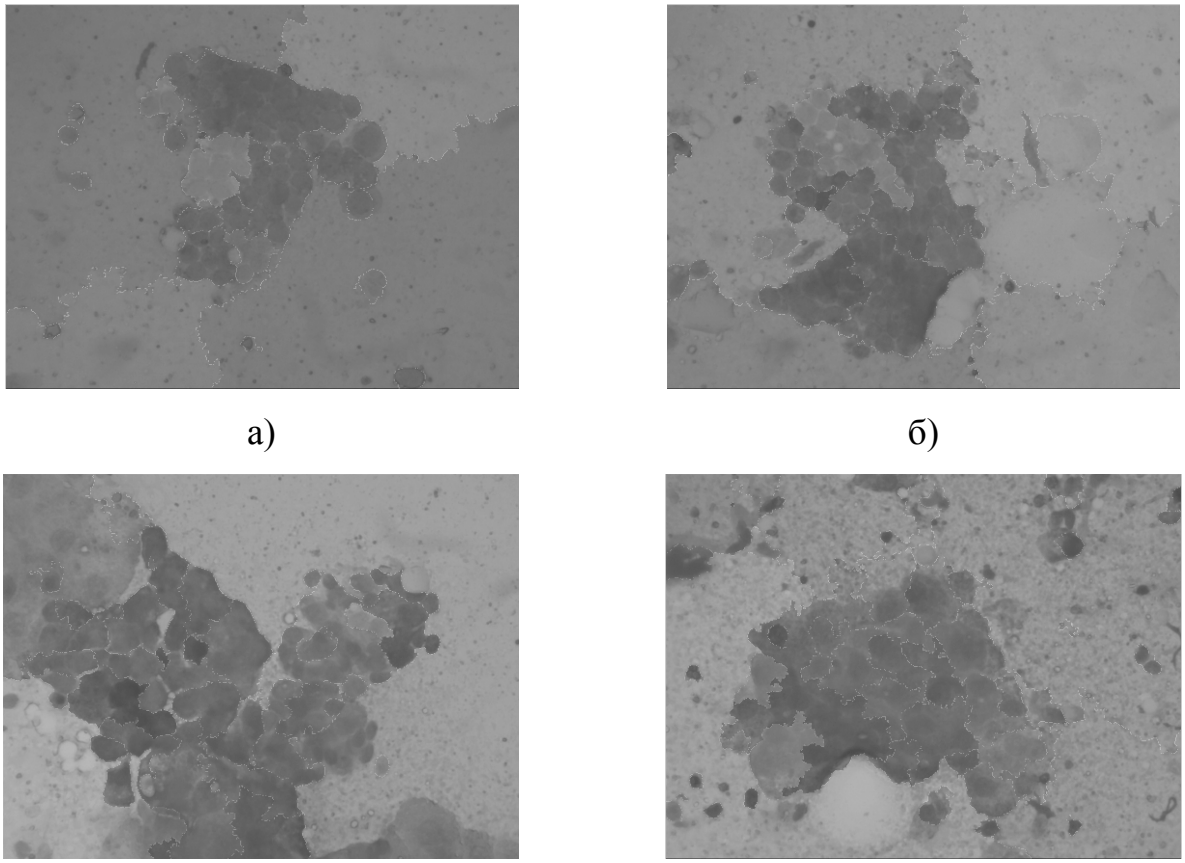
- точки локального мінімуму;
- точки, які знаходяться на схилі (з них вода скочується в один і той же ж локальний мінімум);
- точки, які знаходяться на гребені або піку (вода з однаковою ймовірністю скочується в більш, ніж один мінімум).

Множина точок останнього вигляду утворює лінії водоподілу [1].

Ідея алгоритму наступна:

1) в точках локального мінімуму «проколюються» дірки. Через них поступає і повільно наповняє «рельєф» зображення вода;
2) коли вода в двох сусідніх басейнах близька до того, щоб з’єднатися, в цьому місці ставиться перегородка, яка запобігає цьому;
3) коли заповнення досягає фази, що над водою видно лишень верхівки перегородок, сегментація завершується, а дані перегородки утворюють лінії водоподілу [1].

На рисунку 1.7 (а, б, в, г) представлені зображення із рисунку 1.1 (а, б, в, г), сегментовані методом водоподілу.



а) б)

в) г)

Рисунок 1.7 – Зображення, сегментовані методом водоподілу

Із рисунку 1.7 можна побачити, що деякі мікрооб’єкти, розташовані поряд, виокремлені в різні групи. Тобто, цей алгоритм сегментації дозволяє просегментувати зображення, на яких елементи мають нечіткий контур або розташовані близько.

Перевагою алгоритму є висока точність сегментації та знаходження контуру. Недоліком є велика чутливість алгоритму до шуму. Інколи, після проведення

сегментації методом водоподілу може знадобитися постоброблення зображення (злиття).

На основі проведених експериментів складемо таблицю використання відповідних методів сегментації (таблиця 1.1) [4].

Таблиця 1.1 – Класифікація мікрооб'єктів та методів їх сегментації

Тип зображень	Алгоритм
Мікрооб'єкти одного типу, що розташовані окремо	Порогова сегментація, високочастотна фільтрація
Мікрооб'єкти одного типу, що розташовані окремо (нечіткі границі)	Водоподіл, кластеризація
Мікрооб'єкти різних типів, що розташовані окремо	Порогова сегментація, кластеризація, нарощування областей
Мікрооб'єкти різних типів, що розташовані окремо (нечіткі границі)	Порогова сегментація, кластеризація
Мікрооб'єкти одного типу, що розташовані разом	Кластеризація, високочастотна фільтрація
Мікрооб'єкти одного типу, що розташовані разом (нечіткі границі)	Порогова сегментація, кластеризація
Мікрооб'єкти різних типів, що розташовані разом	Кластеризація, нарощування областей, водоподіл
Мікрооб'єкти різних типів, що розташовані разом (нечіткі границі)	Водоподіл, кластеризація

В результаті досліджень, встановлено, що для гістологічних зображень, наведених на рисунку 1.1, найкращим алгоритмом сегментації є алгоритм k-середніх. Тому, його взято за основу для проведення подальшої роботи.

1.3 Метрики для оцінки подібності зображень

У більшості практичних завдань якістю сегментації вважається ступінь близькості двох зображень: сегментованих експертом та сегментованих

відповідними алгоритмами. Якість сегментації зображення можна визначити як у суб'єктивному (якісному), і на об'єктивному (кількісному) рівні [4].

Суб'єктивні критерії – критерії зорового сприйняття, отримані у процесі дослідження певною групою експертів. Найпопулярніший спосіб оцінки – це, який оцінює такі властивості зображення, як правильність вибору контурів, областей, кольорів тощо.

Об'єктивні критерії – це критерії, отримані шляхом порівняння (пошуку відмінностей) кількісних характеристик сегментованих та еталонних зображень (сегментованих експертом). Критерії ці часто використовуються в автоматизованих системах аналізу зображень визначення кількісного еквівалента якості сегментації [4].

Відомі критерії кількісної оцінки можна розділити на дві групи [23]:

– критерії, які пов'язані з наглядом (несупервізорні);
– критерії спостереження (супервізорні).

Критерії, які пов'язані з наглядом засновані на обчисленні різної статистики та використовуються за відсутності апріорної інформації про сегментовані зображення. Критерії спостереження побудовані на обчисленні виміру відстані за результатами сегментованих та зразкових форм об'єктів. Приклад форми об'єкта визначає експерт. Особливе місце при аналізі якості сегментації займає використання метрик [23].

Метричний простір – це множина елементів для якої відстані між усіма елементами є визначеними. Ці відстані утворюють разом метрику [24]. Математично, метрика представляється наступним чином: нехай X – випадкова множина. Відображення $X \times X \to \mathbb{R}$ називається метрикою (відстанню) в X, якщо воно задовольняє наступні умови (аксіоми метрики):

1) $\rho(x, y) \geq 0 \; \forall x, y \in X$ (аксіома невід'ємності);
2) $\rho(x, y) = 0 \Leftrightarrow x = y \; \forall x, y \in X$ (аксіома рівності);
3) $\rho(x, y) = \rho(y, x) \; \forall x, y \in X$ (аксіома симетрії);
4) $\rho(x, z) \leq \rho(x, y) + \rho(y, z) \; \forall x, y, z \in X$ (аксіома трикутника) [25].

Найбільш поширеною метрикою є евклідова метрика. В даній метриці відстань між двома точками обчислюється за допомогою теореми Піфагора [26].

Якщо є дві точки p та q і p=(p_1, p_2, …, p_n), q=(q_1, q_2, …, q_n), то відстань між ними знаходиться за формулою:

$$d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2}. \quad (1.2)$$

В даній метриці існує лише одна найкоротша відстань від точки p до точки q. І вона є прямою лінією.

Основним елементом вищенаведеної метрики є точка. Наступна метрика – метрика Хаусдорфа – основними елементами має криві. Відстань між двома кривими у даній метриці вимірюється максимальним радіусом кола із центрами у точках однієї кривої, яке міститиме у собі всі точки іншої кривої. Знаходження відстані представляється формулою [27]:

$$\rho_H(\Gamma_1, \Gamma_2) = \max\left\{\max_{y\in\Gamma_2}\min_{x\in\Gamma_1}\rho(x,y), \max_{x\in\Gamma_1}\min_{y\in\Gamma_2}\rho(x,y)\right\}, \quad (1.3)$$

де Γ_1, Γ_2 – криві;

x, y – точки на кривих;

$\rho(x,y)$ – радіус кола із центром в точці x.

Графічне представлення вимірювання відстані Хаусдорфа між двома кривими представлено на рисунку 1.8.

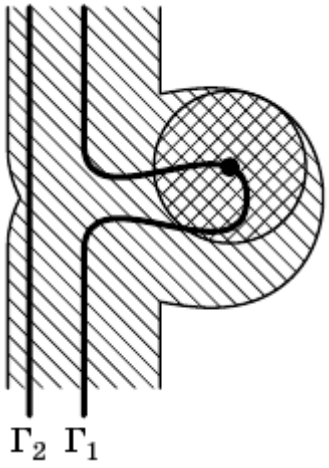


Рисунок 1.8 – Графічне представлення відстані Хаусдорфа

На рисунку 1.8 Γ_1 та Γ_2 – це криві, між якими шукається відстань. Заштрихована область – це шлях кола вздовж кривої Γ_1. Як можна побачити, крива Γ_2 перебуває у заштрихованій області, отже, радіус кола, представленого на рисунку, і буде хаусдорфовою відстанню між кривими. Дана метрика найкраще підходить для порівняння контурів двох зображень, оскільки, контур є кривою. Основним недоліком метрики є те, що враховуються лише масиви точок на порівнюваних, але не їхня послідовність.

Ще одною метрикою є метрика Фреше. Вона, на відміну від метрики Хаусдорфа, приймає до уваги неперервність кривих. Відстань Фреше знаходиться за формулою (1.4) [28].

$$F_r(P,Q) = \inf_{\alpha,\beta} \max_{t\epsilon[0,1]} \left|\left|P\big(\alpha(t)\big) - Q\big(\beta(t)\big)\right|\right|, \tag{1.4}$$

де P і $Q \to \mathbb{R}^2$ – дві криві, задані параметрично,

α, β [0,1] $\to$ [0,1] – безперервні монотонні функції.

Дана метрика використовується при порівнянні контурів двох областей, оскільки, перевіряє неперервність контуру. Для обчислення дискретної відстані Фреше, використовують апроксимацію кривих до полігональних кривих. Тобто, криві дискретизують із деяким інтервалом ε. В такому випадку, криві P та Q представляються наступним чином:

$$P = \big(u_{a_0}, v_{a_0}\big), \big(u_{a_1}, v_{a_1}\big), \dots, (u_{a_p}, v_{a_p}),$$
$$Q = \big(u_{b_0}, v_{b_0}\big), \big(u_{b_1}, v_{b_1}\big), \dots, (u_{b_q}, v_{b_q}),$$

де p та q – довжини кривих P та Q.

В дискретному випадку, відстань Фреше шукається між усіма можливими парами точок кривих P та Q. Перехід від параметричних кривих до полігональних кривих дає значне спрощення обчислень та підвищує швидкодію обчислень (найкращий алгоритм працює має швидкодію $O(p,q)$ [28-32]).

На основі вищерозглянутої інформації, використання метрик при порівнянні двох контурів є набагато вищими, ніж при суб'єктивному порівнянні. Тому, для точного порівняння контурів двох областей необхідно використовувати метрики.

1.4 Аналіз програмних засобів кількісної оцінки алгоритмів сегментації

На даний час існує велика кількість програм для кількісної оцінки сегментації зображення. Серед них є як безкоштовні, так і комерційні зразки.

Першою розглянутою програмою для аналізу зображень є JMicroVision. Дана програма розроблена для опису, кількісного вимірювання та класифікації всіх видів зображень. Має зрозумілий інтерфейс, велику кількість функцій, які допомагають працювати із великими зображеннями. Також, із підключеним мікроскопом, дозволяє проводити динамічний аналіз зразка. Призначений, в основному, для роботи з гірськими породами, але й може використовуватися в інших напрямках [33]. Основні характеристики JMicroVision наступні:

- читання зображень у форматах TIFF, BMP, FlashPix, GIF, JPEG, PNG;
- ефективна система візуалізації;
- кількісні компоненти: об'єкт або фон;
- аналіз об'єктів (розмір, форма, орієнтація, текстури);
- класифікація об'єктів;
- обробка зображень (фільтрація, сегментація і т. п.);
- інструменти для збору даних в одному або двох вимірах [33].

На рисунку 1.9 наведено вигляд вікна програми, в якому здійснюється вимірювання розмірів зерна породи [33].

–

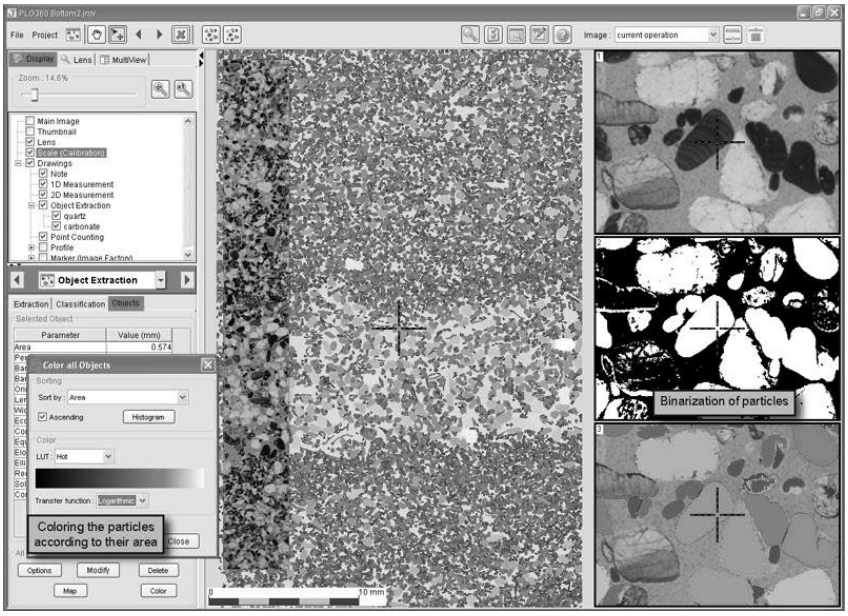


Рисунок 1.9 – Вікно програми JMicroVision

І ще однією програмою для аналізу зображень є NEXSYS ImageExpert Pro 3. Призначена для вирішення задач кількісного аналізу зображень мікроструктур в металографії, металоведенні та медицині. Аналізатор базується на потужних математичних алгоритмах та протестований на більш ніж ста підприємствах. Програма отримує широкий спектр геометричних параметрів, таких як:

– відсоткові долі складових;
– площі, периметри областей;
– мінімальні, максимальні та середні діаметри;
– параметри форми й витягнутості об’єктів;
– характеристики розподілу об’єктів;

характеристики анізотропії структур [34].

