МІНІСТЕРСТВО ОСВІТИ І НАУКИ КРАЇНИ
Західноукраїнський національний університет
Фаультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Клімовський Дмитро Богданович

ПОРІВНЯННЯ БІОМЕДИЧНИХ ЗОБРАЖЕНЬ В МЕТРИЧНИХ ПРОСТОРАХ / COMPARISON OF BIOMEDICAL IMAGES IN METRIC SPACES

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

Кваліфікаційна робота

Виконав студент групи Кім-22
Д.Б. Клімовський
_____

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

Кваліфікаційну роботу
допущено до захисту
«___» _____2021 р.

Завідувач кафедри КІ
О.М.Березький
_____

Тернопіль – 2021

ВСТУП

Актуальність теми. Порівняння зображень – одна з основних проблем в машинному зорі, заснована на виявленні певних критеріїв. Без цього неможливо розпізнання об'єктів. Важливим завданням на даному етапі є вибір критеріїв для подальшої оцінки подібності об'єктів. Перед виконанням порівнняння, зображення має пройти попередню обробку.

Найчастіше на практиці при аналізі конкретного зображення виникає необхідність вибору алгоритму сегментації, найбільш відповідного для даного типу зображень. При цьому, як очевидно, необхідно враховувати як властивості зображення, так і особливості конкретного алгоритму.

Експертна оцінка при порівннянні зображень несе суб’єктивний характер.

Тому виникає необхідність розробки системи порівняння областей зображень і, як наслідок, результатів роботи алгоритмів сегментації зображень за допомогою об’єктивного критерію.

Метою роботи є дослідження та розробка алгоритмів порівняння областей зображень в метриці Громова-Хаусдорфа і розробка програмного засобу для порівняння областей.

Для досягнення поставленої мети необхідно вирішити наступні завдання:

– вибрати алгоритми сегментації зображень для проведення оцінки;

– проаналізувати алгоритми та методі порівняння областей зображень;

– розробити алгорими пошуку відстані Громова-Хаусдорфа між двома областями;

– розробити архітектуру майбутньої програмної системи;

– реалізувати систему на одній з мов програмування;

– провести експерименти та проаналізувати отримані результати.

Об’єктом дослідження є процес порівняння зображень.

Предметом дослідження є алгоритми порівняння зображень.

Наукова новизна полягає у розробці модифікованих алгоритмів для порівняння зображень в метриці Громова-Хаусдорфа.

Методи досліджень базуються на використанні алгоритмів сегментації зображень, контурного аналізу, критеріїв порівняння областей зображень та метрик.

Практичне значення. Розроблено програмний засіб для порівняння областей зображень у метриці Громова-Хаусдорфа .

Публікації та апробація КР. 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) [1, 2]:

1. Полагнюк 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., Полагнюк 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.

Структура роботи.

У першому розділі аналізуються особливості біомедичних зображень. Розглядаються основні алгоритми сегментації, методи порівняння областей зображень та наявні програмні засоби [3].

У другому розділі представлено алгоритми на основі яких буде сформовано програмну систему для порівняння областей зображень. Розглянуто алгоритми обходу контурів та кусково-лінійної апроксимації. Розглянуто метрику Громова-Хаусдорфа та модифіковані алгоритми пошуку відстані Громова-Хаусдорфа.

У третьому розділі описується архітектура системи, виконуються експериментальні дослідження з метою порівняння модифікованих алгоритмів та виявлення їх переваг і недоліків.

Додаток А містить діаграму класів, в додатку Б приведено зображення над якими виконувались порівняння модифікованих алгоритмів пошуку відстані Громова-Хаусдорфа. Додаток В містить таблицю з результатами досліджень алгоритму пошуку відстані з пороговою фільтрацією хорд. В додатку Г міститься код програми і в додатку Д – публікації.

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

1.1 Аналіз біомедичних зображень

Протягом двох останніх десятиліть технологія медичної інтроскопії (medical imaging) або технологія одержання зображень внутрішніх органів та тканин людини пережила ряд принципових змін. Використання комп'ютерних технологій призвело до розвитку нових напрямків інтроскопії, таких як, наприклад, комп'ютерна томографія (CT-computed tomography), магнітна резонансна томографія (MRi- magnetic resonance imaging) і позитронна емісійна томографія (PET-positron emission tomography). За допомогою нової апаратури можна одержати зображення великої кількості перетинів тканин тіла пацієнта – біомедичні зображення, що характеризують особливості його анатомії і фізіології. Ці знімки опосередковано відтворюють стан різних органів [4].

У медичній практиці за характером отримання та галузі використання розрізняють три типи зображень:

- анатомічні;
- гістологічні (включаючи цитологічні);
- фізіологічні.

Фізіологічні зображення мають специфічні методи отримання та обробки, тому розглядатися не будуть. В даний час існує досить багато способів отримання медичних зображень.

Анатомічні зображення можна отримати за допомогою рентгенографії, комп'ютерної томографії, магнітно-резонансної томографії, УЗД, мікроскопії, термографії, ендоскопічними та оптичними методами. Для отримання гістологічних зображень використовуються термографія, електронна та оптична мікроскопія.

Гістологічні зображення одержують методами оптичної мікроскопії. У оптичної мікроскопії існує кілька методів вивчення зображень: світле поле,

темне поле, фазовий контраст, інтерференційні методи контрастування, флуоресцентні методи.

На гістологічних зображеннях оптичної мікроскопії визначити інформативні об'єкти непросто. Це пов'язано із низьким контрастом зображень клітин і клітинних структур, складністю організації тканини і її структури, наявністю в полі зору різної групи клітин і значною неоднорідністю тканини як фону (рисунок 1.1).

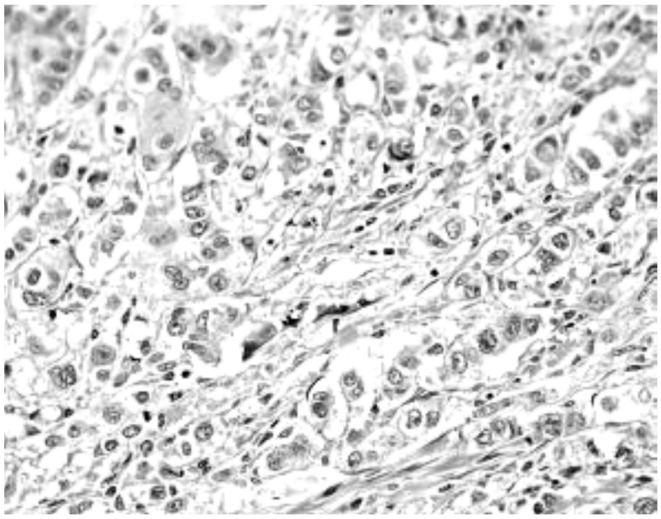


Рисунок 1.1 – Зображення оптичної мікроскопії

Зображення оптичної мікроскопії дуже сильно залежить від якості приготованого препарату і обладнання. Зображення оптичної мікроскопії можуть бути як півтоновими, так і кольоровими (це визначається типом обладнання та поставленими завданнями). Наявність великої кількості різнотипних об'єктів на гістологічних зображеннях є серйозним недоліком при аналізі цих зображень.

Як окремий клас зображень, одержуваних в оптичній мікроскопії, можна виділити цитологічні зображення (рисунок 1.2), які є окремим випадком гістологічних. Переважно це зображення мазків, що складаються з окремо розміщених клітин [5].

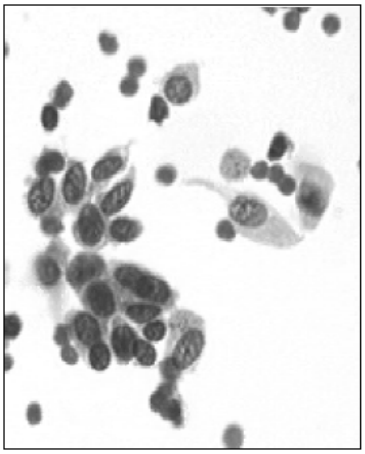


Рисунок 1.2 – Зображення цитології

Біомедичні зображення використовуються під час встановлення діагнозу та в процесі лікування. Слід підкреслити, що нерідко для отримання повної і об'єктивної картини захворювання використовується не один, а цілий комплекс візуально-діагностичних методів. Наприклад, при деяких захворюваннях нирки в єдиний діагностичний процес об'єднуються методи УЗД, КТ та магнітно-резонансної томографії.

При обробці біомедичних зображень оптичної мікроскопії найбільш важливу роль відіграють характеристики зашумлення зображення, однорідність фону, оптичні та геометричні характеристики об'єктів, порівняльна характеристика різних видів біомедичних зображень наведено в таблиці 1.1.

Таблиця 1.1 – Особливості зображень оптичної мікроскопії

Зображення оптичної мікроскопії	Тип зображень	Наявність великих шумів і артефактів	Нечіткі границі об’єктів	Геометричні спотворення	Наявність різно-типних об’єктів в полі зору	Неодно-рідність фону
Гістологічні	Півтонові та кольорові	Є	Є	Можливі	Є	Є
Цитологічні		Можливі	Можливі	Можливі	Є	Є
Анатомічні		Є	Є	Є	Є	Є

З таблиці видно, що найбільшим потенціалом, для автоматизованого аналізу, володіють цитологічні і гістологічні зображення, оскільки доля шумів і спотворень в них менша [5].

В межах даної роботи в основному буде розглянуто гістологічні та цитологічні зображення, так як зображення клітин досить стабільні і зручні для аналізу.

Приблизно в центрі клітини розташовано ядро, яке часто за своїм показником заломлення відрізняється від решти клітини. На забарвлених препаратах видно, що ядро обмежене ядерною мембраною, або оболонкою. У ядрі є одне або кілька округлих темнофарбованих тілець, які називаються ядерця.

Ядро розташоване в цитоплазмі, що займає зовнішню і зазвичай більшу частину клітини. Кожен компонент цитоплазми виконує свої певні функції, причому в різних клітинах ці компоненти мають різний вигляд залежно від цих функцій. Слід вказати, що за допомогою світлового мікроскопа в цитоплазмі виявляються лише деякі з її компонентів. Дослідження ж всіх компонентів і структури цитоплазми проводиться з використанням електронного мікроскопа. Цитоплазма містить безліч спеціалізованих ультраструктур, званих органелами клітини, їх наявність обумовлює оптичну неоднорідність клітини.

Клітини різняться за своїми розмірами, існують деякі верхні і нижні межі розмірів: у більшості випадків діаметр клітини не виходить за межі 0,01 - 0,1 мм [6, 7].

На багатьох гістологічних зображеннях, присутні різного роду шуми. Деякі з них класифікуються як артефакти. У гістології артефактами називають ознаки або структури, що виникають на препаратах випадково або внаслідок поганої обробки. Такі артефакти зменшують вірогідність правильного аналізу властивостей об’єктів на зображення.

1.2 Аналіз алгоритмів сегментації

Сегментація зображень – одна з головних задач розпізнавання зображень. Це розділення зображення на декілька областей, які відрізняються одна від одної елементарними ознаками, такими як яскравість, колір, текстура, форма. Сегментація дозволяє виділити ділянки зображення, які можуть розглядатися однорідними. Неправильне виділення сегментів на зображенні в результаті може відбитися на якості розпізнавання і навіть зробити його неможливим [8-14].

Мета сегментації полягає у спрощенні та/або зміні представлення зображення, щоб його було простіше і легше аналізувати. Сегментація зображень зазвичай використовується для того, щоб виділити об'єкти і межі (лінії, криві, і т.п.) на зображеннях шляхом присвоєння міток кожному пікселю зображення так, щоб пікселі з однаковими мітками мали спільні візуальні характеристики .

Результатом сегментації зображення є множина сегментів, які разом покривають все зображення, або множина контурів, виділених з зображення. Всі пікселі в сегменті схожі за деякою характеристикою, наприклад за кольором, яскравості або текстурою. Сусідні сегменти значно відрізняються по цій характеристиці.

При сегментації зображення повинні виконуватися наступні умови:

– в результаті сегментації зображення розділяється на ряд областей таким чином, щоб кожен його піксель входив би в одну з областей;

– області, які виходять в результаті сегментації, не повинні перетинатися, іншими словами, кожен піксель зображення може входити тільки в одну область;

– всі пікселі, віднесені до однієї області, повинні володіти одними і тими ж властивостями. Наприклад, яскравість або колір пікселів, віднесених до однієї області, повинні лежати в межах, визначених для даної області [15].

Сегментація є складним моментом в обробці та аналізі біомедичних зображень, так як необхідно виділяти області, які відповідають різним об'єктам

або структурам на гістологічних препаратах: клітинам, органелам, артефактам та ін. Це пов'язано з високою варіабельністю їх характеристик, слабкою контрастністю оброблюваних зображень і складною геометричною організацією об'єктів [5].

Відповідно до математичного апарату, що використовується для реалізації методів сегментації, алгоритми діляться на такі види:

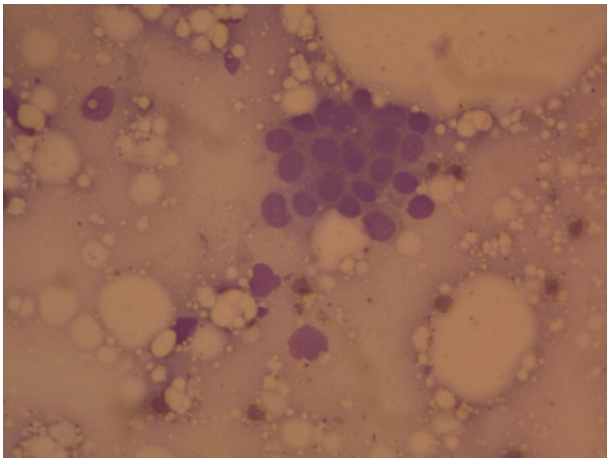
– порогова сегментація [16];
– сегментація методом кластеризації [15];
– сегментація по морфологічних вододілах [17];
– об'єднання (нарощування) областей [18].

У зв'язку з тим, що одним з найважливіших гістологічних об'єктів є клітина, особливості сегментації будуть розглядатися на її прикладі. Основна характеристика зображення клітини – це замкнута границя, що відповідає клітинній мембрані. Незважаючи на те, що основною характеристикою більшості клітин є їх опукла форма, вона не завжди служить визначальним фактором, так як у випадках мітозу і при деяких видах патології ця умова порушується. Крім того, склад клітини залежно від способу забарвлення гістологічного препарату на зображенні відображається по різному. Найчастіше воно має профіль яскравості, що нагадує Гаусовий розподіл. Іноді на зображенні чітко видно елементи, відповідні клітині і ядру. У разі незафарбованих ядер клітина характеризується порожнистим зображенням [5-7].

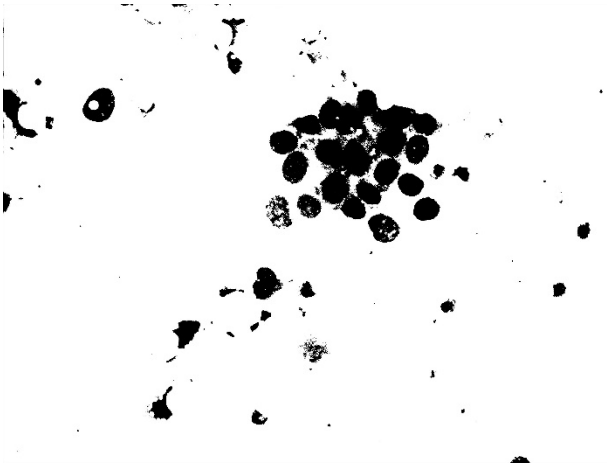
1.2.1 Порогова сегментація зображень клітинних структур

За наявності якісного контрастного зображення клітини (рисунок 1.3) найефективнішим способом обробки є порогова сегментація.

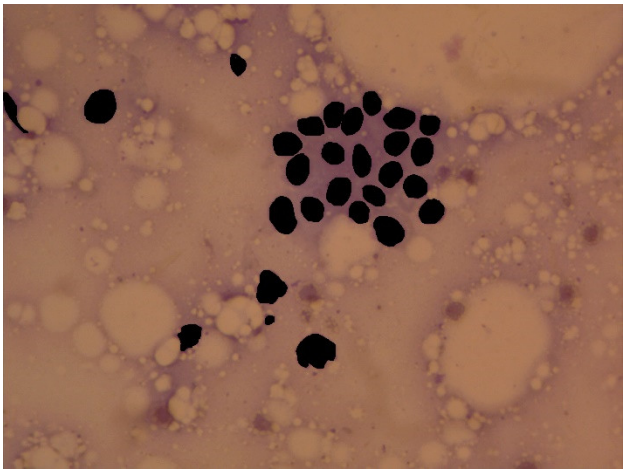
При отриманні багатьох зображень за допомогою світлового мікроскопа задати постійний поріг можна тільки для зображень з високим контрастом.



а)



б)



в)

Рисунок 1.3 – Зображення клітин: а) вихідне зображення; б) бінарне, яке отримане після порогової сегментації; в) сегментоване експертом

У зв'язку з тим що якісні зображення гістологічного препарату відображають клітини на певному фоні, більш ефективно задавати поріг яскравості за допомогою аналізу гістограми (рисунок 1.4).

Однією з причин наявності у гістограми статистичної асиметрії та ексцесу є неоднорідність сукупності. Це означає, що в одну сукупність значень яскравості на зображенні зведені пікселі двох і більше нормальних сукупностей (наприклад, для фону і об'єктів), кожна з яких характеризується своїм набором параметрів і границями [5].

