Міністерство освіти і науки України
Західноукраїнський національний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерної інженерії

Євтушок Михайло Анатолійович

«Алгоритми пошуку біомедичних зображень на основі колірних і текстурних ознак / Biomedical image searching algorithms based on colour and texture features»

Студент групи КІм – 21
Євтушок Михайло Анатолійович
_____

Науковий керівник
к.т.н., доцент Мельник Г.М.
_____

Тернопіль – 2020

РЕЗЮМЕ

Кваліфікаційна робота на тему «Алгоритми пошуку біомедичних зображень на основі колірних і текстурних ознак» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «магістр» написана обсягом 68 сторінок і містить 25 ілюстрацій, 10 таблиць, 2 додатки та 58 джерел за переліком посилань.

Метою кваліфікаційної роботи є розроблення алгоритмів пошуку біомедичних зображень на основі комбінування текстурних і колірних ознак.

Методи дослідження: методи та алгоритми теорії графів, методи комп'ютерного зору, методи штучного інтелекту.

Результати дослідження. Розроблено алгоритм пошуку зображень по вмісту на основі обчислення оцінки подібності зображень запиту. Оцінка подібності є лінійною комбінацією колірної метрики і функцій відстані текстурних ознак з певними вагами. Здійснено програмну реалізацію компонентів системи пошуку по вмісту на мові Python.

Орієнтовні напрямки розвитку досліджень. Перпективним є продовження робіт по аналізу метрик на основі форми областей.

КЛЮЧОВІ СЛОВА: КОЛІР, HSV, ДЕСКРИПТОР, ТЕКСТУРА, ПОШУК ПО ВМІСТУ, БІОМЕДИЧНІ ЗОБРАЖЕННЯ.

RESUME

Qualification work on "Algorithms for searching biomedical images based on color and texture features" in the specialty 123 "Computer Engineering" educational degree "Master" is written in 68 pages and contains 25 illustrations, 10 tables, 2 appendices and 58 sources by reference. .

The purpose of the qualification work is to develop algorithms for searching biomedical images based on a combination of textural and color features.

Research methods: methods and algorithms of graph theory, methods of computer vision, methods of artificial intelligence.

Research results. An algorithm for searching images by content based on the calculation of the evaluation of the similarity of the query images has been developed. Estimation of similarity is a linear combination of color metrics and functions of distance of textural signs with certain weights. The software implementation of the components of the content search system in the Python language has been implemented.

Approximate directions of research development. It is promising to continue work on the analysis of metrics based on the form of areas.

KEY WORDS: COLOR, HSV, DESCRIPTOR, TEXTURE, CONTENT SEARCH, BIOMEDICAL IMAGES.

ЗМІСТ

ВСТУП

Актуальність теми. Різні алгоритми пошуку схожих зображень широко застосовуються в засобах масової інформації, пошукових системах, соціальних мережах, букінгових системах і тому подібне. За допомогою пошуку схожих зображень ефективно вирішуються завдання класифікації зображень, видалення дублікатів, відстеження порушень авторських прав.

Для порівняння зображень різних розмірів можна використовувати приведення розмірів до деякого еталонного вирішенню, наприклад, 16x16 пікселів. За допомогою порівняння одержуваних еталонів можна швидко знайти клас ймовірно схожих зображень, серед яких можна провести більш детальний аналіз, наприклад, порівняти їх за стандартами інших розмірів.

Порівняння еталонів можна проводити за допомогою простих процедур, таких як обчислення норми різниці матриць, що складаються з закодованих числами квітів пікселів зображень. Однак подібні підходи потребують значних витрат часу і з цієї причини не пристосовані для вирішення завдань класифікації зображень.

Ефективним підходом до порівняння еталонних зображень є використання представлення зображень у вигляді невпорядкованих множин деяких заздалегідь виділених елементів. Наприклад, можна розбити всі можливі кольори на кілька груп, і побудувати по заданому зображенню безліч всіх груп кольорів його пікселів. Для оцінки близькості множинних уявлень зображень застосовуються різні коефіцієнти подібності множин, наприклад, коефіцієнт Жаккар. Даний підхід можна використовувати також для порівняння схожості інших об'єктів, що допускають представлення у вигляді множин, таких як текстові документи, відеозаписи, популяції організмів, хімічні речовини і тому подібне.

Обчислення коефіцієнтів подібності зображень безпосередньо є досить трудомістким завданням. Для швидкого наближеного обчислення коефіцієнта Жаккар можна використовувати метод MinHash. При застосуванні цього методу, відповідне зображення, описується за допомогою вектора, кожна компонента якого є мінімальним значенням хеш-функції серед її значень для всіх елементів множини. Кожній компоненті вектора відповідає своя хеш-функція. Вектори, одержувані за допомогою методу MinHash, як правило, мають велику розмірність.

Мета і завдання дослідження**.** Метою роботи є розроблення алгоритмів пошуку біомедичних зображень на основі комбінування текстурних і колірних ознак.

Об’єкт дослідження процес аналізу біомедичних зображень.

Предмет дослідження – методи і алгоритми обчислення текстурних ознак зображень та метрики для їх порівняння.

Для досягнення поставленої мети необхідно розв’язати такі задачі:

- проаналізувати характеристики біомедичних зображень та їх ознаки;
- розробити алгоритм обчислення колірних і текстурних дескрипторів;
- розробити алгоритм обчислення відстані між зображеннями для їх ранжування при пошуку;
- здійснити програмну реалізацію компонентів системи пошуку по вмісту;
- провести експериментальне дослідження розроблених алгоритмів.

Методи досліджень базуються на використанні методів гістограмного аналізу зображень, методів комп’ютерного зору, положень теорії алгоритмів і аналітичної геометрії.

Наукова новизна одержаних результатів. Розроблено алгоритм пошуку біомедичних зображень по вмісту на основі комбінування колірних текстурних ознак мікрооб’єктів зображень що дозволило підвищити оцінку повноти пошуку.

Практичне значення отриманих результатів. Здійснено програмну реалізацію компонентів системи пошуку по вмісту на мові Python. Методи порівняння й пошуку зображень по змісту знаходять широке застосування у медицині, наприклад, в пошуку пацієнта зі схожим діагнозом по біомедичному зображенню.

Публікації та апробація випускної кваліфікаційної роботи. Отримані результати апробовані в межах ІІІ науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі» Тернопільського національного економічного університету та опубліковано дві тези доповіді по темі роботи [1,2].

Кваліфікаційна робота складається із трьох розділів, висновків, списку використаної літератури та додатків. У першому розділі розглянуто структуру та функції програмного забезпечення аналізу біомедичних зображень. Підсумовано переваги і недоліки систем автоматизованої мікроскопії. Проаналізовано структуру і функції систем пошук зображень по вмісту та міри оцінки якості пошуку.

У другому розділі розроблено алгоритми обчислення дескрипторів кольору та текстури. Розроблено алгоритми пошуку по вмісту шляхом лінійного комбінування дескрипторів проіндексованих зображень.

У третьому розділі здійснено програмну реалізацію компонентів системи пошуку зображень по вмісту. Проведено експериментальне дослідження набору алгоритмів обчислення текстурних ознак та оцінку ефективності розроблених алгоритмів пошуку зображень в базі зображень.

1 ПОШУК БІОМЕДИЧНИХ ЗОБРАЖЕНЬ ПО ВМІСТУ НА ОСНОВІ ОЗНАК

1.1 Біомедичні зображення.

Біомедичне зображення – це структурно-функціональний образ органів людини і тварин, призначений для діагностики захворювань і вивчення анатомофізіологічної картини організму. Зображення отримані використанням технічних засобів візуалізації у медицині та біології. Завдяки візуалізації різних процесів, які відбуваються в живих організмах, вдається вивчати механізми функціонування клітин, тканин та органів людини і тварин [1,2].

Біомедична інженерія (англ. biomedical engineering) — галузь науки і техніки, яка поєднує інженерно-технічні та медико-біологічні знання, засоби і методи для створення, вдосконалення і дослідження природних і штучних біологічних об'єктів, техніки, матеріалів і виробів медичного призначення, технологій і технічних систем діагностики, лікування, реабілітації і профілактики захворювань людини, а також програмного забезпечення та інформаційних технологій для вирішення прикладних і фундаментальних проблем біології і медицини.

Перший етап в цифровій обробці зображення - перетворення оптичного зображення в форму, яка може бути збережена в пам'яті комп’ютера. Це перетворення виконується світлочутливою системою, що називається електронно-оптичним цифровим перетворювачем. Цей перетворювач видає закодовані числа, котрі є виміром інтенсивності світла. Перетворювач реєструє оптичне зображення з повторюваними часовими інтервалами, і видає електричний струм. Електричний струм котрий перетворено в рядок дискретних числових значень, представляє розподіл інтенсивностей в оптичному зображенні. Описаний процес називано оцифровуванням

(діджиталізацією), або квантуванням. Числове представлення вихідного оптичного зображення, котре зберігають, називано цифровим зображенням. Цифрове зображення - числова абстракція збережена в пам'яті компютера, котрою можна оперувати, так само будь-якими іншими даними,. Існує величезна кількість операцій, які можуть бути виконані над цифровим зображенням; проте всі такі перетворення можуть бути віднесені до однієї з двох загальних категорій: опрацювання та аналізу зображення.

Алгоритми оброблення що передують аналізу зображень, можуть дуже відрізнятися за складністю і, тому, за вимогами до засобів для їх реалізації. Самі прості алгоритми аналізу зображення розробляються, щоб істотно підвищити продуктивність і точність при вирішенні задач, які могли бути в іншому разі виконані вручну. Типова система такого типу може складатися тільки з цифрового графічного планшета, оптичного проектора і мінікомп'ютера. Альтернативно може складатися з відеокамери, відеомонітора, і мінікомп'ютера. Така система не вимагає особливих витрат і може бути використана негайно мало тренованим персоналом [2,14].

Процедура аналізу зображення належить до процедур, які отримують описову інформацію про задане цифрове зображення. Типовим прикладом може бути визначення кількості клітин в заданому полі зору мікроскопа або вимірювання діаметрів ядер клітин. Такий тип числового аналізу називається виділенням ознак. Отримане число котре описує цифрове зображення, називають характеристикою цього зображення. У найширшому смислі, характеристика зображення - будь-яка ознака, який може використовуватися, щоб описати цифрове зображення, на основі таких, зокрема, властивостей, так само форма або площа ядра клітини, розподіл клітин в змішаній популяції за розмірами, і т.д. Крім того алгоритми аналізу зображення, можуть бути використані для створення логічного аналізу інформації про проаналізоване зображенні, наприклад, чи містить область зображення (поле зору в мікроскопі)

клітину, яка повинна бути проаналізована. Інакеше розглянута область повинна бути відхилена і слід перейти до аналізу іншої наступної області.

Наступним випадком застосування алгоритмів аналізу зображень, що вимагає досить складних апаратних і програмних засобів і високої кваліфікації користувача, можуть стати алгоритми автоматичного розпізнавання і класифікації аномальних клітин в гістологічних препаратах. Така процедура вимагає високошвидкісного вимірювання списку різних характеристик зображення препарату з метою виділення на складному тлі окремих клітин, отримання їх кількісних признаків і віднесення на основі заданих критеріїв до того чи іншого типу. Описані системи, що створюються для такого роду складного аналізу зображень, вимагають, так само мінімум, високоякісної системи вводу зображень в комп'ютер, великий оперативної пам'яті і високошвидкісного процесора. Більшість таких систем включають в себе спеціальні апаратні засоби, котрі виконують опрацювання зображення у багато разів швидше за звичайні програмованих універсальних комп'ютерів. Сучасні системи мають в своєму складі і складні пристрої формування і вводу зображень, зокрема повністю автоматизовані мікроскопи, що дозволяють не тільки позиціонувати роглянуті клітини, а й виконують процедури настроювання оптичної системи приладу для отримання оптимального якості зображення та його реєстрації з високою просторовою і фотометричною роздільністю. До високою основної вартості таких систем додаються також додаткові витрати, пов'язані з їх обслуговуванням, навчанням персоналу, розроблення спеціалізованого програмного забезпечення.

Іншою категорією процедур опрацювання зображень є процедури підняття якості зображення. Вони відрізняються від процедур аналізу зображення тим, що метою опрацювання є не отримання описової інформації про зображення, а перетворення зображення таким чином, щоб зробити його більш інформативним для спостерігача. Приклади функцій підняття якості

зображення: віднімання зафіксованих в різний час зображень, з метою виявлення руху; підняття контрасту; видалення шуму, підкреслення границь об'єктів і т.п. Процедури підняття якості зображення можуть використовуються, щоб компенсувати деградацію оригінального зображення. Деградація може бути викликана дефектами в системі формування зображення, або неоднорідністю освітлення в полі зору мікроскопа. Так як цифрове зображення є абстракцією в пам'яті комп'ютера, то ним можна керувати способами, які не можуть бути реалізовані фізичними пристроями, типу призми чи лінзи. Деякі функції підняття якості зображення використовують корекцію цифрового зображення за допомогою цифрового аналога повної оптичної системи, яка була ретельно розрахована, щоб виправити спотворення зображення [10-18].

Системи підняття якості зображення, повинні задовольняти ряду додаткових вимог - вони повинні бути гнучкими і зручними для користувача, тобто дуже інтерактивними. Система підняття якості зображення - найбільш ймовірний тип для лікаря, який використовує, наприклад, мікроскоп або рентгенівський апарат так само щоденний дослідний інструмент, і чиї вимоги до опрацювання можуть швидко змінюватися. За таких умов система повинна бути досить гнучкою, щоб дозволити користувачеві випробувати багато різних процедур опрацювання і мати можливість спостерігати результати в максимально короткий час. Такі системи зазвичай мають можливість виконання опрацювання в режимі реального часу, тобто в цифровій формі оброблене зображення буде відтворено на відеомоніторі без затримки. Очевидно, що різні застосування методів оброблення зображення висувають різні вимоги до необхідних для їх реалізації апаратних засобів та програмного забезпечення.

Існуючі методи візуалізації грунтуються на різноманітних фізичних взаємодіях електромагнітного випромінювання з органними, тканинними і клітинними структурами.

Сьогодні аналіз медико-біологічних препаратів (цитологічних і гістологічних мазків) в діагностичних лабораторіях проводиться візуально. Цей процес є рутинним і трудомістким. Тому з’явилися системи автоматизованої комп’ютерної мікроскопії (САКМ) – програмно-апаратні комплекси для цифрового оброблення мікроскопічних зображень. САКМ є апаратно-програмними системами, до складу яких входять моторизований керований мікроскоп, відеокамера, комп’ютер, функціональні програми-методики.

САКМ в цілому поділяють на спеціалізовані і дослідницькі [9, 37]. Спеціалізовані САКМ призначені для виконання певного одного стандартизованого клінічного дослідження. Характерною особливістю дослідницьких САМ є використання багатофункціональних мікроскопів і камер із збільшеною чутливістю і роздільністю. Сценарій дослідження не заданий наперед, а формується користувачем. Дослідницькі САКМ використовуються для розроблення нових методів діагностування.