Дані характеристики доступні як для кожного об’єкта, так і для набору об’єктів (у вигляді статистичного набору). Ці характеристики програма може представити відповідно до міжнародних та російських стандартів. Для медичних зображень програма виконує аналіз морфології гістологічних зрізів (відсоткове співвідношення, ареальна гістограма міжклітинного простору). Також, програма виконує наступні операції з обробки зображень: бінаризація (поділ зображення на дві складові – об’єкт та фон), сегментація за кольором (виділення на зображенні

об’єктів, які знаходяться в певному колірному діапазоні), фільтрація об’єктів за геометричними параметрами (виділення об’єктів певного розміру) [34].

Наступною програмою для аналізу зображень є ImageJ. Це програма обробки зображень, написана мовою програмування Java і є у вільному доступі. JmageJ здатна прочитати такі формати, як TIFF, GIF, JPEG, BMP, DICOM, FITS та «raw». [35].

В ImageJ можна обчислювати площі, статистичні показники піксельних значень різних областей на зображенні, виділених вручну чи з допомогою порогових функцій. Програма може вимірювати відстані та кути. ImageJ підтримує стандартні функції обробки зображень, такі як логічні та арифметичні операції між зображеннями, маніпуляції з контрастністю, згортки, Фур’є-аналіз, підвищення чіткості, знаходження границь та медіанний фільтр. Програма дозволяє проводити різноманітні геометричні перетворення, такі як масштабування, поворот чи відображення. Вона підтримує будь-яку кількість одночасно використовуваних зображень. Обмеження пов’язані лишень із об’ємом вільної пам’яті [36].

Переваги ImageJ наступні:

– працює під будь-якою операційною системою – завдяки тому, що програма написана мовою Java, запускається на усіх операційних системах;

– відкритий код – програма доступна безкоштовно;

– наявність мови макросів – дозволяє автоматизовувати завдання користувача використовуючи макромову;

– наявність плагінів – дозволяє розширювати можливості програми вже готовими плагінами або власними;

– швидкодія – ImageJ є найшвидшою програмою для обробки зобарження в світі;

– виділення областей – наявний інструмент для прямокутного, еліптичного виділення або виділення області з неправильною формою;

– редагування зображень – наявні інструменти для вирізання, копіювання та вставки виділених областей; вставлення можна проводити із використанням AND, OR, XOR або змішаних моделей;

– аналіз зображень – інструменти для проведення вимірювання зони, класифікації, вимірювання відхилень та мінімального і максимального значення виділеної області [35].

На рисунку 1.10 представлено панель інструментів програми ImageJ, а на рисунку 1.11 – зображення, відкрите за допомогою цієї програми.

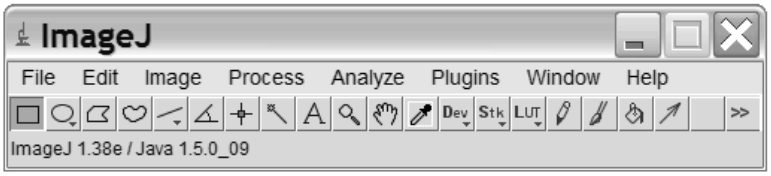


Рисунок 1.10 – Панель інструментів програми ImageJ

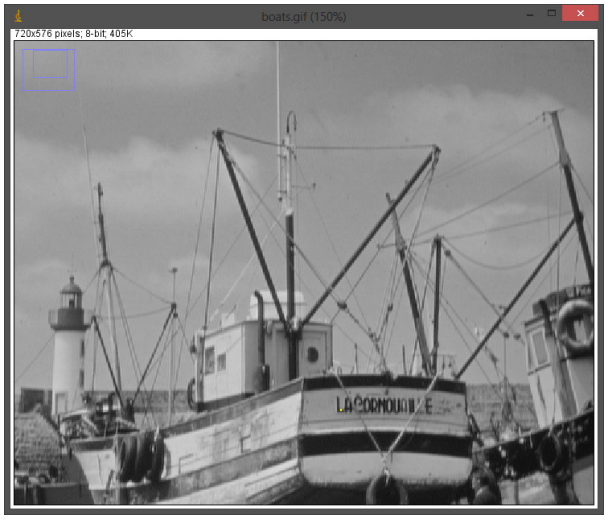


Рисунок 1.11 – Зображення, відкрите у програмі ImageJ

Отже, на основі порівняння переваг програм для обробки зображень, встановлено, що найкращою програмою є ImageJ. Вирішальними факторами є кросплатформність, багатопоточність, велика кількість реалізованих інструментів та безкоштовна ліцензія.

1.5 Висновки до розділу 1

У розділі проаналізовані біомедичні зображення на прикладі гістологічних зображень.

Біомедичні зображення – це растрові зображення, отримані з використанням будь-яких біомедичних методів, що використовуються для візуального та автоматичного аналізу в медицині та біології [37]. Для проведення експериментів з біомедичними цитологічними зображеннями дослідник повинен вручну виділити області, що цікавлять (клітини і цитоплазма). В даний час ця робота потребує багато часу. Щоб зменшити кількість і складність роботи, необхідно використовувати алгоритми сегментації зображень виділення ними однорідних ділянок. Проте основна проблема алгоритмів сегментації – це розрив між поділом алгоритмів автоматичної сегментації та сегментацією експертів.

У більшості практичних завдань якістю сегментації вважається ступінь близькості двох зображень:

У більшості практичних завдань якістю сегментації вважається ступінь близькості двох зображень: сегментованих експертом та сегментованих відповідними алгоритмами. Якість сегментації зображення можна визначити як у суб'єктивному (якісному), і на об'єктивному (кількісному) рівні [8]. Суб'єктивні критерії – критерії зорового сприйняття, отримані у процесі дослідження певною групою експертів. Об'єктивні критерії – це критерії, отримані в результаті порівняння (пошуку відмінностей) кількісних характеристик сегментованих та еталонних зображень (сегментованих експертом) [8]. Один із методів об'єктивного порівняння – використання метрик.

Зазвичай обрис області є кривою. Відстань між двома кривими переважно треба знаходити в метриці Фреше, тому що ця метрика дозволяє порівнювати дві криві один з одним з максимальною точністю. На відміну від аналогічних метрик вона враховує порядок точок на кривих і, отже, є хорошим інструментом для об'єктивного порівняння контурів зображення [8]. Щоб знайти найменшу відстань між кривими, необхідно використовувати модифікацію метрики Фреше – метрику Громова-Фреше.

2 АЛГОРИТМИ ПОРІВНЯННЯ КОНТУРІВ У МЕТРИЦІ ГРОМОВА-ФРЕШЕ

2.1 Метрика Громова-Фреше

У графіці плоскі фігури представляються кривими. Головною проблемою при порівнянні та розпізнаванні є встановлення подібності між двома заданими кривими [23]. Як згадувалося раніше, для об’єктивного порівняння двох кривих варто використовувати метрики.

Двома основними метриками для порівняння кривих є метрики Хаусдорфа та Фреше. Основною їхньою відмінністю є те, що друга метрика враховує порядок розташування точок. На рисунку 2.1 наведено дві криві, які порівнюються у даних метриках.

Рисунок 2.1 – Порівняння двох кривих у метриках

У метриці Хаусдорфа відстань між кривими, представленими на рисунку 2.1, буде рівною δ. Це пояснюється тим, що відстань Хаусдорфа бере до уваги набір точок двох кривих, але не враховує їхній напрямок [8]. Якщо важливим є порядок проходження по кривих, необхідно використати метрику Фреше. Дана метрика дозволяє не лише порівняти дві криві між собою, а й врахувати їхній напрямок.

Відстань Фреше представляється наступним чином: чоловік вигулює собаку на повідку: чоловік може рухатися по одній кривій, а собака – по іншій; обоє можуть рухатися з різними швидкостями, але повернення не дозволяється. В такому разі, відстань Фреше – це найкоротша довжина повідка, потрібна для проходу обома кривими [8]. Графічно дане пояснення зображено на рисунку 2.2.

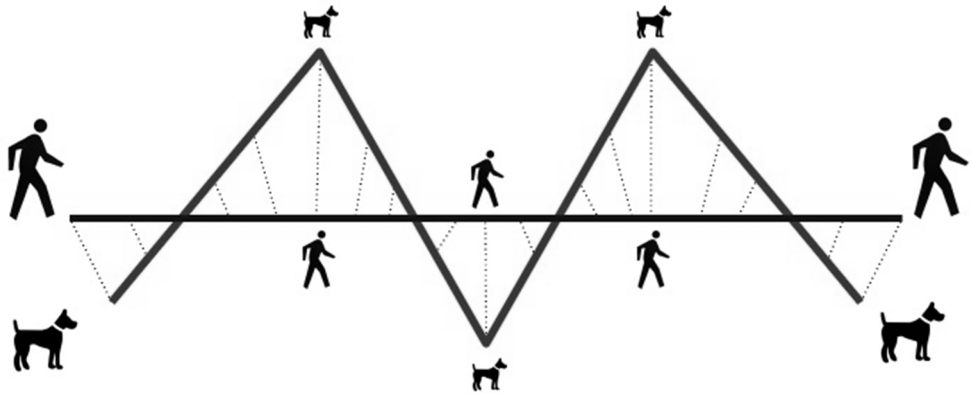


Рисунок 2.2 – Графічне представлення відстані Фреше

Математично, відстань Фреше описується так: нехай V – метричний простір і криві $f: [a, b] \to V$ та $g: [a', b'] \to V$ розміщені в даному просторі. Відстань Фреше для цих кривих знаходиться як мінімум усіх замін α і β на відрізку [0,1] від максимумів усіх відстаней $t \in [0,1]$, які належать V vs; $f(\alpha(t))$ і $g(\beta(t))$. Це виражається формулою (2.1):

$$d_F(f, g) = \inf_{\alpha,\beta} \max_{t\epsilon[0,1]} d\left(f\left(\alpha(t)\right), g\left(\beta(t)\right)\right), \tag{2.1}$$

де $d_F(f, g)$ – відстань Фреше між кривими f та g,

α і β – неперервні неспадні функції на відрізку [0, 1],

d – функція відстані метрики V.

Можна вважати, що параметр t – це час, $f(\alpha(t))$ – позиція собаки, а $g(\beta(t))$ – її господаря у час t. Довжина мотузки між ними в час t – це відстань між $f(\alpha(t))$ і $g(\beta(t))$. Отримуємо мінімум усіх можливих замін на відрізку [0,1], аналогічно до вибору проходження вздовж даного шляху, де максимальна довжина мотузки є мінімізованою. Обмеження полягає в тому, що α і β повинні бути неспадними функціями. Мається на увазі, що ні собака, ні її хазяїн не можуть повертатися назад [38].

Основною проблемою класичного визначення відстані Фреше є використання параметричних функцій при її знаходженні. Для підвищення швидкодії обрахунків та спрощенні створення програмного забезпечення, проводиться перехід від кривих до полігональних кривих. Полігональна крива – це

представлення кривої, шляхом розбиття її на відрізки. На рисунку 2.3 представлено апроксимацію частини синусоїди до полігональної кривої.

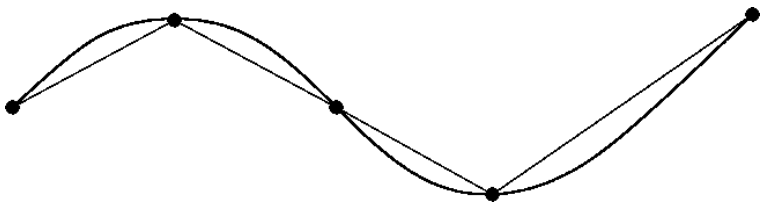


Рисунок 2.3 – Апроксимація синусоїди до полігональної кривої

Апроксимація кривої відбувається із відстанню, не більшою за деяке ε. Залежно від задання розміру ε, можна отримати полігональні криві різної детальності та подібності до оригіналу. На рисунку 2.4 представлено апроксимацію кривої із різними значеннями ε.

Рисунок 2.4 – Приклади апроксимації кривої з різними значеннями ε

У цьому випадку зображення відстані за допомогою собаки та її господаря замінюється двома жабами, які можуть розташовуватися відповідно на m-му та n-му камінні [39]. Камені є послідовністю точок відповідних ламаних f і g. Жаби стрибають із каменю на камінь, і повернення не допускається. Мінімальна довжина мотузки, яка з'єднує їх, але дозволяє стрибати по трасі, називатиметься дискретною відстанню Фреше. Математично дискретна відстань Фреше розраховується за формулою.

$$d_{dF}(f,g) = max\begin{Bmatrix} d_E(f_n, g_m) \\ min\begin{Bmatrix} d_{dF}(\langle f_1 \dots f_{n-1}\rangle, \langle g_1 \dots g_m\rangle), \forall n \neq 1 \\ d_{dF}(\langle f_1 \dots f_n\rangle, \langle g_1 \dots g_{m-1}\rangle), \forall m \neq 1 \\ d_{dF}(\langle f_1 \dots f_{n-1}\rangle, \langle g_1 \dots g_{m-1}\rangle), \forall n \neq 1, m \neq 1 \end{Bmatrix} \end{Bmatrix}, \quad (2.2)$$

де $d_{dF}(f, g)$ – дискретна відстань Фреше,

$d_E(f_n, g_m)$ – евклідова відстань між точками.

Дискретна відстань Фреше відрізняється від класичної відстані Фреше. Результат розрахунку лежатиме в діапазоні:

$$d_F(f, g) \leq d_{dF}(f, g) \leq d_F(f, g) + L_{max},$$

де L_{max} – довжина найдовшого відрізка ламаної.

Існує кілька алгоритмів обчислення дискретної відстані Фреше. Перший – був запропонований Альтом і Годом [27]. Однак цей алгоритм дуже складно зрозуміти та запрограмувати. Його обчислювальна складність становить $O(pq\ log^2 pq)$. Ще більш поширений метод був запропонований Томасом Ейтером та Хейкі Манілою [28]. Його легко запрограмувати, а складність виконання – $O(pq)$. Нижче наведено псевдокод цього алгоритму.

```
Function dF(P,Q): real;
  input: polygonal curves P = (u1, ..., up) and Q = (v1,
..., vq).
  return: _dF (P,Q)
  ca : array [1..p, 1..q] of real;
  function c(i, j): real;
    begin
      if ca(i, j) > -1 then return ca(i, j)
      elsif i = 1 and j = 1 then ca(i, j) := d(u1, v1)
      elsif i > 1 and j = 1 then ca(i, j) := max{ c(i - 1,
1), d(ui, v1) }
```

```
        elsif i = 1 and j > 1 then ca(i, j) := max{ c(1, j -
    1), d(u1, vj) }
        elsif i > 1 and j > 1 then ca(i, j) :=
          max{ min(c(i - 1, j), c(i - 1, j - 1), c(i, j - 1)),
      d(ui, vj) }
        else ca(i, j) = 1
       return ca(i, j);
       end; /* function c */
     begin
       for i = 1 to p do for j = 1 to q do ca(i, j) := -1.0;
       return c(p, q);
end.
```

Даний алгоритм можна описати наступним чином: нехай P [0, n], Q[0, m] – полігональні криві. $\sigma(P) = (u_1, u_2, \ldots, u_p)$, $\sigma(Q) = (v_1, v_2, \ldots, v_q)$ – сегменти відповідних кривих. Сукупність L між P та Q є наступною послідовністю:

$$L = (u_{a_1}, v_{b_1}), (u_{a_2}, v_{b_2}), \ldots, (u_{a_m}, v_{b_m}), \text{ де}$$

$a_1 = 1, b_1 = 1, a_m = p, b_m = q.$

Сукупність довжин послідовності L позначається $||L||$ та знаходиться за наступною формулою:

$$\|L\| = \max_{i,j=1,\ldots,m} d(u_{a_i}, v_{b_j}),$$

де $d(u_{a_i}, v_{b_i})$ – відстань між точками u_{a_i} та v_{b_j}.