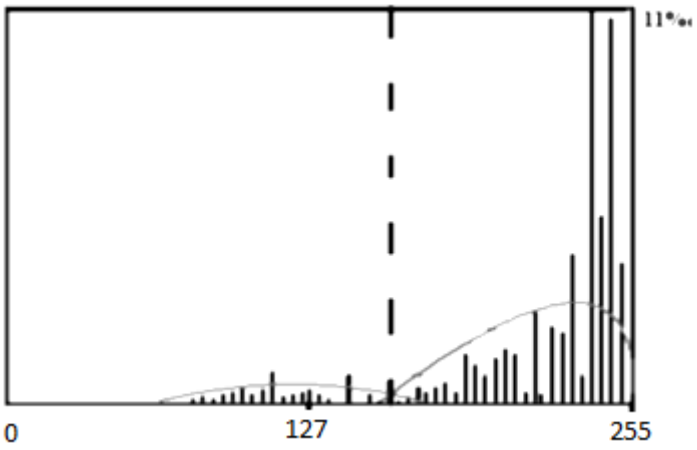


Рисунок 1.4 – Гістограма для півтонового зображення нервової клітини, поріг позначений пунктирною лінією

Методи з використанням порогової сегментації дуже ефективні, коли порівнюються з іншими методами сегментації зображень, тому що вони вимагають тільки один прохід по пікселям. У цьому методі гістограма обчислюється за всіма пікселях зображення і її мінімуми і максимуми використовуються, щоб знайти пороги між сегментами на зображенні.

Покращення цього методу дозволяє рекурсивно застосовувати його до сегментів на зображенні для того, щоб поділити їх на більш дрібні кластери. Процес повторюється з усе меншими і меншими сегментами до тих пір, коли перестануть з'являтися нові сегменти.

Один недолік цього методу – те, що ним може бути важко знайти значні мінімуми і максимуми на зображенні [19].

1.2.2 Сегментація зображень методом кластеризації

В загальному випадку метод кластеризації або k-метод сегментує зображення на k різних кластерів, розміщених на деякій відстані один від одного, за певним критерієм. В якості такого критерію може бути вибраний колір, геометрична відстань та ін. За замовчуванням при реалізації цього методу для вимірювання відстаней використовується Евклідова метрика.

Метод сегментації на основі кластеризації реалізується шляхом двоетапного ітеративного алгоритму, котрий мінімізує суму відстаней «точка - центроїд» отриману шляхом сумування за всіма кластерами. Іншими словами, ціллю роботи алгоритму є мінімізація мінливості пікселів в середині кластера і максимізація мінливості між кластерами.

Алгоритм починає свою роботу з k випадково вибраних розміщень центроїдів кластерів, а потім змінює приналежність точок до кластерів. Тобто переміщує точки з одних кластерів в інші, щоб отримати найбільш підходящий результат.

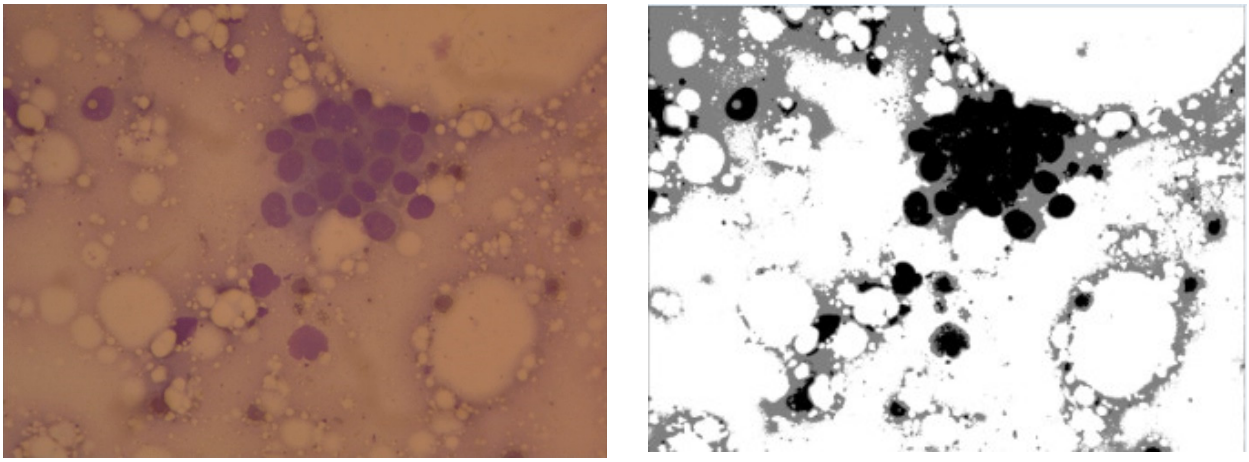
Під час першого етапу при кожній ітерації всі точки одразу перегруповуються таким чином, щоб вони розміщувались якнайближче до своїх центроїдів, після чого перераховуються координати центроїдів кожного кластера. Ця частина алгоритму дозволяє швидко знайти вирішення задачі сегментації, але наближене, котре є вхідними даними для наступного етапу.

Під час другого етапу алгоритму точки по черзі перегруповуються, в тому випадку, якщо це приводить до зменшення суми відстаней, а координати центроїдів кластерів перераховуються після перегрупування кожної точки. Кожна ітерація під час другого етапу складається тільки з одного проходу по всіх точках.

По закінченні виконання описаного алгоритму програма сегментації може видавати додаткові дані, такі як:

– суми відстаней «точка – центроїд»;
– координати центроїдів.

Алгоритм кластеризації може сходитися до локального оптимуму, якщо при розділенні точок, переміщення будь-якої точки в інший кластер збільшує результуючу суму відстаней. Ця проблема може бути вирішена шляхом розумного (вдалого) вибору початкових точок [15].



а) б)

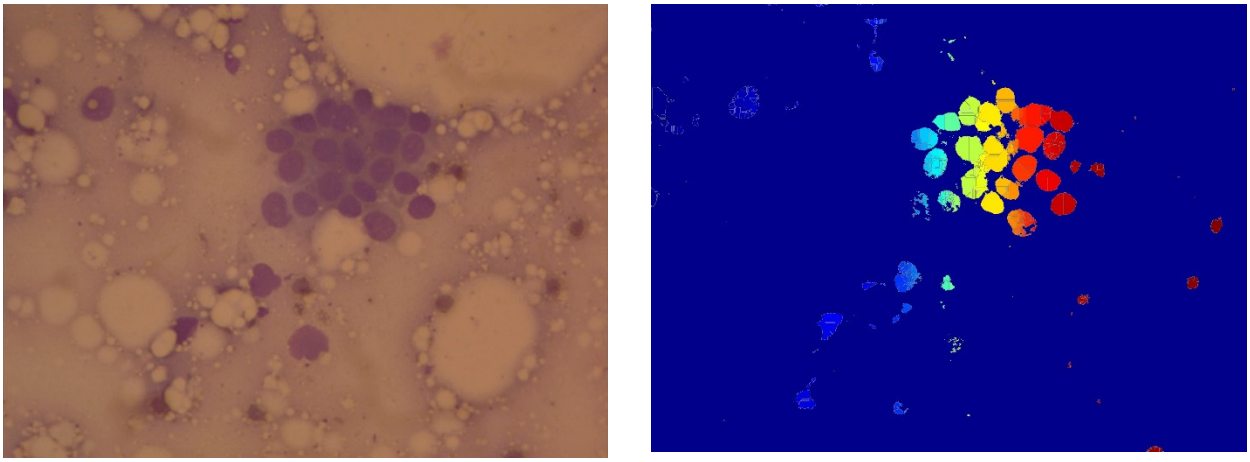
Рисунок 1.5 – Зображення клітин: а) вихідне зображення; б) півтонове, отримане після сегментації методом кластеризації

На рисунку 1.5 представлено результат сегментації k-методом. В результаті сегментації крім необхідних об’єктів є багато помилково сегментованих, майже неможливо побачити границі між клітинами, крім того даний алгоритм має значну обчислювальну складність.

1.2.3 Сегментація зображень за морфологічними водоподілами

Поняття водоподілу грунтується на представленні зображення тривимірною поверхнею, з двома просторовими координатами і рівнем яскравості. Для такої постановки розглядаються точки 3-х типів: (а) точки локального мінімуму; (б) точки, що знаходяться на схилі. Для таких вода стікає в один локальний мінімум; (в) точки, що знаходяться на піку. Для таких вода стікає більш ніж в один локальний мінімум. Для кожного локального мінімуму точки умови (б) називаються басейном або водозбором. Точки умови (в) – лінії водоподілу. Головна мета – знаходження ліній водоподілу.

На рисунку 1.6 представлено результат сегментації за водоподілами.



а) б)

Рисунок 1.6 – Зображення клітин: а) вихідне зображення; б) отримане після сегментації методом водоподілу

Основна ідея в тому, що у локальних мінімумах проколоті отвори і знизу підступає вода. Коли вода в двох сусідніх басейнах буде близькою до того, щоб злитися, ставиться перегородка. У міру наповнення водою залишаються видними тільки перегородки. На практиці метод сегментації за водоподілами часто застосовується не до самого зображення, а до його градієнту. Локальний мінімум – мале значення градієнта.

1.2.4 Сегментація зображень за принципом нарощування областей

Метод нарощування областей полягає в тому, що сусідні елементи зображення, характеристики яких задовольняють якусь умову, групуються разом і утворюють область [18]. Умови угруповання залежать від завдань сегментації. У самому простому випадку групують елементи за яскравістю, але цей спосіб сильно поступається за швидкістю виконання сегментації іншим. Для нарощування областей часто використовуються функції енергії [20], функції Байєса, вейвлет і властивості фракталів, а також апарат нейронних мереж.

Нарощування областей може відбуватися по-різному. Найбільш простий спосіб – поточкове нарощування [18]. Початковий піксель визначається як

область, і якщо він задовольняє необхідним умовам, до нього приєднується сусідня область. Інший спосіб заснований на моделях, подібних активному контуру, і моделі «Змій» [20]. Тут розраховують згладжують і розтягують сили, що діють на контур області.

Об'єднання областей можна виконати з урахуванням різноманітних умов в залежності від завдань сегментації, наприклад, таких, як порівняння середніх значень півтонових величин в областях [18], розподіл ймовірності, фрактальної розмірності, текстурних примітивів, ентропії, енергії [20].

Часто структура тканини є фоном і не дозволяє виділити клітину, так як її елементи мають яскравість і рівні перепадів яскравості, що збігаються з яскравістю фону (рисунок 1.7). Це не дає можливість виділити зовнішній контур клітини або сегментувати його за обраними порогами. В такому випадку найчастіше використовується метод нарощування областей.

У більшості випадків він починається з операції визначення випадковим чином маленьких областей розміром у кілька пікселів, які в ході роботи алгоритму ростуть і утворюють інформаційні області [21-24]. Для зображень з плавно змінюваною яскравістю це не дуже ефективно, так як результат виділення залежить від типу зображення та кількості пікселів при ініціалізації.

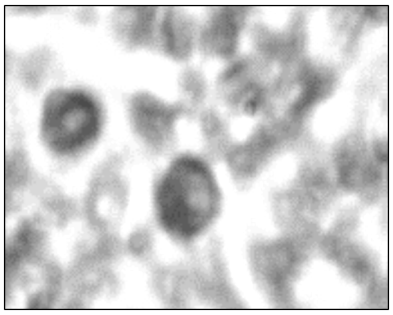
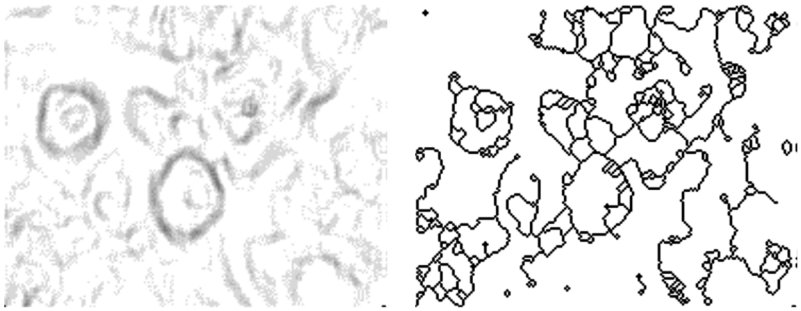


Рисунок 1.7 – Фрагмент зображення гістологічного препарату

Якщо використовувати замість вихідних пікселів бінарні області, отримані в результаті послідовного виконання наступних операцій: морфологічний градієнт (рисунок 1.8 а), півтонове утончування, бінаризація, інверсія (рисунок 1.8, б), то можна зменшити час обробки.

У результаті виходить бінарне зображення, що складається з щільно прилеглих областей. Ці області розділені лініями товщиною в один піксель, що дозволяє обмежуватися тільки об'єднанням областей і ігнорувати їх зростання. Об'єднання проводиться на основі обчислення характеристик областей. Для цієї операції необхідно мати вихідне півтонове зображення та організувати стек, що характеризує області.



а) б)

Рисунок 1.8 – Фрагмент зображення гістологічного препарату: а) результат виконання операції морфологічного градієнта; б) результат утончування

Обчислення характеристик виконується за допомогою процесу ідентифікації – визначення та класифікації об'єктів на вихідному бінарному зображенні. Бінарне зображення має дві градації яскравості, які відображають об'єкти і фон.

Припустимо, що фон відображається пікселями зі значеннями, рівними 0, а об'єкти – зі значеннями, рівними 1. У ході ідентифікації кожен об'єкт повинен отримати свій індивідуальний колір. Ідентифікація виконується за допомогою сканування зображення. Як тільки зустрічається піксель, рівний 1, координати його 8-сусідів, які теж рівні 1, заносяться в стек. Потім триває сканування поточної лінії за пікселями та фарбування їх в колір оброблюваного об'єкта. одночасно відбувається накопичення проміжних параметрів, необхідних в наступних вимірюваннях. Як тільки зустрічається піксель фону, з стека береться наступний піксель об'єкта та сканується горизонтальна лінія пікселів, рівних 1, елементом якої він є. Координати бінарних 8-сусідів крайніх пікселів також

заносяться в стек. Ці операції виконуються доти, поки стек не опиниться порожнім.

У результаті виходить об'єкт, виділений індивідуальним кольором, за яким, використовуючи відповідність пікселів на мультифазному і півтоновому зображеннях, обчислюються такі характеристики областей, як середнє значення (1.1) і дисперсія (1.2).

$$M = \frac{1}{\text{число пікселів}} \cdot \sum_{\text{об'єкт}} \text{значення пікселя}. \quad (1.1)$$

$$\sigma = \frac{1}{\text{число пікселів}} \cdot \sum_{\text{об'єкт}} \text{значення пікселя}^2 - \text{M}^2. \quad (1.2)$$

Будемо вважати область, яка ідентифікується, першою батьківською. Її ідентифікація проводиться двічі. Вдруге при ідентифікації визначаються сусідні області. Після цього їх характеристики порівнюються з батьківської областю за такими умовами:

– різниця дисперсії для півтонової величини не повинна перевищувати заданого значення, що визначає відмінності клітини від тканини;

– середнє значення яскравості в кожній області не повинно виходити за межі, обмежені дисперсією іншої області.

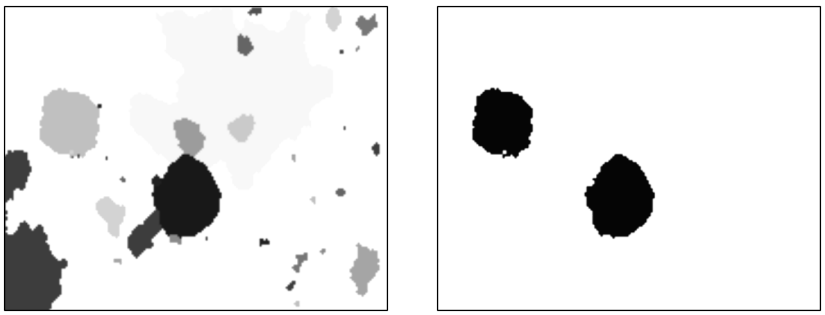
При виконанні цих умов області об'єднуються шляхом їх перевизначення, в результаті чого область, яка ідентифікується, забарвлюється кольором батьківської області, а також шляхом морфологічної операції замикання, виконаної для батьківської області на загальному мультифазному зображенні. Граничні розділяючі лінії між областями видаляються. Потім обчислюються характеристики нової об'єднаної області:

$$\begin{cases} M_{12} = \frac{A_1 \cdot M_1 + A_2 \cdot M_2}{A_1 \cdot A_2} \\ \sigma_{12} = \frac{1}{A_1 \cdot A_2} \left(A_1 \cdot \sigma_1^2 + \frac{A_1 \cdot A_2 \cdot (M_1 - M_2)^2}{A_1 \cdot A_2} + A_2 \cdot \sigma_2^2 \right), \end{cases} \quad (1.3)$$

де M_1, M_2 – середні значення півтонових величин для батьківської і сусідньої області; σ_1, σ_2 – дисперсії півтонових величин для батьківської і сусідньої області; A_1, A_2 – площі батьківської і сусідньої області; M_{12} – середнє значення півтонової величини для об’єднаних областей; σ_{12} – дисперсія півтонових величин для об’єднаних областей [5].

При виконанні всіх умов область заново аналізується скануванням для пошуку сусідніх необроблених областей. Якщо всі області оброблені, одна з сусідніх областей аналізується як батьківська.

У результаті виходять ряд областей, що відповідають різним ділянкам зображення (рисунок 1.9,а). Для визначення приналежності області обчислюються її геометричні характеристики, за якими проводиться класифікація області як клітини, тканини або іншого гістологічного об'єкта. Області, що не відповідають клітині, видаляються, а ті, що залишилися, заливаються відповідним кольором. У результаті виходить бінарне зображення клітин (рисунок 1.9,б).