Для задач мікробіології на світовому ринку пропонують САКМ або окремі їх компоненти наступні фірми: Carl Zeiss (Німеччина), Leica, Olympus, Nikon, Микромед (Росія), Motic (Китай), Konus (Італія).

Представимо узагальнений процес дослідження з допомогою САКМ. На першому етапі препарат встановлюється на предметний стіл мікроскопа. Зображення поля зору виводиться на екран монітора. У реальному режимі часу проводяться всі необхідні налаштування і захоплення зображення. Зображення може бути оброблене за допомогою фільтрів. На зображення можна нанести текстові анотації і калібрувальний маркер. САКМ дозволяє детектувати мікрооб'єкти на зображенні в напівавтоматичному, автоматичному або ручному режимах. Детектовані мікрооб’єкти вимірюються автоматично і результати вимірювань відображаються у вигляді таблиці. По будь-якому із виміряних параметрів об'єкти інтересу можуть класифікуватися. Результати аналізу зберігаються в базі даних і друкуються.

Для побудови узагальненої структури САКМ було досліджено склад і функції існуючих програмних засобів (таблиця 1.1).

Зокрема порівняння проводилось за наступними параметрами:

– спосіб вводу інформації: зображення приймається із давача в реальному часі (підтримка технології MCI/TWAIN) чи завантаження з диску;

– режими роботи алгоритмів сегментації: ручний (оператор в ручному режимі виділяє мікрооб’єкти), автоматизований (оператор попередньо проводить навчання алгоритму) чи автоматичний (параметри алгоритму вибираються автоматично);

– попереднє оброблення зображення: зменшення шумів, корекція яскравості, контрасту, фільтрація, виділення області інтересу, тощо;

– обчислення числових ознак мікрооб’єктів: периметр, площа, ядерно-цитоплазматичне віднощення, діаметр, кут між двома відрізками, тощо;

– обчислення статистичних ознак: математичне сподівання, середньоквадратичне відхилення, максимальне (мінімальне) значення, тощо;

– представлення результатів у вигляді діаграм, гістограм, графіків, тощо;

– взаємодія з іншим програмним забезпеченням: MS Word, MS Excel, MS Access, FoxPro, тощо;

– використання скриптів – присутність вбудованої мови для пакетного оброблення даних;

– наявність детальної технічної документації.

Типова структура апаратної частини САКМ складається з комп’ютера, монітора, мікроскопа, фотокамери та принтера. Елемент вводу зображення складається з світлового мікроскопа, камери, фотоадаптера, засобу заміни поточного об’єктива, засобу фокусування, засобу переміщення предметного стола, засобу подачі зразків та засобу освітлення.

Таблиця 1.1 – Порівняння функцій САКМ

№	Назва програми	Підтримка додаткових форматів файлів	Конвертування форматів	Редагування зображення	Конвертація в інші кольорові базиси	Виділення мікрооб’єктів в ручному / автоматизованому / автоматичному режимах	Визначення координат обмежуючого прямокутника	Застосування сценаріїв	Формування звітів	Передача даних в програмне забезпечення 3-х сторін	Друк	Налаштування параметрів роботи програми	Калібрування камери	Система допомоги
1	ImageTool v.5	+	+	+	+	+/-/-	-	+	+	-	+	+	-	+
2	Micromed Images	+	+	+	-	+/-/-	-	+	+	-	+	+	-	+
3	„ИМАДЖЕР-ЦГ”	-	+	+	-	+/-/-	-	-	+	-	+	+	-	+
4	BioVision	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
5	ВидеоТесТ-Морфология 5.0	+	+	+	+	+/+/+	-	+	+	+	+	+	+	+
6	ВидеоТесТ-Морфо 3.2	+	+	+	+	+/+/-	-	+	+	-	+	+	-	+
7	ScreenMeter	-	-	-	-	+/-/-	-	-	+	-	-	+	-	+
8	ImageExpert Pro 3	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
9	ImageExpert™ Gauge	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
10	AnalySIS Five	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
11	MetaMorph 7.5	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
12	QCapture PRO 8.0	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
13	Motic Images Advanced 3.2	+	+	+	+	+/+/+	+	+	+	+	+	+	-	+
14	MCID™ Core	-	+	+	+	+/+/-	-	+	+	-	+	+	-	+
15	Image-Pro Plus 6.5	+	+	+	+	+/+/-	-	+	+	-	+	+	-	+
16	NIH Image	-	+	+	+	+/+/+	+	+	+	+	+	+	-	+
17	ImageJ (NIH, USA)	-	+	+	+	+/-/-	-	+	+	-	+	+	-	+
18	Pixels	-	-	+	-	+/-/-	-	-	+	-	+	+	-	+
19	ImageWarp	-	+	+	-	+/-/-	-	+	+	-	+	+	-	+
20	Xite в 6.45	-	-	+	-	+/-/-	-	-	+	-	+	+	-	+
21	ДиаМорф	-	+	+	+	+/+/+	+	+	+	+	+	+	-	+

Світловий мікроскоп прохідного світла складається із штатива, до якого кріпляться всі основі вузли: система освітлення, предметний стіл, системи

відтворення та візуалізації зображення. Головними характеристиками мікроскопа є його клас, кратність збільшення, оптична роздільна здатність, розмір лінійного поля на предметі і ступінь виправлення аберацій. У САКМ для галузей цитології і гістології застосовують біологічні прямі мікроскопи робочого, лабораторного та дослідницького класів. Розглянуто структуру та функції програмного забезпечення аналізу біомедичних зображень. Підсумовано переваги і недоліки систем автоматизованої мікроскопії.

1.2 Огляд систем пошуку зображень за змістом

Традиційна архітектура систем пошуку зображень по вмісту представлена на рисунку 1.1. Так само і в класичних системах інформаційного пошуку, у системах CBIR виділяють два модулі: індексування й пошуку. Перший відповідає за опрацювання даних і побудову індексних структур, що дозволяють значно прискорити пошук. Другий бібліотека займається безпосередньо пошуком по запиту користувача. Під час індексування обчислюються вектори ознак для всіх зображень колекції й далі по них будується індекс. У випадку, коли пошук відбувається по зображенню-зразку, що не входить у колекцію, на етапі пошуку необхідно обчислити ознаки для зображення запиту, використовуючи ті ж самі алгоритми, за допомогою яких обчислювалися ознаки для зображень колекції. Далі пошук проводиться по отриманих векторах ознак зображення-запиту [6-10].

Деякі системи використовують механізм зворотного зв'язку для наступного уточнення запиту, при якому процес пошуку стає ітеративним і інтерактивним [13]. Після того, як система видала результати пошуку, користувачеві надається можливість відзначити серед знайдених зображень

релевантні й нерелевантні запиту. На основі цієї інформації система може автоматично уточнити первісний запит і на наступній ітерації видати результат, який краще відповідає інформаційній потребі користувача.

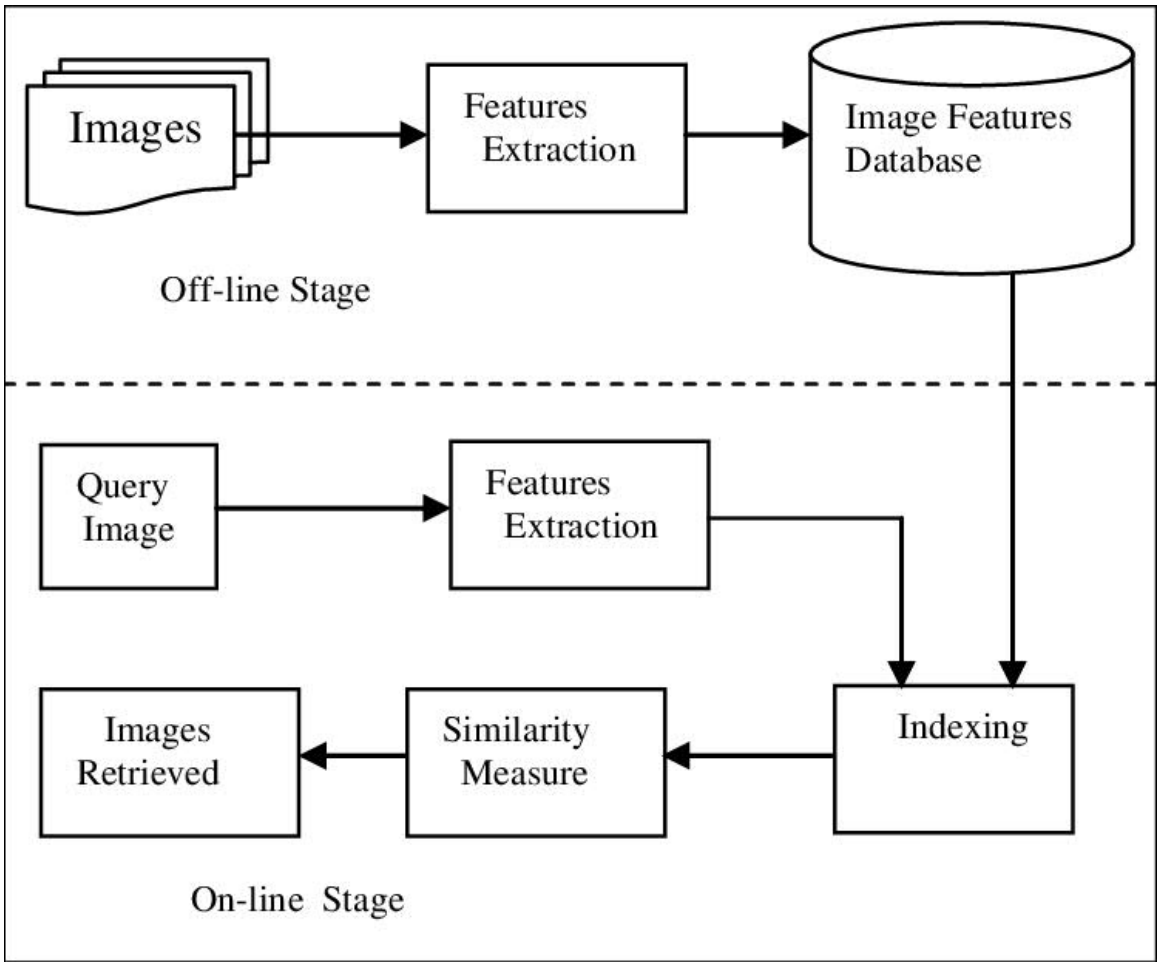


Рисунок 1.1 - Традиційна архітектура систем CBIR

Усього виділяють три основні напрямки досліджень в області CBIR.

– Побудова просторів ознак. Розробка алгоритмів обчислення векторів ознак по матриці значень пікселів зображення й метрик для їхнього порівняння.

– Багатомірне індексування. Розробка алгоритмів багатомірного індексування, що підходять для задач CBIR, для яких характерна висока розмірність і великі обсяги індексованих даних.

– Проектування систем пошуку. Важливою особливістю будь-якої системи є її ергономічність - зручність роботи для користувача. Для систем CBIR цей параметр відіграє особливу роль через складність таких систем. Задачі візуалізації результатів пошуку, побудови інтерфейсу для зручної й ефективної роботи користувача із системою становлять даний напрямок.

Від алгоритмів побудови векторів ознак залежить якість пошуку. Багатомірне індексування дозволяє зробити пошук швидким. Ергономічний інтерфейс системи пошуку допоможе користувачеві правильно сформулювати запит, уточнити його на наступних кроках спілкування із системою, полегшити роботу з пошуковою системою в цілому.

Основною проблемою пошуку по вмісту на сьогоднішній день більшість дослідників визнають так званий семантичний розрив. Людина, порівнюючи два зображення, у першу чергу порівнює їхнє значеннєве наповнення - семантику, у той час так само оцінка системи ґрунтується на порівнянні векторів ознак, відповідних до візуальних характеристик зображення.

Незручність запиту-зразка. Запит-Зразок є найпоширенішим видом запиту для систем пошуку зображень по вмісту. Однак його не можна назвати зручним для користувача. У користувача може не бути такого зразка - навпаки, він звертається по допомогу до пошукової системи, щоб знайти таке зображення. Багато систем пропонують користувачеві вибрати з випадкового набору зображень бази те, яке може виступати в ролі запиту. Такий випадковий набір може не містити підходящого зображення, і користувач змушений переглядати набір за набором у пошуках потрібного запиту. Запит-ескіз у цьому змісті виграє в запиту-зразка. Але не кожний користувач здатний скласти ескіз шуканого зображення. Для користувача кращий текстовий запит. Людині найпростіше описати словами зміст того зображення, яке він шукає. Однак для можливості пошуку по текстовому запиту система повинна мати яку-небудь текстову інформацію про зображення.

Обсяги даних і швидкість пошуку. Пошук по вмісту має на увазі пошук по низькорівневим характеристикам зображень, а точніше, по описуючим ці характеристики векторам ознак. Чим більше розмірність вектора, тем докладніше він описує характеристику зображення. Але при цьому тим більший час витрачає система на порівняння векторів між собою, і тим більші обсяги пам'яті потрібні для їхнього зберігання.

Обчислення векторів ознак також може бути досить ресурсномістким. І хоча для кожного зображення досить обчислити його вектор усього одні раз - коли зображення попадає в колекцію, у випадку більших колекцій зображень (кілька мільйонів) задача оптимізації обчислення векторів ознак стає досить актуальною. У тому випадку, коли система дозволяє використовувати в якості запиту зображення, надане користувачем, і не міститься в колекції, час обчислення вектора прямо впливає на швидкість відповіді системи.

При пошуку по різнорідній колекції зображень більшість систем аналізує відразу кілька характеристик для покращення якості пошуку, що збільшує час відгуку системи. Таким чином, підняття розмірності й числа векторів ознак робить пошук якісніше, але в той же час негативно позначається на швидкодії системи. Перед дослідниками й розроблювачами систем CBIR стоїть непроста задача одночасного підняття якості й збільшення швидкості роботи алгоритмів пошуку. Релевантність в інформаційному пошуку - семантична відповідність пошукового запиту отриманому документу.

1.3 Існуючі методи та засоби класифікації зображень

Класифікація зображень за допомогою методів комп’ютерного зору.

Ознаки Хаара - ознаки цифрового зображення, використовувані в розпізнаванні образів. Своєю назвою вони зобов'язані інтуїтивним схожістю з

вейвлетами Хаара. Ознаки Хаара використовувалися в першому детекторі осіб, що працює в реальному часі.

Історично склалося так, що алгоритми, що працюють тільки з інтенсивністю зображення (наприклад значення RGB в кожному пікселі), мають велику обчислювальну складність. В роботі Папагеоргіу, була розглянута робота з безліччю ознак, заснованих на вейвлет Хаара. Віола і Джонс адаптували ідею використання вейвлетів Хаара та розробили те, що було названо ознаками Хаара. Ознака Хаара складається з суміжних прямокутних областей. Вони позиціонуються на зображенні, далі сумуються інтенсивності пікселів в областях, після чого обчислюється різниця між сумами. Ця різниця і буде значенням певної ознаки, визначеного розміру, певним чином позиційований на зображенні.