$d(u_{a_i}, v_{b_i})$ знаходиться із наступних умов:

1) якщо i=1 та j=1, то дана відстань знаходиться, як Евклідова відстань між точками: $dE = \sqrt{(v_{b_j} - u_{a_i})^2}$;

2) якщо i>1 та j=1, тоді відстань знаходиться за наступною формулою: $\max\{d(u_{a_{i-1}}, v_1), dE(u_{a_i}, v_1)\}$;
3) якщо i=1 та j>1, тоді відстань знаходиться за наступною формулою: $\max\{d\left(u_1, v_{b_{j-1}}\right), dE\left(u_1, v_{b_j}\right)\}$;
4) якщо i>1 та j>1, тоді відстань знаходиться за наступною формулою: $\max\{\min\left(d\left(u_{a_{i-1}}, v_{b_j}\right), d\left(u_{a_{i-1}}, v_{b_{j-1}}\right), d\left(u_{a_i}, v_{b_{j-1}}\right)\right), dE\left(u_1, v_{b_j}\right)\}$;

Таким чином, шукаємо максимальну з відстаней між першою точкою з кривої P та усіма точками кривої Q. У даному випадку відстань Фреше знаходиться за формулою:

$$\delta_{dF}(P, Q) = \min_{i=1,\dots,m} \|L\|_i.$$

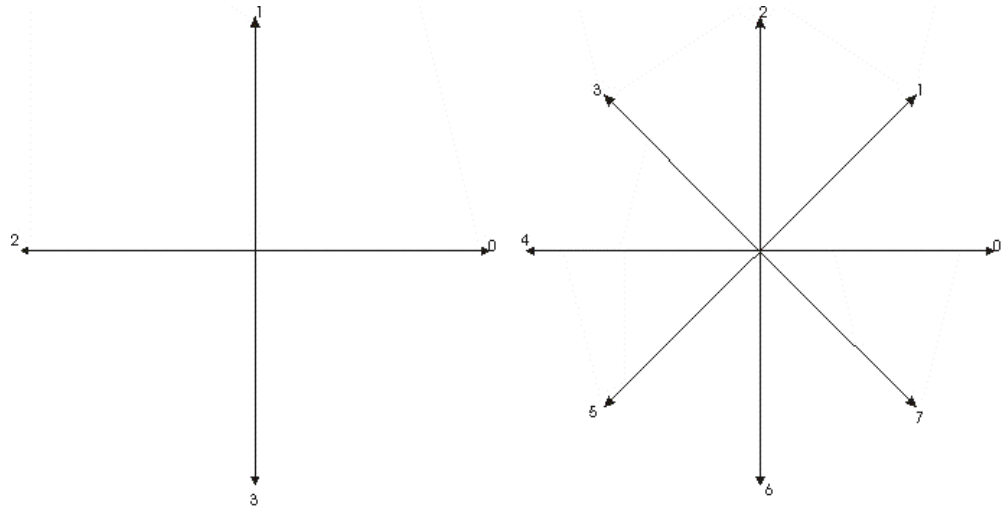
В результаті, отримуємо мінімальну зі всіх максимальних відстаней і вона є відстанню між кривими у метриці Фреше.

Метрика Громова-Фреше – це вдосконалення класичної метрики Фреше, яке зроблене Михайлом Громовим. На відміну від класичної відстані Фреше, дана метрика дозволяє знайти найменшу відстань між двома кривими. Основною модифікацією є введення ізометричних перетворень, які необхідно виконати перед пошуком відстані Фреше. До цих перетворень належать переміщення та поворот на кут.

Порівняно із алгоритмом пошуку відстані Фреше, алгоритм Громова-Фреше включає додатково дві операції для виконання ізометричних перетворень. Тому, останній алгоритм є набагато складніший. Для зменшення його обчислюваної складності, спочатку необхідно визначити, який із контурів містить меншу кількість точок. Над цим контуром і необхідно виконувати перетворення.

Щоб мати змогу порівнювати контури зображень, спочатку необхідно знайти контури. Контур – це множина точок, які мають по сусідству як фонові пікселі, так і пікселі об’єкта. Для знаходження контуру існує декілька алгоритмів, які базуються на попіксельному проходженні по зображенню та його розподіл на контур та фон. Цифрові зображення, зазвичай, реєструються та обробляються у

вигляді сітки з однаковим кроком дискретизації в напрямках осей x та y. Контур при цьому представляють ланцюговим кодом, який будують шляхом прослідковування границі, наприклад, за годинниковою стрілкою. Напрямки зв'язків між сусідніми пікселями нумеруються відповідно до схеми, представленої на рисунку 2.5 [1].



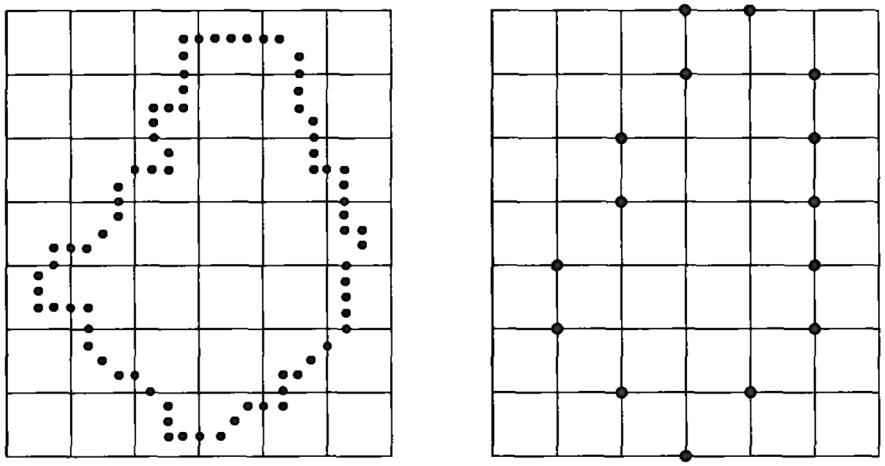
а б

Рисунок 2.5 – Приклад нумерації зв'язків між сусідніми пікселями: *а* – 4-зв'язний код; *б* – 8-зв'язний код

В такого методу є два основних недоліки:

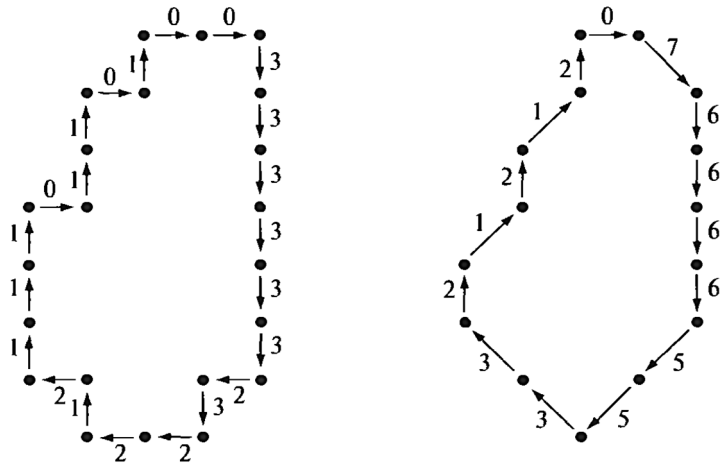
1) ланцюговий код може вийти дуже довгим;
2) наявність шумів призводить до зміни форми границі об'єкта.

Для обходу таких проблем, використовують повторну дискретизацію зі збільшенням кроку сітки. Після цього знову проводиться обхід знайденого контуру, але вже відносно нової сітки. На рисунку 2.6 представлено проходження даного процесу [1].



а *б*

Рисунок 2.6 – Знаходження контуру при збільшеному кроці сітки: *а* – початковий контур; *б* – новий контур



а *б*

Рисунок 2.7 – Представлення контуру: *а* – 4-зв’язним; *б* – 8-зв’язним ланцюговим кодом

При цьому, даний контур можна представити як 4-зв’язним, так і 8-зв’язним кодом. На рисунку 2.7 наведено контур, представлений 4-зв’язним та 8-зв’язним ланцюговим кодом [1].

2.2 Алгоритми обходу та проріджування контуру

Для знаходження контуру, крім представлення, необхідно здійснити його обхід. Для обходу контуру існує декілька алгоритмів. Першим розглянутим алгоритмом є алгоритм квадратів. Основна ідея полягає в тому, якщо під час попіксельного обходу контуру перебуваємо на пікселі фону, потрібно повернути праворуч. Якщо на пікселі об'єкта – ліворуч. Виконується доти, поки не досягнемо початкового пікселя. Вхідними даними для алгоритму квадратів є зображення T, яке містить пікселі об'єкта (наприклад, чорного кольору) P. Псевдокод даного алгоритму наведено нижче [40].

Begin

1. *B* присвоюється порожня множина.
2. Зверху-донизу та зліва-направо виконати прохід по пікселям із *T*, поки не буде знайдено піксель чорного кольору *s* із множини *P* не буде знайдено.
3. Додаємо s до B.
4. Маркуємо поточний піксель p стартовим пікселем s.
5. Повертаємо наліво, перевіряючи лівого сусіда пікселя p.
6. Оновлюємо p, замінюючи його поточним пікселем.
7. while p != s do

if p is black

Додаємо p до B і повертаємо наліво.

Оновлюємо p, замінюючи його поточним пікселем.

else

Повертаємо направо.

Оновлюємо p, замінюючи його поточним пікселем.

end while

End

В результаті роботи такого алгоритму, у множині B будуть розміщені точки контуру об'єкта. Даний алгоритм є дуже простим у реалізації. Основним недоліком алгоритму квадратів є проблема із визначенням завершення роботи. Оскільки,

можуть бути випадки, коли алгоритм повернеться в початкову точку так і не обійшовши весь контур. В такому випадку є два варіанти вирішення цієї проблеми:

1) зупиняти алгоритм, коли він доходить до початкової точки два або більше рази;
2) зупиняти алгоритм при доходженні до початкової точки другий раз, при цьому, з такої ж передостанньої точки [40].

Наступним алгоритмом знаходження контурів є окіл Мура (Moore-Neighbor Tracing). Даний алгоритм базується на понятті околу Мура. Для центрального пікселя *P* це буде набір із восьми пікселів $P_1 ... P_8$, які його оточують (рисунок 2.8).

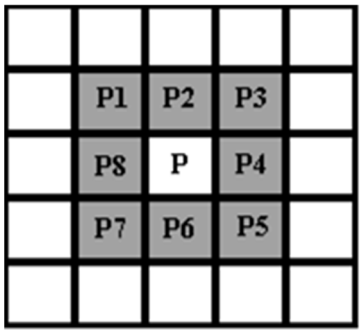


Рисунок 2.8 – Окіл Мура

Для роботи алгоритму необхідне зображення, представлене групами пікселів чорного (об'єкт) та білого (фон) кольорів. Спочатку, необхідно знайти перший піксель об'єкту та маркувати його як стартовий. Із цього пікселя необхідно розпочати обхід по околу Мура за чи проти годинникової стрілки. Коли буде знайдено наступний піксель чорного кольору, він маркується контуром та стає центром наступного околу Мура. Обхід контуру завершується, коли стартовий піксель буде відвіданий двічі.

Алгоритм знаходження контуру за допомогою околу Мура наступний [41]:

Begin

1. *B* (послідовність контурних пікселів) присвоюється порожня множина.
2. Зверху-донизу та зліва-направо виконати прохід по пікселям із *T*, поки не буде знайдено піксель чорного кольору *s* із множини *P* не буде знайдено.
3. Додаємо *s* до B.
4. Маркуємо поточний піксель p стартовим пікселем *s*.

5. Повертаємося назад, тобто, повертаємося на піксель, з якого попали на *s*.
6. Присвоїмо *c* наступний піксель із околу Мура *M(p)* за годинниковою стрілкою.
7. while c != s do

   if *c* is black

   Додаємо *c* до B

   Встановлюємо *p=c*

   Повертаємося назад, на піксель, з якого прийшли в *c*

   else

   Пересуваємо поточний піксель *c* до наступного за годинниковою стрілкою пікселя із *M(p)*

   end while

End

Щоб запобігти зациклюванню при проходженні стартового пікселя нескінченну кількість разів, Якоб Еліософ запропонував додати такий критерій зупинки: завершувати обхід контуру, якщо стартовий піксель досягнутий двічі однаковим способом [41]. Таким чином, алгоритм обходу контуру околом Мура є найбільш ефективним для знаходження контурів об'єктів на зображеннях.

Після знаходженні контуру об'єкта, ми отримаємо множину точок. Проте, для великого контуру цих точок буде дуже багато. Тож, необхідно використати алгоритм проріджения контуру. Найпростішим методом проріджения контуру є видалення фіксованої кількості точок. Наприклад, видаляти п'ять пікселів через один. Проте, при такому підході, можна втратити деякі деталі зображення.

Надійнішим методом проріджения контуру є кусково-лінійна апроксимація. Під цим поняттям розуміють розбиття кривої на сегменти та послідовну заміну кожного з них прямим відрізком. При цьому, необхідно, щоб кількість сегментів було якомога меншим, а відмінність між сегментом кривої та відповідним прямолінійним відрізком не перевищувало заданий поріг. При неправильно обраному порозі крива може бути апроксимована неточно. Для прикладу, на рисунку 2.9 представлено неправильно апроксимовану криву [42]. Тут криву було апроксимовано до прямої лінії.

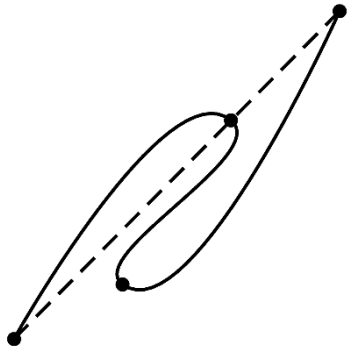


Рисунок 2.9 – Приклад неправильної апроксимації

Найефективнішим алгоритмом апроксимації кривих є алгоритм Рамера-Дугласа-Паркера. Він призначений для зменшення кількості точок у кривій. Основна мета алгоритму полягає, при вхідній кривій, яка складається із лінійних сегментів, знайти подібну криву з меншою кількістю точок. Алгоритм визначає «неподібність» базуючись на максимальній дистанції між оригінальною та спрощеною кривими. Спрощена крива складається із підмножини точок, які належать оригінальній кривій.

Алгоритм прорідження наступний: на вхід поступає початкова крива (впорядкований набір точок) та максимальна відстань ε. Далі, перша та остання точки кривої з'єднуються прямою. Визначається точка, яка знаходиться найдалі від прямої. На наступному кроці прямою з'єднується перша точка та точка, обрана на попередньому кроці. Якщо точки знаходяться під або над цією прямою на відстані меншій за ε, тоді вони видаляються. Інакше, дана точка маркується для залишення та до неї проводиться пряма і алгоритм повторяється. На виході, отримуємо криву, яка складатиметься тільки із тих точок, які марковані для залишення.