а) б)

Рисунок 1.9 – Фрагмент зображення гістологічного препарату: а) результат об’єднання областей; б) бінарне зображення клітин виявлених при порівнянні площ

Такий підхід має ряд переваг. Результат сегментації стабільний і залежить тільки від порога дисперсії, так як об'єднуються конкретні області, які не залежать від особливостей розміщення об’єктів на зображенні. Він швидше класичних алгоритмів зростання областей, оскільки заснований тільки на

об'єднанні областей. Окремі області, що виділяються завдяки утонченню півтонового градієнта зображення, належать клітинам або їх фрагментами, що значно покращує якість визначення клітини. Тому цей алгоритм можна визнати досить ефективним для сегментації клітин.

Недоліком даного методу є велика обчислювальна складність і складність алгоритму та його програмної реалізації.

1.3 Методи оцінки подібності зображень, метрики

На сьогоднішній день існує чимало алгоритмів сегментації зображень, які використовують різні ознаки і характеристики зображень.

Варто відзначити, що оцінка результатів сегментації може бути проведена візуально, однак при цьому остаточні висновки виявляються досить суб'єктивними. Відомий альтернативний підхід, в якому оцінка якості алгоритмів сегментації зображень проводиться за кінцевим результатом роботи технічної системи, наприклад, в системах технічного зору. Однак даний підхід є скоріше якісним, ніж кількісним, оскільки в цілому задовільні результати роботи технічної системи аж ніяк не означають, що обраний найкращий алгоритм.

Необхідно відзначити, що при дослідженні алгоритмів сегментації зазвичай виникають наступні проблеми:

– Проблема вибору найкращого алгоритму сегментації, відповідного класу аналізованого зображення;

– Проблема знаходження об’єктивного критерію, що дозволяє оцінювати обґрунтованість вибору алгоритму.

Існує багато підходів до класифікації методів кількісної та якісної оцінки алгоритмів сегментації, проте найбільш загальним способом їх класифікують на суб’єктивні і об’єктивні.

Суб'єктивні (вони ж візуальні) – найбільш широко використовувані методики оцінки. Їх основний недолік, власне, відображений у назві цього класу – оцінка якості дається людиною, тому у різних експертів ця оцінка може кардинально відрізнятися.

Об’єктивні (емпіричні) – методи оцінки засновані на певних математичних критеріях оцінки. Дані критерії кількісної оцінки якості сегментації можна поділити на дві основні групи:

1. Несупервізорні критерії – ті, що базуються на обчисленні різноманітних статистик, дані критерії використовуються при відсутності апріорної інформації про об’єкти отримані в результаті сегментації.

2. Супервізорні критерії, базуються на обчисленні міри відмінності сегментації від істинної форми об’єкта – еталона. В даному випадку еталон задається експертом.

Для того що вирішити вище описані проблеми найкращим варіантом є використання супервізорних критеріїв. Для кількісної оцінки якості сегментації відомі такі супервізорні критерії: FOM, критерій Хаусдорфа, RMS та ін. [27].

Розглянемо деякі з вище перелічених критеріїв.

Критерій FOM (Figure of Merit), запропонований Праттом, дозволяє знайти відстань між двома об’єктами представленими в вигляді контурів X, Y.

$$FOM(X,Y) = \frac{1}{\max\{card(X),card(Y)\}} \cdot \sum_{i=1}^{card(Y)} \frac{1}{1+d^2(i)}, \qquad (1.4)$$

де X – множина точок еталонного контуру; Y – множина точок контуру отриманого в результаті сегментації; card(X) I card(Y) відповідно кількості пікселів в множинах X і Y; d(i) відстань між і-м пікселем множини X і найближчим пікселем множини Y.

Критерій RMS (root mean squared error) – середньо квадратична похибка, запропонована в роботі [28].

$$RMS(I_1, I_2) = \left[\frac{1}{card(X)} \cdot \sum_{x \in X}(I_1(x) - I_2(x))^2\right]^{\frac{1}{2}}, \qquad (1.5)$$

де $I_1(x)$ інтенсивність пікселя x в I_2, X множина пікселів на сегментованому зображенні.

Критерій Хаусдорфа – один із супервізорних критерії, що базується на використанні метрик і поняття відстані в метричному просторі. Перевагою даного методу є те, що він порівнює форму об'єктів і дозволяє знайти максимальну відмінність між ними.

Метрики – важливий інструмент вирішення багатьох завдань розпізнавання образів та інтелектуального аналізу даних. Наявність метрики в просторі дозволяє приймати рішення про належність до множини або про подібність множин на основі кількісного показника. Величина метрики (відстані) часто безпосередньо пов'язана з імовірнісними характеристиками віднесення елемента до класу.

Метрика – функція, яка кожній впорядкованій парі точок x і y простору, ставить у відповідність дійсне число d(x,y). При цьому функція d(x,y) має наступні властивості:

d(x, y) ≥ 0, d(x, y) = 0 тоді і тільки тоді, коли x = y;

d(x, y) = d(y, x);

d(x, y) ≤ d(x, z) + d(z, y).

Введення метрики d(x,y) в просторі зображень дозволяє стверджувати про близькість точок в тому чи іншому просторі або про міру подібності чи розбіжності між аналізованими зображеннями.

Нехай задано деяку кінцеву множину $S = \{S_1, S_2, \dots S_n\}$ вхідних зображень, кожне з яких є точкою в n-вимірному просторі зображень. Міру подібності зображень можна ввести як функцію двох аргументів $L(S_k, S_i)$, де $S_k, S_i \in S$. При цьому функція $L(S_k, S_i)$ має такі властивості:

1. властивість симетрії, тобто $L(S_k, S_i) = L(S_i, S_k)$;

2. областю значень функції є множина невід'ємних чисел, тобто $L(S_k, S_i) \geq 0, k, i = 1,2 \ldots, n$;
3. міра подібності зображення з самим собою приймає екстремальне значення в порівнянні з будь-яким іншим зображенням, тобто залежно від способу введення міри схожості виконується одне з двох співвідношень:
   a. $L(S_k, S_k) = \max\big(L(S_k, S_i)\big)$,
   b. $L(S_k, S_k) = \min\,(L(S_k, S_i))$;
4. у разі компактних образів функція $L(S_k, S_i)$ є монотонною функцією видалення точок S_k і S_i одну від одної в n-вимірному просторі.

Аналіз властивостей метрики і міри схожості зображень показує, що вимоги до функції $L(S_k, S_i)$ неважко виконати в метричних просторах. Зокрема, якщо в метричному просторі введено відстань, то її може бути використано у вигляді міри схожості зображень.

1.3.1 Метрика Евкліда

Це, мабуть, найбільш часто використовувана міра відстані. Вона є геометричною відстанню в багатовимірному просторі і обчислюється так:

$$L = \sqrt{\sum_{i=1}^{N}(A_i - B_i)^2}\ ,$$

де: L - Відстань між об'єктами Ai B; A_i- Значення і-властивості об'єкта A;

B_i- Значення і-властивості об'єкта B.

З геометричної точки зору, евклідова міра відстані може виявитися безглуздою, якщо ознаки виміряні в різних одиницях. Щоб виправити становище, вдаються до нормування кожної ознаки. Застосування евклідової відстані виправдано в таких випадках, коли властивості (ознаки) об'єкта

однорідні за фізичним змістом і однаково важливі для класифікації і простір ознак збігається з геометричним простором [25, 29].

1.3.2 Метрика Хаусдорфа

Метрика Хаусдорфа – це метрика, визначена на множині всіх компактних підмножин метричного простору. Таким чином, метрика Хаусдорфа перетворює множину всіх компактних підмножин метричного простору в метричний простір.

Нехай X і Y це дві компактні підмножини метричного простору M. Тоді відстань по Хаусдорфу $d_H(X,Y)$ між точками X і Y це мінімальне значення r, таке що замкнутий r-окіл X містить Y і також замкнутий r-окіл Y містить X.

Іншими словами, якщо |xy| позначають відстань між точками x і y в M то:

$$d_H^Z(X,Y) = max\left\{\sup_{x\epsilon X}\inf_{y\epsilon Y}|xy|, \sup_{y\epsilon Y}\inf_{x\epsilon X}|xy|\right\}$$

Іноді метрика Хаусдорфа розглядається на множині всіх замкнутих підмножин метричного простору, в цьому випадку відстань між деякими підмножинами може дорівнювати нескінченності [17].

1.4 Аналіз програмних засобів кількісної оцінки алгоритмів сегментації

Відмітимо, що оцінка результатів сегментації може бути проведена візуально, однак при цьому кінцеві висновки є дуже суб’єктивними. Відомий альтернативний підхід, в якому оцінка якості алгоритмів сегментації проводиться по кінцевому результаті роботи технічної системи, наприклад в системах технічного зору. На сьогоднішній день існує велика кількість

програмного забезпечення для аналізу цифрових зображень, наприклад: ImageJ, Image-Pro Plus, ImageTool, PhotoLib, ScopoTek та ін..

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. 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.

На рисунку 1.10 наведено вигляд вікна програми, в якому здійснюється вимірювання розмірів зерна породи.

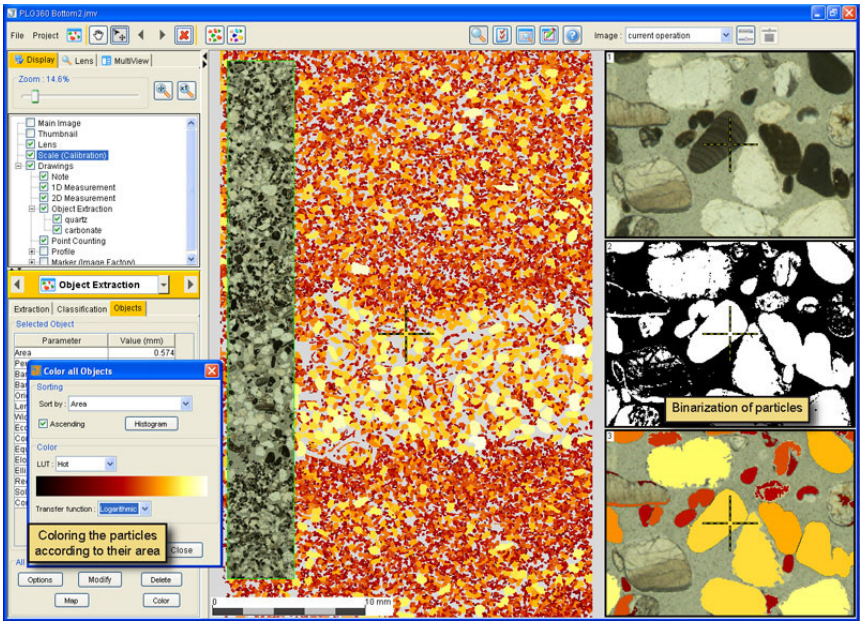


Рисунок 1.10 – Вікно програми JMicroVision

ImageJ – це програма з відкритим вихідним кодом для аналізу і обробки зображень. Написана мовою Java співробітниками National Institutes of Health і поширюється без ліцензійних обмежень як суспільне надбання. Відкритий API дозволяє гнучко нарощувати функціонал за рахунок додаткових плагінів, а вбудована макромова - автоматизувати складні повторювані дії. ImageJ широко застосовується в біомедичних дослідженнях, астрономії, географії та інших дисциплінах, пов'язаних з аналізом зображень .

Плагіни сторонніх розробників охоплюють широке коло завдань аналізу і обробки зображень: дозволяють проводити тривимірну візуалізацію в діапазоні від клітин до рентгенологічних зображень, автоматичні порівняння аж до створення автоматизованих систем вивчення, наприклад, в гематології.

Архітектура плагінів ImageJ і вбудована в програму система розробки робить цю платформу вельми популярною для роботи і викладання аналізу та обробки зображень [26].

1.5 Висновки до розділу 1

У розділі проаналізовані біомедичні зображення на прикладі гістологічних зображень. Також проведено експерименти з алгоритмами сегментації біомедичних зображень. Здійснено аналіз метрик для порівняння зображень. Проведено аналіз програмних засобів порівняння зображень.

2 ОПТИМАЛЬНИЙ АЛГОРИТМ ПОРІВНЯННЯ ОБЛАСТЕЙ ЗОБРАЖЕНЬ В МЕТРИЦІ ГРОМОВА-ХАУСДОРФА

2.1 Метрика Громова-Хаусдорфа

Громов модифікував метрику Хаусдорфа в 1981 р. Метрика Громова-Хаусдорфа – спосіб визначити відстань між двома компактними метричними просторами.

Метрикою Громова-Хаусдорфа називається метрика на множині всіх ізометричних класів компактних метричних просторів, що задається для будь-яких двох класів X^* і Y^* з представниками X і Y відповідно як:

$$d_{GH}(X^*, Y^*) = inf\ d_H^Z(f(X), g(Y)), \qquad (2.1)$$

де d_H^Z – відстань Хаусдорфа (формула 1.5), а мінімум береться по всіх метричних просторах Z та ізометричних вкладеннях f : X→Z, g : Y→Z. Відповідний метричний простір називається простором Громова-Хаусдорфа [32-33].

За допомогою модифікації Громова можна знайти мінімальну відмінність між двома контурами, якщо обчислити відстані Хаусдорфа при всіх можливих розміщеннях двох контурів. При цьому необхідно використовувати ізометричні перетворення, такі як зміщення і поворот на кут.

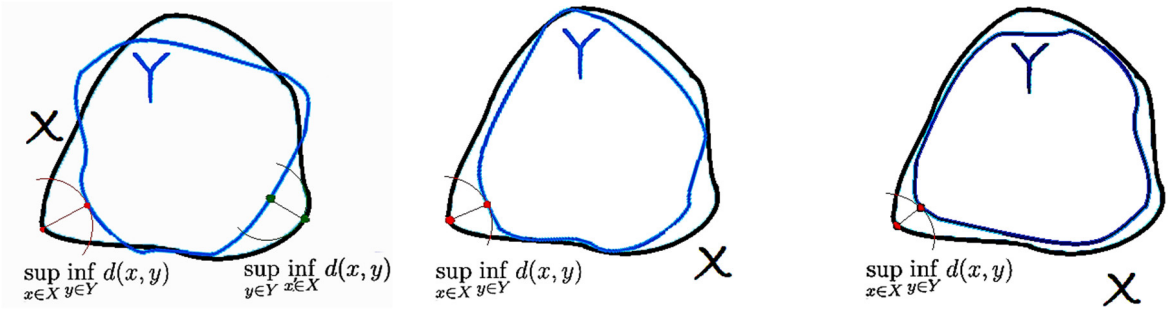
На рисунку 2.1 показано спрощену ілюстрацію принципу пошуку відстані Громова-Хаусдорфа.

На рисунку 2.1 а) показано початкове розміщення двох контурів X та Y, згідно метрики Хаусдорфа, максимальна відстань шукається спочатку від контуру X до Y ($\sup\limits_{x\epsilon X} \inf\limits_{y\epsilon Y}|xy|$), а потім навпаки від Y до X ($\sup\limits_{y\epsilon Y} \inf\limits_{x\epsilon X}|xy|$), з двох отриманих значень вибирається максимальне, що і є відстанню Хаусдорфа.

На рисунку 2.1 б) показано якою є відстань Хасдорфа якщо виконати ізометричне перетворення одного з контурів (в даному випадку Y) при повороті

контуру Y відносно центру мас ми отримали таке розміщення контурів при якому відстань Хаусдорфа зменшилася.

На рисунку 2.1 в) показано таке розміщення контурів при якому відстань Хаусдорфа буде найменшою, в даному випадку це і буде відстанню Громова-Хаусдорфа.



а) б) в)

Рисунок 2.1 – Ілюстрація пошуку відстані Громова-Хаусдорфа; а) вихідне розміщення контурів; б), в) контур Y повернутий на деякий кут

Доцільно спочатку дізнатися потужності множин X та Y, щоб в подальшому виконувати ізометричні перетворення над одним контуром, тобто тим, в якого менше точок. Таким чином можна зменшити загальну обчислювальну складність алгоритму.

Як видно з рисунків, основною перевагою метрики Громова-Хаусдорфа є можливість отримати точніші дані в порівнянні з метрикою Хаусдорфа. Недоліком є підвищена обчислювальна складність [2].

2.2 Алгоритм знаходження контуру зображення

Процес виявлення контурів є одним із найважливіших етапів в порівнянні зображень. Маючи контури зображень можна значно зменшити обчислювальну складність алгоритму порівняння цих зображень, адже в такому випадку немає

необхідності розглядати точки, що знаходяться в середині контуру. Крім того контур дає цілком точний опис форми об’єкта і на його основі можна з легкістю знайти площу об’єкта.

Контуром зв’язаної множини пікселів R називається множина всіх пікселів, що належать R, кожен з яких має, як мінімум, одного безпосереднього сусіда, розміщеного поза множиною R, якщо R – множина пікселів сегментованого об’єкта. Пікселі називається безпосередніми сусідами (б-сусідом), якщо відповідні елементи мають спільну сторону. Непрямі сусіди (н-сусіди) – ті, що дотикаються тільки кутами. Термін N - сусід , де 0 < = N < = 7, буде використовуватися для позначення пікселя, позиція якого визначається його положенням у відповідності із різними значеннями N (рисунок 2.2).

3	2	1
4	p	0
5	5	7

Рисунок 2.2 – Маска, що визначає положення пікселів відносно положення p

Варто відмітити, що б-сусіди є N-сусідами при парних значеннях N, а н-сусіди відповідають непарним N [16].