Для прикладу розглянемо базу даних з людськими обличчями. Загальним для всіх зображень є те, що область в районі очей темніше, ніж область в районі щік. Отже загальним ознакою Хаара для осіб є 2 суміжних прямокутних регіону, що лежать на очах і щоках.

На етапі виявлення в методі Віоли - Джонса вікно встановленого розміру рухається по зображенню, і для кожної області зображення, над якою проходить вікно, розраховується ознака Хаара. Наявність або відсутність предмета в вікні визначається різницею між значенням ознаки і учнем порогом. Оскільки ознаки Хаара мало підходять для навчання або класифікації (якість трохи вище ніж у випадкової нормально розподіленої величини), для опису об'єкта з достатньою точністю необхідна більша кількість ознак. Тому в методі Віоли - Джонса ознаки Хаара організовані в каскадний класифікатор.

Ключовою особливістю ознак Хаара є найбільша, в порівнянні з іншими ознаками, швидкість. При використанні інтегрального представлення зображення, ознаки Хаара можуть обчислюватися за постійний час (приблизно 60 процесорних інструкцій на ознака з двох областей).

Машинне навчання успішно використовується в задачах комп'ютерного зору, роботи з текстами, медицини, безпілотного керування дронів та ін. Ядром для багатьох з цих додатків є методи та засоби штучного інтелекту, такі так само класифікація, локалізація і виявлення. За декілька попередніх років нейронні мережі дуже добре себе показали у задачах, де дані представлені у вигляді текстів чи зображень, відео, тому зараз їх використовують досить часто для таких задач. Для задач класифікації можна використовувати класичні методи машинного навчання: наївний Баєсовий класифікатор, метод опорних векторів, логістичну регресію, випадкові ліси чи метод стимулювання градієнта.

MNIST— об'ємна база даних зразків рукописного написання цифр. База даних є стандартом, запропонованим Національним інститутом стандартів і технологій США з метою калібрації і зіставлення методів розпізнавання зображень за допомогою машинного навчання в першу чергу на основі нейронних мереж. Дані складаються з заздалегідь підготовлених прикладів зображень, на основі яких проводиться навчання та тестування систем. База даних MNIST містить 60000 зображень для навчання і 10000 зображень для тестування. На цьому наборі даних метод наївного байєса працює досить точно, а саме за джерелом вдалось отримати 85% точності. Так як наївний баєсовий класифікатор — модель, що базується на теоремі Баєса для визначення ймовірності приналежності екземпляра до одного з класів C за умови того, що залежні змінні приймають задані значення. Це означає що, якщо на основі відомих змінних можна точно визначити, до якого класу належить екземпляр, баєсовий класифікатор підсумує, що ймовірність приналежності до даного класу рівна 1. У інших випадках, коли екземпляр може з різною ймовірністю належати до різних класів, результатом роботи класифікатора буде набір ймовірностей приналежності до певного класу. Виходячи з визначення і самого набору даних, можна пояснити чому ця модель дала прийнятну точність, а саме, що зазвичай люди пишуть схоже і так як зображення на вхід ми подаємо у

вигляді пікселів, то основна частина буде розміщатись у одного типу цифр в тих же місцях.

Але, на реальних даних, зокрема і в цій роботі, краща точність була отримана за допомогою згорткових нейронних мереж. Детальніше про ці алгоритми у наступному розділі. Але варто зазначити, що на наборі даних ImageNet - набір даних, за станом на 2017 рік там було 14 197 122 зображення, розбитих на 21 841 категорію, за допомогою різних архітектур нейронних мереж та підходів було отримано на 2017 рік точніть більше 95%.

Отже, класичні методи машинного навчання, такі як - наївний Баєсовий класифікатор, метод опорних векторів, логістичну регресію, випадкові ліси чи метод стимулювання градієнта, а також методи глибинного навчання, дають хороші результати для класифікації зображень. Звісно, метод потрібно вибирати виходячи зі специфіки даних та інших характеристик, таких як - задовільна точність алгоритму, час передбачення і тд. Зазвичай на реальних даних надають перевагу згортковим нейронним мережам, так як класичні методи не здатні виявити всі необхідні закономірності.

Комбінація методів комп’ютерного зору та нейронних мереж.

Класифікація цифрових зображень, що зберігаються в базах даних, з використанням традиційних алгоритмів машинного навчання характеризується високою трудомісткістю, що обумовлено великою кількістю зображень і деталей, якими описуються зображення. Зазначені алгоритми характеризуються невисокою стабільністю при класифікації зображення з великих баз даних. Крім того, класифікація з використанням цих алгоритмів вимагає великих тимчасових витрат. Існуючі системи зберігання зображень, такі як (QBIC и VisualSEEK, які обмежують методи класифікації способами опису зображень, заснованими на формі, текстурі і колірній інформації.

Одним з методів, що використовуються для розпізнавання, класифікації та відновлення зображень, є метод, заснований на нейронних мережах. Для того

щоб зменшити число вхідних нейронів мережі, система класифікації зображень зазвичай розташовується на кроці предобробки. Одним з кроків предобробки цифрових зображень є вейвлет-перетворення. В даний час вейвлет-перетворення є широко відомим методом, застосовуваним для аналізу зображень та отримання таких характеристик зображення, як форма і текстура.

Алгоритм, заснований на комбінації вейвлет-перетворення Хаара і нейронної мережі для класифікації цифрових зображень з бази даних. Кольорове зображення ділиться на три RGB-компоненти. Моменти кольору першого порядку і коефіцієнти розкладання вейвлет перетворення Хаара трьох RGB-компонентів зображень є вхідним вектором багатошарової нейронної мережі, навченої алгоритмом зворотного поширення помилки.

Подання вмісту цифрових зображень. Зміст і контури цифрових зображень зазвичай використовуються в класифікації зображень. У алгоритмі моменти кольору і вейвлет-коефіцієнти розкладання використовуються для представлення вмісту цифрових зображень.

Моменти кольору. Моменти кольору використовуються в багатьох системах відновлення кольорового зображення, особливо в тих випадках, коли зображення містить тільки один об'єкт. Моменти кольору першого, другого і третього порядків є ефективними для подання розподілу кольору зображень.

Вейвлет-перетворення зазвичай використовується в системах відновлення вмісту зображень. На кожному рівні вейвлет-перетворення сигнал розкладається на чотири піддіапазони частот (квадранта) LLn, LHn, HLn, HHn (рис 1.1) , де L - низька частота; H - висока частота; n - рівень розкладання. На рис. 1.1 представлені стандартні позначення квадрантів перетвореного зображення: LL, LH, HL, HH. Квадрант LLn являє собою зображення з низьким дозволом (cAn), HLn - вертикальні деталі зображення (cVn), ЛНн - горизонтальні деталізображення (cHn), HHn - діагональну інформацію зображення (cDn). У алгоритмі використовується вейвлет-перетворення Хаара

по шести рівням розкладання. Отримані вейвлет-коефіцієнти подаються на входи нейронної мережі.

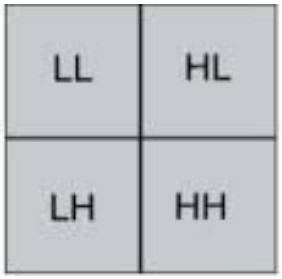


Рисунок 1.2 - Одноразове застосування двовимірного вейвлет-перетворення до квадратного зображення

У роботі [11] зазначалось, що максимальна отримана точність класифікації була 88%.

Розглянемо існуючі рішення

За останнє десятиліття було опубліковано значну кількість статей і наукових робіт про алгоритми класифікації зображень у різних областях застосування, від класифікації предметів побуту, тварин до класифікації ракових пухлин чи знімків із космосу. Крім того, було запропоновано наступні підходи: класифікація зображень, де на виході ми використовуємо лнттте мітку класу або ж використання ймовірності того, що на даному зображені клас з певною міткою. Останній підхід буде частіше використовуватись у системах, де помилки є не припустимими, наприклад, при класифікації ракових клітин. У статті[1] було описано, що класичні підходи до вирішення таких задач на наборі даних, який вони використовували дали 85% точності. У роботі[2], де потрібно було класифікувати 53 класи на їхньому наборі даних використовували специфічну опрацювання даних за рахунок чого їм і вдалось отримати 87% точності. Також була розглянуто рішення[3], де було поєднано два вище зазначених підходи, що дозволило покращити результати. Звісно були і статті,

результати яких відтворити не вдалось, так як деякі з них використовували закритий набір даних чи дуже багато обчислювальних потужностей.

1.3 Оцінка ефективності пошуку

Для оцінки ефективності пошуку автори використовували міру ефективності (retrieval efficiency) η_T, яка обчислюється в такий спосіб:

$$\eta_T = \begin{cases} n/N, якщо\ N \le T \\ n/T, якщо\ N > T \end{cases} \quad (1.1)$$

де n - кількість релевантних зображень серед перших T у вибірці,

N - кількість релевантних зображень у всій базі. Нескладно помітити, що при $N \le T$ значення ефективності збігається з повнотою (recall), а при $N > T$ - з точністю (precision), традиційними заходами в інформаційному пошуку.

У таблиці 1.2 показані результати порівняння, отримані Melitre і співавторами й представлені в роботі [9]. Згідно з отриманими результатами, ланцюгові коди значно поступаються по ефективності більш складним ознакам форми. Також можна відзначити, що не можна дати однозначну відповідь про перевагу ознак якогось одного класу - зовнішніх (по границі) або внутрішніх (по всій області). Комбінування ж тих і інших (у цьому випадку дескрипторів Фур'є й інваріантів моментів) дозволяє значно підвищити ефективність пошуку.

Комбінування різних методів пошуку.

Більшість дослідників сходиться в думці, що для створення ефективної системи пошуку по колекції різнорідних зображень недостатньо якогось одного простору ознак. Необхідно комбінувати результати пошуку, отримані в різних

просторах, що відбивають властивості різних характеристик зображення. Традиційно пошук у кожному із просторів ознак проводиться незалежно, після чого використовуються методи синтезу (комбінування) даних (Data Fusion) для побудови результуючої видачі.

Таблиця 1.2 – Значення середньої ефективності для різних ознак форми.

Методи	T=5	T=10	T=15	T=20
Ланцюгові коди	55,1%	47,6%	50,0%	60,6%
Дескриптори Фур'є	72,2%	76,9%	75,9%	74,9%
UNL features	81,3%	79,9%	83,7%	89,3%
Інваріанти моментів	84,7%	86,3%	86,6%	87,7%
Моменти Зерніке	66,9%	66,5%	70,4%	78,2%
Моменти псевдо Зерніке	66,9%	66,5%	70,4%	78,2%
MI and FD	93,8%	87,3%	87,1%	89,6%
MI and UNL	93,3%	89,2%	89,3%	91,1%

Задача синтезу даних полягає в тому, щоб маючи результати роботи деяких пошукових алгоритмів, одержати один загальний результат, по тем або іншим критеріям переважаючий вихідні. Кожний вихідний алгоритм пошуку в сукупності з даними, з якими він працює, прийнято називати свідком пли джерелом.

Існує велика кількість робіт [12-23], що розглядають задачу синтезу в контексті текстового пошуку. В 1994 році Fox і Shaw [42] представили кілька простих функцій синтезу: CombMAX, CombMIN, CombSUM, CombMED, CombANZ і CombMNZ (див. таблицю 1.3). Тут $S(d)$ - результуюче значення міри подібності документа d запиту; S_n - значення міри подібності документа

запиту згідно із джерелом *n*; *N* - загальне число джерел; *nz* - число джерел, згідно з якими значення міри подібності документа *d* запиту ненульове.

Для порівняння методів він запропонував використовувати два показники: коефіцієнт зміни числа релевантних об'єктів ($R_{overlap}$) і коефіцієнт зміни числа нерелевантних об'єктів ($N_{overlap}$) у випадку синтезу двох джерел. Ці коефіцієни розраховуються по наведених нижче формулах.

Таблиця 1.3 – Функції синтезу

Назва	Визначення
CombMAX	$S(d) = \max_{n=1\ldots N}(S_n)$
CombMIN	$S(d) = \min_{n=1\ldots N}(S_n)$
CombSUM	$S(d) = \sum_{n=1\ldots N}(S_n)$
CombANZ	$\sum_{n=1\ldots N}(S_n)/nz$
CombMNZ	$\sum_{n=1\ldots N}(S_n)*nz$

$$R_{overlap} = \frac{2R_{common}}{R_1 + R_2}, \tag{1.2}$$

$$N_{overlap} = \frac{2N_{common}}{N_1 + N_2}, \tag{1.3}$$

де R_{common}/N_{common}, R_1/N_1, R_2/N_2- число релевантних/не релевантних відповідей у результуючій видачі, у першому й у другому джерелах відповідно.

У ряді робіт розглядаються функції синтезу, що представляють собою лінійну комбінацію оцінок подібності, отриманих від різних джерел.

Дослідженню методів синтезу в контексті задачі пошуку зображень у літературі приділене менше уваги. Більшість дослідників запозичили найпростіші методи синтезу, використані при текстовому пошуку.

Перші системи пошуку зображень використовували булівську модель або її розширення для комбінованого пошуку по різних наборах ознак [19, 25].

Іншим розповсюдженим розв'язком для оцінки ступеня подібності зображень є використання зваженої суми відстаней у різних просторах ознак [9,47,6,40]. Ваги відстаней можуть задаватися експертами або обчислюватися за допомогою методів машинного навчання. Ряд алгоритмів припускає використання механізму зворотному зв'язку для автоматичної оптимізації ваг на етапі оброблення пошукового запиту [17,19].

На сьогоднішній день існує велика кількість систем пошуку зображень по змісту. Більша частина з них являє собою розробки дослідницьких лабораторії, але також відомі й комерційні системи.

Ми розглянемо тільки чотири системи: QBIC [37], VisualSEEk [15,17], Netra [10,29] і Mars [25]. Зведена інформація про використовувані ознаки в кожній із цих систем представлено в таблиці 1.4.

Таблиця 1.3 - Ознаки в системах пошуку зображень по змісту.

	Колір	Текстура	Форма
QBIC	Гістограми (HSV), $d^2(Q,I) = (H_Q\text{-}H_I)^T * A * (H_Q\text{-}H_I)$	Зображення Тамури, Евклідова відстань	Геометричні ознаки, інваріанти моментів
VisualSEEK	Гістограми (HSV), колірні множини, просторове розташування		
Netra	Гістограми (HSV), Color codebook	Фільтри Габора	Дескриптори Фур'є
Mars	Гістограми (HSV), $d(Q,I) = 1 - \sum_{i=1}^{N} \min(h_i^Q, h_i^I)$	Зображення Тамури, 3D-Гістограма	MFD (дескриптори Фур'є)

1.4. Висновки до розділу

Розглянуто структуру та функції програмного забезпечення аналізу біомедичних зображень. Підсумовано переваги і недоліки систем автоматизованої мікроскопії.