Алгоритм можна представити наступним псевдокодом:

PointList – набір точок кривої

dmax – максимальна дистанція

index – номер точки для перевірки

end = length(PointList) – кінцева точка кривої

Begin

for i = 2 to (end - 1)

```
    d = shortestDistanceToSegment(PointList[i], Line(PointList[1], PointList[end]))
    if ( d > dmax )
      index = i
      dmax = d
  // Якщо відстань більша за максимальну відстань, викликаємо рекурсивно цей алгоритм
    if ( dmax > epsilon )
      recResults1[] = DouglasPeucker(PointList[1...index], epsilon)
      recResults2[] = DouglasPeucker(PointList[index...end], epsilon)
      // Зберігаємо список результатів
      ResultList[] = {recResults1[1...length(recResults1)-1], recResults2[1... length(recResults2)]
    else
      ResultList[] = {PointList[1], PointList[end]
  // Повертаємо результат
    return ResultList[]
  end
```

Вхідні дані для роботи алгоритму – це упорядкований набір точок та максимальна відстань ε. Алгоритм рекурсивно ділить контур. На вхід алгоритму подаються координати всіх точок, включно з першою і останньою, а також відстань ε. Першу і останню точки залишаємо незмінними. За алгоритмом знаходиться точка, яка найбільш віддалену від відрізка, кінці якого перша і остання точки. Якщо точка знаходиться на відстані, меншій, ніж ε, то всі точки, які ще алгоритм не позначив для збереження, викидаються з набору, і отримуємо пряму, яка згладжує криву з точністю не нижчою ε. Якщо ж відстань є більшою за ε, то алгоритм працює на наборі точок відрізків з кінцями від початкової до даної і від даної до кінцевої точок (дана точка буде позначена для збереження). По закінченню всіх рекурсивних викликів результуюча контур будується тільки з тих точок, що були позначені для збереження [43]. На рисунку 2.10 представлено вигляд кривої на кожному з етапів спрощення алгоритмом Рамера-Дугласа-Паркера [44-45].

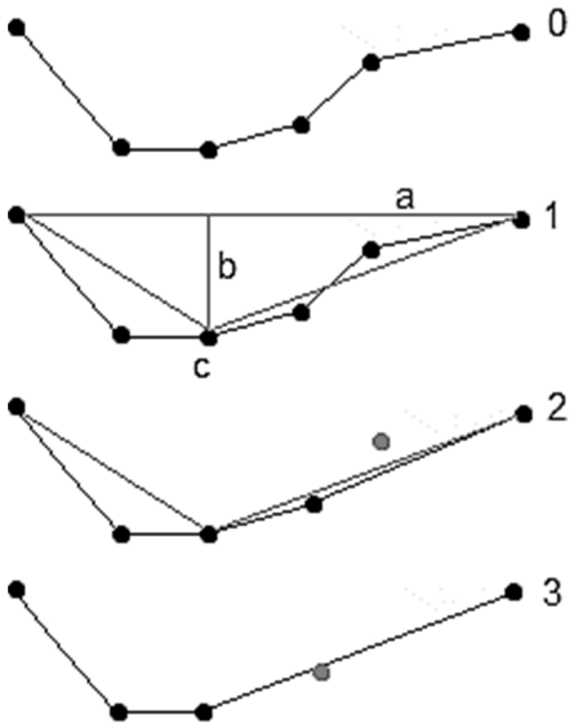
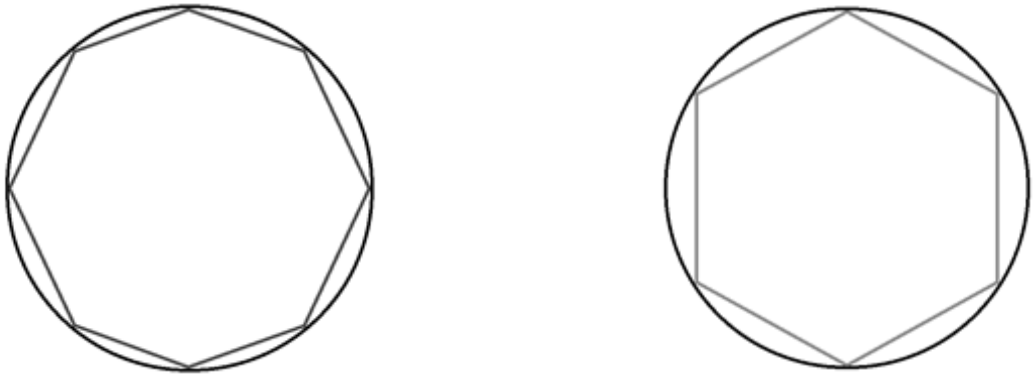


Рисунок 2.10 – Спрощення полігональної кривої алгоритмом етапи Рамера-Дугласа-Паркера

Загалом, алгоритм проріджения контуру може значно зменшити кількість точок, які утворюють контур. Але, при цьому, може втрачатися оригінальна форма контуру. Для того, щоб зберегти форму та зменшити кількість точок в контурі, необхідно підібрати оптимальне ε. Складність даного алгоритму становить $O(n \log n)$.

Основним недоліком алгоритму Рамера-Дугласа-Паркера є те, що він не зберігає топологію кривої, тобто, на виході можна отримати лінію із самопересіченням. На додачу, результат може бути не завжди оптимальним. На рисунку 2.11 представлено результат роботи алгоритму (*а*) та оптимальну апроксимацію (*б*).



а б

Рисунок 2.11 – Неоптимальна апроксимація кола (*а*) та оптимальне рішення (*б*)

Таким чином, при незадовільному результаті спрощення варто виконати повторний прохід зі збільшеним максимальним порогом ε. Таким чином можна добитися значного спрощення полігонального ланцюга.

2.3 Алгоритми проведення ізометричних перетворень

Для пошуку відстані в метриці Громова-Фреше, необхідно провести ізометричні перетворення: паралельний перенос та поворот на кут. Це необхідно виконати для знаходження такого розміщення контурів, при якому їхнє накладання буде максимальним.

Паралельний перенос – це перетворення, при якому всі точки об’єкта переміщуються в одному напрямку та на однакову відстань. На рисунку 2.12 зображено паралельний перенос контуру X в позицію X'. Математично паралельний перенос пікселів можна представити наступною формулою:

$$(x', y') = (x + a, y + b),$$

де *x’, y’* – нове положення пікселя,

x, y – вхідне положення пікселя,

a, b – відстані, на які потрібно змістити *x* та *y* відповідно.

Для виконання операції накладання двох контурів, спочатку необхідно обчислити їхні центри мас. Координати центру мас $(C_x;C_y)$ знаходяться як середнє арифметичне усіх координат контурів:

$$C_x = \frac{\sum_{i=1}^{N}(x_i)}{N}$$
$$C_y = \frac{\sum_{i=1}^{N}(y_i)}{N} \quad (2.3)$$

Наступним кроком є накладання центру мас контурів на позицію (0;0), тобто, на початок координат 2D-простору. Для контуру *P* дана операція матиме вигляд:

$$P'[i] = P[i] - \left[C_x C_y\right]^T, \quad (2.4)$$

де *P'[i]*– трансформований результат переносу *P[i]*.

Таким чином, центри мас різних контурів можна накласти одне на одного. Наступним кроком, для отримання найбільшого перекриття між контурами, є поворот одного з контурів навколо центру мас. Для виконання повороту на кут θ необхідно перемножити координати пікселя на матрицю повороту (2.5).

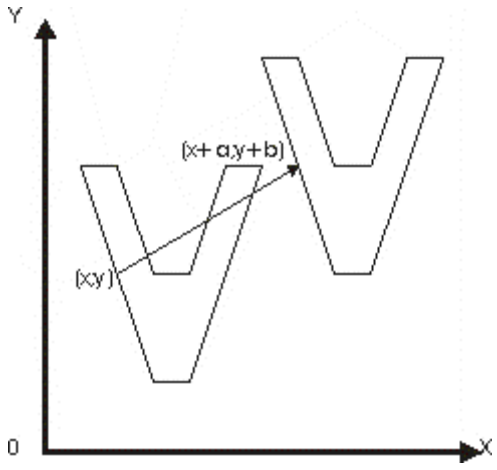


Рисунок 2.12 – Паралельний перенос контуру

$$R = \begin{pmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{pmatrix} \quad (2.5)$$

Після кожного повороту на 1 градус, виконується обчислення відстані Фреше між контурами. Зі всіх обчислених відстаней знаходиться мінімальна. Така відстань і буде відстанню між контурами в метриці Громова-Фреше.

Алгоритм пошуку відстані між контурами в метриці Громова-Фреше наступний:

1) проводиться знаходження контуру зображень;
2) контури проріджуються;
3) із двох контурів обирається той, кількість пікселів у якому найменша;
4) обчислюються центри мас контурів;
5) проводиться паралельне перенесення обраного контуру;
6) обчислюється кут повороту θ;
7) виконується поворот контуру на кут θ;
8) знаходиться відстань Фреше в даній позиції.

В результаті, отримане число і буде відстанню в метриці Громова-Фреше. Процес порівняння контурів у метриці Громова-Фреше представлено на рисунку 2.13.

а б в

Рисунок 2.13 – Процес порівняння контурів у метриці Громова-Фреше. *а)* контури двох зображень A і B; *б)* накладання та поворот другого контуру на перший; *в)* знаходження відстані Фреше в позиції максимального перекриття

Основним недоліком такого методу є те, що поворот та перерахунок значень займатимуть велику кількість часу. Тому, необхідно спростити цей процес.

2.4 Алгоритми скелетонізації області

Для спрощення проведення операцій ізометричних перетворень запропоновано використовувати попередню скелетонізацію області, обмеженої контуром. Таким чином, область можна представити у вигляді графа і тоді виконувати ізометричні перетворення уже з графом.

Скелет двовимірної замкнутої області – це сукупність усіх точок області, кожна з яких має принаймні дві еквідистантні точки, що є найбільш близькими до контуру цієї області [46]. Щоб знайти каркас (скелет) ділянки необхідно використовувати алгоритми проріджування. Добре відомі методи побудови каркасу поділяються на методи, що розглядають контур області як набір багатокутників, та методи, що працюють з растровим представленням області. Растрові методи називаються методами скелетонізації або потоншення (проріджування), а побудований за ними каркас складається з множини пікселів і є деякою апроксимацією каркаса двомірної замкнутої області.

Скелет, побудований методами скелетонізації, має бути товщиною в один піксель і має бути з'єднаний для створення пов'язаної двомірної області [47].

Для роботи із опуклими областями найкраще підходять алгоритми, які працюють із растровим поданням області. Такі алгоритми точно дають результат за кінцеву кількість кроків. Одним із методів скелетонізації – це перетворення по головним осям, яке було запропоноване Блюмом. В даному випадку розглядається область G із границею C. Для кожної із точок $p \in G$ шукають найближчу точку із B. Якщо таких точок більше, ніж одна, то p лежить на серединній осі G [1]. На рисунку 2.14 представлені приклади скелетів областей.

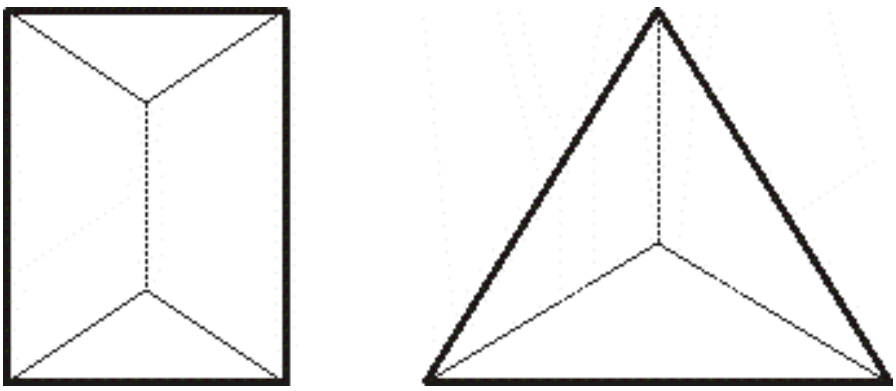


Рисунок 2.14 – Приклади скелетів прямокутної та трикутної областей

Робота алгоритму перетворення по головним осям приводить до отримання хороших скелетів. Проте, для виконання цього алгоритму витрачається велика кількість часу, оскільки, необхідно обчислювати відстані від кожної з внутрішніх точок області до всіх точок контуру. Для підвищення швидкодії роботи такого алгоритму можна поступово видаляти точки області, якщо вони задовольняють такі умови: не є кінцевими точками; після видалення область залишається зв'язною; видалення не призводить до надлишкової ерозії області [1].

Основним поняттям при знаходженні скелетону є особливий піксель. Це такі пікселі, в яких відбувається видима зміна форми області. Для знаходження особливих пікселів використовують наступний алгоритм:

1) Вхід: *C* – точки контуру, *s* – інтервал, *Ф* – порогове значення кута.
2) for *p* in *C*
– знаходяться пікселі *(p-s)* та *(p+s)*;
– знаходиться кут між ними;
– якщо кут більший за *Ф*, то *p* – особлива точка
3) end for

Таким чином, на виході отримаємо масив особливих точок контуру.

Для знаходження особливих точок існує велика кількість алгоритмів. Першим розглянутим алгоритмом є алгоритм Моравеца. Він є дуже простим у реалізації. В даному алгоритмі розглядається зміна яскравості квадратного вікна (3x3, 5x5 чи 7x7 пікселів) відносно досліджуваної точки при зсуві вікна на один піксель у восьми напрямках [48]. Алгоритм наступний:

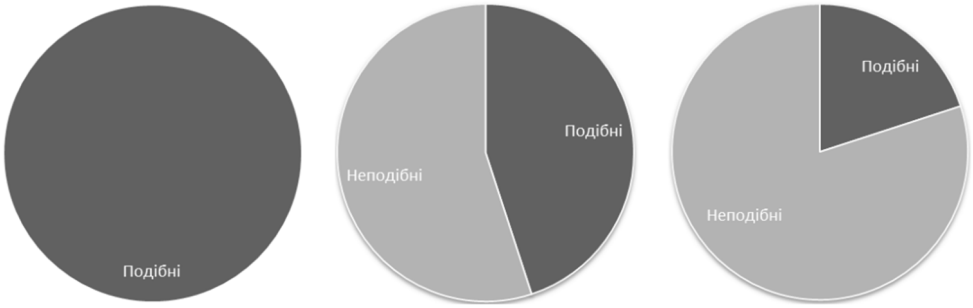
1) для кожного пікселя контуру *(x, y)* знаходиться зміна яскравості за формулою:

$$V_{u,v}(x,y) = \sum_{\forall a,b \in W}\big(I(x+u+a, y+v+b) - I(x+a, y+b)\big)^2;$$

2) знайти напрями ребер кута: $C(x,y) = \min\left(V_{u,v}(x,y)\right)$;
3) відсіяти пікселі, значення *C(x,y)* яких менше за поріг *T*;

4) видалити кути, які повторюються.

Іще одним алгоритмом знаходження особливих пікселів є алгоритм SUSAN. В ньому для кожного пікселя розглядається кругова область фіксованого радіусу. Центральний піксель називається ядром, і його значення яскравості запам'ятовується. Усі інші пікселі поділяються на дві групи: схожі та несхожі, залежно від подібності до яскравості ядра. Якщо круг складається повністю із подібних пікселів, то знайдено піксель із однорідної області. На гранях таке співвідношення між схожими та несхожими пікселями становитиме приблизно 50%, а на кутах – падатиме до приблизно 25%.



а б в

Рисунок 2.15 – Приклад роботи алгоритму SUSAN: *а* – однорідна область, *б* – грань, *в* – кутовий піксель

Алгоритм роботи наступний [49]:

1) розмістити центр кругової маски в ядро;
2) всередині маски знайти кількість пікселів, які мають схожу яскравість до яскравості ядра за наступною формулою:

$$c(\vec{r}, \overrightarrow{r_0}) = e^{\left(\frac{I(\vec{r}) - I(\overrightarrow{r_0})}{t}\right)^6},$$

де $\vec{r}$ – точка всередині маски,

$\overrightarrow{r_0}$ – центр ядра,

$I(\vec{r})$ – яскравість точки $\vec{r}$,

t – різниця яскравостей,

$c(\vec{r}, \overrightarrow{r_0})$ – результат порівняння.