Обхід пікселів контуру можна виконувати у відповідності з маршрутом, причому для такого обходу завжди можна використовувати деякий замкнутий маршрут. Спочатку знаходимо початковий піксель A, який можна знаходити кількома способами, в тому числі і за допомогою обходу площини зверху вниз та зліва на право.

Процедура пошуку контуру закінчується, коли наступним пікселем, що розглядається в процесі роботи алгоритму, є початковий.

Вводиться булева змінна «First» для того, щоб можна було розрізняти початок алгоритму і повернення в початковий піксель.

Алгоритм пошуку контуру включає наступні операції:

//Оголошуємо змінні

A – початкова точка контура множини R;

C – поточна точка, яка досліджується;

S – напрямок пошуку, що виражається кодом N (рисунок 2.2);

first – булева зміна, котра істинна якщо A == C

found – булева змінна, яка істинна коли знайдено чергову точку контуру.

// початок роботи алгоритму

0. В контурі вибирається точка A, така, щоб її 4-сусід не належав до до множини R.
1. Точці A присвоюється значення поточної точки C, напряму пошуку S – 6, first = true;
2. while (C != A OR first == true)

   do

   begin
3. found = false;
4. while (found == false) /* поки не знайшли точку контуру, цикл виконується не більше 3 разів */

   do

   begin
5. if B is (S-1)-сусід C AND B нележить R then
6. B = C, S -= 2, found = true;
7. else if B is S-сусід C AND B належить R then

   B = C, found = true;
8. else if B is (S+1)-сусід C AND B належить R then

   B = C, found = true;
9. else S+=2;

   end;
10. first = false;
11. end; //кінець алгоритму.

Цикл, що передбачений на 5-9 кроках алгоритму виконується не більше 3 разів, щоб не відбувався обхід навколо множини, що складається з одного пікселя [16].

За допомогою вище описаного алгоритму можна обійти всі точки контуру. В даному випадку доцільно всі знайдені точки записувати в масив, щоб забезпечити можливість подальших маніпуляції з контуром. Щоб це реалізувати необхідно оголосити масив на початку процедури і на кроках алгоритму 6,7,8 додавати знайдену точку в масив.

2.3 Алгоритм спрощення полігонального ланцюга

Знайдений, за допомогою алгоритму обходу контуру, ланцюг містить набір точок, які точно повторюють форму вихідного об’єкта. Але в такому випадку, порівнюючи два контури, необхідно опрацьовувати дуже велику кількість точок, що в свою чергу значно збільшує час обчислення відстані між контурами. Щоб цього уникнути використовуються різні підходи проріджения контуру, наприклад апроксимація сплайнами та кусково-лінійна апроксимація.

На даний час найчастіше використовується алгоритм алгоритм Дугласа-Рамера-Пекера, який ще називають алгоритмом ітеративної найближчої точки чи алгоритмом розбиття і злиття [34].

Алгоритмом задається початкова полілінія (контур, полігональний ланцюг) і максимальна відстань, яка може бути між вихідною і спрощеною полілініями, тобто максимальна відстань від точок вихідної до найближчої ділянки отриманої полілінії. Спрощена полілінія складається з підмножини точок, які визначаються з вихідної.

Початковий контур являє собою упорядкований набір точок. Алгоритм рекурсивно ділить полілінію. На вхід алгоритму подаються координати всіх точок, включно з першою і останньою, а також відстань ε. Перша і остання точка

зберігаються незмінними. Після чого алгоритм знаходить точку, найбільш віддалену від відрізка, що складається з першої і останньої. Якщо точка знаходиться на відстані, менше, ніж ε, то всі точки, які ще не були позначені для збереження, можуть бути викинуті з набору, і отримана пряма згладжує криву з точністю не нижче ε. Якщо ж відстань більше ε, то алгоритм рекурсивно викликає себе на наборі від початкової до даної і від даної до кінцевої точок (дана точка буде позначена для збереження).

По закінченню всіх рекурсивних викликів результуюча полілінія будується тільки з тих точок, що були позначені для збереження [34]. На рисунку 2.3 показано поетапне прорідження полігонального ланцюга за допомогою алгоритму Дугласа-Рамера-Пекера.

Опис вище представленого алгоритму за допомогою псевдокоду:

```
DouglasPeucker (int first, int last, double eps)
beigin
    max = -infinity
    index = -1
    for (int i = first + 1; i < last; i ++)
        distance = dist (points [i], segment (points [first], points [last]))
        if (distance > max && distance > eps)
        max = distance, index = i
    if (index == -1)
        return
    else
        rezult [index] = true
        DouglasPeucker (first, index, eps)
        DouglasPeucker (index, last, eps)
end
```

Розглянемо приклад для точок, заданих на рисунку 2.3, де суцільна лінія відображає вихідну лінію, і $\varepsilon = \sqrt{2}$.

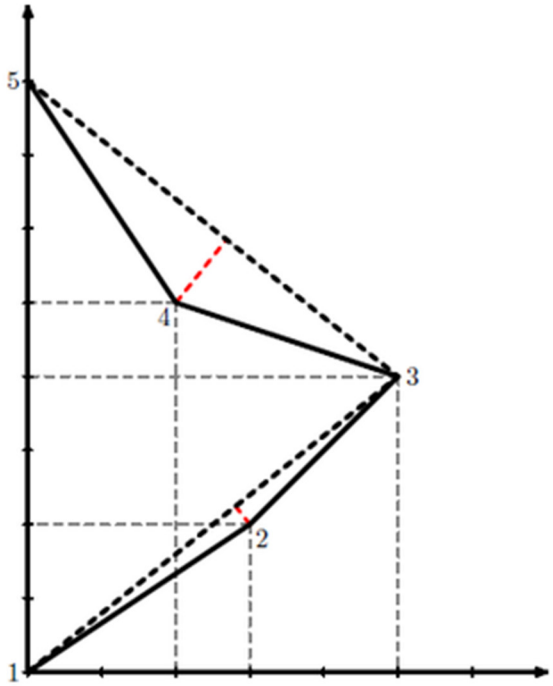


Рисунок 2.3 – Приклад проріджения контуру за допомогою алгоритму Дугласа-Рамера-Пекера

Для ламаної, представленої на рисунку 2.3 алгоритм Дугласа-Рамера-Пекера буде складатися з 9 кроків:

1) Знаходимо найбільш віддалену точку від відрізка 1 – 5, це точка 3.
2) Відстань до точки 3 більша за ε отже додаємо її в результат.
3) Виконуємо алгоритм для точок 1 і 3.
4) Знайдемо найбільш віддалену точку від відрізка 1 – 3, це точка 2.
5) Відстань до точки 2 менша за ε, отже повертаємося.
6) Виконуємо алгоритм для точок 3 і 5
7) Знайдемо найбільш віддалену точку від відрізка 3 – 5, це точка 4.
8) Відстань до точки 4 менша за ε, повертаємося.
9) Кінець алгоритму.

Як бачимо вхідний ланцюг, що складався з 5 точок вдалося скоротити до ланцюга з 3 точок. Звичайно в такому випадку форма частково спотворюється, але варто врахувати що в вихідному контурі зображення таких точок може бути декілька тисяч. І якщо правильно підібрати ε, можна досягнути оптимального результату проріджения і точності повторення форми вихідного контуру.

2.4 Алгоритм знаходження найкращого взаємного розміщення контурів

Як було сказано вище, для знаходження відстані Громова-Хаусдорфа треба знайти таке взаємне розміщення контурів, коли значення відстані Хаусдорфа буде мінімальне або максимально наближене до нього.

Необхідно відмітити, що пошук найкращого розміщення можна виконувати різними способами і кожен з них має свої особливості. Нижче буде розглянуто декілька способів які різняться за швидкістю виконання та кінцевою точністю результатів.

2.4.1 Алгоритм повного перебору

Для знаходження мінімальної відстані використаємо алгоритм, який приймає на вхід два проріджених контури X та Y і знаходить відстань в метриці Хаусдорфа для найкращого взаємного розміщення контурів за допомогою повного перебору.

Алгоритм складається з наступних кроків:

1) Обчислюються центри мас двох контурів.
2) Матриці контурів зсуваються так, щоб їх центри мас співпадали.
3) Обчислюється відстань Хаусдорфа.
4) Зсуваємо контур по осі x на 1 вліво
5) Знаходимо відстань Хаусдорфа. Якщо знайдена відстань менша за відстань знайдену на попередньому кроці, то повертаємося до кроку 4. Інакше переходимо до кроку 6.
6) Зсуваємо контур по осі x на 1 вправо
7) Знаходимо відстань Хаусдорфа. Якщо знайдена відстань менша за відстань знайдену на попередньому кроці, то повертаємося до кроку 6. Інакше переходимо до кроку 8.
8) Фіксуємо поточну координату x, і виконуємо кроки 4 -7 для осі y.

9) Після знаходження обох координат, обчислюється відстань Хаусдорфа і додається до результуючого масиву.

Недоліком вище описаного алгоритму є те, що він не виконує поворот контуру на кут, що є необхідною умовою для знаходження відстані Громова-Хаусдорфа, тому для вирішення цієї проблеми необхідно додати ще один крок під номером 10, на якому буде відбуватися поворот контуру. А сам алгоритм виконувати в циклі, обчислюючи відстані для кожного повороту на кут.

Для знаходження координат повернутої точки відносно довільної використовується формула 2.2 та 2.3 відповідно для кожної координати.

$$x' = x0 + (x - x0) \cdot \cos(a) - (y - y0) \cdot \sin(a), \quad 2.2$$

$$y' = y0 + (y - y0) \cdot cos(a) + (x - x0) \cdot sin(a), \quad 2.3$$

де x', y' - координати повернутої точки, $x0, y0$ координати довільної точки.

Отже на кроці 10 необхідно виконати обчислення повороту для кожної точки контуру на 1° відносно геометричного центру мас контуру. В результаті буде отримано нові координати повернутих точок, тобто повернутий контур, який буде виступати в якості вхідних даних на наступній ітерації циклу. Цикл відбувається поки виконується наступна умова: $0 < \theta < 360$, де θ кут повороту.

Перевагою даного алгоритму є максимальна точність результатів, а недоліком надзвичайно велика обчислювальна складність.

2.4.2 Алгоритм пошуку кута повороту за допомогою найдовшої хорди

Алгоритм приймає на вхід два проріджених контури X та Y і знаходить таке розміщення контурів при якому відбувається їх максимальне перекриття. Метрика Громова-Хаусдорфа передбачає виконання ізометричних перетворень над контурами для знаходження максимального перекриття.

В даному випадку необхідно знайти кут на який потрібно повернути один з контурів щоб досягти максимального перекриття. Дане завдання пропонується

вирішити за допомогою хорд проведених між вершинами контуру. Опустивши перпендикуляр на вісь ординат з верхньої точки хорди можна знайти кут повороту контуру (рисунок 2.4).

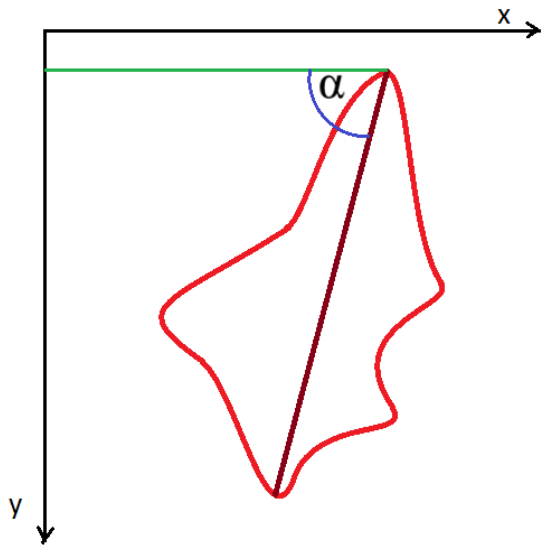


Рисунок 2.4 – Ілюстрація знаходження кута повороту фігури по хорді і перпендикуляру опущеному на вісь ординат

Пошук хорд потрібно виконувати над еталонним контуром і контуром що порівнюється. Для того щоб відкинути менш інформативні хорди і зменшити кількість обчислень, пошук найдовшої хорди контуру відбувається на опуклому контурі. Далі необхідно співставити контури так, щоб вони співпадали їх геометричні центри мас і повернути менший контур так, щоб хорди були паралельні.

Останнім етапом буде обчислення відстаней Хаусдорфа, яке буде виконуватися 2 рази, після першого обчислення один з контурів необхідно повернути на 180º, оскільки невідомі напрями хорд.

За допомогою псевдокоду даний алгоритм можна описати так:

//Оголошуємо змінні

contour1, contour2 – списки точок контурів після проріджения;

contour1Centroid, contour2Centroid – центри мас відповідних контурів;

contour1Chorde, contour2Chorde – найдовші хорди контурів;

distance – відстань Громова-Хаусдорфа

// початок роботи алгоритму

0. Знаходимо центри мас і опуклі контури:

```
contour1Centroid = getCentroid(contour1);
contour2Centroid = getCentroid(contour2);
contour1 = getConvex(contour1);
contour2 = getConvex(contour2);
```

1. Знаходимо найдовшу хорду першого контуру:

```
maxDist1 = 0;// відстань між  2 точками що хорди
for (int i = 0; i < contour1.size(); i++) {
    for (int j = i + 2; j < contour1.size(); j++) {
        if (i==0 AND j == contour1.size()-1) {continue;}
            tmpDist = euclideanDist(contour1.get(i), contour1.get(j));
            if (maxDist1 <  tmpDist) {
                maxDist1 = tmpDist;
                //створюємо об'єкт хорди
                contour1Chorde = new Chorde(contour1.get(i), contour1.get(j));
        }
    }
}
```

2. Виконуємо аналогічно кроку 1, операції для другого контуру.
3. Співставляємо контури:

```
//зсув другого контуру так щоб центи мас співпадали
shiftX = contour1Centroid.x – contour2Centroid.x;
shiftY = contour1Centroid.y – contour2Centroid.y;
contour2 = contour2.shift(shiftX, shiftY);
//повертаємо 2 контур і обчислюємо відстань
contour2 = rotate(contour2Chorde.getAngle() – contour1Chorde.getAngle);
distance = distHausdorff(contour1, contour2);
```

4. Повертаємо на 180º:

```
contour2 = rotate (contour2Chorde.getAngle() – contour1Chorde.getAngle);
newDistance = distHausdorff(contour1, contour2);
if (newDistance < distance){
    distance = newDistance;
}
```

5. Кінець алгоритму. Знайдено результуючу відстань distance, що і є відстанню Громова-Хаусдорфа.

Вище описаний алгоритм характеризується високою швидкодією, проте він розглядає тільки два можливі розміщення контурів, що негативно впливає на точність результатів, тому даний алгоритм може бути використаний лише для зображень певного класу.

2.4.3 Алгоритм пошуку кута повороту за допомогою трьох точок

Даний підхід спирається на алгоритм пошуку кута повороту за допомогою найдовшої хорди, тобто можна сказати що він є модифікацією. Суть даного алгоритму в тому, що необхідно знайти найдовшу хорду і найдовший перпендикуляр, опущений на середину хорди (рисунок 2.5).

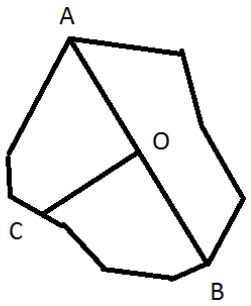


Рисунок 2.5 – Ілюстрація ключових точок

Найскладніше, в даному алгоритмі, знайти перпендикуляр, оскільки необхідно опустити перпендикуляр в конкретну точку – середину хорди, точка перетину перпендикуляру з контуром – невідома.

Щоб вирішити дану проблему необхідно використати аналітичний підхід до пошуку точки перетину перпендикуляру і контуру, даний алгоритм можна описати так:

0. Нехай задано ходу з точками A(x1;y1) B(x2; y2);
1. Знаходимо коефіцієнти рівняння прямої що проходить через хорду:

```
if(x1 > x2){
    switch(x1,x2);
    switch(y1,y2);
}
k = 0;
if(y2 != y1){
    k =  ( y2 - y1 ) / ( x2 - x1 );
 }
 b = y1 - k * x1;
//знайдено коефіцієнти рівняння y=kx+b
```

2. Знаходимо коефіцієнти рівняння прямої, що проходить через перпендикуляр опущений на середину хорди:

   k_{per} = -1 / k;

   b_{per} = (y1+y2)/2 //коефіцієнт b рівний координаті y точки середини хорди
3. Маючи коефіцієнти прямої, виконуємо перевірку з усіма відрізками контуру на наявність перетину:

   C – точка перетину; D = 0; – довжина перпендикуляру.

   for (int i =0; i < contour.size() – 1; i++){

   знаходимо коефіцієнти прямої ($k_{contLine}$, $b_{contLine}$) що проходить через відрізок контуру заданий точками A2 = contour.get(i) і B2 = contour.get(i+1) як показано в 1 кроці. Перевіряємо умову паралельності.

   if(k_{per} == $k_{contLine}$){continue;}

   знаходимо точку перетину двох прямих:

   x = (b2 - b1) / (k_{per} - $k_{contLine}$);

```
y =  k_per *(x) + b1;
перевіряємо умову чи точка лежить на відрізку контуру:
if((x >= min(A2.x,B2.x)) && (x <= max(A2.x,B2.x))
        && (y >= min(A2.y,B2.y)) && (y <= max(A2.y,B2.y))) {
        tmpD = euclideanDist(P, middle(A,B));
        if(tmpD > D){D = tmpD; C =  new Point(x,y);}
  }
}
```

4. Кінець алгоритму, точка C буде точкою перетину найдовшого перпендикуляру і контуру.

Коли відомий найдовший перпендикуляр, можна накласти два контури так, щоб їх хорди були паралельні, а напрямки перпендикулярів співпадали.