Проаналізовано структуру і функції систем пошук зображень по вмісту. Проаналізовано міри оцінки якості пошуку.

2 АЛГОРИТИМ ВИЗНАЧЕННЯ ТЕКСТУРНИХ І КОЛІРНИХ ДЕСКРИПТОРІВ

2.1 Дескриптор колірного вмісту.

Використовуючи кольорову гістограму як дескриптор зображення, ми врахуємо розподіл кольору зображення. Через це, нам доведеться зробити важливе припущення відносно нашого система пошуку зображення:

Наше припущення в тому, що зображення із подібними статистичними розподілами кольорів, вважатимуться релевантними один для одного. Навіть якщо зображення мають дуже різний вміст, вони все ще вважатимуться "подібними" за умови, що їх кольорові розподіли подібні також. Це є дійсно важливим припущенням, але справедливим є припущення використати кольорові гістограми як дескриптори зображення.

Замість використання стандартної колірної гістограми, ми застосуємо модифіковану. Наш дескриптор зображення буде 3D гістограмою кольору в колірному просторі (Відтінок, Насиченість, Значення) HSV. Зазвичай, зображення представляються як 3-мірний кортеж червоного, зеленого, і синього (RGB). Ми часто представляємо колірний простір RGB як "куб", як показано на рисунку 2.1,а нижче.

Проте, хоча RGB значення прості для розуміння, колірний простір RGB не в змозі передати те, як люди сприймають колір. Замість цього, ми збираємося користуватися колірним простором HSV, який проектує інтенсивності пікселів на циліндрі (рисунку 2.1,б).

Є інші колірні простори, які роблять навіть краще імітують те, як люди сприймають колір, як наприклад CIE L*a*b* і CIE XYZ. Проте ми обрали відносно просту колірну модель для проекту.

Так тепер, коли обрано колірний простір, треба визначити число відліків

для нашої гістограми. Гістограми використані, щоб представити (грубо) щільність інтенсивностей пікселів на зображенні. По суті, гістограма оцінює щільність імовірності функції, або, в нашому випадку, імовірність *P* появи кольору пікселя *C*, в зображенні *I*.

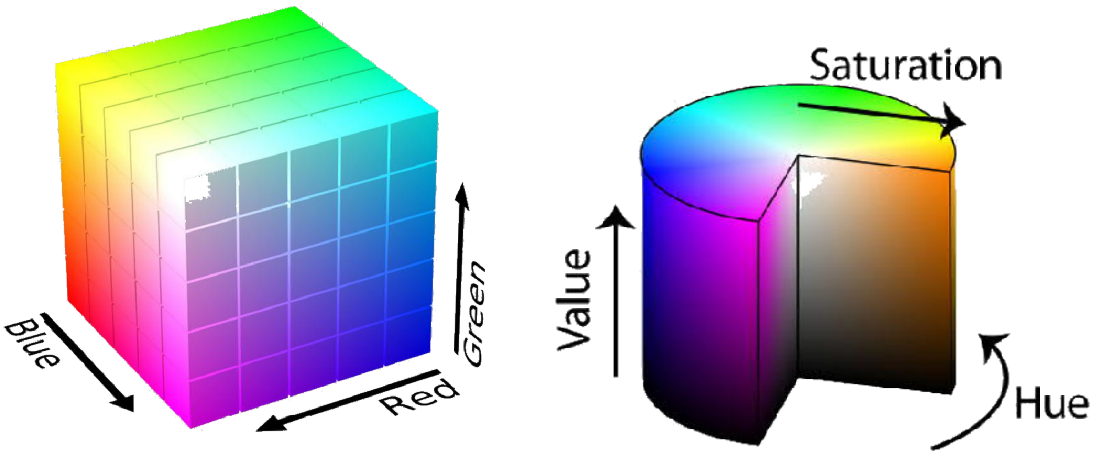


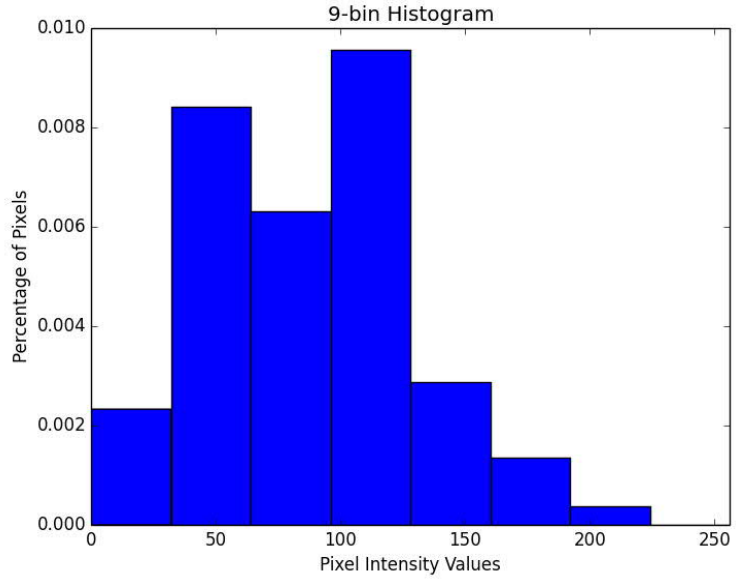
Рисунок 2.1 - Приклад куба RGB і циліндра HSV

Важливо звернути увагу, що є компроміс з числом відліків, які ми оберемо для гістограми. Якщо оберемо занадто багато відліків, то гістограма матиме менше компонентів і не в змозі зняти неоднозначність порівняння зображень з істотно різним статистичним розподілом кольорів. Також, якщо ми оберемо занадто багато відліків, гістограма матиме багато компонентів і зображення з дуже подібним вмістом, можливо, виявляться "не подібними", коли фактично вони є побідними.

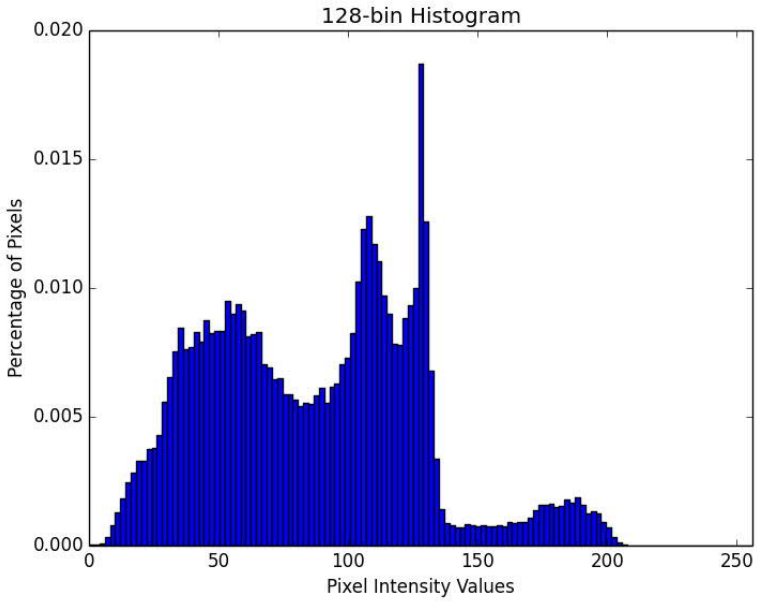
Приклад гістограми з декількома відліками показано на рисунку 1.2,а. Приклад гістограми з великою кількістю відліків показано на рисунку 1.2,б.

Ефективним є ітеративний, експериментальний підхід до налаштування числа відліків. Цей ітеративний підхід зазвичай базується на розмірі набору даних. Чим менше цей набір даних, тим менше відліків я використовую. І якщо

мій набір даних великий, я користуюся більшою кількістю відліків, що робить гістограми більш і більш дискримінуючими.



а)



б)

Рисунок 1.2- Приклад гістограми з 9 відліками (а) і з 128 відліками (б)

Потрібно експериментувати з числом відліків для побудови дескриптора кольорової гістограми, оскільки він залежить від розміру набору даних і подібності розподілів кольору

Для системи пошуку зображення, ми використовуватимемо «тривимірну» гістограму кольору в колірному просторі HSV з 8 відліками для каналу відтінку, 12-ма відліками для каналу насиченості, і 3 відліками для каналу значення. Отже, повна кількість значень вектора ознак буде 8 * 12 * 3 = 288. Це означає, що для кожного зображення в наборі даних, незалежно чи його розмір 36 x 36 пікселів чи 2000 x 1800 пікселів, усі зображення будуть абстрактно представлені і списком 288 чисел із плаваючою крапкою.

2.2 Ознаки текстури на основі локальних бінарних шаблонів

Використаємо локальні бінарні шаблони (ЛБШ, ЛБШ) для побудови текстурних ознак для зображень в колекції. ЛБШ, уперше описаний в [14], являє собою простий і потужний інструмент для розпізнавання різних елементів зображення. У якості простору ознак використовуються гістограми так званих ЛБШ кодів. Завдяки високій швидкості обчислення деяких типів кодів ЛБШ став широко застосовуватися для розпізнавання образів.

ЛБШ являє собою фільтр, що позначається як ЛБШ $_{R,P}$(x, y), який для кожної точки зображення розраховує код на основі значень точок у деякій околиці цієї точки. У цьому випадку P – число точок, R – радіус околиці

Точки околиці позначимо як g_i, де $i = \overline{0, P-1}$. При цьому координати точки розраховуються як

$$(R \cdot \cos(\frac{i}{2\pi}); R \cdot \sin(\frac{i}{2\pi})).$$

Позначимо зображення $f(x,y)$ або $f(g)$, якщо g це точка. Нехай

$$s_i(x, y) = \begin{cases} 1, f(g_i) > f(g_c) \\ 0, інакше \end{cases}$$

де g_c - точка з координатами (x, y). Тоді

$$LBP_{P,R}(x, y) = \sum_{i=0}^{P-1} 2^i \cdot s_i(x, y)$$

Таким чином, послідовність $s_i(x, y)$, де $i = \overline{0, P-1}$, являє собою двійкову послідовність коду ЛБШ. Отже, $LBP_{P,R}(x,y) \in [0, 2^P - 1]$.

Для порівняння двох зображень як векторів ознак використовуються гістограми кодів ЛБШ. У загальному випадку для кожного зображення будується гістограма $H(l)$ для значень $LBP_{P,R}(x,y)$, де $l = \overline{0, P-1}$. Існує кілька методів розрахунків відстані між гістограмами. Наприклад, відстань Хі-квадрат:

$$\chi^2(H_1, H_2) = \sum_{i=0}^{B-1} \frac{(H_1(i) - H_2(i))^2}{H_1(i) + H_2(i)},$$

де B - число кодів.

Існують також модифікації фільтра, описані в роботах [10, 13-15]:

Деякі коди несуть у собі більше інформації, ніж інші. Коди, у двійковому циклічному записі яких число переходів між послідовностями «1» і «0» не перевищує двох, позначаються як «uniform», що відповідає слову «рівномірний» [3]. Для заданого P існує P*(P-1)+2 рівномірних значень. Модифікований фільтр $LBP_{P,R}^{u2}$ у цьому випадку повертає коди рівномірних значень, додаючи тільки один код для нерівномірних значень.

$$LBP_{P,R}^{u2}(x,y) = \begin{cases} \textit{індекс коду, якщо рвіномірний} \\ P \cdot (P-1) + 2, \quad \textit{інакше} \end{cases},$$

4. Т.к. околиця являє собою коло, то можна знайти групи кодів, інваріантних до повороту. Для кожного коду ЛБШ існує P кодів, інваріантних до повороту, одержуваних шляхом циклічного зсуву P -бітового числа. Для кожної такої групи у фільтр попадає мінімальне значення кодів даної групи. Задача визначення кількості кодів, інваріантних до повороту є нетривіальною. Фільтр позначається як $LBP_{P,R}^{ri}$.

5. З врахуванням попередніх двох властивостей визначається також рівномірний фільтр, інваріантний до повороту. ЛБШ кодів, що володіють одночасно двома властивостями всього P + 2, які відрізняються друг від друга числом біт, рівних «1». У цьому випадку фільтр $LBP_{P,R}^{riu2}$ задається в такий спосіб:

$$LBP_{P,R}^{riu2}(x,y) = \begin{cases} \textit{число одиниць, якщо рвіномірний} \\ P+1, \quad \textit{інакше} \end{cases}.$$

У роботі [14] був запропонований метод порівняння гістограм мікрооб’єктів. Зображення особи розбивається на k*k ділянок, для кожного з

яких обчислюється гістограма. Підсумкова гістограма зображення мікрооб'єктів визначається як конкатенація гістограм ділянок зображення.

У роботі [26] був запропонований розширений метод порівняння гістограм мікрооб'єктів, заснований на зваженій матриці. Задається матриця ваг *k*k*, кожний елемент якої відповідає ділянці зображення. Позначимо гістограму *j*-ї ділянки зображення як H^j,. $j = \overline{0, k^2 - 1}$ Для кожної *j*-ї ділянки задається вага w_j. Тоді можна визначити модифіковану (зважену) відстань Хі-квадрат у такий спосіб:

$$\chi_w^2(H_1, H_2) = \sum_{j=0}^{k^2-1} w_j \cdot \sum_{i=0}^{B-1} \frac{(H_1^j(i) - H_2^j(i))^2}{H_1^j(i) + H_2^j(i)},$$

де *B* - число кодів ЛБШ.

Процес офлайнового обчислення дескриптора текстур, який у загальному випадку містить у собі наступні кроки:

1. Задається набір текстур, кожна з яких належить одному з декількох заданих класів.

2. Кожному класу присвоюється набір міток.

3. Для кожного зображення виконується пошук текстур із заданого набору.

4. Для кожної знайденої текстури до зображення додається набір міток відповідного класу.

Для виконання текстурного пошуку потрібен класифікатор. У даній роботі пропонується класифікатор на основі методу моментів.

У роботі [17] був запропонований метод сегментування текстур на основі моментів і описаний відповідний математичний апарат. Нехай *f*(*x, y*) являє собою зображення довжиною *W* і шириною *H*. Область значень *f*(*x,y*) - [0;1], що

відповідає області [0; 255] звичайного монохромного зображення. Метод моментів полягає в тому, що для кожної точки зображення $f(x, y)$ розраховується набір моментів і похідних від них характеристик у межах деякого вікна із центром у даній точці. Таким чином, формується набір зображень, відповідних до набору ознак.

Момент $M_{p\ q}$ з розміром вікна W_M розраховується в такий спосіб:

$$M_{p,q}(i, j) = \sum_{(a,b)\in W_{i,j}^{M}} f(a,b) \cdot x_a^p \cdot y_b^q$$

де

$$x_a = \frac{a-i}{W_M / 2};\ y_b = \frac{b-i}{W_M / 2}$$

де $W_{i,j}^{M}$ - вікно розміром W_M із центром у точці (i,j).