3) обчислити розмір області точок зі схожою яскравістю за формулою:

$$R(\overrightarrow{r_0}) = \begin{cases} g - n(\overrightarrow{r_0}), \text{якщо } n(\overrightarrow{r_0}) < g \\ 0, \text{в інших випадках} \end{cases},$$

де $R(\overrightarrow{r_0})$ – значення відгуку краю; чим область подібних пікселів менша, тим відгук більший,

g – геометричний поріг,

$n(\overrightarrow{r_0})$ – кількість подібних пікселів.

4) вибрати особливі точки на основі максимумів функції відгуку.

Після знаходження особливих точок, необхідно використати стандартний алгоритм скелетонізації. Даний алгоритм полягає у послідовному проходженні по всім точкам контуру маскою 3×3, представленою на рисунку 2.7. При цьому, точка, яка відповідає наступним умовам, видаляється:

- $2 \le N(P) \le 6, N(P) = \mathrm{p}_1 + \mathrm{p}_2 + \ldots + \mathrm{p}_8;$
- $T(P) = 1;$
- $p_2 \times p_4 \times p_6 = 0;$
- $p_4 \times p_6 \times p_8 = 0.$

На другому кроці роботи алгоритму, перші дві умови залишаються такими ж, а останні замінюються на наступні:

- $p_2 \times p_4 \times p_8 = 0;$
- $p_2 \times p_6 \times p_8 = 0.$

Точки, призначені для видалення, встановлюються в 0. Точки скелету – в 1. В результаті, отримується скелетон області. Основною перевагою такого алгоритму є зменшення обчислювальної складності операцій ізометричного перетворення, оскільки кількість точок в скелетоні є меншою, ніж у контурі.

2.5 Алгоритм знаходження оптимального кута повороту

Скелетонізація області – процес складний та довгий. Для пониження складності програми та підвищення її швидкодії необхідний кращий алгоритм знаходження кута повороту. Найпростішим та найшвидшим способом знайти оптимальний кут повороту області є алгоритм максимальної хорди. Даний алгоритм представлено нижче:

1) шукається максимальна відстань між точками контуру (максимальна хорда);

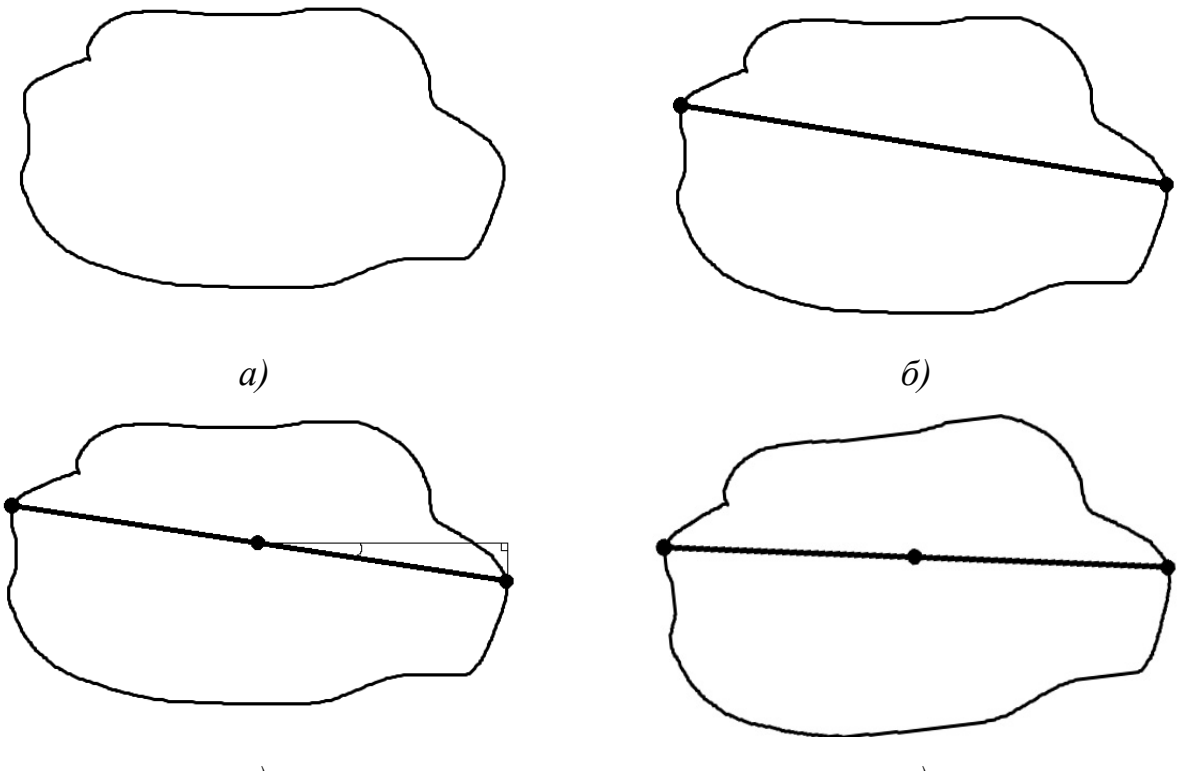
2) опускається перпендикуляр з одного кінця відрізку на вісь *OX*;

3) знаходиться кут, утворений одним із кінців відрізка та його серединою до осі *OX*;

4) проводиться корекція знаку кута для повороту;

5) отриманий кут є кутом повороту до осі абсцис.

Якщо накласти конури по центрах їхніх мас, а тоді паралельно повернути на отриманий кут, в результаті отримаємо максимальне перекриття контурів. Процес роботи алгоритму знаходження кута повороту представлено на рисунку 2.16.



а) *б)*

в) *г)*

Рисунок 2.16 – Знаходження кута повороту: *а)* вхідне зображення; *б)* знаходження максимальної відстані між точками контуру; *в)* знаходження кута повороту та коригування; *г)* виконання повороту

Завдяки виконанню повороту контуру лише один раз, складність такого алгоритму становитиме *O(2*p*q)*, оскільки потрібно повернути обидва контури.

Після вибору необхідних для використання алгоритмів, алгоритм порівняння контурів зображень в метриці Громова-Фреше наступний:

1) проводиться знаходження контуру зображень (алгоритм околу Мура);
2) контури проріджуються (алгоритм Рамера-Дугласа-Паркера);
3) проводиться паралельне перенесення першого контуру на другий із накладанням їхніх центрів мас;
4) знаходиться кут повороту θ контурів до осі *OX* (алгоритм максимальної хорди);
5) виконується поворот скелетона на кут θ;
6) знаходиться відстань Фреше в даній позиції.

Отже, в результаті аналізу метрики Громова-Фреше встановлено, що для порівняння двох зображень у даній метриці, необхідно виділити контури об’єктів на них. Для знаходження та представлення контурів зображень у вигляді числової послідовності обрано алгоритм обходу контуру околом Мура. Даний алгоритм є точнішим та швидшим, порівняно з іншими алгоритмами обходу контуру. Для прорідження контуру обрано алгоритм Рамера-Дугласа-Паркера, оскільки він дає оптимальні результати за малий час роботи. Для виконання ізометричних перетворень обрано алгоритми знаходження центру мас, паралельного перенесення та повороту на кут. Для знаходження кута повороту контуру до осі *OX* обрано алгоритм максимальної хорди. Також, представлено алгоритм порівняння контурів зображень в метриці Громова-Фреше, що дає змогу створити програмне забезпечення для цих цілей.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ДОСЛІДЖЕННЯ

3.1 Вимоги до програмного продукту

Необхідною умовою побудови будь-якого програмного забезпечення є встановлення вимог до нього. Дані вимоги представлені у відповідності до шаблону SRS Секції 3, організованої режимом: Версія 1.

1. Основні функціональні вимоги до комп'ютерної програми:
   1.1. Робота із зображеннями форматів JPEG, PNG та GIF;
   1.2. Необхідно прийняти два зображення як вхідні дані та виконати їх паралельну обробку;
   1.3. Дослідження та спрощення контурів на сегментованих зображеннях;
   1.4. Порівняння контурів зображень у метриці Громова-Фреше;
   1.5. Результат роботи програми повинен відображатись у консолі;
   1.6. Шляхи введення зображення вводяться через консоль.
2. Вимоги до надійності:
   2.1. Програма має працювати з великими зображеннями без збоїв;
   2.2. Запобігти помилкам при нестачі пам'яті;
   2.3. Спрогнозувати результат помилок, коли вони виникнуть.
3. Вимоги до робочого місця:
   3.1. Система повинна працювати на IBM-сумісних робочих станціях;
   3.2. Мінімальні вимоги до робочих станцій: процесор із частотою 1,8 ГГц із числом ядер більше 1, клавіатура, екран;
   3.3. Оперативна пам'ять від 1 Гб;
   3.4. Наявність дискретної відео карти nVidia з підтримкою технології CUDA;
4. Програмні вимоги:
   4.1. Операційна система – система Unix, Windows;
   4.2. Доступність Java;
   4.3. Розмір програми до 10МБ;

4.4. Код програмних модулів повинен мати коментарі, необхідні для розуміння.

3.2 Архітектура системи

Проаналізувавши технічні вимоги, проводиться проектування та опис архітектури системи. Для цього необхідно розділити програму на незалежні компоненти, які виконуватимуть певну частину від поставленого завдання. На рисунку 3.1 представлено приблизний розподіл програми для порівняння контурів зображень на окремі модулі.

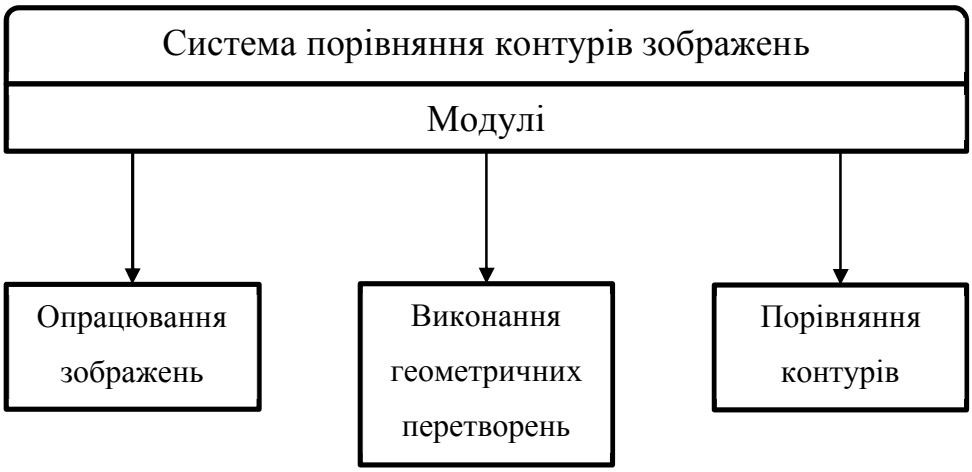


Рисунок 3.1 – Розподіл системи на модулі

Кожен із представлених модулів виконує набір незалежних операцій над вхідними зображеннями. Модуль опрацювання зображень проводить завантаження, порогову обробку, пошук контурів та їхнє прорідження. Модуль виконання геометричних перетворень виконує операції зсуву та повороту даного на вхід контуру. Модуль порівняння контурів знаходить відстань між вхідними контурами в метриці Громова-Фреше. Результат роботи останнього модуля і буде результатом роботи програми.

Для пришвидшення проектування та реалізації програми, вирішено використовувати бібліотеку для роботи із зображеннями OpenCV. Дана бібліотека має набір реалізованих методів для знаходження контурів на зображенні, їхнього прорідження, виводу на екран та інші. Також, у ній реалізовані найшвидші та найкращі алгоритми для здійснення цих операцій. Саме тому, OpenCV є хорошим варіантом для використання у системі [50].

На рисунку А.1 представлена діаграма класів для системи порівняння контурів зображень в метриці Громова-Фреше. Головний клас програми – QualityEstimator. Він приймає на вхід два зображення (еталонне та сегментоване автоматичним алгоритмом сегментації) і повертає числове значення схожості цих зображень [51-54]. Основним та єдиним методом даного класу є метод main, який і виконує виклик допоміжних методів.

Наступним модулем є ImageOperations. В ньому зібрані методи для виконання операцій над вхідними зображеннями. В таблиці 3.1 наведено повний опис методів класу.

Таблиця 3.1 – Клас ImageOperations

Назва методу	Опис
private Mat readImage	readImage(String imgName) Зчитує зображення по вказаному шляху String imgName – шлях до зображення
public Mat convertToBinary	convertToBinary(Mat img) Перетворює вхідне зображення на чорно-біле Mat img – матриця вхідного зображення
public List<MatOfPoint> findContours	findContours(Mat img) Знаходить контури зображення Mat img – матриця вхідного зображення

Продовження таблиці 3.1

public List<Point> reduceContour	reduceContour(MatOfPoint contour, int pointsToLeave)

	Спрощує вхідний контур до заданої кількості точок MatOfPoint contour – контур для спрощення int pointsToLeave – кількість точок, які необхідно залишити
public List<Point> getMaxLengthPoints	getMaxLengthPoints(List<Point> reduceContour) Знаходить дві максимально віддалені одне від одної точки контуру List<Point> contour – масив точок контуру
public List<Contour> proccess	proccess(Mat img, boolean reduce, boolean doRotation, int numOfPoints) Викликає операції оброки зображення та повертає масив із точками контурів Mat img – вхідне зображення boolean reduce – визначає, чи потрібно проводити спрощення контурів boolean doRotation – визначає, чи потрібно проводити поворот контуру int numOfPoints – кількість точок, які потрібно залишити в контурі (якщо контур спрощується)

Наступним модулем є клас GeometryUtils. Він призначений для виконання геометричних перетворень, таких як зсув, поворот контуру та інших. Опис даного класу наведено в таблиці 3.2.

Таблиця 3.2 – Клас GeometryUtils

Назва методу	Опис
public static double getEuclideanDistance	getEuclideanDistance(Point p1, Point p2)

	Повертає евклідову відстань між двома переданими точками Point p1, p2 – координати першої та другої точок
public static double getTriangleArea	getTriangleArea(Point a, Point b, Point c) Повертає площу трикутника, сформованого трьома точками Point a, Point b, Point c – координати вершин трикутника
public Point getCenter	getCenter(List<Point> points) Повертає середину відрізка List<Point> points – масив із двох точок – кінців відрізку
public double getAngle	getAngle(List<Point> points) Повертає кут, на який відрізок відхиляється від осі абсцис List<Point> points – масив із двох точок – кінців відрізку
public List<Point> rotate	rotate(List<Point> points, Point center, double angle) Повертає контур навколо центру на заданий кут List<Point> points – масив точок контуру Point center – точка, навколо якої відбувається поворот double angle – кут, на який необхідно повернути контур

Продовження таблиці 3.2

public List<Point> move	move(Point center, List<Point> toMove, Point centerToMove)

	Виконує накладання центру вхідного контуру на центр іншого контуру Point center – координати центру, на який потрібно накласти List<Point> toMove – масив точок контуру, який потрібно перемістити Point centerToMove – центр переміщуваного контуру

Ще одним модулем є клас Distance. Він призначений для знаходження відстані Фреше між двома контурами. В таблиці 3.3 представлено повний опис класу.