Перевагою такого алгоритму є те, що відстань Хаусдорфа вимірюється тільки один раз, але через специфічність зображень він може давати гірші результати в порівнянні з методом найдовшої хорди.

2.4.4 Алгоритм пошуку кутів повороту за допомогою методу січних

Даний алгоритм приймає на вхід контур зображення і крок формування січних хорд, що задається в градусах. На рисунку 2.6 зображено деякий контур з відкладеними на ньому січними. В даному алгоритмі важливо вибрати 2 ключові точки, перша точка A – це вершина многокутника, що найбільш віддалена від центру мас O, а точка B – найближча.

Таким чином точка A вважається точкою від якої проводять січні, що перетинають контур. Точка B – це початкова точка через яку проводиться перша січна. Наступні січні обчислюються після повороту початкової точки на заданий кут.

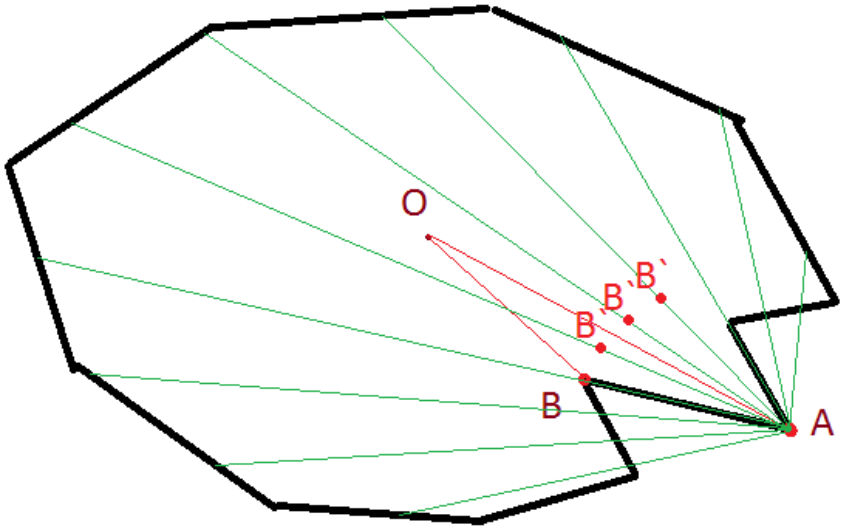


Рисунок 2.6 – Ілюстрація методу січних

Алгоритм пошуку січних хорд можна описати так:

rotationAngle - заданий кут повороту.

0. Знаходимо ключові точки A, B;

   Chordes[]; оголошення масиву хорд.

1. Виконуємо повороти точки B на кут rotationAngle

   Маючи точку B` і A можна знайти коефіцієнти рівняння прямої, що проходить через них і в результаті знайти точку перетину з контуром:

```
for(int i = rotationAngle; i <= 180;i+=rotationAngle){
    B` = rotate(B, rotationAngle);
    P = getIntersectionPoint(contour, B`,A);
    if(P){
            Chordes.add(new Chorde(A,P));
    }
}
```

2. Кінець алгоритму.

Пошук точки перетину здійснюється аналогічно принципу описаного в методі трьох точок. Знайдені січні на одному контурі порівнюються з кожною на іншому контурі, накладаються контури так щоб дві січні були паралельні, центри мас співпадали і напрями січних були однаковими. Напрям січної

знаходиться при порівнянні координат початкової точки A і точки перетину з контуром.

Перевагою даного алгоритму є більша точність за рахунок перебору більшої кількості хорд. Максимальна кількість січних хорд що може бути проведена залежить від заданого кута і форми фігури. Чим більше форма об’єкту наближається до круга, тим більше буде січних хорд, і навпаки чи більш видовжена форма, тим менше січних хорд. В гіршому випадку кількість рівна $180/\theta$, де θ – це заданий кут. Максимально можлива кількість обчислень відстані Хаусдорфа рівна $n = 2 \times 180/\theta$.

Недоліком даного підходу є більша обчислювальна складність, за рахунок перебору січних і пошуку точок перетину січної з контуром.

2.4.5 Алгоритм пошуку кутів повороту з пороговою фільтрацією хорд

Даний алгоритм приймає на вхід контур зображення і два параметри: поріг перекриття хордою контур і поріг коефіцієнту довжини хорди. Даний алгоритм передбачає перебір усіх можливих унікальних хорд проведених через вершини апроксимованого контура приведеного до опуклого. Приведення до опуклого необхідно для того щоб відкинути хорди які являються менш інформативними.

Для кожної хорди відбувається обчислення коефіцієнту перекриття об’єкту і коефіцієнт довжини, який вираховується з відношення довжини найдовшої хорди до довжини поточної.

Маючи задані пороги по двох коефіцієнтах можна додатково відкидати хорди і тим самим зменшувати обчислювальну складність.

Для того щоб обчислити коефіцієнт перекриття необхідно перевірити скільки дискретних точок, що складають хорду, входять в об’єкт. Тобто для початку необхідно растеризувати лінію, задану двома точками.

Існує декілька алгоритмів растеризації ліній, найпоширеніші з них це: алгоритм DDA-лінії, Алгоритм Брезенхейма, і модифікований алгоритм Брезенхейма, котрий додає згладжування.

Алгоритм DDA-лінії використовує обчислення з плаваючою крапкою, через що він являється повільнішим за алгоритм Брезенхейма. Скористаємося звичайним алгоритмом Брезенхейма (рисунок 2.7), так як в даному випадку немає необхідності використовувати модифікований алгоритм.

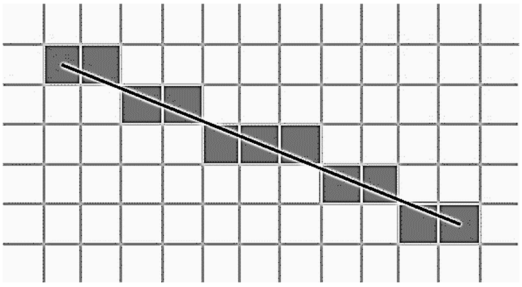


Рисунок 2.7 – Ілюстрація роботи алгоритму Брезенхейма

Алгоритм Брезенхейма – алгоритм, який визначає які точки в *n*-вимірному растрі треба накреслити для формування близького наближення для прямої лінії між двома заданими точками. Алгоритм загально використовується для рисування ліній на екрані по тій причині, що він використовує тільки цілочисельну суму, віднімання та бітові операції. Ці операції є дуже швидкими в стандартній архітектурі комп'ютерів. Алгоритм Брезенхейма став одним з перших алгоритмів, що розроблені в галузі комп'ютерної графіки [35].

Маючи алгоритм растеризації його можна використовувати при обчисленні коефіцієнту перекриття лінії і об’єкту. Для знаходження цього коефіцієнту необхідно порівняти значення кожного пікселя растеризованої лінії з відповідним пікселем на зображенні і встановити відношення кількості знайдених пікселів до кількості пікселів в лінії, дане відношення і буде шуканим коефіцієнтом.

Отже тепер можна перейти до пошуку хорд контуру.

Псевдокод алгоритму:

0. Оголошення та ініціалізація змінних.

   OverlapThreshold - поріг перекриття хордою об’єкту.

   DistThreshold – поріг довжини хорди.

   maxChorde – максимальні хорда

chordesList – набір хорд

1. Знаходимо хорди контуру і виявляємо максимальну хорду:

```
maxDist = 0;// відстань між  2 точками що хорди
for (int i = 0; i < points.size(); i++){
      for(int j = i + 2; j < points.size(); j++){
         if(i==0 && j == points.size()-1){continue;}
         line = rasteredLine(points[i], points[j]);
         dist = euclideanDist (points[i], points[j]);
         overlapCoef = getChordeOverlapCoef(line);
         chorde = new Chorde(i,j,dist, overlapCoef);
         chordesList .add (chorde);
         if(dist > maxDist) {
            maxDist = dist;
            maxChorde = chorde;
         }}
}
```

2. Сортуємо хорди по показнику, що рівний добутку обох коефіцієнтів.

Сортування необхідно для того, щоб при виборі занадто великих коефіцієнтів можна було залишити ту хорду яка має найбільшу суму цих коефіцієнтів. В іншому випадку було б відкинуто всі хорди і подальші обчислення стали б неможливими.

```
firstChorde = chordesList.get(0);
```

3. Для кожної хорди перевіряємо її коефіцієнти із заданими пороговими.

```
tmpChordes список результуючих хорд
foreach(chordesList as chorde){
overlapCoef = chorde.getOverlapCoef();
if(overlapCoef >= OverlapThreshold  ) {
distCoef = chorde.getDistance()/maxDist;
if(distCoef >= this._distCoef) { tmpChordes.add(this._chordes.get(i)); }} }
```

4. Якщо tmpChordes пустий список, то необхідно додати хоча б одну хорду.

```
tmpChordes.add(firstChorde);
```

5. Кінець алгоритму.

В результаті при певних заданих порогових коефіцієнтах будуть відсіяні деякі хорди, при заданих коефіцієнтах перекриття і довжини (0,0) відповідно буде максимальна кількість хорд і максимальна точність, а при коефіцієнтах (1,1) точність буде мінімальна, тобто залишиться лише одна хорда, але результат буде відрізнятися від методу найдовшої хорди, оскільки в даному випадку враховується ще й перекриття.

Суттєвою перевагою даного підходу є можливість відкидання хорд, але основний недолік полягає в необхідності пошуку такої комбінації порогових коефіцієнтів при яких буде спостерігатись максимальна точність і мінімум хорд.

2.5 Алгоритм пошуку відстані Громова-Хаусдорфа

На сьогоднішній час немає ефективного алгоритму порівняння областей зображень в метриці Громова-Хаусдорфа. Сама метрика Громова-Хаусдорфа залишається слабо вивченою, а алгоритм обчислення вимагає надзвичайно великих затрат обчислювальних ресурсів. Тому, зібравши найкращі алгоритми, що описані вище, можна сформувати кінцевий алгоритм пошуку відстані Громова-Хаусдорфа.

Алгоритм складається з чотирьох основних етапів:

1) Підготовка вхідних даних (контурів).
2) Оголошення масиву відстаней Хаусдорфа.
3) Знаходження відстаней Хаусдорфа, при всіх можливих розміщеннях за допомогою одного з вище описаних алгоритмів.
4) Вибір мінімального значення з масиву відстаней.

Псевдокод алгоритму пошуку відстані Громова-Хаусдорфа:

```
Begin
  Знаходження контурів об’єктів. X, Y – набір точок багатокутників
  Проріджения контурів X, Y, використовуючи алгоритм Дугласа Рамера Пекера
  array distances
  Порівнюємо 2 об’єкти:
  Співставляємо центри мас
  for(i=0; i < count(etalonChordes);i++)
      for(j=0; j<count(testChordes);j++)
          співставляємо контури
          distances.add(distHausdorff(X,Y))
      end for
  end for
  DGromovHausdorff = min(distances)
end begin
```

Даний алгоритм можна легко розпаралелити на рівні циклів, тобто кожна ітерація циклу for може виконуватися на окремому ядрі, це дасть змогу пришвидшити час знаходження відстані якщо використовувати багатоядерну чи багатопроцесорну апаратну платформу.

2.6 Висновки до розділу 2

У розділі проаналізована відстань Громова-Хаусдорфа. Проаналізовані алгоритми виділення контуру зображення. Розроблені алгоритми накладання областей зображень.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Вимоги до програмного та апаратного забезпечення

В реалізованому програмному продукті повинні враховуватися функціональні та нефункціональні вимоги, тобто вимоги сумісності програмного і апаратного забезпечення, виконання покладених завдань коректно і в повному обсязі.

Функціональні вимоги:

- кінцевий програмний продукт повинен мати можливість приймати на вхід зображення різних форматів, такі як png, jpg, jpeg, bmp;
- в програмному продукті повинні бути реалізовані методи знаходження контурів областей зображень;
- проріджения контурів областей зображень;
- перетворення неопуклого контуру в опуклий;
- пошук найкращого взаємного розміщення контурів за допомогою декількох алгоритмів;
- виконання ізометричних перетворень над контуром таких як поворот на кут та зміщення;
- визначення метричних відстаней між контурами областей.

Нефункціональні вимоги:

- можливість використання програмного продукту як модуль до інших систем, зокрема графічного редактора ImageJ.
- кросплатформність;
- програма не повинна завершуватись аварійно;
- програма не повинна призводити до зависання операційної системи;
- повне і безпомилкове виконання заявлених функцій.

Мінімальні вимоги до середовища виконання ПЗ:

- ОС: Windows (XP, Server 2003), Linux-подібні системи з підтримкою JAVA (Linux 5.5+, Red Hat 5.5+, Suse Server 10.x+, Ubuntu 10.04), Mac OS X 10.7.3.
- RAM 512 MB.
- Вільний простір на HDD: 200 MB.
- Процесор 800 MHz Intel Pentium III або 1000 MHz AMD Athlon.

Рекомендовані вимоги до середовища виконання ПЗ:

- ОС: Windows (7, Server 2008), Linux (Linux 5.5+, Red Hat 5.5+, Suse Server 10.x+, Ubuntu 10.04 і вище), Mac OS (Mac OS X 10.7.3 (Lion) і пізніші версії);
- RAM 1024 MB;
- Вільний простір на HDD: 400 MB;
- Процесор Intel Pentium 4 Processor 2.80 GHz або новіший.

3.2 Розробка архітектури системи

Програмний продукт базується на використанні бібліотеки для роботи з графікою – OpenCV 3.0.0. В проекті буде реалізовано більшість методів саме з використанням OpenCV [36]. OpenCV (англ. Open Source Computer Vision Library, бібліотека комп'ютерного зору з відкритим вихідним кодом) - бібліотека алгоритмів комп'ютерного зору, обробки зображень та чисельних алгоритмів загального призначення з відкритим кодом. Реалізована на C / C ++, також розробляється для Python, Java, Ruby, Matlab, Lua та інших мов. Може вільно використовуватися в академічних та комерційних цілях - поширюється в умовах ліцензії BSD.

Реалізація будь-якої складної системи починається з проектування класів з допомогою UML, що зображена в додатку А.

Клас PolySimplifier призначений для проріджения контурів областей (таблиця 3.1), містить у собі два статичних методи спрощення – кусково-лінійної апроксимації контуру алгоритмами Дугласа-Рамера-Пекера і Visvalingam-Whyatt.

Таблиця 3.1 – Клас PolySimplifier

Методи	
Модифікатор і тип	Назва і опис
`public static List<Point>`	`reduceRDP(List<Point> contour, double epsilon)` Повертає набір точок прорідженого контуру. Проріджения реалізоване на алгоритмі Рамера-Дугласа-Пекера. `contour` – набір точок вхідного контуру; `epsilon` – максимальна відстань відхилення від перпендикуляра.
`Public static MatOfPoint`	`reduceRDP(MatOfPoint contour, double epsilon)` Повертає набір точок прорідженого контуру. Проріджения реалізоване на алгоритмі Рамера-Дугласа-Пекера. `contour` – набір точок вхідного контуру представлений об'єктом класу MatOfPoint; `epsilon` – максимальна відстань відхилення від перпендикуляра.
`Public static MatOfPoint`	`reduceVW(MatOfPoint mop, int numberToKeep)` Повертає набір точок прорідженого контуру. Проріджения реалізоване на алгоритмі Visvalingam-Whyatt. `contour` – набір точок вхідного контуру представлений об'єктом класу MatOfPoint; numberToKeep – кількість точок яка залишиться в кінцевому контурі

Клас DistHromivHausdorff відповідає за пошуку відстані Громова-Хаусдорфа. В класі реалізовано алгоритми, що були описані в підрозділі 2.4.

В таблиці 3.2 наведено методи класу DistHromivHausdorff, всі вони приймають на вхід два об'єкти контурів і повертають відстань Громова-Хаусдорфа.

Таблиця 3.2 – Клас DistHromivHausdorff

Методи	
Модифікатор і тип	Назва і опис
public static double	distBy360Rotation (Contour contour1, Contour contour2) Повертає відстань Громова-Хаусдорфа між двома контурами. Реалізований на алгоритмі повного перебору. contour1, contour2 – об'єкти класу Contour, що представляють області зображень;
public static double	distByLongestChord(Contour contour1, Contour contour2) Повертає відстань Громова-Хаусдорфа між двома контурами. Реалізований на алгоритмі пошуку кутаповороту за допомогою найдовшої хорди.
public static double	distByListOfChordes(Contour contour1, Contour contour2) Повертає відстань Громова-Хаусдорфа між двома контурами. Реалізований на алгоритмі пошуку кутів повороту з пороговою фільтрацією хорд
public static double	distByLongestChordAndPerpendiqular(Contour2 contour1, Contour2 contour2) Повертає відстань Громова-Хаусдорфа між двома контурами. Реалізований на алгоритмі пошуку кута повороту за допомогою 3 точок. contour1, contour2 – об'єкти класу Contour2, що представляють області зображень;
public static double	distBySecantMethod(Contour3 contour1, Contour3 contour2) Повертає відстань Громова-Хаусдорфа між двома контурами. Реалізований на алгоритмі пошуку кутів повороту за допомогою методу січних. contour1, contour2 – об'єкти класу Contour3.

Клас GeometryUtils містить допоміжні методи, в даному класі реалізовано методи пошуку контурів на зображенні, пошук геометричного центру мас, відстаней Евкліда і Хаусдорфа. Крім того реалізовано метод пошуку точки перетину прямої і контуру. В таблиці 3.3 описано методи класу GeometryUtils.