У тій же роботі [17] стверджувалося, що набір моментів в «чистому» виді не годиться для сегментування, тому автором був запропонований покращений набір ознак:

$$F_{p,q}(i, j) = \frac{1}{W_F^2} \sum_{(a,b)\in W_{i,j}^{F}} \left|\tanh(\sigma(M_{p,q}(a,b) - \overline{M_{p,q}}))\right|,$$

$$\overline{M_{p,q}}(i, j) = \frac{1}{W_F^2} \sum_{(a,b)\in W_{i,j}^{F}} M_{p,q}(a,b),$$

де $W_{i,j}^{F}$ - вікно розміром W_F із центром у точці (i,j).

Ступені моментів p і q задаються таким чином, щоб їх сума не перевищувала деякого значення O, яке являє собою порядок моментів. Число

ознак, таким чином, залежить від O. У даній роботі розглядалися моменти порядку не вище 2-го.

Іншими параметрами для розрахунків характеристик є розміри вікон W_M і W_F і коефіцієнт C. Авторами даного методу запропоновані значення даних параметрів: 9, 49 і 0.01 відповідно. На жаль, дані значення були обґрунтовані методом «проб і помилок» без якого-небудь аналітичного або практичного обґрунтування. Тому в даній роботі був проведений аналіз впливу значень даних параметрів на результат сегментації.

Для експериментів використовувався алгоритм сегментації, похідний від алгоритму, запропонованого в [17].

1. Тестове зображення «склеюється» з декількох текстур (приклад на рисунку 4). Кожна текстура представлена квадратною ділянкою. Площі ділянок текстур однакові.

2. Розраховуються характеристики, що представляють собою N зображень, де N - число характеристик.

3. Випадково відбираються 6 % точок, рівномірно розподілених по зображенню.

4. Для обраних точок проводиться кластеризація методом k-середніх. Число кластерів відповідає числу текстур. Автори методу запропонували алгоритм кластеризації CLUSTER [12].

5. Отримані центри кластерів як результат роботи алгоритму k-середніх використовуються для сегментації всього зображення. Кожному пікселю ставиться у відповідність номер того кластера, до якого ближче всього розташовується відповідний вектор ознак.

6. Визначимо точність сегментації p як частка правильно сегментованих пікселів. Оскільки алгоритм k-середніх довільно визначає початкові центри кластерів, до алгоритму додається проміжний крок, на якому визначається відповідність номеру текстури й номеру кластера. Будемо вважати, що кластер з

номером *i* буде відповідати текстурі *j*, якщо більша частина пікселів класу *i* буде розташовуватися усередині ділянки текстури *j* .

2.3 Порівняння зображень з допомогою коефіцієнтів подібності

Для порівняння зображень різних розмірів можна використовувати приведення розмірів до деякої еталонної роздільності, наприклад, 16x16 пікселів. За допомогою порівняння одержуваних еталонів можна швидко знайти клас ймовірно схожих зображень, серед яких можна провести більш детальний аналіз, наприклад, порівняти їх за стандартами інших розмірів.

Порівняння еталонів можна проводити за допомогою простих процедур, таких як обчислення норми різниці матриць, що складаються з закодованих числами кольорів пікселів зображень. Однак подібні підходи потребують значних витрат часу і з цієї причини не пристосовані для вирішення завдань класифікації зображень.

Ефективним підходом до порівняння еталонних зображень є використання представлення зображень у вигляді невпорядкованих множин деяких заздалегідь виділених елементів. Наприклад, можна розбити всі можливі кольори на кілька груп, і побудувати по заданому зображенню множину всіх груп кольорів його пікселів. Для оцінки близькості множинних представлень зображень застосовуються різні коефіцієнти подібності множин, наприклад, коефіцієнт Жаккар.

Обчислення коефіцієнтів подібності зображень безпосередньо є досить трудомістким завданням. Для швидкого наближеного обчислення коефіцієнта Жаккар можна використовувати метод MinHash. При застосуванні цього методу множина, відповідна зображенню, описується за допомогою вектора, кожна

компонента якого є мінімальне значення хеш-функції серед її значень для всіх елементів множини. Кожній компоненті вектора відповідає своя хеш-функція. Вектори, одержувані за допомогою методу MinHash, як правило, мають велику розмірність.

Locality-sensitive hashing (LSH, локально-чутливе хешування) -імовірнісні метод зниження розмірності багатовимірних даних. Цей метод дозволяє будувати структуру для швидкого наближеного (імовірнісного) пошуку *n*-мірних векторів, «схожих» на шуканий шаблон. Він часто використовується, коли є надзвичайно великі обсяги даних, які повинні бути зрівняні. LSH має велику кількість застосувань. Він використовується в таких областях, як розпізнавання образів, обчислювальна геометрія і стиснення даних, в класифікації і кластеризації.

Проаналізуємо використання методу MinHash і LSH для сигнатур, одержуваних за допомогою цього методу. Розглядаються загальні класи хеш-функцій, які можна використовувати для зниження розмірності даних.

Нехай *S* і *T* - дві одночасно непустих множини. Визначимо коефіцієнт Жаккар для множин *S* і *T* як відношення числа елементів, що входять в перетин множин S і T до числа елементів, що входять в об'єднання цих множин, т. Е. За формулою:

$$\frac{|S \cap T|}{|S \cup T|}. \qquad (2.1)$$

Розглянемо природне уявлення множини. Нехай - універсальна множина, елементи якого довільно впорядковані. Нехай безліч *S* з *U*. Безліч *S* може бути представлено вектором довжини \ *U* \ з нулів і одиниць, в якому *1* означає, що відповідний елемент з універсальної множини присутній в *S*, і *0* означає

відсутність цього елементу в *S*. З кінцевим числом множин з *U* можна пов'язати матрицю характеристик, де кожен стовпець представляє собою вектор, що відповідає певному множини і кожен рядок відповідає елементу *U*.

Наприклад, розглянемо таблицю 2.1. Тут чотири множини S_i (що представляють сім'ї) і універсальне безліч *U*, що складається з п'яти елементів (варіанти відпочинку). З характеристичної матриці слід, що безліч S_1 воліє круїз і сафарі, а S_2 любить відвідувати тільки курорти.

Таблиця 2.1. – Матриця характеристик

	S_1	S_2	S_3	S_4
Круїз	1	0	0	1
Лижі	0	0	1	0
Курорти	0	1	0	1
Сафарі	1	0	1	1
Залишитися вдома	0	0	1	0

Одним з ефективних способів знаходження сигнатури для набору множесті є застосування методу MinHash: спочатку рядки характеристичної матриці випадковим чином переставляються, потім для кожного множини S_i (стовпець в характеристичній матриці), можна знайти MinHash- значення *h* як номер (починаючи з 0) першого рядка, в якій є 1 (якщо в стовпці немає одиниць, в якості значення *h* до можна взяти $|U|$).

Використовуючи попередній приклад, припустимо, що результат перестановки рядків представлений в таблиці 2.2.

Таблиця 2.2. – Характеристична матриця після перестановки рядків

	S_1	S_2	S_3	S_4
Круїз	0	0	1	0
Лижі	1	0	1	1
Курорти	0	0	1	0
Сафарі	0	1	0	1
Залишитися вдома	1	0	0	1

Тоді значення MinHash для відповідних множин

$$h(S_1)=1, h(S_2)=3, h(S_3)=0, h(S_4)=1. \quad (2.3)$$

Наступна лема показує, що значення коефіцієнта Жаккар двох множин збігається з ймовірністю того, що після випадкової перестановки рядків характеристичної матриці їх значення MinHash виявляться рівними.

Лемма 1.1. Для будь-яких двох множин S_i та S_j ймовірність того, що $h(S_i)=h(S_j)$ дорівнює коефіцієнту Жаккар цих множин.

Доведення [32]. Уявімо множини S_i та S_j у вигляді стовпців перед випадковою перестановкою. Тоді рядки можна розділити на три типи:

а) рядки мають в обох стовпчиках 0;

б) рядки мають в обох стовпчиках 1;

в) рядок має в одному стовпці 1, в іншому стовпці - 0.

Нехай X - число рядків типу (b), Y - число рядків типу (c). Зауважимо, що коефіцієнт Жаккар множин S_i та S_j, дорівнює $\frac{x}{x+y}$, так як X - розмір перетину $S_i \cap S_j$, $X + Y$ – розмір об'єднання $S_i \cup S_j$.

Тепер розглянемо вірогідність того, що $h(S_i)=h(S_j)$. Нехай рядки переставляються в довільному порядку. Потім будемо переглядати рядки зверху вниз. Тоді ймовірність зустріти тип (*b*) перед типом (*c*) дорівнює $\frac{x}{x+y}$. Але якщо перший рядок зверху, крім (*a*) рядків є типом (*b*), то, очевидно, що $h(S_i)=h(S_j)$. З іншого боку, якщо перший рядок буде типу (*c*), то безліч з *1* отримує цей рядок в якості MinHash, а безліч з *0* в цьому рядку, має *1* в іншому рядку, що йде далі. Отже, $h(S_i)\neq h(S_j)$, якщо перший рядок буде типу (*c*).

Таким чином, показано, що ймовірність того, що $h(S_i)=h(S_j)$ дорівнює $\frac{x}{x+y}$, що збігається з коефіцієнтом Жаккара для множин S_i та S_j. Лемма доведена.

Розглянемо MinHash-сигнатури, які мають менші розміри матриці, отримані шляхом кількаразового застосування методу MinHash до характеристичної матриці. Нехай *M* позначає характеристическую матрицю для множини системи *S* універсальної множини *U*. Виберемо набір *n* випадкових перестановок рядків характеристичної матриці.

Для кожної множини в *S* можна обчислити його *h*-значення по відношенню до кожного набору з *n* перестановок. Це призводить до сигнатурної матриці - матриці, що складається з |*S*| стовпців і *n* рядків, де (*i*,*j*) - му елементу відповідає сигнатура *j*-го множини щодо *i*-ї хеш-функції. Зазвичай сигнатурної матриця повинна бути значно меншого розміру, ніж *M*.

Представимо приклад мінімальної хеш-сигнатури, отриманої з таблиці 2.1 за допомогою *n* = 2 перестановок (задаються функціями *h1* (*r*) = *r + 1 mod 5*, і *h2* (*r*) = *3r + 1 mod 5*, де *r* - номер рядка таблиці 1.1, починаючи з 0).

Таблиця 2.3 – Сигнатурної матриця для чотирьох множин

	S_1	S_2	S_3	S_4
h_1	1	3	0	1
h_2	0	2	0	0

Обчислення сигнатурної матриці має бути зроблено в електронному вигляді: використовуючи описаний метод, необхідно провести п випадкових перестановок характеристичної матриці *M* досить великого розміру, і це є трудомістким завданням. Можна використовувати *n* випадкових хеш-функ- ций, переставляють рядки з *M*. Будемо вважати, що відображення з |*U*| елементів на себе не викличе багато збігів, проте деякі збіги можуть бути.

Таким чином, *h*(*r*) Для рядка з номером *r* (починаючи з *0*) з матриці *M* позначає новий номер рядка після перестановки. Нижче наведено короткий опис алгоритму для обчислення сигнатурної матриці.

Крок 1. Вибрати *n* випадкових хеш-функцій h_1, ..., h_n.

Крок 2. Створити масив *H* з *n* рядків і |*S*| стовпців для зберігання сигнатурної матриці і заповнити його ∞.

Крок 3. Виконати наступні кроки для кожного рядка з номером *r* з матриці *M*.

1. Обчислити значення $h_1(r)$, ..., $h_n(r)$.

2. Для кожного $c = 1, ..., |S|$, якщо $M[r,c] = 1$, то привласнити

$H[i, c] = min\ (H\ [i,c], h_1(r))$ для $i = 1, ..., n$.

Зауважимо, що для кожного ненульового елемента з *M*, проводиться *O* (*n*) обчислень, і, крім пам'яті для збереження *n* хеш-функцій, ніяка додаткова пам'ять не потрібно.

Незважаючи на те, що сигнатурної матриця має менший розмір порівняно з характеристичною матрицею, її розмір, як і раніше може бути досить великим. До того ж, якщо документи порівнюються попарно, то це займе багато часу.

Для швидкого попарного порівняння можна застосувати LSH-метод хешування, що збільшує ймовірність колізії об'єктів, які мають шанси бути схожими.

2.4 Алгоритм пошуку по вмісту за дескрипторами

Для порівняння двох гістограм використовують критерії подібності на основі таких метрик: Хі-квадрат, відстань Бхаттачарія, перетин:

$$d_h(H_1, H_2) = \sum_{i=1}^{h} \left(\frac{\left(H_1(i) - H_2(i)\right)^2}{H_1(i) + H_2(i)} \right),$$

$$d_b(H_1, H_2) = \sqrt{1 - \sum_{i=1}^{h} \left(\frac{\sqrt{H_1(i) \cdot H_2(i)}}{\sqrt{\sum_i H_1(i) \cdot \sum_i H_2(i)}} \right)},$$

$$d_i(H_1, H_2) = \sum_{i=1}^{h} \min\left(H_1(i), H_2(i)\right),$$

де h – кількість інтервалів групування гістограми.

Основними характеристиками біомедичних зображень є колір і текстурні ознаки ЛБШ . Загальноприйнятою практикою є незалежна оцінка подібності зображень по кольору й текстурі й використання лінійної комбінації для одержання загальної оцінки подібності. Більшість існуючих алгоритмів комбінування результатів різних методів пошуку не залежать від зображення-запиту. При використанні лінійної комбінації вихідних дескрипторів подібності як функції синтезу оптимальні коефіцієнти визначаються експертом заздалегідь

і однакові для всіх запитів [1,6,9]. Для оптимізації ваг під конкретний запит використовується механізм зворотного зв'язку [37,39].

Результат пошуку можна покращити, якщо виділити залежності між ознаками запиту й оптимальними для нього коефіцієнтами для комбінування колірної оцінки і текстурних ознак. У якості алгоритму синтезу використовуються лінійні комбінації відповідних метрик (будемо далі називати лінійну комбінацію вихідних метрик з певними вагами змішаною метрикою).

Будемо обчислювати підсумкову оцінку подібності зображень запиту-зразку за допомогою метрик D_{mixed}, які є лінійною комбінацією колірної метрики і функцій відстані текстурних ознак з певними вагами

$$D_{mixed}(I,Q) = \alpha * D_{color}(I,Q) + \gamma * D_{GLCM}(I,Q) + (1 - \gamma) * D_{LBP}(I,Q), \quad (2.5)$$

де I і Q – зображення;

D_{color} - колірна метрика;

D_{GLCM} – функція відстані на основі МРРС;

D_{LBP} – функція відстані на основі ЛБШ;

γ - параметр із відрізка [0,1].