Таблиця 3.3 – Клас Distance

Назва методу	Опис
Властивості	
private double ca[][]	Масив результуючих значень порівнянь
private List<Point> contourA	Перший контур для порівняння
private List<Point> contourB	Другий контур для порівняння
Методи	
private double c	c(int i, int j) Повертає результат порівняння для однієї ітерації int i, int j – індекси масиву ca
public double frechet	frechet(List<Point> contourA, List<Point> contourB) Знаходить та повертає відстань між двома контурами в метриці Фреше List<Point> contourA – перший контур для порівняння List<Point> contourB – другий контур для порівняння

Окрім основних класів, у програмі використовуються допоміжні класи. Першим із них є клас PolySimplifier. Даний клас має в наявності метод для прорідження контуру. Опис класу в таблиці 3.4.

Таблиця 3.4 – Клас PolySimplifier

Назва методу	Опис
public static List<Point> reduceVW	reduceVW(List<Point> inputPointList, int numberToKeep) Виконує проріджування контуру List<Point> inputPointList – масив точок контуру int numberToKeep – кількість точок, яку потрібно залишити в контурі

Наступним допоміжним класом є Contour. Даний клас призначений для збереження усієї інформації про контур зображення. Його опис наведений в таблиці 3.5.

Таблиця 3.5 – Клас Contour

Назва методу	Опис
Властивості	
private MatOfPoint contour	Масив точок контуру
private List<Point> reduceContour	Масив точок проріджуваного контуру
private List<Point> maxLengthPoints	Масив двох найбільш віддалених між собою точок контуру
private double maxLength	Довжина між найбільш віддаленими точками контуру
private Point center	Координати центру відрізка maxLengthPoints

Продовження таблиці 3.5

public void setContour	setContour(MatOfPoint contour) Зберігає масив точок контуру MatOfPoint contour – масив точок контуру
public void setReduceContour	setReduceContour(List<Point> contour)

	Зберігає масив точок прорідженого контуру List<Point> contour – масив точок прорідженого контуру
public void setMaxLengthPoints	setMaxLengthPoints(List<Point> maxLengthPoints) Зберігає масив найбільш віддалених точок контуру List<Point> maxLengthPoints – масив найбільш віддалених точок контуру
public void setMaxLength	setMaxLength(double length) Зберігає максимальну відстань double length – максимальну відстань
public void setCenter	setCenter(Point center) Зберігає координати центру Point center – координати центру
public MatOfPoint getContour	getContour() Повертає масив точок контуру
public List<Point> getReduceContour	getReduceContour() Повертає масив точок прорідженого контуру
public List<Point> getMaxLengthPoints	getMaxLengthPoints() Повертає масив координат найбільш віддалених точок контуру
public double getMaxLength	getMaxLength() Повертає довжину відрізку між найбільш віддаленими точками контуру
public Point getCenter	getCenter() Повертає координати центра
public String toString	toString() Перевизначає метод для перетворення об’єкта типу Contour в стрічку

Після опису класів необхідно описати процес роботи програми. В додатку А зображено модель протікання роботи системи. Вхідна точка – метод main із класу QuantityEstimator. Даний метод спочатку викликає метод process, який зчитує зображення, знаходить контури на них, проводить спрощення та поворот. Тоді, метод main в циклі виконує накладання кожного контуру першого зображення на

кожен із контурів другого зображення та знаходження відстані Фреше між ними. Результат виконання цих операцій зберігається у змінній result, яка після успішного завершення, виводиться у консоль. Тестування роботи розробленого програмного забезпечення проводиться у наступному підрозділі.

3.3 Тестування розробленого програмного забезпечення

Для перевірки відповідності розробленого програмного забезпечення поставленим вимогам, проведено його тестування. Першою вимогою для тестування є робота із зображеннями у форматах JPEG, PNG та GIF. Наступний код виконує завантаження зображення по вказаному шляхові та вивід його на екран. Результат виконання даних команд наведено на рисунку 3.2.

```
Mat imgA = imageOperations.readImage("images\\etalon.jpg");
GUI.displayImage(imgA, "Etalon image");
```

Аналогічно, даний код може завантажити зображення будь-якого типу. Тому, перша функціональна вимога до програми задоволена.

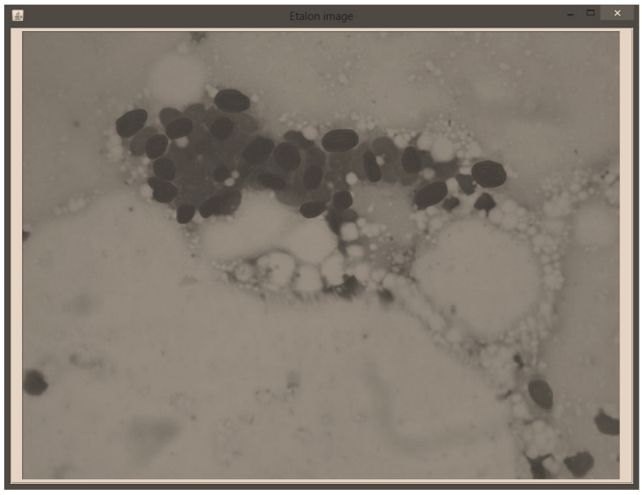


Рисунок 3.2 – Завантажене зображення

Наступним кроком є тестування перетворення кольорового вхідного зображення на чорно-біле. Код, наведений нижче, виконує дані операції та виводить результуюче зображення на екран. На рисунку 3.3 представлено перетворене чорно-біле зображення.

```
Imgproc.cvtColor(img, img, Imgproc.COLOR_RGB2GRAY);
Mat imgBinary = new Mat(img.size(), CvType.CV_8UC1);
Imgproc.threshold(img, imgBinary, 66, 255, Imgproc.THRESH_BINARY_
INV);
GUI.displayImage(imgBinary, "Binary image");
```

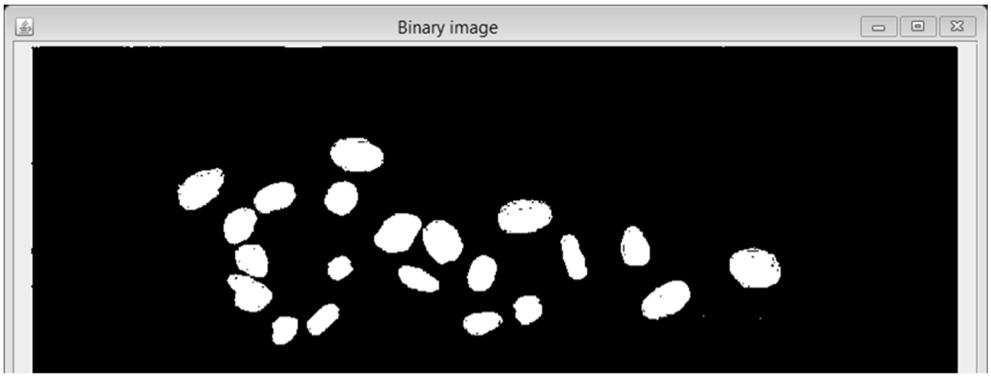


Рисунок 3.3 – Перетворене в чорно-біле вхідне зображення

Після обробки зображення алгоритмом порогової сегментації, необхідно протестувати знаходження контурів на ньому. Для знаходження контурів використовується код, наведений нижче. Результат його роботи представлено на рисунку 3.4.

```
List<MatOfPoint> contours = new ArrayList<>();
Imgproc.findContours(img, contours, new Mat(), Imgproc.RETR_ EXTERNAL,
Imgproc.CHAIN_APPROX_SIMPLE);
```

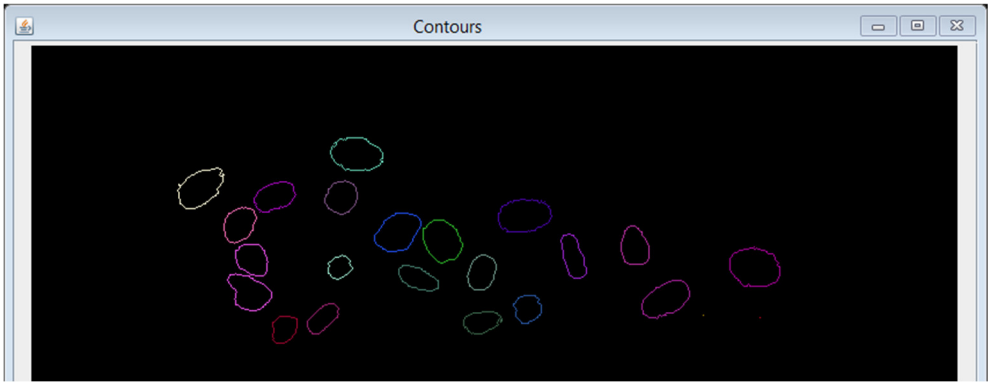
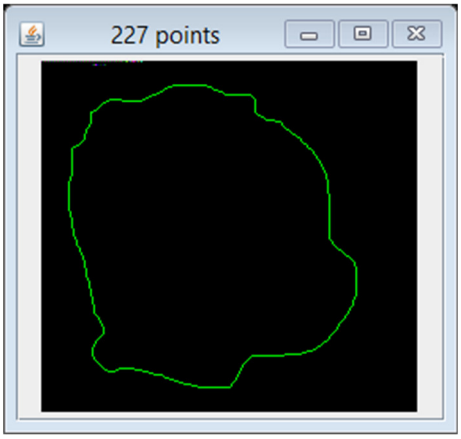


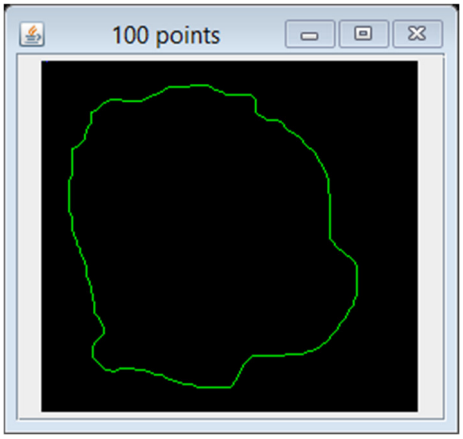
Рисунок 3.4 – Знайдені контури на зображенні

Після знаходження контурів, необхідно виконати їхнє спрощення. Це робиться тому, що у знайденому контурі може бути більше 1000 точок, що значно підвищить складність подальших обчислень. Спрощення контурів виконується за допомогою класу PolySimplifier методом reduceVW. На рисунку 3.5 представлено вхідний контур та його варіанти зі зменшеною кількістю точок.

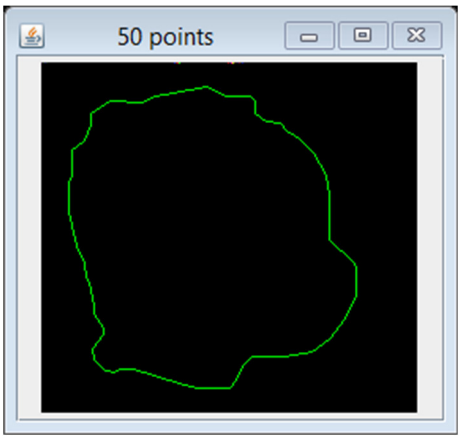
Як можна побачити із рисунку 3.5, оптимальною кількістю точок для залишення є 30. Оскільки, при 10 та 25 точок відбуваються уже значні спотворення форми, а 30 та 50 точок практично ідентично передають форму вхідного контуру.

Після спрощення контуру необхідно здійснити операцію пошуку двох найбільш віддалених між собою точок контуру. Дана операція виконується шляхом повного перебору та знаходження евклідової відстані між точками. Хоча, повний перебір є неоптимальною операцією, проте, після проріджения контуру до 30 точок, пошук евклідової відстані потрібно виконати лише 2450 разів. Врезультаті, отриманий відрізок буде слугувати віссю для повороту та накладання контурів. На рисунку 3.6 зображена головна вісь контуру.

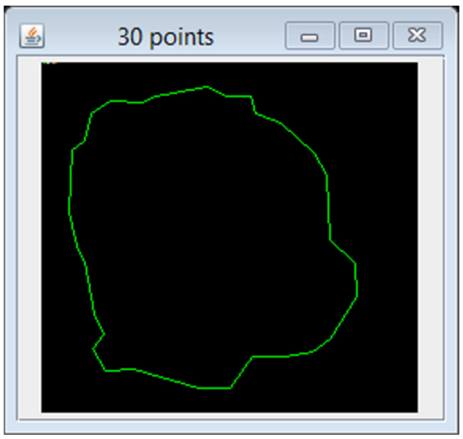




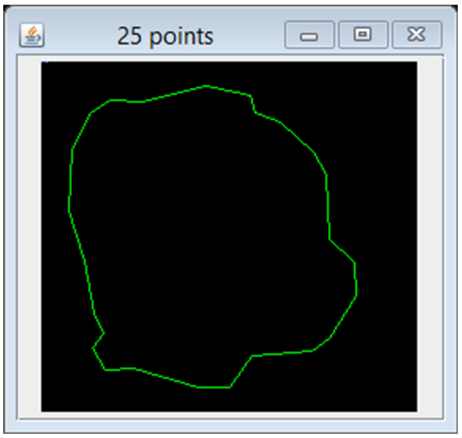
а



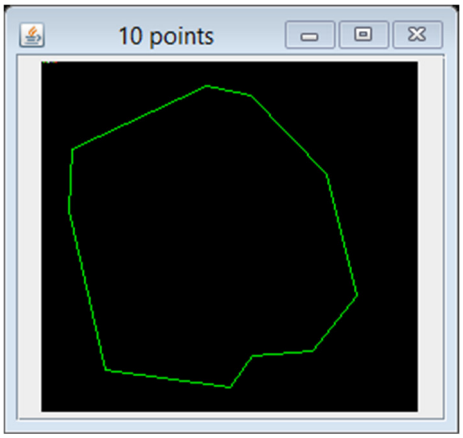
б



в



г



д

е

Рисунок 3.5 – Спрощення контурів: *а)* оригінальний контур; *б)* спрощений до 100 точок; *в)* до 50 точок; *г)* до 30 точок; *д)* до 25 точок; *е)* до 10 точок

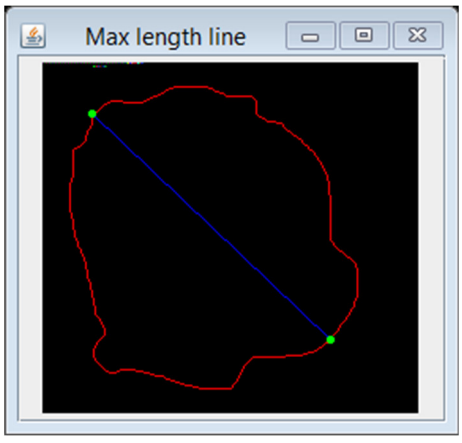
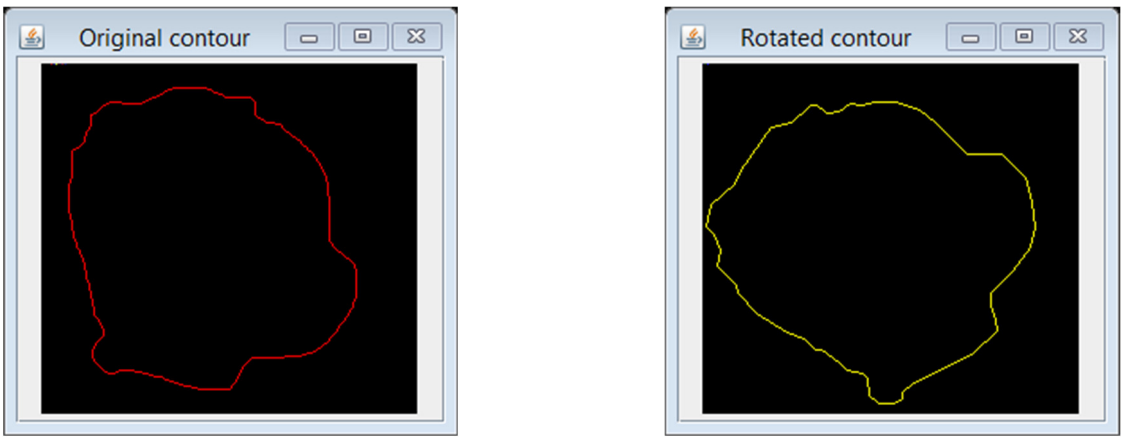