Таблиця 3.3 – Клас GeometryUtils

Методи	
Модифікатор і тип	Назва і опис
`public static Point`	`getCentoid(MatOfPoint contour)` Повертає точку - геометричний центр мас
`public static double`	`euclideDist(Point a,Point b)` – Повертає відстань Евкліда між двома точками a, b.
`public static double`	`distHausdorff(MatOfPoint contour1, MatOfPoint contour2)` – Повертає відстань Хаусдорфа між двома контурами. `contour1, contour2` – Контури об'єктів
`public static ArrayList <Double>`	`getStraightCoefficients(Point A, Point B)` – Повертає коефіцієнти рівняння прямої, що проходить через точки дві точки A і B.
`@Nullable public static Point`	`getIntersectionPointOfLines(Point A1, Point B1, Point A2, Point B2)` – Повертає точку перетину прямої, що проходить через відрізок A1, B1 і відрізка A2, B2. Якщо така точка не існує, то повертає `null`
`public static List<Contour>`	`getContours(Mat image, String msg,double distCoef,double overlapCoef)` – Повертає список контурів, представлених за допомогою класу Contour. `image` – вхадне зображення; `msg` – назва зображення; `distCoef` – поріг коефіцієнту відстані хорди; `overlapCoef` – поріг перекриття об'єкту хордою.

Крім вище описаних GeometryUtils містить аналогічні методи getContours2 і getContours3 , що відповідають за створення об'єктів класів Contour2 і Contour3.

Клас Contour містить набір властивостей і методів для реалізації можливості пошуку хорд і як наслідок кутів повороту (таблиця 3.4).

Таблиця 3.4 – Клас Contour

Властивості	
Модифікатор і тип	Назва і опис
`protected MatOfPoint`	`_contour` – об’єкт контуру.
`private MatOfPoint`	`_convexContour` – об’єкт опуклого контуру.
`private List<Chorde>`	`_chordes` – список об’єктів хорд.
`private Chorde`	`_maxChorde` – максимальна хорда
`protected Point`	`_centroid` – центр мас
`protected String`	`_name` – назва контуру
`protected Mat`	`_image` - вихідне зображення на якому було знайдено контур
`private double`	`_overlapCoefThreshold` – поріг коефіцієнту перекриття
`private double`	`_distCoefThreshold` – поріг коефіцієнту довжини хорди
Конструктори	
`public Contour`	`Contour(Mat image,MatOfPoint contour, String name,double distCoefThreshold, double overlapCoefThreshold)` Ініціалізує властивості класу, викликаються інші методи класу, що відповідають за пошук хорд. `image` – вхідне зображення; `contour` – контур;
`public Contour`	Contour(Contour contour) - Конструктор що використовується для клонування об’єкту
Методи	
`private void`	`setChordes()` – Виконує пошук хорд в контурів і додає їх до списку.
`protected double`	`getChordeIntersectionCoef(List<Point> line)` Повертає коефіцієнт перекриття. `line` – растеризована хорда.
`private void`	`getConvex()` – Знаходить опуклий контур і записує відповідну властивість в об’єкт.

Продовження таблиці 3.4

`public Contour`	`shift(double x, double y)` – Зсуває кожну точку контуру на певну відстань, задану двома параметрами. Повертає об'єкт Contour.
`private List<Point>`	`rasteredLine(Point p1, Point p2)` – Растеризує лінію що задана двома точками p1 і p2.
`public Contour`	`rotate(double angle)` – Повертає контур на кут. `angle` – кут в градусах.
`private void`	`sortChordes()` – Сортує і відкидає хорди використовуючи порогові коефіцієнти.

Крім того в даному класі реалізовані get і set («геттери» і «сеттери») методи для доступу до властивостей класу. За допомогою класу Contour реалізовано алгоритми повного перебору, найдовшої хорди і алгоритм порогової фільтрації хорд. Алгоритм пошуку кута повороту за трьома точками спирається на клас Contour2, котрий є дочірнім класом від Contour. В ньому використовується хорда, що представлена класом Chorde2. Також в класі Contour2 реалізовано пошук точки перетину контуру найдовшого перпендикуляру, що опускається в середину хорди. Опис класу Contour2 представлено в таблиці 3.5.

Таблиця 3.5 – Клас Contour2

Властивості	
Модифікатор і тип	Назва і опис
`private Chorde2`	`_maxChorde2` – Максимальна хорда і перпендикуляр
Конструктори	
`public Contour2`	`Contour2(Mat image,MatOfPoint contour, String name)` – Ініціалізує властивості класу, викликає допоміжні методи класу.
Методи	
`private void`	`setMaxChorde ()` – Знаходить максимальну хорду з перпендикуляром.

Продовження таблиці 3.5

private void	setPerpendicular (Chorde2 ch2) – Знаходить перпендикуляр і записує в об'єкт ch2.
public boolean	isMiddleHigher() – Використовується для виявлення напряму перепендикуляру. Повертає true якщо точка середини хорди має більші значення координат ніж у точки з якої опущено перпендикуляр.

Клас Contour3 є похідним від класу Contour2. В Contour3 реалізовано пошук січних хорд. Опис класу представлено в таблиці 3.6.

Таблиця 3.6 – Клас Contour3

Властивості	
Модифікатор і тип	Назва і опис
private List <Chorde3>	_chordes – Список січних хорд.
Конструктори	
public Contour3	Contour3(Mat image,MatOfPoint contour, String name) - Ініціалізує властивості класу, викликає допоміжні методи класу.
public Contour3	Contour3(Contour3 contour3) – Використовується для клонування об'єкту
Методи	
private void	setChordes(int rotationAngle) – Знаходить січні і додає до списку. rotationAngle – кут, що є кроком проведення січних.
private Point	getInitialBasePoint(List<Point> contourPoints) – Повертає вершину контуру, котра найбільш віддалена від центру мас. contourPoints – список точок контуру.
private Point	private Point getInitialPoint(List<Point> contourPoints) – Повертає вершину котра найближча до центру мас.
protected Point	_rotatePoint(Point base,Point p,double angle)- Повертає точку p відносно точки на кут angle.

Усі три класи контурів використовують різні класи хорд, оскільки форми представлення хорд в різних алгоритмів відрізняються. Клас Chorde представлений в таблиці 3.7, містить індекси точок хорд, а не самі точки. Такий підхід дозволяє відкинути необхідність в перерахунку координат точок хорд при повороті контуру.

Таблиця 3.7 – Клас Chorde

Властивості	
Модифікатор і тип	Назва і опис
`protected int`	`chordePoint1` – Індекс першої точки хорди.
`protected int`	`chordePoint2` – Індекс другої точки хорди.
`protected double`	`distance` – довжина хорди
`protected double`	`angle` – кут повороту хорди
`private double`	`overlapCoef` – коуфіцієнт перекриття
`private double`	`Coefficient` – добуток коефіцієнів перекриття і довжини
Конструктори	
`public Chorde`	`Chorde(int a,int b,double dist, double overlapCoef,Contour contour)` - Ініціалізує властивості класу, викликає допоміжні методи класу. `a,b` – індекси точок хорд; `dist` – довжина хорди, `overlapCoef` – коефіцієнт перекриття, `contour` – контур.
Методи	
`protected void`	`setEngle(List<Point> line)` – Знаходить кут нахилу хорди, що задана точками списку line.
`public MatOfPoint`	`public MatOfPoint getMOP()` – Повертає об’єкт MatOfPoint, що представляє собою лінію з двох точок.

Клас Chorde має похідний клас Chorde2, котрий суттєво не відрізняється від батьківського, в ньому змінений конструктор, оскільки в методі трьох точок немає коефіцієнтів перекриття. Також додано властивість maxPerpendicular типу MatOfPoint.

В Класі Chorde3, що є похідним від Chorde2, точки хорд більше не представлені індексами вершин контуру. Крім того реалізовано додаткові методи для повороту і зсуву хорд.

Для відлагоджування і відображення результатів, передбачені класи GUI (таблиця 3.8) та ImageConverter. В класі GUI зосереджені статичні методи які дозволяють виводити на дисплей різні елементи: контури, точки, відрізки і т.д. Клас ImageConverter призначений для перетворення зображення типу Mat, що використовує OpenCV в об'єкт типу BufferedImage, що використовується в java.

Таблиця 3.8 – Клас GUI

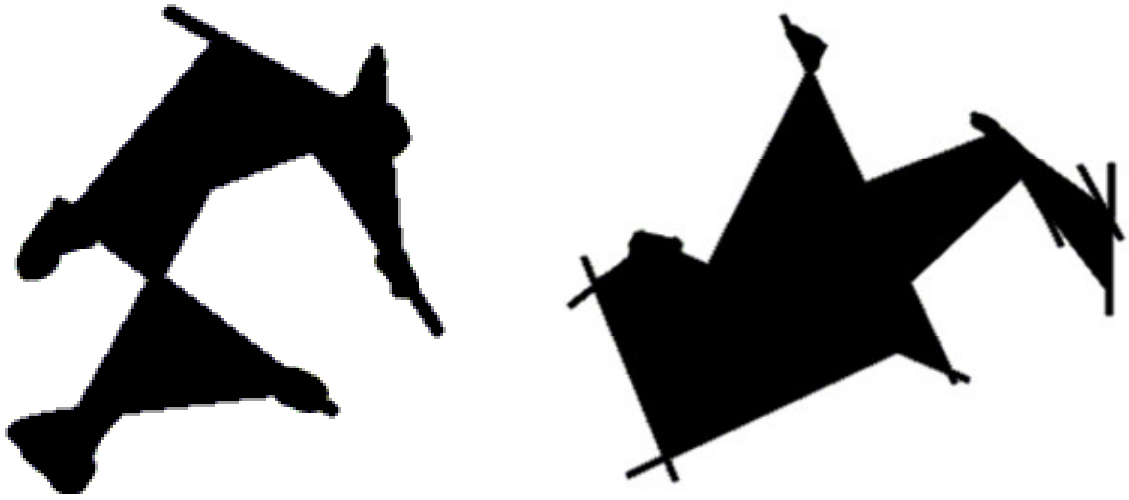
Методи	
Модифікатор і тип	Опис
`public static void`	`displayImage(Image img)` – виводить зображення на екран.
`Private static Image`	`scaleImage2ScreenSize(Image img)` – масштабує зображення до розмірів дисплею.
`public void`	`drawContour(MatOfPoint contour,int height, int width, String msg)` – Виводить контур на екран. `height` - висота вікна `width` - ширина
`public void`	`drawContours(List<MatOfPoint> contours,int height, int width)` – виводить список контурів на екран `contours` – список контурів

В додатку А представлено діаграму класів, де видно зв’язки між всіма класами. Робота програми починається в функції Main() класу Main. Спочатку відбувається створення об’єктів контурів шляхом виклику необхідної функції з класу GeometyUtils. Далі пара контурів передаються в відповідний метод класу DistHromivHausdorff, в результаті виконання якого повертається відстань Громова-Хаусдорфа.

3.3 Підбір порогових коефіцієнтів експериментальним способом

Алгоритм пошуку кутів повороту за допомогою порогової фільтрації задання порогових коефіцієнтів. В такому випадку виникає питання. Які саме вибрати пороги коефіцієнтів перекриття і довжини, щоб досягнути максимальної точності і мінімуму хорд? Дану проблему вирішено за допомогою експериментальних досліджень.

Було опрацьовано 441 комбінації порогових коефіцієнтів для однієї пари зображень, що представлені на рисунку 3.1.



а) б)

Рисунок 3.1 – Дослідні зображення: а), б) об’єкти, між яким відбувався пошук відстані Громова-Хусдорфа

На рисунку 3.2 показано залежність відстані Громова-Хаусдорфа від значень порогових коефіцієнтів для пари зображень.

В результаті аналізу даних було виявлено що при виборі порогу перекриття в межах від 0 до 0.55 і порогу довжини хорди від 0 до 0.8 спостерігається мінімальна відстань Громова-Хаусдорфа.

Зменшення кількості обчислень при максимальних значеннях порогів 0,55 і 0,8 спостерігається на рівні 79%, тобто це показник що відображає кількість відкинутих хорд в середньому для пари контурів.

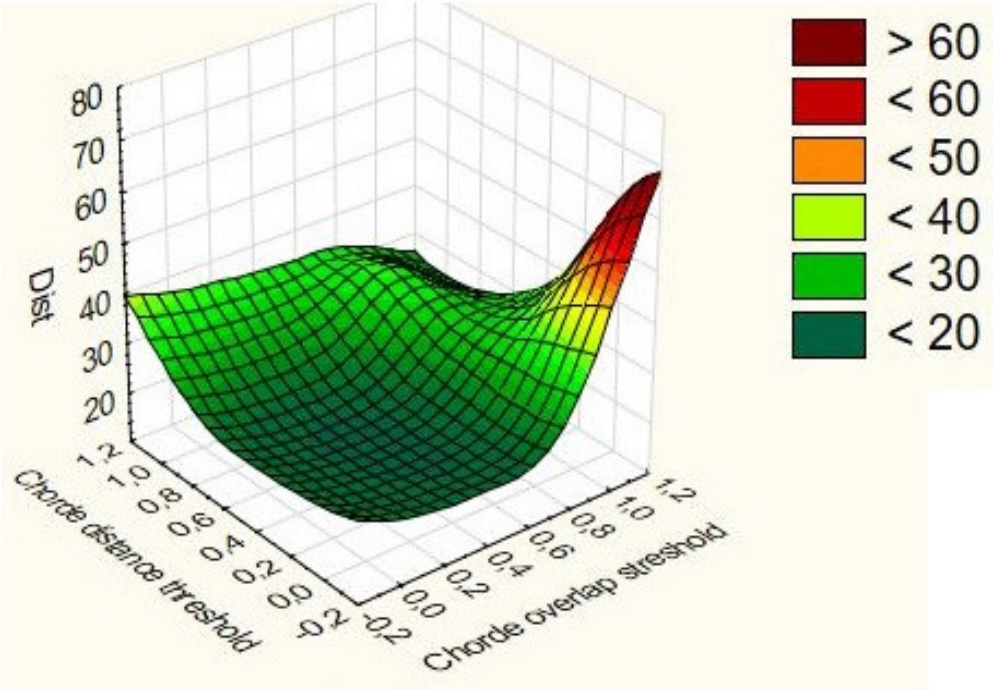


Рисунок 3.2 – Поверхнева діаграма відстані Громова-Хаусдорфа при різних порогових коефіцієнтах

Для знаходження оптимальних коефіцієнтів було виконано 32 досліди з різними парами зображень (додаток Б). В додатку В представлено таблицю з результатами досліджень. Проаналізувавши дані можна виявити середні значення коефіцієнтів, котрі будуть оптимальними.

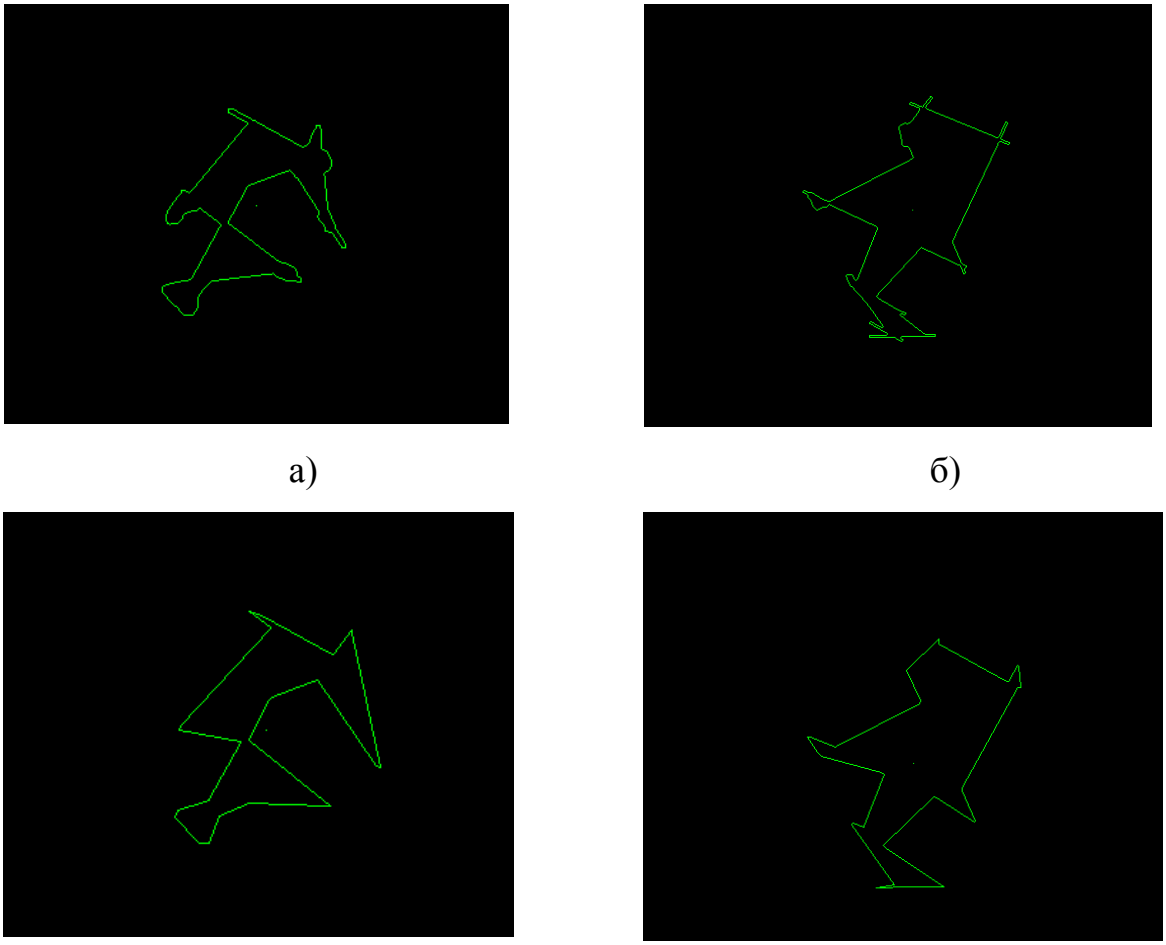
Отже оптимальними пороговими коефіцієнтами можна вважати 0,47 і 0,63 відповідно для коефіцієнтів перекриття і довжини, в такому випадку відкидається в середньому 62% хорд.