Ми припускаємо, що для схожих зображень-запитів оптимальне значення параметра γ буде однаковим. У такому випадку можна виділити класи семантично близьких зображень, визначити оптимальне значення даного параметра для кожного класу й визначити загальні ознаки зображень, що належать одному класу. Далі, під час пошуку по запиту визначати, до якого з виділених класів відноситься зображення-зразок, і використовувати оптимальне для даного класу значення параметра а для обчислення змішаної метрики.

Таким чином, задача зводиться до наступних підзадач:

1) Перевірка існування й визначення єдиного оптимального значення параметра γ для класів семантично близьких зображень.

2) Класифікація довільного запиту-зразка по виділених класах зображень із ідентичним оптимальним значенням параметра γ.

Можливо, не для кожного із класів семантично близьких зображень вдасться визначити оптимальне значення параметра γ й обчислити спільні ознаки. У такому випадку для зображення-зразка, який неможливо віднести до жодного з певних класів з відомим значенням параметра γ, під час пошуку слід використовувати деяке середнє значення даного параметра.

Вибір оптимальних ваг для комбінування результатів пошуку по кольору й дескриптору структурних змін залежно від запиту-зразка. Перевіримо гіпотезу про те, що можна визначити оптимальні коефіцієнти для комбінування оцінок по кольору й дескрипторі структурних змін залежно від ознак запиту.

Для оцінки ефективності розроблених ознак і алгоритмів буде обчислено повноту.

Для отриманих результатів пошуку була обчислені значення середньої повноти на рівні N для кожного із кластерів і кожної зі змішаних метрик, де N розмір відповідного кластера. Для кожного із зображень кластера *cluster*, що виступив у ролі запиту при пошуку з використанням змішаної метрики *mixed-metrics*, була обчислена повнота результату в такий спосіб:

$$Recall_{cluster,mixed-metrics} = \frac{\left|cluster \cap retrived_{mixed-metrics}\right|}{\left|cluster\right|}$$

- відношення числа зображень, що входять у той же кластер *cluster*, що й запит, серед перших N результатів пошуку із застосуванням заданої змішаної метрики ($\left|cluster \cap retrived_{mixed-metrics}\right|$) до загального числа зображень у даному кластері |cluster| = N.

Далі була обчислена середня для даного кластера повнота за результатами для кожного із зображень кластера. У якості найкращої змішаної метрики для кластера була обрана метрика з максимальною повнотою на даному кластері.

Функція подібності є лінійною комбінацією текстурних ознак з певними вагами. По ній буде обчислюватися підсумкова оцінка подібності запиту-зразку і зображень в БД.

2.5 Висновки до розділу

Для системи пошуку зображення використовуватимемо «тривимірну» гістограму кольору в колірному просторі HSV з 8 відліками для каналу відтінку, 12-ма відліками для каналу насиченості, і 3 відліками для каналу значення.

Для порівняння текстур як векторів ознак використовуються гістограми кодів локальних бінарних шаблонів.

Розроблено алгоритми обчислення дескрипторів кольору та текстури. Розроблено алгоритми пошуку по вмісту шляхом лінійного комбінування дескрипторів проіндексованих зображень.

3 ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ РОЗРОБЛЕНИХ АЛГОРИТМІВ

3.1 Інструментальні засоби

Виберемо програмні засоби для реалізації розроблених алгоритмів. Існує два великих класи програмного забезпечення для комп’ютерної графіки [26, 27]: пакети спеціального призначення і загальні програмні пакети. Інтерфейс пакету спеціального призначення, як правило, представляє собою набір меню, яке дозволяє користувачу взаємодіяти з програмою на своїй власній мові. В якості прикладів таких програм можна назвати програми малювання для художників і різні архітектурні, ділові, медичні та інженерні системи автоматизованого проектування. Пакети загального призначення навпаки пропонують бібліотеку графічних функцій, які можна використати в мовах програмування, таких як C, C++, Java чи Fortran. До числа основних функцій звичайної графічної бібліотеки відносяться функції для опису елементів малюнка (прямих ліній, багатокутників, кіл та інших об’єктів), призначення кольорових характеристик, виконання поворотів чи інших перетворень. До графічних програмних пакетів загального призначення відносяться, OpenCL, VRML (Virtual-Reality Modeling Language), Java 2D, Java 3D, OpenCV.

Для реалізації алгоритмів та систем опрацювання зображень більшу роль відіграють інтерпретовані мови. Зумовлено зростаючою міццю комп'ютерів і хмарних сервісів, що починають забезпечувати достатню швидкість виконання інтерпретованих програм. Все ще істотною перевагою компільованих мов програмування є високошвидкісний код. Якщо швидкість виконання додатку не є критичною, найбільш правильним вибором буде інтерпретована мова, котра є більш простим і гнучким інструментом програмування. Для програмної

реалізації розроблених алгоритмів та сценарію для тестування ми обрали інтерпретовану мову Python. Розглянемо її переваги.

При реалізації програмного забезпечення створено узагальнену структуру програмного засобу та показані використані інструментальні бібліотеки (рисунок 3.2). Бібліотека SciPy містить модулі для інтегрування, оптимізації, спеціальних функцій, обробки зображень, обробки сигналів, генетичних алгоритмів, розв'язування звичайних диференціальних рівнянь та інших задач, які розв'язуються в науці і при інженерній розробці. Бібліотека NumPy надає бібліотеку високорівневих математичних функцій для операцій над великими багатомірниими масивами і матрицями. Бібліотека CV дозволяє підключати в бібліотеку алгоритмів опрацювання зображень OpenCV. Бібліотека scikit-image пизначена для обробки зображень, і реалізує алгоритми і утиліти для використання в науково-дослідному, освітньому і промисловому ПЗ. Бібліотека scikit-learn реалізує основні алгоритми машинного навчання: алгоритми навчання без учителя (Unsupervised Learning), алгоритми навчання зі вчителем (Supervised Learning). Бібліотека matplotlib використовується для візуалізації наукових даних у вигляді графіків.

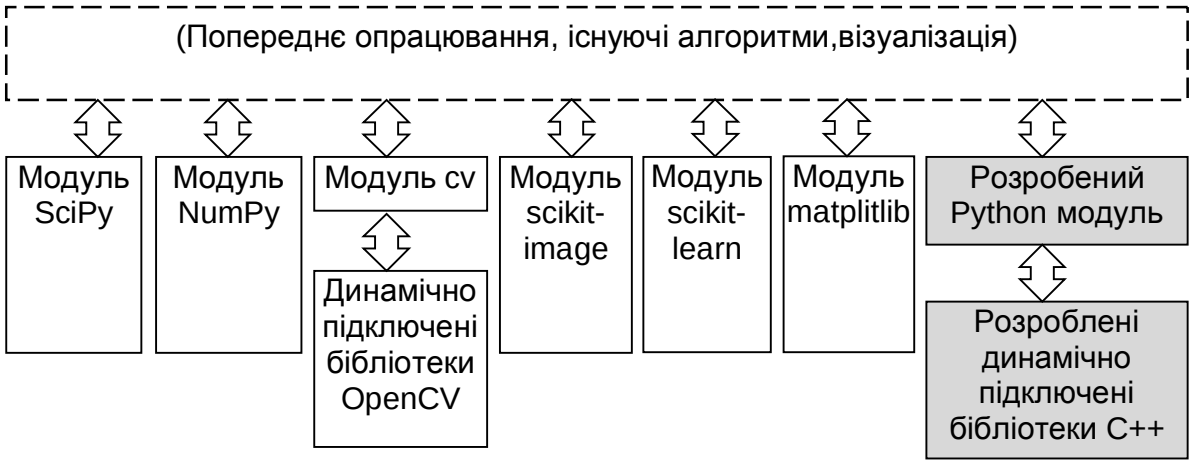


Рисунок 3.1 – Структура програмного засобу оброблення асиметричних зображень

Проектування архітектури програмного забезпечення – це процес розробки, що виконується після етапу аналізу і формулювання вимог. Задача такого проектування – перетворення вимог до системи у вимоги до програмного забезпечення і побудова на їхній основі архітектури системи. Побудова архітектури системи здійснюється шляхом визначення цілей системи, її вхідних і вихідних даних, декомпозиції системи на підсистеми, компоненти або модулі та розроблення її загальної структури.

Проектування виконується для певної моделі даних. Для реляційної меделі даних логічне проектування полягає у створенні реляційної схеми, визначенні числа і структури таблиць, формуванні запитів до БД, визначенні типів звітних документів, розробці алгоритмів обробки інформації, створенні форм для вводу і редагування даних в БД і рішенні цілого ряду інших задач. Концептуальні моделі за певними правилами перетворюються в логічні моделі даних. Коректність логічної моделі перевіряю за допомогою правил нормалізації, які дозволяють переконатися в структурній узгодженості, логічній цілісності і мінімальній збитковості прийнятої моделі даних. Модель також проходить перевірку з метою виявлення транзакцій, які будуть задаватися користувачами і можуть бути виконані.

Рисунок 3.2 – Клієнт-серверна дворівнева архітектура

Структурна схема програмного продукту зображена на рисунку 3.3.

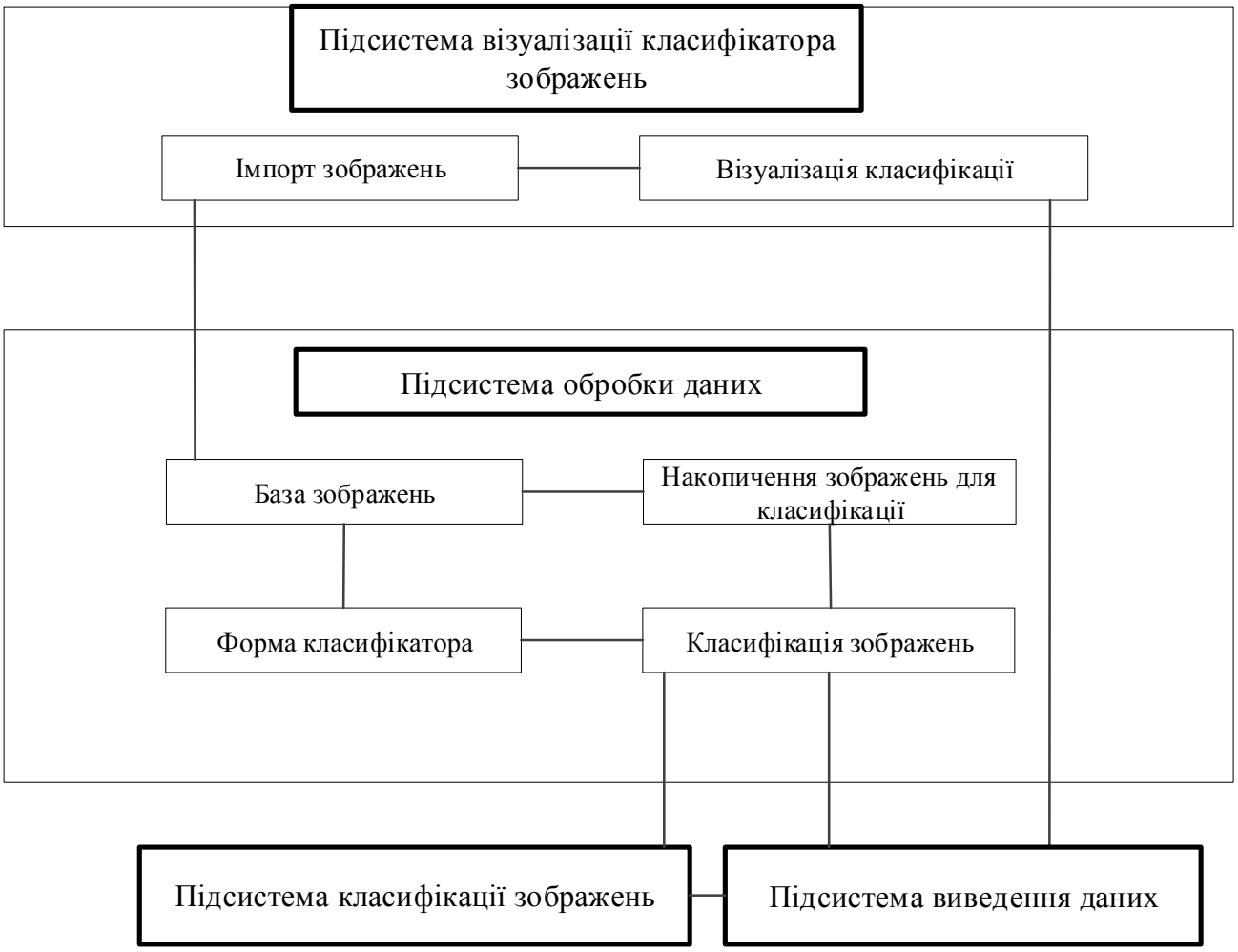


Рисунок 3.3 – Структурна схема програмного продукту

Алгоритм – набір впорядкованих правил необхідних для виконання певного процесу, які призведуть до розв’язання певної задачі за скінченне число операцій. Побудова логічної моделі БД заснована на перетворенні спрощеної концептуальної моделі БД в логічну модель REPL. При написанні комп’ютерних програм алгоритм описує логічну послідовність операцій. Для візуального зображення алгоритмів часто використовують блок-схему.

Кожен алгоритм є списком добре визначених інструкцій для розв’язання задачі. За допомогою інструкцій алгоритму описується процес обчислення від початкового стану, що проходить через послідовність станів, які закінчуються кінцевим станом.

Кожен алгоритм передбачає існування початкових (вхідних) даних та в результаті роботи призводить до отримання певного результату. Робота кожного алгоритму відбувається шляхом виконання послідовності деяких елементарних дій. Ці дії називають кроками, а процес їхнього виконання називають алгоритмічним процесом. В такий спосіб відзначають властивість дискретності алгоритму.

Алгоритм розроблюваного програмного продукту починається запуском програми. Після запуску програми користувач вибирає необхідну дію, або в будь-який момент може вийти з програми. Якщо користувач має за мету оформити звіт або додати, видалити чи редагувати дані по будь-якій з доступних категорій, то він вибирає категорію і на формі вводу даних проводить необхідні маніпуляції.

Для розуміння процесу функціонування системи побудовано діаграми станів для кожного модуля. Діаграму активності для основних модулів зображено на рисунку 3.6.

Наведена вище діаграма показує зміну станів об’єкта «користувач» в циклі роботи програми.

Після запуску програми відбувається авторизація. Якщо дані введені вірно відбуваються процеси машинного навчання, а також імпорту та класифікації зображень. В випадку неправильних авторизаційних даних, відбувається повторний їх ввід.

3.2 Програмна реалізація прототипу системи пошуку по вмісту

Існує три типи систем пошуку зображень: пошук по метаданих, пошук по зразку, і гібридний підхід на основі попередніх двох.