Рисунок 3.6 – Головна вісь контуру

Після знаходження головної осі, виконується поворот контуру, щоб головна вісь розміщувалася паралельно осі абсцис. Це робиться для того, щоб обробка двох

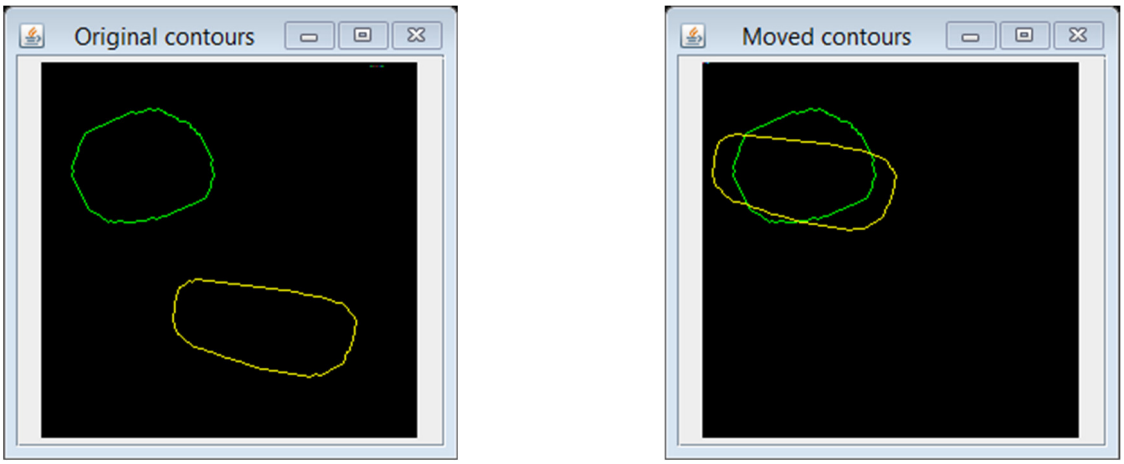
зображень виконувалася паралельно, незалежно одна від одної. На рисунку 3.7 представлено оригінальний та повернутий контур.



а) б)

Рисунок 3.7 – *а)* оригінальний контур; *б)* повернутий паралельно осі абсцис

Для знаходження відстані в метриці Громова-Фреше, повернуті контури потрібно максимально накласти одне на одного. Для цього використовується метод move класу GeometryUtils. На рисунку 3.8 представлено знайдені контури та накладені по головних осях.



а) б)

Рисунок 3.8 – *а)* оригінальне розміщення контурів; *б)* накладені контури

Після виконання даних операцій, виконується підрахунок дискретної відстані Фреше між накладеними контурами. Якщо відстань рівна 0, контури ідентичні. В іншому разі, між контурами є відмінності. На основі результатів роботи програми

та оцінок експерта складено таблицю відповідності результатів роботи мірі подібності зображень (таблиця 3.6).

Таблиця 3.6 – Результуючі показники подібності зображень

Результат програми	Міра подібності
0 – 1	Ідентичні зображення
1 – 15	Подібні зображення
15 – 50	Зображення зі значними відмінностями
50 і більше	Різні зображення

За результатами роботи програми та показниками із таблиці 3.6, можна однозначно стверджувати про подібність зображень між собою в метриці Громова Фреше.

3.4 Експерименти

Для перевірки роботи розробленого програмного забезпечення вирішено провести його порівняння із програмним забезпеченням для порівняння контурів у метриці Фреше. Спочатку, подамо на вхід два однакових зображення (рисунок 3.9). Обрано частину оригінального зображення, представленого на рисунку 3.2. Це зроблено для спрощення суб’єктивного порівняння зображень.

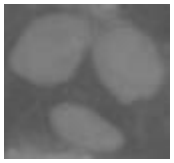


Рисунок 3.9 – Досліджуване зображення

В результаті порівняння одного і того ж зображення самого із собою, результат роботи програми повинен бути рівним 0. На рисунку 3.10 представлені результати роботи обох програм та швидкість виконання обчислень.

а) б)

Рисунок 3.10 – Результати порівняння контурів у метриці *а)* Фреше; *б)* Громова-Фреше

Як можна побачити із рисунку 3.10, результатом порівняння ідентичних зображень в обох метриках є 0. Відрізняється лише швидкість обчислення. Завдяки відсутності операцій зсуву та повороту, метрика Фреше показала результат на 3 мс кращий за Громова-Фреше.

Наступним кроком є подавання на вхід однакових зображень але зі зміщеними клітинами. Для цього, клітини із рисунку 3.9 зміщено відносно початкового їхнього положення (рисунок 3.11).

Рисунок 3.11 – Зміщені позиції клітин

На рисунку 3.11 можна побачити результати роботи обох програм. На основі аналізу результатів та показників, наведених в таблиці 3.6, можна побачити, що у метриці Фреше зображення є зі значними відмінностями. Натомість, у метриці Громова-Фреше ці зображення є подібними. Також, на обробку зображення в досліджуваній метриці потрачено 90 мс (на відміну від 74 мс у метриці Фреше).

```
"C:\Program ...
Program in Frechet mode
The result of comparation is: 24.33741204199716
All program time: 74
```

а)

```
"C:\Program ...
Program in Gromov-Frechet mode
The result of comparation is: 4.4306882975936
All program time: 90
```

б)

Рисунок 3.12 – Результат порівняння зміщених контурів у метриці *а)* Фреше; *б)* Громова-Фреше

Для наступного експерименту, клітини із рисунку 3.9 були повернуті на випадковий кут. Їхнє розміщення при цьому залишилося тим самим (рисунок 3.13). На рисунку 3.14 представлено результати порівняння даного зображення із оригіналом.

Рисунок 3.13 – Повернуті на випадковий кут клітини

```
"C:\Program ...
Program in Frechet mode
The result of comparation is: 29.1875767564848
All program time: 73
```

а)

```
"C:\Program ...
Program in Gromov-Frechet mode
The result of comparation is: 8.98750303973209
All program time: 76
```

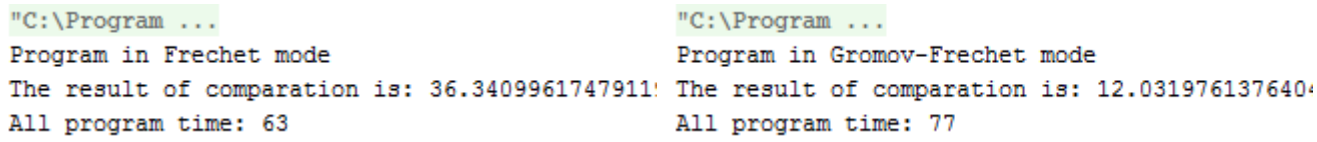
б)

Рисунок 3.14 – Результати порівняння повернутих контурів у метриці *а)* Фреше; *б)* Громова-Фреше

За результатами, наведеними на рисунку 3.14 та на основі показників із таблиці 3.6 можна зробити висновок, що в метриці Фреше дані зображення мають значні відмінності. А у метриці Громова-Фреше вони залишаються подібними. Це твердження і є близьким до істини. Отже, програма працює вірно.

Наступним експериментом є одночасна зміна позиції та поворот клітин. На рисунку 3.15 представлено досліджуване зображення, а на рисунку 3.16 – результати порівняння його з оригіналом.

Рисунок 3.15 – Поворот та зміщення клітин оригінального зображення



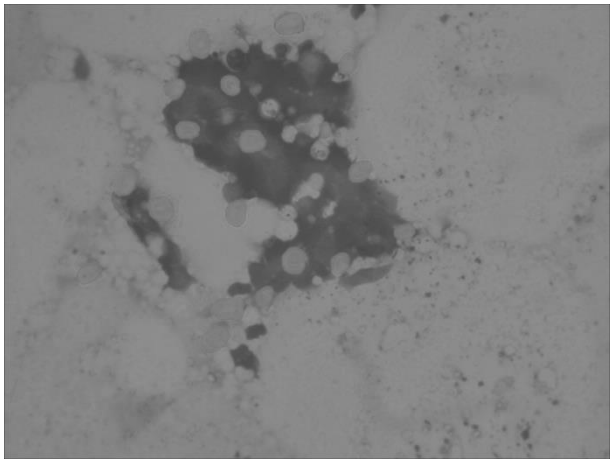
а) б)

Рисунок 3.16 – Результати порівняння контурів зображень у метриці *а)* Фреше; *б)* Громова-Фреше

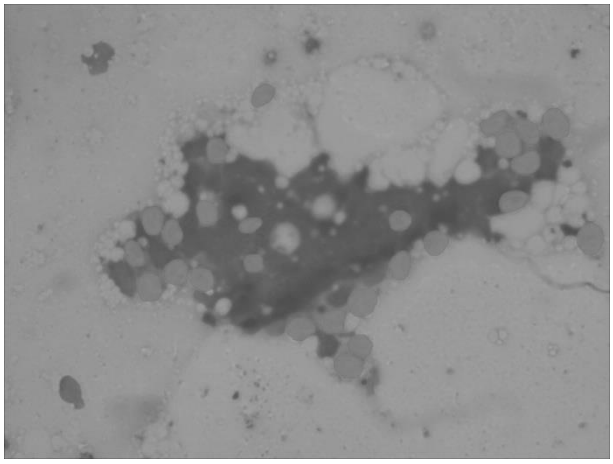
Як можна побачити із рисунку 3.16, результати порівняння у метриці Фреше є найвищими за всі тести. Це означає, що два вхідні зображення мають значні відмінності. А, оскільки, метрика Громова-Фреше враховує переміщення та накладання контурів, у ній дані зображення є подібними.

Далі, проведено дослідження роботи розробленого програмного продукту на повномасштабних сегментованих зображеннях. На рисунках 3.17 – 3.20 наведені порівнювані зображення та результати їхнього порівняння у метриках Фреше та Громова-Фреше. Підсумки проведених експериментів підведені в додатку Б.

Із додатку Б можна побачити, що відстань у метриці Фреше знаходиться швидше і вона набагато більша, за метрику Громова-Фреше. Метрика Фреше передає реальну оцінку подібності двох зображень, тоді як Громова-Фреше – призначена для коригування результатів попередньої. Тому, дані дві метрики повинні працювати в межах одного модуля та мати певне відсоткове співвідношення важливості результатів вимірювання. Тоді, результат порівняння контурів зображень можна вважати вірним.



а)



б)

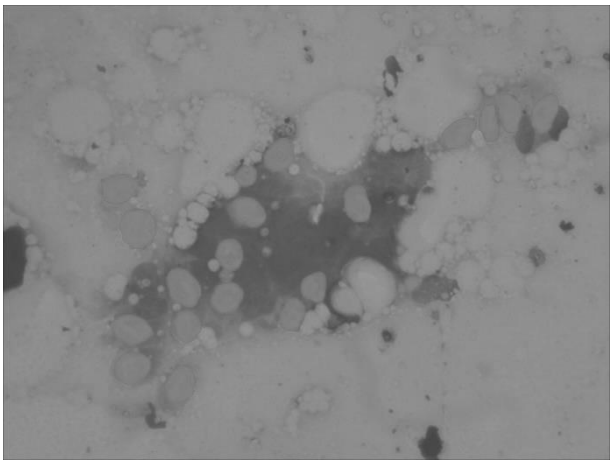
```
"C:\Program ...
Program in Frechet mode
The result of comparation is: 945.033861827183
All program time: 8415
```

в)

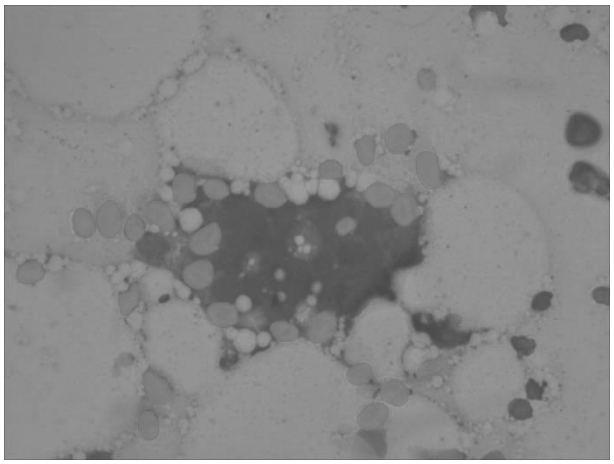
```
"C:\Program ...
Program in Gromov-Frechet mode
The result of comparation is: 25.7848591172753
All program time: 11350
```

г)

Рисунок 3.17 – Експеримент 1: *а), б)* – вхідні зображення; *в)* метрика Фреше; *г)* метрика Громова-Фреше



а)



б)

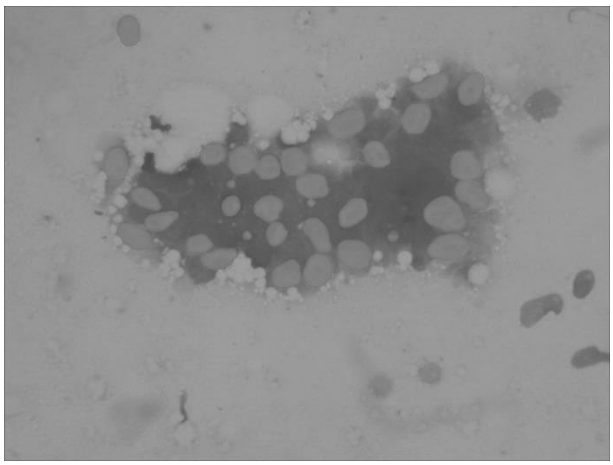
```
"C:\Program ...
Program in Frechet mode
The result of comparation is: 458.46266432379
All program time: 11128
```

в)

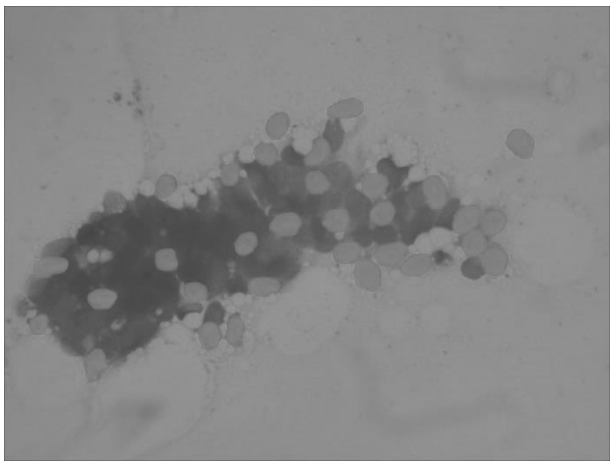
```
"C:\Program ...
Program in Gromov-Frechet mode
The result of comparation is: 145.99285990049074
All program time: 18381
```

г)

Рисунок 3.18 – Експеримент 2: *а), б)* – вхідні зображення; *в)* метрика Фреше; *г)* метрика Громова-Фреше



а)



б)