Всі подальші дослідження будуть виконуватися з оптимальними пороговими коефіцієнтами.

3.4 Тестування функцій системи

Протестуємо роботу деяких методів пошуку відстані Громова-Хаусдорфа.

Спробуємо знайти від стань Громова-Хаусдорфа для двох неопуклих контурів, базуючись на найдовшій хорді контуру. На рисунку 3.3 показано процес спрощення контуру за алгоритмом Рамера-Дугласа-Пекера.



а) б)

в) г)

Рисунок 3.3 – Процес спрощення контуру: а),б) вхідні контури; в),г) – відповідні спрощені контури

На рисунку 3.4 показано знайдені найдовші хорди на обох контурах.

а) б)

Рисунок 3.4 – Процес пошуку хорд: а),б) контури з найдовшими хордами

На рисунку 3.5 відображено накладання двох контурів при співпадінні їх центрів мас і кутів нахилу хорд.

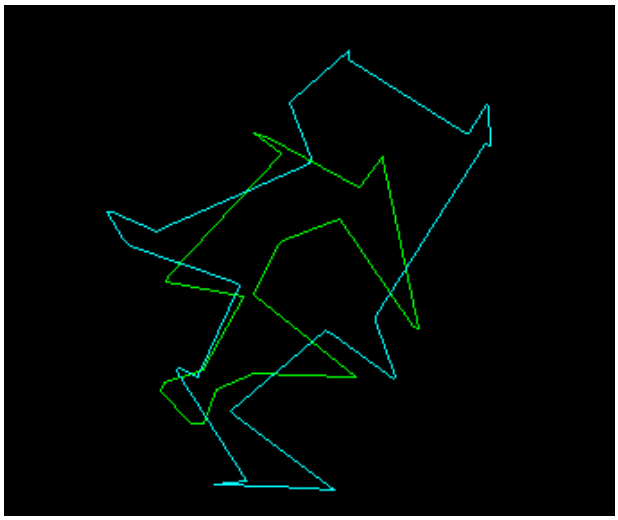


Рисунок 3.5 – Накладання двох контурів

В результаті було накладено два об’єкти і отримано відстань Громова-Хаусдорфа, яка складає ~40.0124 пікселі.

Тепер протестуємо пошук відстані за допомогою методу трьох точок. На рисунку 3.6 показано два тестові контури з проведеними хордами та опущеними перпендикулярами в середину хорди.

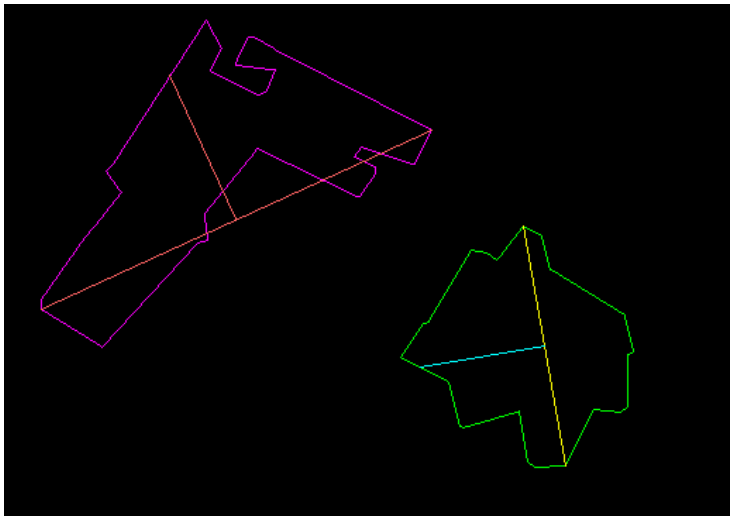


Рисунок 3.6 – Тестові контури та їх ключові точки

Наклавши зображені вище контури (рисунок 3.7) обчислюємо відстань Громова-Хаусдорфа, що складає 19,647 пікселі.

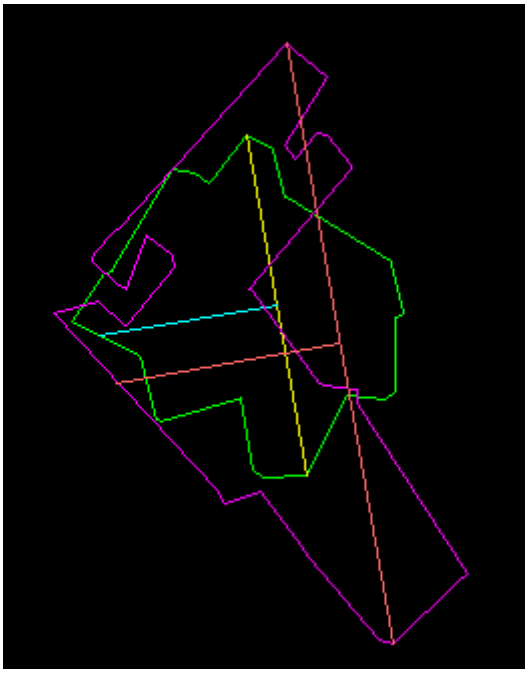


Рисунок 3.7 – Накладання контурів за допомогою трьох точок

На рисунку 3.8 показано знайдені січні хорди для двох довільних контурів. Згідно алгоритму контури будуть накладені по центрах мас а кути повороту буде знайдено за допомогою січних, зокрема їх нахилів. На рисунку 3.9 показано процес перебору січних і накладання контурів так, щоб січні були паралельні і їх

напрями співпадали. Під напрямом розуміється промінь від спільної точки січних до точки перетину з контуром.

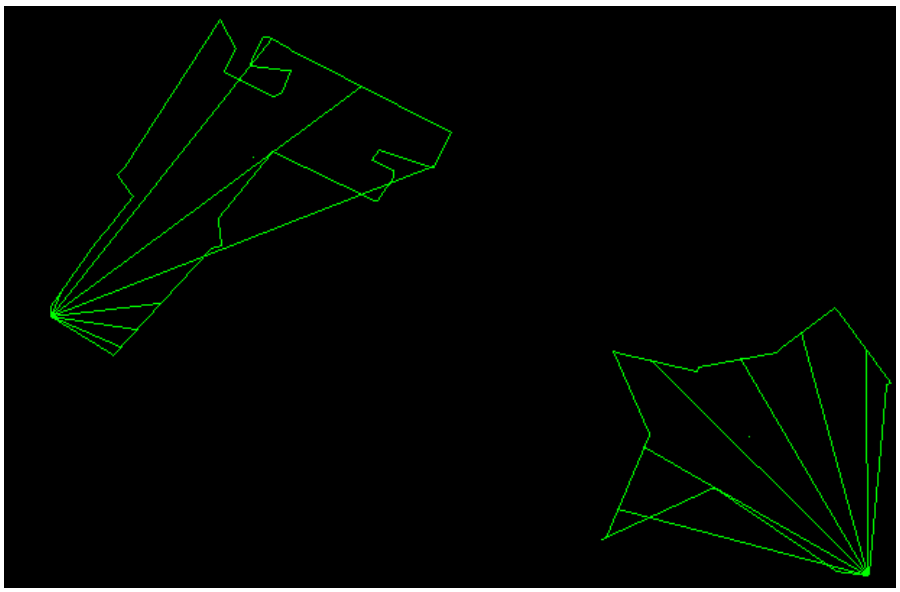


Рисунок 3.8 – Контури та їх січні хорди

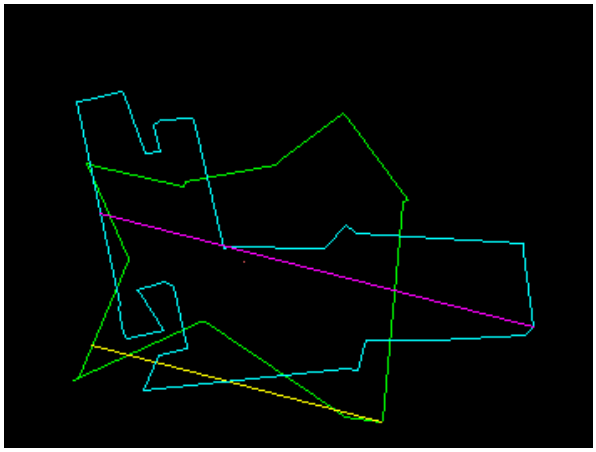


Рисунок 3.9 – Процес накладання контурів, одна з можливих комбінацій

Як видно з рисунку 3.8, кількість знайдених січних залежить лише від кута відсікання і форми об’єкту, чим більш видовжена форма тим менше в нього спостерігається січних.

3.5 Порівняння алгоритмів пошуку оптимального накладання

В процесі виконання кваліфікаційної роботи було розглянуто і реалізовано п’ять алгоритмів співставлення контурів [37, 38]. Щоб виявити якісні показники кожного з них було проведено досліди на 32 парах різних зображень, котрі представлено в додатку Б, результати дослідів показано в таблиці 3.9.

Дослідження проводились з метою виявлення швидкодії і точності кожного з методів. На рисунку 3.10 представлено графік відхилень відстані Громова-Хаусдорфа від результату отриманого повним перебором.

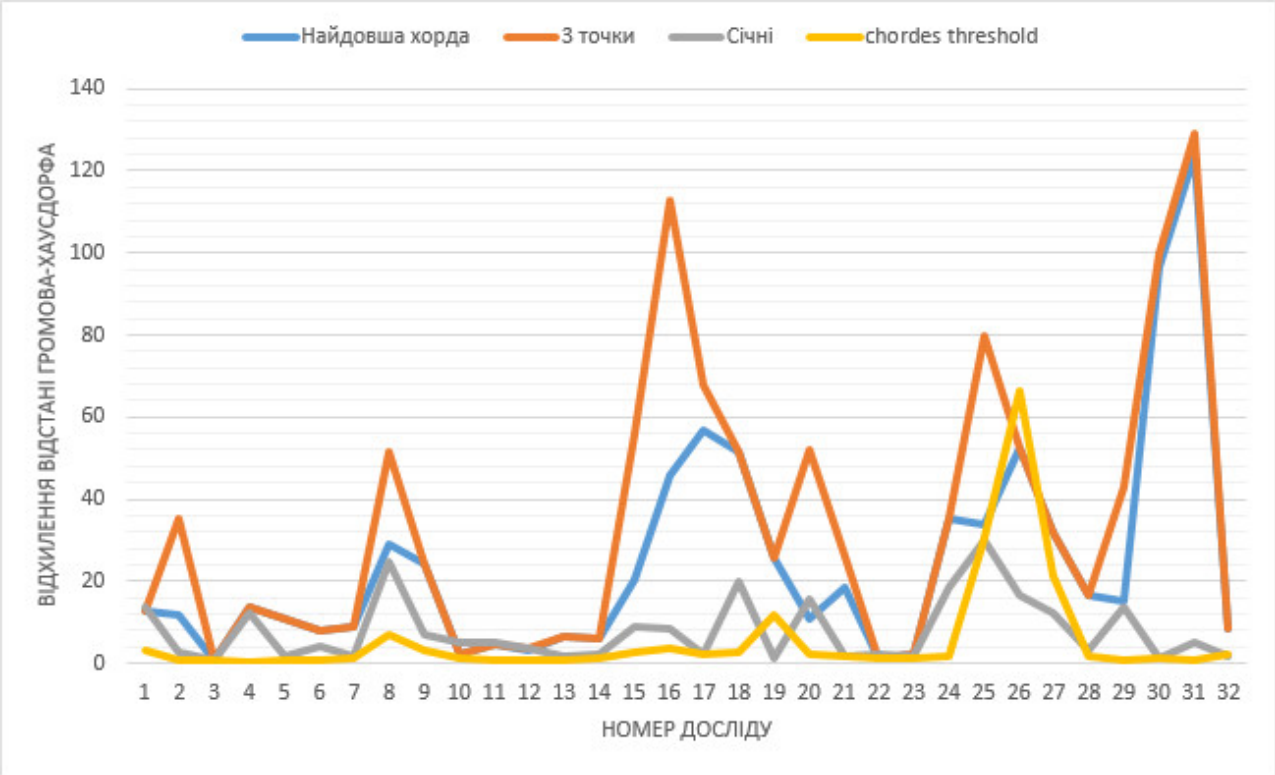


Рисунок 3.10 – Графік відхилень відстані Громова-Хаусдорфа від повного перебору

З діаграми видно, що метод найдовшої хорди і трьох точок в більшості випадків мають гірші результати в порівнянні з рештою алгоритмів. Середні відхилення від мінімального в обох алгоритмів складають 24,65696635 і 33,74386386 пікселі відповідно.

Таблиця 3.9 – Порівняння показників відхилень для різних алгоритмів

Номер досліду	Найдовша хорда	3 точки	Січні	Порогова фільтрація хорд
1	12,63278235	12,63278235	13,67573639	3,096870021
2	11,72923908	35,2076399	2,528559429	0,889284062
3	0,711079166	0,711079166	0,901459664	0,711079166
4	13,6468827	13,6468827	12,02775638	0,32455532
5	10,86577281	10,86577281	1,889882148	0,976156999
6	7,753359046	7,753359046	4,15119162	0,586724213
7	8,832975678	8,832975678	1,906112267	1,384709653
8	29,04410416	51,75689087	24,55470441	7,024594537
9	24,14555402	24,14555402	6,99530054	3,145331115
10	2,235384062	2,235384062	4,9544984	1,027756377
11	4,458885617	4,458885617	4,854673128	0,732809336
12	3,331979599	3,447045572	3,857298779	0,581098561
13	6,5916182	6,5916182	1,828501756	0,726922136
14	5,980811042	5,980811042	2,191230989	1,144296014
15	20,58442046	55,93895689	9,068116844	2,85532896
16	45,59644256	112,8812962	8,453624047	3,602325267
17	57,00694411	67,92767934	2	2,117242769
18	51,29007974	51,29007974	19,94092858	2,609684253
19	25,45565982	25,45565982	1,043457733	11,72741841
20	10,9546685	52,05412892	15,81093656	2,225400619
21	18,7039895	26,32110764	1,679169294	1,539036252
22	1,337581188	1,337581188	2,409462831	1,337581188
23	2,256031054	2,256031054	1,835927058	1,493752234
24	35,28523088	35,28523088	18,39789633	1,970460171
25	33,57693735	79,53665321	29,98125073	30,48903112
26	52,62281078	52,62281078	16,47433026	66,32794466
27	31,83784863	31,83784863	12,4222051	21,43074903
28	16,55045351	16,55045351	3,127011085	1,640723978
29	15,13212016	42,71767546	13,56648418	0,607954159
30	96,58342714	100,0159418	1,397140589	1,33176852
31	123,7417295	128,9617066	5,288746303	0,949532082
32	8,546120631	8,546120631	1,851144907	2,04866112
Середнє відхилення	24,65696635	33,74386386	7,845773073	5,583024447

Алгоритм порогової фільтрації хорд в переважні більшості випадків дає найбільшу точність, за винятком дослідів 25-27, де зустрічалися порожнисті об’єкти. Середнє відхилення складає 5,583024447 пікселя.

Алгоритм січних дає більш стабільний результат, проте в більшості випадків меншу точність в порівнянні з алгоритмом порогової фільтрації хорд. Середнє відхилення 7,845773073.

Тепер проаналізуємо часові показники для всіх алгоритмів (рисунок 3.11).