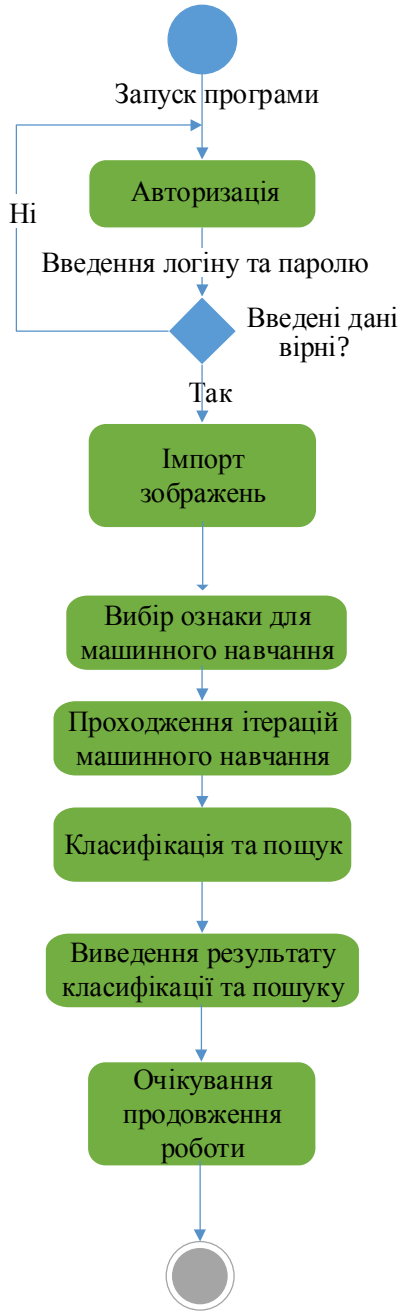


Рисунок 3.4 – Діаграма активності основних модулів програми

Пошук по метаданих, лише незначно відрізняється від стандартних пошукових систем на основі ключових слів. Пошук в системах по метаданих рідко включає зміст самого зображення. Замість цього він покладається на текстові підказки, такі як (1) ручне анотування і тагування, що виконується

людьми, а також (2) на основі автоматизованих контекстних підказок, як текст поруч із зображенням на веб-сторінці. Коли користувач виконує пошук в системі по метаданих, то вони пропонують запит, такий самий, як при традиційні системи текстового пошуку. У видачі повертають зображення, які мають такі ж теги або анотації.

В CBIR використовується певний алгоритм, щоб виділити "ознаки" (тобто список чисел для кількісного абстрактного представлення зображення) із самого зображення. Потім, коли користувач надсилає зображення-запит, з нього виділяються ознаки і порівнюються їх з базою даних ознак для знаходження схожих зображень. Системи пошуку на основі прикладу покладаються виключно на змісті зображення. Ці типи систем, як правило, надзвичайно важко побудувати і масштабувати, але вони дозволяють використати повністю автоматичні алгоритми для управління пошуком - не потрібно ніякого втручання людини.

При побудові системи пошуку зображення спочатку потрібно індексувати наш набір даних. Індексування набору даних є процесом кількісної оцінки набору даних, використовуючи дескриптор зображення для виділення ознак з кожного зображення. Дескриптор зображення визначає алгоритм, який використовується, щоб описати зображення. Наприклад:

– середнє значення і стандартне відхилення кожного каналу червоного, зеленого і синього відповідно;

– статистичні моменти зображення для характеристики форми;

– величина градієнту і орієнтації, щоб описати форму і текстуру.

Дескриптор зображення визначає як кількісно оцінити зображення. Ознаки, з іншого боку, є результатом роботи дескриптора зображення. Коли ми подаємо зображення на дескриптор, то отримуємо ознаку на виході. Отже,

ознаки (або вектори ознак) це лише список чисел, використовуваних для абстрактного представлення і кількісної оцінки зображення (рисунок 3.5).

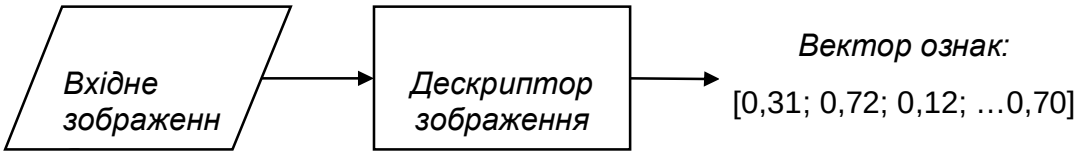
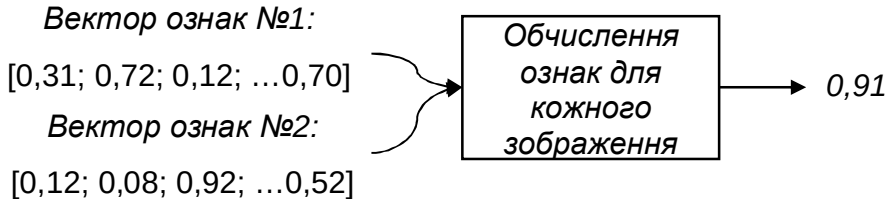


Рисунок 3.5 - Схема дескриптора зображення

Вектори ознак можна порівняти по подібності за допомогою використання метрики або функції подібності. Метрики і функції подібності беруть два вектори ознак в якості вхідних даних, а потім виводять число, що показує наскільки "подібні" два вектора ознак. На рисунку 3.6 показано процес порівняння двох зображень.



Рисунку 3.6 – Схема порівняння двох зображень

Для порівняння двох зображень, передаються відповідні вектори ознак в функцію відстані (метрики) або подібності. Результатом є значення, яке використовується для представлення та кількісної оцінки, наскільки "однакові" два зображення. Результатом функції відстані є одне число з плаваючою крапкою, що використовується для позначення подібності між двома зображеннями.

Послідовність із 4 кроків для побудови системи CBIR:

1. Визначення дескрипторів зображень. На цьому етапі необхідно вирішити, який аспект зображення нам потрібно описати. Першою основною характеристикою ГЗ є колір. Другою основною ознакою, як було виявлено, є структура мікрооб’єктів на ГЗ.

2. Індексація набору даних. Для кожного нового зображення в наборі даних потрібно застосувати дескриптор зображення. Обчислені ознаки записати в сховище (наприклад, CSV файл, RDBMS, Redis, і т.д.), так, щоб їх можна було пізніше порівнювати.

3. Визначення метрики подібності. Побудувати функції порівняння. Популярні варіанти включають евклідова відстань, косинусну відстань, і відстань Хі-квадрат. Фактично вибір сильно залежить від (1) набору даних і (2) типу виділених ознак.

4. Пошук. Останній крок полягає у виконанні власне процесу пошуку. Користувач надає зображення-запит до системи (засобами інтерфейсу), а система (1) обчислює ознаки для нього, а потім (2) застосовує функцію подібності для порівняння його із уже проіндексованими. Далі повертаються найбільш релевантні результати відповідно до функції подібності.

Оскільки системи стають все більш складними і використовують різні ознаки, число кроків росте і потрібно буде додавати значну кількість допоміжних кроків для кожного зазначеного вище етапу. На рисунку 3.7 показано кроки 1 і 2:

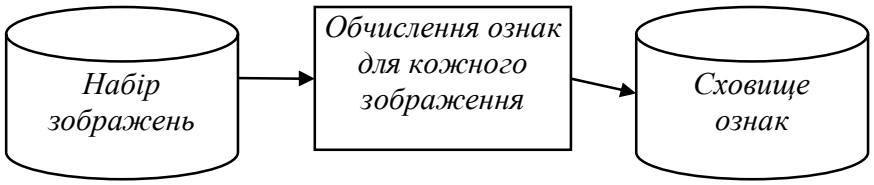


Рисунок 3.7 - Схема, що представляє процес виділення ознак з кожного зображення в наборі даних.

Почнемо з виділення ознак з кожного зображення в наборі даних. Далі ці ознаки зберігаються в базі даних. Далі (рисунок 3.8) можна перейти до функції пошуку (кроки 3 і 4).

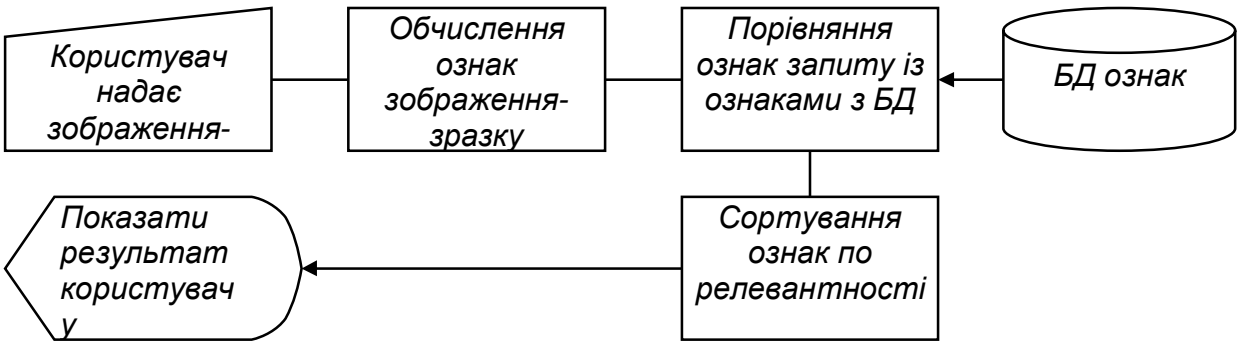


Рисунок 3.8 - Схема пошуку в системі CBIR

Користувач відправляє запит, зображення-запит описується, ознаки запиту порівнюються з існуючими ознаками в базі даних, результати впорядковуються за релевантністю, а потім представляються користувачеві.

Ми використовуватимемо набір даних зображень Breast Cancer Dataset. З огляду на наш набір даних, ми хочемо зробити цей набір даних "можливим для пошуку" та реалізувати пошук "найбільш подібного" - це буде система типу "пошук по зразку". Наприклад, якщо користувач запитує зображення більш нормальних тканин то і релевантними будуть зображення нормальних тканин.

Опишемо програмну реалізацію алгоритму обчислення дескриптора кольору описаного в підрозділі 2.3. Відкриємо новий файл у редакторі, назвемо colordescriptor.py

```
# import the necessary packages
import numpy as np
import cv2
```

```
class ColorDescriptor:
   def __init__(self, bins):
     # store the number of bins for the 3D histogram
     self.bins = bins

   def describe(self, image):
     # convert the image to the HSV color space and initialize
     # the features used to quantify the image
     image = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
     features = []

     # grab the dimensions and compute the center of the image
     (h, w) = image.shape[:2]
     (cX, cY) = (int(w * 0.5), int(h * 0.5))
```

В стрічці 5 Ми визначаємо клас ColorDescriptor. Цей клас інкапсулює усю необхідну логіку, щоб будувати 3D HSV гістограми кольору зображень. Метод Init__ приймає тільки єдиний аргумент, кількість відліків для колірної гістограми. В стрічці 10 визначаємо метод describe. Цей метод вимагає зображення яке ми хочемо описати. Всередині методу describe, ми перетворимо з зображення з RGB кольору (точніше з колірного простору BGR що є характерним для OpenCV) в колірний простір HSV. Завершуємо метод ініціалізацією списку ознак для представлення зображення.

В стрічках 17 і 18 обчислено виміри зображення і обчислено координати центру center(x, y).

Замість обчислення 3D HSV колірної гістограми для цілого зображення, ми обчислимо її для різних областей зображення. Використання гістограм на основі областей замість глобальної гістограми дозволяє нам імітувати просторову локальність розподілу кольорів. Користуючись глобальною гістограмою, ми були б не в змозі визначити, де в зображенні розміщені області різного кольору. Ми знали б тільки, що існує деякий відсоток різних кольорів. Щоб вирішити цю проблему, ми можемо обчислити кольорові гістограми в областях зображення (рисунок 3.9).

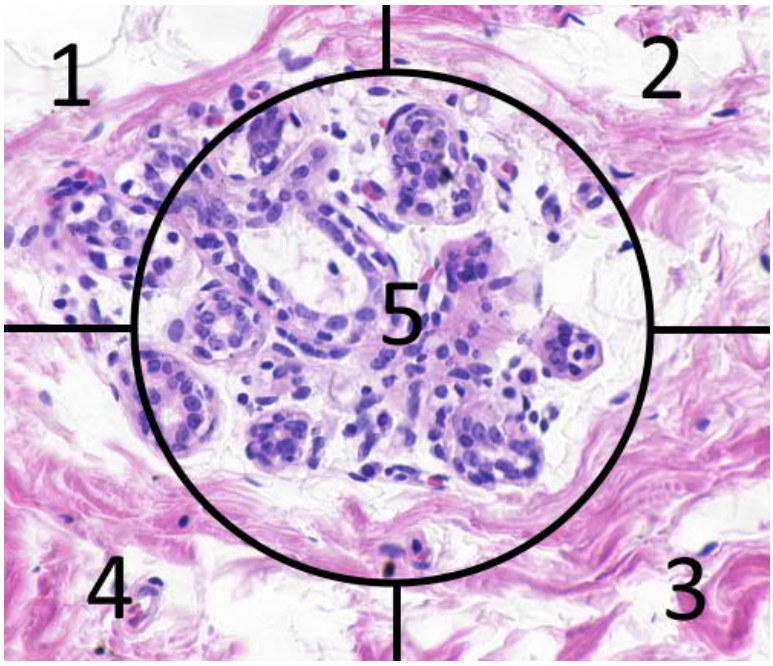


Рисунок 3.9 - Приклад ділення зображення на 5 різних сегментів

Для дескриптора зображення, ми збираємося ділити зображення на п'ять різних областей:(1) верхній лівий кут,(2) верхній правий кут,(3) нижній правий кут,(4) лівий нижній кут, і (5) центр зображення. Використовуючи ці області, ми зможемо реалізувати просту локалізацію. Приведемо код колірного дескриптора на основі областей:

```
    # розділити зображення на чотири прямокутники / сегменти (верхній лівий,
    # верхній правий, правий нижній, лівий нижній)
    segments = [(0, cX, 0, cY), (cX, w, 0, cY), (cX, w, cY, h),
    (0, cX, cY, h)]

    # сконструювати еліптичну маску, що представляє центр
    # зображення
    (axesX, axesY) = (int(w * 0.75) / 2, int(h * 0.75) / 2)
    ellipMask = np.zeros(image.shape[:2], dtype = "uint8")
    cv2.ellipse(ellipMask, (cX, cY), (axesX, axesY), 0, 0, 360,
255, -1)

    # цикл над сегментами
    for (startX, endX, startY, endY) in segments:
```

```
        # сконструювати маску для кожного кута зображення,
віднімаючи
        # еліптичний центр від цього
        cornerMask = np.zeros(image.shape[:2], dtype = "uint8")
        cv2.rectangle(cornerMask, (startX, startY), (endX, endY),
255, -1)
        cornerMask = cv2.subtract(cornerMask, ellipMask)

        # обчислити колірну гістограму зображення, потім обновити
        # вектор ознак
        hist = self.histogram(image, cornerMask)
    f  eatures.extend(hist)

    # обчислити гістограму кольорів еліптичної області
    # update the feature vector
    hist = self.histogram(image, ellipMask)
    features.extend(hist)

    # повернути вектор ознак
    return features
```

В стрічках 22 і 23 визначено індекси верхнього лівого, верхнього правого, нижнього правого і нижнього лівого сегменту, відповідно.