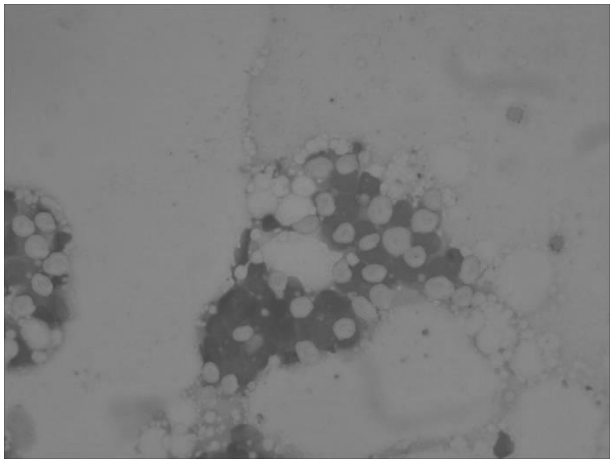
```
"C:\Program ...
Program in Frechet mode
The result of comparation is: 882.7144890499524
All program time: 11046
```

в)

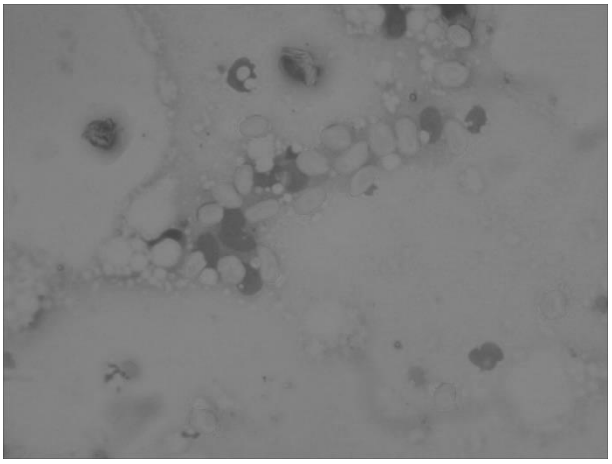
```
"C:\Program ...
Program in Gromov-Frechet mode
The result of comparation is: 77.3001187394586
All program time: 16300
```

г)

Рисунок 3.19 – Експеримент 3: *а), б)* – вхідні зображення; *в)* метрика Фреше; *г)* метрика Громова-Фреше



а)



б)

```
"C:\Program ...
Program in Frechet mode
The result of comparation is: 1118.73407203261
All program time: 5045
```

в)

```
"C:\Program ...
Program in Gromov-Frechet mode
The result of comparation is: 52.1070144264842
All program time: 8445
```

г)

Рисунок 3.20 – Експеримент 4: *а), б)* – вхідні зображення; *в)* метрика Фреше; *г)* метрика Громова-Фреше

ВИСНОВКИ

1. Було проаналізовано алгоритми сегментації зображень та проведено комп'ютерні експерименти для сегментації цитологічних зображень на основі алгоритмів сегментації порога, збільшення (нарощення) площі, k-середнього та сегментації вододілу. Суб’єктивний аналіз алгоритмів сегментації показав відмінність результатів сегментації експерта і її програмної реалізації, що зумовило необхідність розроблення об’єктивних критеріїв оцінки результатів сегментації;

2. В результаті аналізу метрики Фреше встановлено, що для порівняння контурів двох зображень і знаходження найменшої відстані між ними необхідно виконати ізометричні перетворення, що зумовило введення метрики Громова-Фреше;

3. Розроблено алгоритми знаходження та представлення контурів зображень, що базуються на алгоритмі обходу контуру околом Мура, складність якого становить $O(p \cdot q)$

4. Розроблено алгоритм ізометричних перетворень, що дозволило реалізувати порівняння контурів зображень у метриці Громова-Фреше;

5. Розроблено алгоритм порівняння контурів зображень у метриці Громова-Фреше на основі алгоритму дискретного визначення відстані Фреше і алгоритму ізометричних перетворень, що дало можливість оцінити найменшу відстань між двома контурами;

6. На основі поставлених вимог до програмного забезпечення, проведено формалізацію функціональних та не функціональних вимог та розроблено архітектуру програм, що дало можливість спроектувати та програмно реалізувати модуль порівняння контурів зображень в метриці Громова-Фреше;

7. Проведено тестування розробленого модуля та комп’ютерні експерименти, що дало можливість оцінювати найменшу відстань між контурами зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс. – М.: Вид-во «Техносфера», 2005. – 1072 с.

2. Березький О. М. Методи сегментації біомедичних зображень / О. М. Березький, Ю.М. Батько, Г.М.Мельник // Вісник Хмельницького національного університету. Технічні науки. – 2010. – №1. – С.189- 197.

3. Березький О. М. Метод вибору алгоритмів автоматичної сегментації біомедичних зображень / О. М. Березький, Ю. М. Батько // Системи обробки інформації. – 2013. – Випуск 2 (109). – С. 246-251.

4. Березький О. Методи кількісної оцінки якості сегментації зображень / Олег Березький // Матеріали дванадцятої всеукраїнської міжнародної конференції «Оброблення сигналів і зображень та розпізнавання образів» (УкрОБРАЗ’2014), Київ, 3-7 листопада 2014 р. – К., 2014. – С. 51–54.

5. Березский О. Н. Количественная оценка качества сегментации изображений на основе метрик / О. Н. Березский, Е. Н. Березская // Управляющие системы и машины. – 2015. – №6. – С.59-65.

6. Березький О.М. Аналіз алгоритмів співставлення областей зображень для кількісної оцінки результатів сегментації / О.М. Березький, Г.М. Мельник, Ю.М. Батько, О.Й. Піцун // Матеріали Міжнародної наукової конференції «Інтелектуальні системи прийняття рішень та проблеми обчислювального інтелекту» (ISDMCI’2016), м. Залізний Порт, 24–28 травня 2016 р. – Херсон: ПП Вишемирський В.С., 2016. – С. 252-253.

7. Березький О.М. Адаптивний метод сегментації зображень на основі метрик / О.М. Березький, О.Й. Піцун // Науковий вісник НЛТУ України: збірник науково-технічних праць. Львів: РВВ НЛТУ України. – 2018. – №. 28(3). – С.122-126.

8. Березький О. М. Дослідження похибки перетворення контурів біомедичних зображень [Текст] / О. М. Березький // Науковий вісник НЛТУ України. – 2013. – № 23.17 .– С. 352-359.

9. Палагнюк І. В., Клімовський Д. Б., Вдодович О. В., Бучинський Т. Б. Сегментація зображень з використанням метричних мір. *Інтелектуальні комп'ютерні системи та мережі* : тези доп. V Наук.-практ. конф. молодих вчених і студентів (2 груд. 2021 р.). Тернопіль : ЗУНУ, 2021. С.

10. Вдодович О. В., Клімовський Д. Б., Палагнюк І. В., Бучинський Т. Б. Алгоритми порівняння зображень в метричних просторах. *Інтелектуальні комп'ютерні системи та мережі* : тези доп. V Наук.-практ. конф. молодих вчених і студентів (2 груд. 2021 р.). Тернопіль : ЗУНУ, 2021. С.

11. Березький О. М., Дубчак Л. О., Мельник Г. М. Методичні рекомендації до виконання кваліфікаційної роботи з освітнього ступеня "Магістр". Спеціальність: 123 - Комп'ютерна інженерія. Магістерська програма – «Комп'ютерна інженерія». Тернопіль : ЗУНУ, 2021. 32 с.

12. Цифрове зображення [Електронний ресурс]. – Режим доступу: http://uk.wikipedia.org/wiki/Цифрове_зображення.

13. Biomedical Imaging [Електронний ресурс]. – Режим доступу: http://seas.yale.edu/faculty-research/research-areas/biomedical-imaging.

14. Гістологія людини / О. Д. Луцик, А. Й. Іванова, К. С. Кабак, Ю. Б. Чайковський / За ред. О. Л. Босецького. – К.: «Книга плюс», 2003. – 592 с.

15. Гистология, цитология и эмбриология [Електронний ресурс] .– Режим доступу: http://www.morphology.dp.ua/_mp3/intro.php.

16. Методи гістологічних досліджень [Електронний ресурс] .– Режим доступу: http://intranet.tdmu.edu.ua/data/kafedra/internal/histolog/classes_stud/uk/stomat/ptn/1/01 гістологічна техніка. методи гістологічних досліджень. основи цитології.загальна організація клітини. поверхневий комплекс.htm.

17. Новак В. П. Цитологія, гістологія, ембріологія. /В. П. Новак, А. П. Мельниченко .– Біла Церква, 2005. – 256 с.

18. Автандилов Г.Г. Основы количественной паталлогической анатомии / Г.Г. Автандилов. – М. : Изд-во "Медицина", 2002. – 238 с.

19. Березький О. М. Перетворення цитологічних зображень із заданою похибкою [Текст] / О. М. Березький // Науковий вісник НЛТУ України. – 2013. – № 23.12 .– С. 357-362.

20. Ткаченко О. М. Метод кластеризації на основі послідовного запуску k-середніх з удосконаленим вибором кандидата на нову позицію вставки / О. М. Ткаченко, О. Ф. Грійо Тукало, О. В. Дзісь, С. М. Лаховець // Інформаційні технології та комп'ютерна техніка .– 2012 .– №2 .– С. 1-10.

21. Батько Ю. М. Метод і алгоритми сегментації біомедичних зображень на основі попередніх розміток [Текст] / Ю. М. Батько // «Штучний інтелект». – 2010. – № 4 .– С. 140-149.

22. Березский О. Н. Топологические методы и алгоритмы преобразования контуров и областей плоских изображений / О. Н. Березский // Проблемы информатики и управления. – 2010. – № 5. – С.123-131.

23. Кутузов А. С. Метрические пространства / А. С. Кутузов. – Троицк.: Челябинский государственный университет, 2012. – 100 с.

24. Metric space [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Metric_space.

25. Евклидова метрика [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Евклидова_метрика.

26. Скворцов В. А. Примеры метрических пространств / В. А. Скворцов. – М.: Издательство Московского центра непрерывного математического образования, 2002. – 24 с.

27. Alt H. Computing the Fréchet distance between two polygonal curves [Text] / H. Alt, M. Godau // International Journal of Computational Geometry & Applications. – 1995. – Vol. 05, No. 01n02 : pp. 75-91.

28. Eiter Th. Computing Discrete Fréchet Distance [Text] / Thomas Eiter, Mannila Heikki // International Journal of Computational Geometry & Applications. – 1994: pp. 1-7.

29. Berezsky O. Gromov-Fréchet distance between curves / O. Berezsky, M. Zarichnyi // Matematychni Studii. – 2018. – Vol. 50, No.1. – P. 88-92.

30. Berezsky O. Image Segmentation Metric-Based Adaptive Method / Oleh Berezsky, Oleh Pitsun, Natalia Batryn, Kateryna Berezska, Nadiya Savka, Taras Dolynyuk // Proceedings of the 2018 IEEE Second International Conference on Data

Stream Mining & Processing (DSMP), Lviv, August 21-25, 2018. – Lviv, 2018. – P. 554-557.

31. Berezsky O. Fréchet distance between weighted rooted trees / O. Berezsky, M. Zarichnyi // Matematychni Studii. – 2017. – Vol. 48, No.2. – P. 165-170.

32. Berezsky O. Fréchet Metric for Trees / Oleh Berezsky // Proceedings of the 2016 IEEE First International Conference on Data Stream Mining & Processing (DSMP), Lviv, August 23-27, 2016. – Lviv, 2016. – P. 213-217.

33. JMicroVision [Електронний ресурс]. – Режим доступу: http://www.jmicrovision.com.

34. NEXSYS ImageExpert Pro 3 [Електронний ресурс]. – Режим доступу: http://www.nexsys.ru/iepro3x.htm.

35. ImageJ Features [Електронний ресурс]. – Режим доступу: http://rsb.info.nih.gov/ij/features.html.

36. ImageJ [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/ImageJ.

37. Березький О. М. Системи автоматизованої мікроскопії: стан та перспективи розвитку / О. М. Березький, О.Й. Піцун, С. О. Вербовий // Вісник Хмельницького національного університету, 2016. - №2 (235). - С. 61-68.

38. Frechet distance [Електронний ресурс]. – Режим доступу: http://en.wikipedia.org/wiki/Frechet_distance.

39. Pankaj K. Agarwaly Computing the discrete Frechet distance in subquadratic time [Text] / Pankaj K. Agarwaly, Rinat Ben Avrahamz, Haim Kaplanx, Micha Sharir // International Journal of Computational Geometry & Applications. – 1995. – Vol. 05, No. 01n02 : pp. 75-91.

40. Square Tracing Algorithm [Електронний ресурс]. – Режим доступу: http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/square.html

41. Moore-Neighbor Tracing [Електронний ресурс]. – Режим доступу: http://www.imageprocessingplace.com/downloads_V3/root_downloads/tutorials/contour_tracing_Abeer_George_Ghuneim/moore.html

42. Шлезингер М.И. Распознавание сходства многоугольников в усиленной хаусдорфовой метрике [Текст] / М.И. Шлезингер, Е.В.Водолазский, В.М.Яковенко // Кибернетика и Системный Анализ. - 2014.- № 3. - С. 174 - 187.

43. Алгоритм Рамера–Дугласа–Пекера [Електронний ресурс]. – Режим доступу: https://ru.wikipedia.org/wiki/Алгоритм_Рамера_—_Дугласа_—_Пекера

44. Javascript implementation of the Ramer Douglas Peucker Algorithm [Електронний ресурс]. – Режим доступу: http://karthaus.nl/rdp/Javascript implementation of the Ramer Douglas Peucker Algorithm.

45. Berezsky. O. Development of a metric and the methods for quantitative estimation of the segmentation of biomedical images / Berezsky O., Zarichnyi M., Pitsun O. // Eastern-European Journal of Enterprise Technologies. – 2017. – V. 6, № 4. – P. 4–11.

46. Местецкий Л. Скелет многосвязной многоугольной фигуры / Л. Местецкий // Труды 15 междунар. конф. ГРАФИКОН-2005. – Новосибирск. – С. 242-249

47. Давидов М. В. Вдосконалений метод скелетонізації двовимірної області, що враховує особливості контуру / М.В. Давидов // Вісник Національного університету "Львівська політехніка". – 2013. – № 770 : Інформаційні системи та мережі. – С. 36–42

48. Moravec H. Rover visual obstacle avoidance // Proc. Intl. Joint Conference on Artificial Intelligence. – 1981. – P. 785–790.

49. Детекторы углов [Електронний ресурс]. – Режим доступу: http://habrahabr.ru/post/244541.

50. OpenCV. URL: https://uk.wikipedia.org/wiki/OpenCV.

51. Березький О. М. Інтелектуальна система аналізу зображень ауто- та ксеногенних тканин / О. М. Березький, Г.М. Мельник, К. М. Березька, Т.В. Дацко // Науковий вісник НЛТУ України: зб. наук.-техн. праць. – Львів: РВВ НЛТУ України. – 2014. – Вип. 24.11. – С. 323-330.

52. Березький О. М. Інтелектуальна система автоматизованої мікроскопії аналізу гістологічних та цитологічних зображень / О.М. Березький, О.Й. Піцун,

П.Б. Лящинський, Г.М. Мельник // Штучний інтелект, Київ, 2017. - №2 (76). - С. 128-140.

53. Berezsky O. Modern automated microscopy systems in oncology / O. Berezsky, O.Pitsun, N. Batryn, T. Datsko, K. Berezska, L. Dubchak // Proceedings of the 1st International Workshop on Informatics & Data-Driven Medicine, Lviv, Ukraine, 28-30 november 2018. – P. 311-325.

54. Методи, алгоритми і програмні засоби опрацювання біомедичних зображень / Березький О. М., Батько Ю.М., Березька К.М., Вербовий С.О., Дацко Т.В., Дубчак Л.О., Ігнатєв І.В., Мельник Г.М., Николюк В.Д., Піцун О.Й. – Тернопіль: Економічна думка, ТНЕУ, 2017. – 330 с.