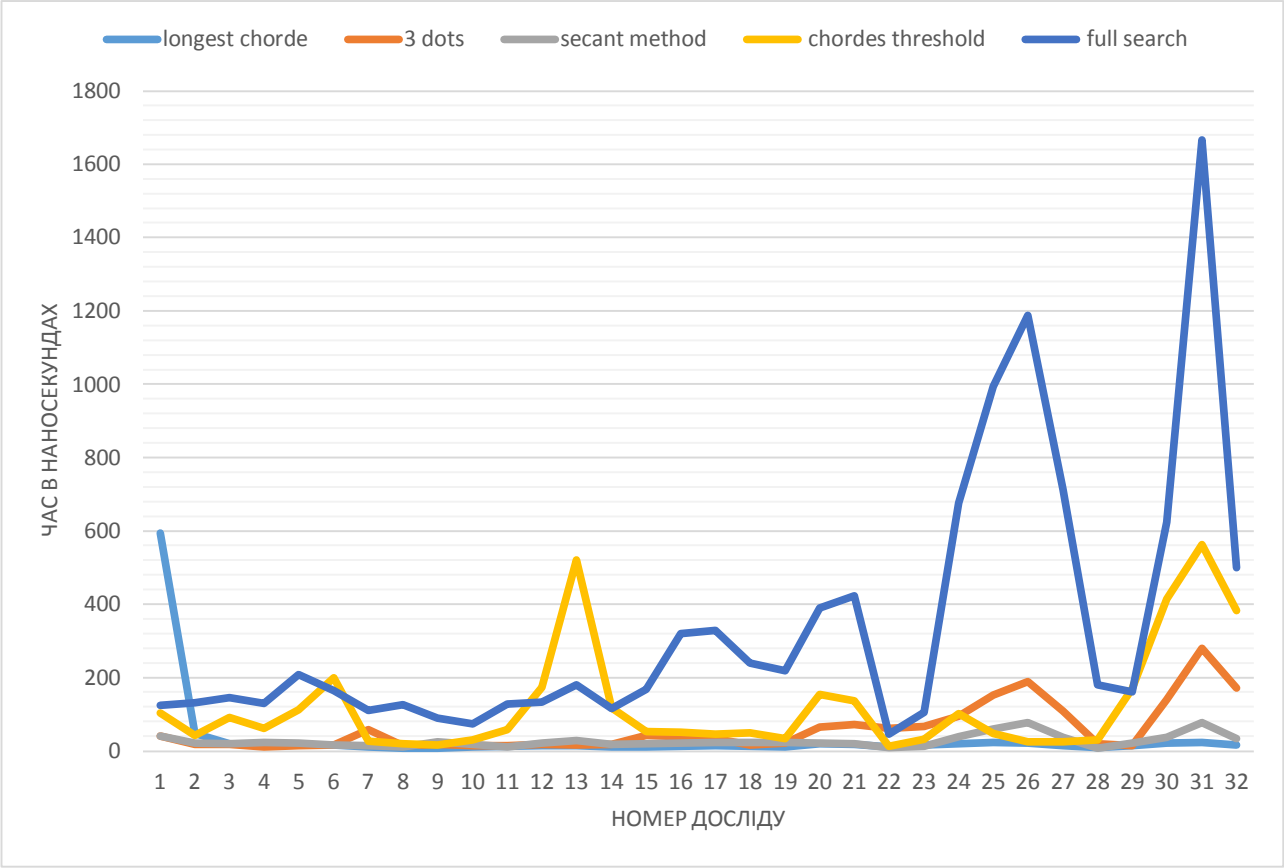


Рисунок 3.11 – Часові показники алгоритмів

Проаналізувавши діаграму, зображену на рисунку 3.11, можна стверджувати, що найоптимальнішим є алгоритм параметричної фільтрації хорд, оскільки він має більш ніж в два рази менший середній час роботи, а саме 121,5886682 наносекунд, в порівнянні з повним перебором 337,2666429, крім того в нього не спостерігаються великі піки по часу і точність його результатів найбільш наближена до точності при повному переборі.

3.6 Порівняння результатів алгоритмів сегментації з еталоном

Для порівняння результатів сегментації використаємо алгоритм порогової фільтрації хорд, оскільки вище було доведено його ефективність. Проте даний алгоритм працює тільки пари контурів, а не зображень цілком.

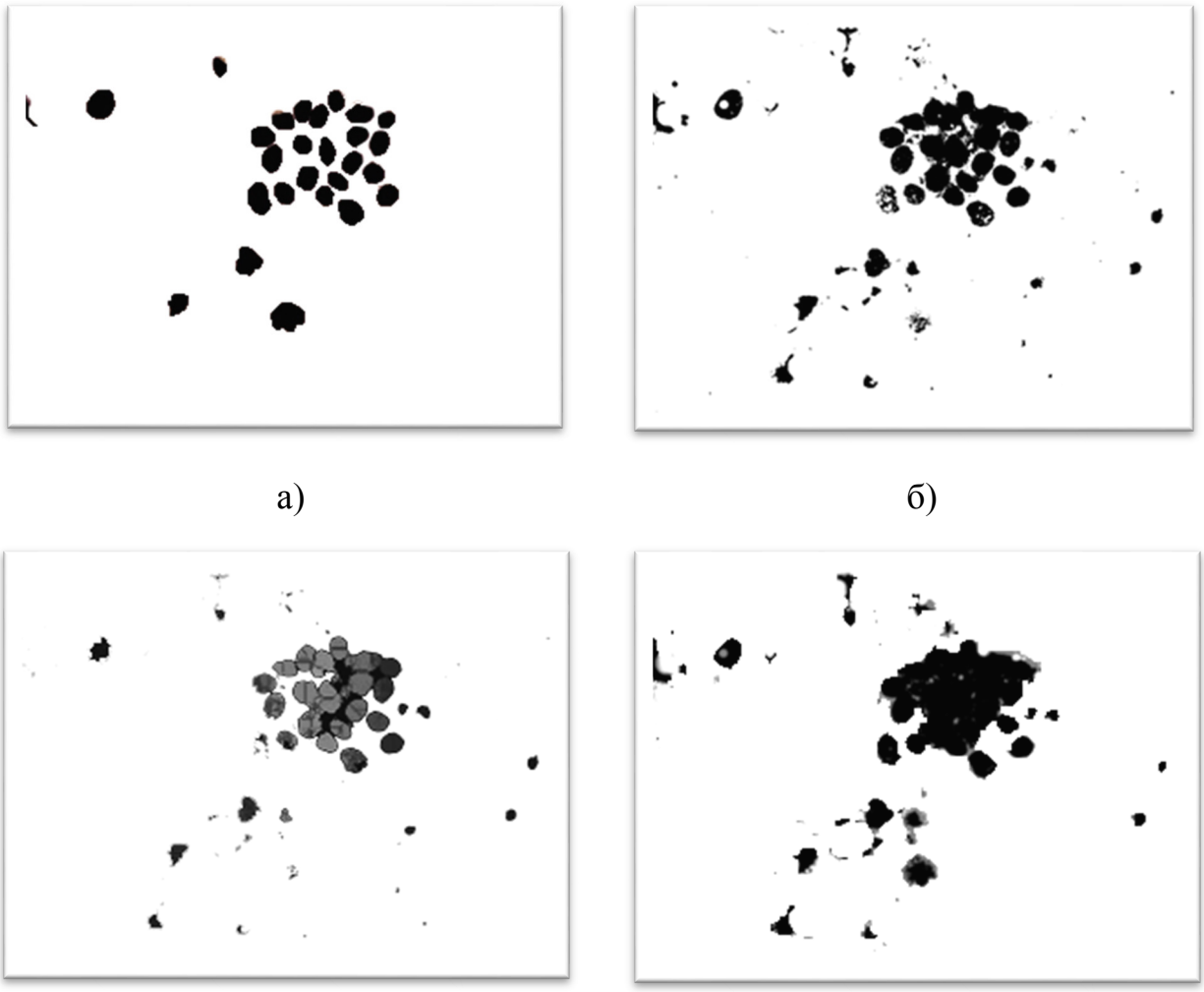
Вирішити дану проблему пропонується за допомогою наступного методу:

```
public static void compareImages(Mat im1 , Mat im2){
     List<Contour> img1Cntrs =
     GeometryUtils.getContours(im1.clone(), "Image1", 0.63, 0.47);
     List<Contour> img2Cntrs =
     GeometryUtils.getContours(im2.clone(),   "Image2", 0.63,
     0.47);
     double dist = 0;
     for(int i = 0; i < img1Cntrs.size(); i++) {
         double minDist = Double.MAX_VALUE;
         for(int j = 0; j < img2Cntrs.size(); j++) {
          double currentDist =
          DistHromovHausdorff.distByListOfChordes
          (img1Cntrs.get(i), img2Cntrs.get(j), im1, im2, false);
           if(currentDist < minDist)
               minDist = currentDist;
         }
         if(minDist > dist) dist = minDist;
     } System.out.println("Distance = "+ dist ); }
```

Вище представлений метод приймає на вхід два зображення і виводить в консоль відстань між ними. Процес пошуку відстані включає перебір всіх комбінацій контурів першого зображення з контурами другого.

У внутрішньому циклі шукається мінімальна відстань між і-тим контуром і всіма контурами другого зображення. А у зовнішньому циклі відбувається пошук максимального значення відстані серед мінімальних.

На рисунку 3.12 представлено гістологічні зображення на основі яких проводився дослід. В результаті виконання досліду виявлено що відстань між еталонним зображенням і сегментованим методом кластеризації дає найгірший результат ~27.785, метод водоподілу дає відстань ~21.024, а найкраще проявив себе метод порогової сегментації ~9.472.



а) б)

в) г)

Рисунок 3.12 – Тестові зображення: а) сегментоване експертом; б) порогова сегментація; в) сегментація водоподілом; г) кластеризація

Як видно з рисунку 3.12, методи сегментації за своєю якістю дають результати, що далекі від експертного, тому для того щоб покращити результат, необхідно застосовувати попередню обробку зображень перед сегментацією, а після сегментації додатково обробити зображення відкинувши надто дрібні області та розмежувати об’єкти що зливаються.

3.7 Висновки до розділу 3

Розроблено архітектуру програмної системи, протестовано алгоритми накладання областей зображень, проведено порівняння про сегментованих гістологічних зображень.

ВИСНОВКИ

1. Проаналізовано особливості біомедичних зображень, досліджено основні проблеми повязані з сегментацією цитологічних зображень і проведено порівняльну характеристику зображень.

2. Проаналізовано алгоритми сегментації зображень, алгоритми та програмні засоби для порівняння областей зображення, розглянуто основні критерії і методи порівнняння областей, в якості об’єктивного критерію було обрано метрику Громова-Хаусдорфа.

3. Проаналізовано алгоритми знаходження контурів областей зображень, виявлено, що найкращий результат кусковолінійної апроксимації спостерігається при використанні алгоритму Дугласа-Рамера-Пекера.

4. Розглянуто алгоритми пошуку оптимального накладання контурів, виявлено їх переваги та недоліки.

5. В результаті експериментальних досліджень було виявлено, що для зменшення обчислювальної складності алгоритму пошуку відстані Громова-Хаусдорфа можна використовувати хорди контурів, які несуть інформацію про кут повороту об’єкту.

6. Експериментальним способом було встановлено, що метод порогової фільтрації хорд є найоптимальнішим способом для пошуку кутів повороту.

7. В результаті реалізації програмної системи було проведено дослідження методів накладання об’єктів, тестування ключових функцій системи, та виявлено переваги і недоліки різних методів обчислення максимального перекриття двох контурів. Виконано порівняння алгоритмів сегментації з еталонним зображенням.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Полагнюк 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., Полагнюк 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.

3. 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.

4. Абакумов В.Г. Використання методів сегментації у процедурах обробки біомедичних зображень. / В.Г. Абакумов, Ю.В. Муха // Электроника и связь. – 2009. - №1. – с. 34-37.

5. Абламейко С.В. Обработка оптических изображений клеточных структур в медицине. / С.В. Абламейко, А.М. Недзьведь – Мн.: ОИПИ НАН Беларуси, 2005. – 155с.

6. Автандилов Г.Г. Медицинская морфометрия. / Г.Г. Автандилов – М.: Медицина, 1990. – 384с.

7. Хем А. Гистология: В 5 т. / А. Хем, Д. Кормак / – М.: Мир, 1982.

8. Самойленко Д.Е. Структурная сегментация изображений / Д.Е. Самойленко // Искусств. интеллект. - 2004. - № 4. - С. 521-528.

9. 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.

10. Березький О.М. Segmentation algorithms of biomedical images: development and quantitative evaluation / О.М.Березький, Ю.М.Батько , Г.М. Мельник, С.О.Вербовий // Штучний інтелект, Київ, 2016. - №3 (73). - С. 104-116.

11. Березький О. М. Текстурна сегментація біомедичних зображень на основі просторових моментів / О. М. Березький, Г.М.Мельник, Ю.М. Батько // Матеріали 4-ї Міжнародної науково-технічної конференції "Комп'ютерні науки та інформаційні технології 2009", 15-17 жовтня, 2009, м. Львів. – Львів: ПП «Вежа і Ко», 2009 – С.42-45.

12. 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.

13. 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.

14. Березький О. М. Статистичне оброблення цитологічних зображень / О. М. Березький, К. М. Березька, С. Ю. Попіна // Вісник Хмельницького національного університету: зб. наук.-техн. праць. – Сер.: Технічні науки. – 2012. – № 5. – С. 161–164.

15. Красильников Н.Н. Цифровая обработка 2D- и 3D-изображений: учеб. Пособие / Н.Н. Красильников. — СПб.: БХВ-Петербург, 2011. — 608 с.

16. Саху П.К. Обзор по пороговым методам : кибернетический сборник / П.К. Саху, С. Солтани, Ф.Л. Вонг // Под ред. О.Б. Лупанова, О.М. Касим-Заде. Вып. 27. – М.: Мир, 1990. – С. 139-173.

17. Meyer F. Morphological segmentation / F. Meyer, S. Beucher // Journal of Visual Communication and Image Representation. – 1990. – V. 1. – № 1. – P. 21-46.

18. Претт У. Цифровая обработка изображений: В 2 кн. / У. Претт. - М.: Мир, 1982. - 790с.

19. Сегментация (обработка изображений) это. – режим доступу: http://dic.academic.ru/dic.nsf/ruwiki/1623093. - Заголовок з екрану.

20. Walker R.F. Classification of cervical cell nuclei using morphological segmentation and textural feature extraction / R.F. Walker, P. Jackway, B. Lovell et al. // Second Australian and New Zealand conference on Intelligent information systems. – Brisbane, 1994. – P. 183-189.

21. Недзьведь А.М. Сегментация изображений волокон и сосудов при большом увеличении/ А.М. Недзьведь, С.В. Абламейко // Цифровая обработка изображений. Вып. 4. – Мн.: Ин-т техн. кибернетики НАН Беларуси, 1999. – С. 167-176.

22. Zhang H. Image segmentation evaluation: A survey of unsupervised methods. / H. Zhang J.E.Fritts, S.A.Goldman. // Computer Vision and Image Understanding, Vol.110, Issue 2, 2008, pp.260-280.

23. Деза Е.И. Энциклопедический словарь расстояний / Е.И. Деза, М.М Деза. – М.: Мир, 2008, - 444с.

24. Гонсалес Р. Цифровая обработка изображений. / Р. Гонсалез, Р. Вудс - М.: Техносфера, 2005. - 1072 с.

25. Портал искусственного интеллекта. [електронний ресурс] : Автоматическая классификация. Мера расстояния. – режим доступу: http://www.aiportal.ru/articles/autoclassification/measure-distance.html. - Заголовок з екрану.

26. ImageJ. – режим доступу: https://imagej.nih.gov/ij/.

27. Левашкина А. О. Исследование супервизорных критериев оценки качества сегментации изображений / А. О. Левашкина, С. В. Поршнев // Известия Томского политехнического университета. – 2008. – Т. 313, №5. – С. 28-33.

28. Zhang Y.J. Advances in image and video segmentation./ Y.J. Zhang – IBM Press, 2006. – 473 p.

29. Павлидис Т. Алгоритмы машинной графики и обработки изображений: Пер. с анг. / Т. Павлидис – М.:Радио и связь, 1986. - 400 с.

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. 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.

32. 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.

33. Методи, алгоритми та 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.

34. Алгоритм Рамера — Дугласа — Пекера. – режим доступу: https://ru.wikipedia.org/wiki/Алгоритм_Рамера_—_Дугласа_—_Пекера.

35. Роджерс Д. Алгоритмические основы машинной графики./ Д. Роджерс — М.: Мир, 1989. - С. 54-63.

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

37. 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.

38. 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.

39. Dzung L. Pham, Chenyang Xu, and Jerry L. Prince (2000): «Current Methods in Medical Image Segmentation», Annual Review of Biomedical Engineering, volume 2, pp 315 – 337.

40. Журавлев Ю.И. Распознавание образов и распознавание изображений: Распознавание, классификация, прогноз. Математические методы и их приминение. / Ю.И. Журавлев, Гуревич И.Б. // Ежегодник. – М.: Наука, 1989. – Вып.2. – 302 с.

41. Дуда Р. Распознавание образов и анализ сцен: Пер. с анг. / Р. Дуда, Харт П. – М.: Мир, 1976. – 511 с.

42. Фисенко В.Т. Компьютерная обработка и распознавание изображений: учеб. пособие. / В.Т. Фисенко, Т.Ю.Фисенко. – СПб: СПбГУ ИТМО, 2008. – 192 с.

43. Memoli F. On the use of gromov-hausdorff distances for shape comparison. In Proceedings of PBG. / F. Memoli - Prague, Czech Republic, 2007.

44. Fu K.S. A survey on image segmentation./ K.S. Fu, J.K.Mui. // Pattern Recognition, Vol.13, No.1, 1981, pp.3-16.

45. Виро О. Я. Элементарная топология. / О. Я. Виро, О. А. Иванов, Н. Ю. Нецветаев, В. М. Харламов // М.: МЦНМО, 2012. - 358с.

46. Сидоров Д.В. Оценка качества изображений с использованием вейвлетов. / Д.В. Сидоров, А.Н. Осокин, Н.Г. Марков // Известия Томского политехнического университета [Известия ТПУ]. — 2009. — Т. 315, № 5: Управление, вычислительная техника и информатика. — С. 104-107.

47. Ласло М. Вычислительная геометрия и компьютерная графика на C++: Пер. с англ. / М. Ласло – М.: БИНОМ, 1997. — 304 с.

48. Shnayderman A. An SVD-Based Gray-Scale Image Quality Measure for Local and Global Assessment. / A. Shnayderman, A. Gusev, A.M. Eskicioglu. // IEEE Transactionson image processing. – February 2006. – Vol. 15, №. 2.

49. Bankman I.N. Handbook of Medical Imaging. A volume in Biomedical Engineering. / I.N. Bankman – Academic Press. 2000. p 895.

50. Rieffel M.A. Gromov-Hausdorff distance for quantum metric spaces./ M.A. Rieffel // arXiv:math.OA/0011063 v2, 2001.

51. Gromov M. Metric structures for Riemannian and nonRiemannian spaces, volume 152 of Progress in Mathematics. / M. Gromov. - Birkhauser Boston Inc., Boston, MA, 1999.

52. Pattern recognition. - available at: URL: https://en.wikipedia.org/wiki/Pattern_recognition.

53. Pishchulin L. Matching algorithms for image recognition./ L. Pishchulin - RWTH Aachen University Aachen. 2010.