Тепер треба побудувати еліпс, щоб представити центральну частину зображення. Ми зробимо це визначивши радіус еліпса, який складає 75% від ширини і висоти зображення в стрічці 27. Далі ініціалізуємо порожнє зображення (наповнене нулями, щоб представити чорний фон) з тими ж розмірами які і вихідне зображення в стрічці 28. Нарешті, рисуємо фактичний еліпс в стрічці 29, користуючись функцією cv2.ellipse. Ця функція вимагає вісім різних параметрів:

1) ellipMask: зображення, на якому потрібно нарисувати еліпс. Ми користуватимемося концептом "масок";

2) (cX, cY): 2-арний кортеж, що представляє (x, y) координати центру зображення;

3) (axesX, axesY): 2-кортеж, що представляє довжину осей еліпса. В цьому випадку, еліпс матиме 75% ширини і висоти зображення;

4) 0: кут повороту еліпса. В цьому випадку, ніякого обертання не потрібно і ми залишаємо значення 0 градусів;

5) 0: стартовий кут для побудови еліпса;

6) 360: кінцевий кут для побудови еліпса. Дивлячись на попередній параметр, ми вказуємо, що еліпс побудується від 0 до 360 градусів (повне "коло");

7) 255: колір еліпса. Значення 255 вказує "білий". Еліпс буде нарисовано білим на чорному фоні;

8) 1: розмір границі еліпса. Позитивне ціле значення *r* вказує товщину границі *r* пікселів. Вказання негативного значення для *r* нарисує еліпс "заповненим".

Опрацьовуємо кожен з кутовий сегмент індивідуально, видаляючи центр еліпса з прямокутника на кожній ітерації. Причина в тому, що нам потрібна маска, для передачі в функцію гістограми OpenCV, котра вказує де саме побудувати кольорову гістограму. Наша мета - описати кожен з цих сегментів індивідуально. Найефективніший шлях представлення кожного з цих сегментів - користуватися маскою. Тільки для координат (x, y) в зображенні, які мають відповідне розташування із координатами (x, y) всередині маски з білим (255) значенням пікселя, буде обчислено гістограму. Якщо значення пікселя для координати (x, y) в масці має значення чорного кольору (0), піксель буде проігноровано.

В стрічці 41 для кожного з сегментів ми викликаємо процедуру гістограми. Обчислюємо колірну гістограму із зображення, що є першим аргументом, і маски, відповідної області, котра є другим аргументом.

В стрічках 46 і 47 будується колірна гістограма для центральної області (еліпс) і оновлюється список ознак. В стрічці 50 повертається вектор ознак до викликаючої функції.

Метод побудови гістограм наступний.

```
def histogram(self, image, mask):
# extract a 3D color histogram from the masked region of the
# image, using the supplied number of bins per channel;
# нормалізувати гістограму
hist = cv2.calcHist([image], [0, 1, 2], mask, self.bins,
[0, 180, 0, 256, 0, 256])
hist = cv2.normalize(hist).flatten()

# повернути гістограму
return hist
```

Метод гістограми вимагає два значення: перше - зображення, яке ми хочемо описати, і друге - маска, яка представляє регіон зображення, який ми хочемо описати. Обчислення гістограми області-маски зображення показано в стрічках 56 і 57 шляхом виклику cv2.calcHist, і передачею числа відліків з конструктора класу. В стрічці 58 колірна гістограма нормалізується, щоб отримати інваріантність до масштабу. Це означає, що, якщо ми обчислили колірну гістограму для двох ідентичних зображень, одне з яких на 50% більше, ніж друге, наші кольорові гістограми будуть приблизно ідентичні. Нарешті, нормалізована 3D HSV гістограма повертається викликаючій функції в стрічці 61.

Виділення ознак із набору даних

Тепер, коли ми визначили дескриптор зображення, ми можемо піти далі, і виділити ознаки (тобто кольорові гістограми і коефіцієнт спотворення із розділу 2) із кожного зображення в нашому наборі даних. Процес обчислення ознак і зберігання їх в постійному сховищі зазвичай називають "індексування".

Створимо файл index.py для кодування процедури індексувати набору даних.

```
# імпорт потрібних бібліотек
from pyimagesearch.colordescriptor import ColorDescriptor
import argparse
```

```
import glob
import cv2

# парсер аргументів і процес парсингу аргументів
ap = argparse.ArgumentParser()
ap.add_argument("-d", "--dataset", required = True,
   help = "Path to the directory that contains the images to
be indexed")
ap.add_argument("-i", "--index", required = True,
   help = "Path to where the computed index will be stored")
args = vars(ap.parse_args())

# initialize the color descriptor
cd = ColorDescriptor((8, 12, 3))
```

Ми розпочнемо з імпортування пакетів, які нам будуть потрібні. Нам буде також потрібний argparse для граматичного аналізу аргументів командного рядка, glob для захоплення шляхів зображень файлів, і cv2 для функцій OpenCV.

Граматичний аналіз аргументів командного рядка наведено в стрічках 7-13. Нам будуть потрібні два вимикачі, --dataset, який є шляхом до нашої директорії з фото, і -- index , для вихідного CSV файлу, котрий містить ім'я файлу зображення і ознаки, що асоціюються з кожним зображенням.

Ініціалізуємо ColorDescriptor в стрічці 16, користуючись 8 відліками для відтінку, 12 відліками для насиченості, і 3 відліками для значення.

Тепер, коли все ініціалізовано, ми можемо виділити ознаки з нашого набору даних :

```
# open the output index file for writing
output = open(args["index"], "w")

# use glob to grab the image paths and loop over them
for imagePath in glob.glob(args["dataset"] + "/*.png"):
   # extract the image ID (i.e. the unique filename) from the
image
   # path and load the image itself
   imageID = imagePath[imagePath.rfind("/") + 1:]
   image = cv2.imread(imagePath)
```

```
        # describe the image
        features = cd.describe(image)

        # write the features to file
        features = [str(f) for f in features]
        output.write("%s,%s\n" % (imageID, ",".join(features)))

# закрити файл індексу
output.close()
```

В стрічці 19 відкриваємо вихідний файл для запису, потім створюємо цикл по усіх зображеннях в нашому наборі даних в Стрічці 22.

Для кожного із зображень ми створюємо imageID, який є просто ім'ям файлу зображення. В нашій базі зображень усі імена файлів унікальні. Завантажимо зображення з диску в Стрічці 26.

Коли зображення завантажене застосуємо наш дескриптор зображення і обчислимо його ознаки в стрічці 29. Метод describe класу ColorDescriptor повертає список дробових чисел, для представлення і кількісного опису зображення.

Цей список чисел, або вектор ознак містить представлення для кожної з 5 частин зображення, описаних вище. Кожну область представляє гістограма з 8 x 12 x 3 = 288 значень. Для 5 частин повний вектор ознак складатиме 5 x 288 = 1440 значень. Отже, колір кожного зображення представлений 1440 значеннями.

В Стрічці 32 і 33 запишемо ім'я файлу зображення і асоційований вектору ознак до файлу.

Щоб проіндексувати наш набір даних фото, відкриваємо командний рядок і виконуємо наступну команду:

```
$ python index.py --dataset dataset --index index.csv
```

Виконання цього сценарію не повинно зайняти більше, ніж декілька секунд. Після закінчення матимемо новий файл, index.csv. Якщо відкрити файл,

користуючись текстовим редактором. побачимо, що для кожного ряду в .csv файлі, за ім'ям файлу слідує список чисел. Ці числа є векторами ознак.

Клас для пошуку

Коли отримано ознаки з набору даних, нам потрібний метод, щоб порівняти ці ознаки по схожості. На Кроці 3 потрібно створити клас, який обчислить метрику схожості між двома зображеннями. Створити новий файл, назвати msearcher.py:

```
# імпорт бібліотек
import numpy as np
import csv

class Searcher:
   def __init__(self, indexPath):
     # store our index path
     self.indexPath = indexPath

   def search(self, queryFeatures, limit = 10):
     # initialize our dictionary of results
     results = {}
```

Імпортуємо NumPy для числової обробки і csv, для граматичного аналізу файлу index.csv . Оголошуємо клас Searcher в стрічці 5. Конструктор для нашого Searcher вимагатиме тільки єдиний аргумент, indexPath, який є шляхом до файлу index.csv на диску. Для того, щоб фактично виконати пошук, ми викличемо метод search в Стрічці 10. Цей метод отримує два параметри:

– queryFeatures обчислюються з зображення-запиту системи CBIR,

– limit максимальне число результатів що повертається.

Ініціалізуємо словник результатів в стрічці 12. Словник - хороший тип даних в цій ситуації, що дозволить користуватися (унікальним) imageID для вхідного зображення як ключ і схожість із запитом як значення.

```
# відкрити фацл індексу для  читання
with open(self.indexPath) as f:
```

```
# initialize the CSV reader
reader = csv.reader(f)
# цикл по стрічках у індексі
for row in reader:
# parse out the image ID and features, then compute the
# chi-squared distance between the features in our index
# and our query features
features = [float(x) for x in row[1:]]
d = self.chi2_distance(features, queryFeatures)
# now that we have the distance between the two feature
# vectors, we can udpate the results dictionary -- the
# key is the current image ID in the index and the
# value is the distance we just computed, representing
# how 'similar' the image in the index is to our query
results[row[0]] = d
# close the reader
f.close()
# sort our results, so that the smaller distances (i.e. the
# more relevant images are at the front of the list)
results = sorted([(v, k) for (k, v) in results.items()])
# повернути (обмежені) результати
return results[:limit]
```

Ми відкриваємо index.csv файл в Стрічці 15, визначаємо хендл для читача CSV в Стрічці 17, а потім запускаємо цикл над кожним рядом файл index.csv в Стрічці 20. Для кожного рядку, ми обчислюємо колірні гістограми для кожного індексованого зображення, а потім порівнюємо із ознаками зображення-запиту використовуючи chi2_distance (стрічка 25).

Словник результатів оновлюється в Стрічці 32, використовуючи унікальне ім'я файлу зображення як ключ і схожість зображення-запиту до індексованого зображення як значення.

Тепер потрібно сортувати словник результатів згідно зі значенням схожості у порядку зростання. Зображення, які мають схожість 0, відносно метрики chi-squared вважатимуться ідентичними. Для цієї метрики значення зростає із зменшення схожості.

Створимо функцію обчислення метрики:

```
def chi2_distance(self, histA, histB, eps = 1e-10):
     # compute the chi-squared distance
```

```
        d = 0.5 * np.sum([((a - b) ** 2) / (a + b + eps)
             for (a, b) in zip(histA, histB)])

        # return the chi-squared distance
        return d
```

Наша функція chi2_distance вимагає два аргументи, дві гістограмами, які ми хочемо порівняти по схожості. Необов'язковий параметр eps використовується, щоб запобігти помилкам поділу на нуль. Взагалі, різниця між великою і малою кількістю відліків менш важлива і має бути зважена, як це і зроблено в функції відстані Хі квадрат.

Виконання пошуку.

Виконання фактичного пошуку - найлегша частина. Фактично, це процедура, яка імпортує усі пакети і пов’язує один з одним, щоб побудувати Content-Based Image Retrieval System. Останній файл назвемо search.py,

```
# import the necessary packages
from pyimagesearch.colordescriptor import ColorDescriptor
from pyimagesearch.searcher import Searcher
import argparse
import cv2

# побудова парсера аргументів та сам розбір
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--index", required = True,
   help = " Шлях зберігання обчисленого індексу")
ap.add_argument("-q", "--query", required = True,
   help = "Path to the query image")
ap.add_argument("-r", "--result-path", required = True,
   help = "Path to the result path")
args = vars(ap.parse_args())

# ініціалізувати дескриптор зображення
cd = ColorDescriptor((8, 12, 3))
```

Спершу імпортуємо необхідні пакети. Імпортуємо ColorDescriptor, щоб обчислити ознаки з зображення-запиту. Імпортуємо Searcher, для виконання

фактичного пошуку. Далі аналізуємо аргументи командного рядка в стрічках 8-15. Нам буде потрібний --index, що є шляхом файлу індекс.csv.

Нам буде також потрібний --query, який є шляхом до нашого зображення-запиту. Це зображення буде порівняно з кожним зображенням в нашому індексі. Ціль знайти зображення в індексі, що подібні до зображення-запиту.

Ключ --result-path необхідний, щоб показати фактичні зображення результату до користувачу.

Ініціалізуємо дескриптор зображення в стрічці 18, користуючись точно тими ж параметрами, як і на етапі індексування. Це гарантує, що зображення описуються в однаковій формі і отже порівнянні. Скрипт, щоб виконувати фактичний пошук:

```
# завантажити зображення-запит і описати його
query = cv2.imread(args["query"])
features = cd.describe(query)

# виконати пошук
searcher = Searcher(args["index"])
results = searcher.search(features)

# відобразти на екран запит
cv2.imshow("Query", query)

# цикл по результатах
for (score, resultID) in results:
   # завантажити зображення-результат і відобразити його
   result = cv2.imread(args["result_path"] + "/" + resultID)
   cv2.imshow("Результат", result)
   cv2.waitKey(0)
```

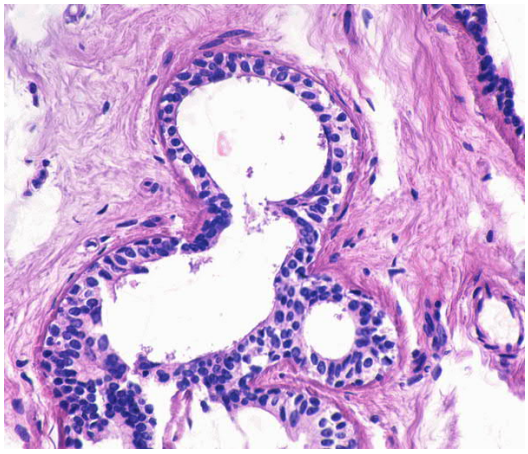
Ми завантажуємо свій запит-зображення з диску в Стрічці 21 і отримуємо ознаки в стрічці 22. В стрічках 25 і 26 виконується пошук, на основі ознак із зображення-запиту, і повертається список ранжованих результатів. Зображення-запит показуємо користувачу в стрічці 29. А потім створюємо цикл по нашими результатам пошуку в стрічках 32-36 і показуємо їх на екрані.

Тестовий запуск системи. Відкриємо термінал, перейдемо до директорії, де розміщено код, і запустимо наступну команду. Зображення-запит 108100.png.

```
$ python search.py --index index.csv --query queries/108100.png --result-path dataset
```

На рисунку 3.10 показано результати тестового запуску системи. Як видно система видає релевантні результати. В наступному підрозділі проведемо оцінку ефективності пошуку.

Запит



Результати пошуку

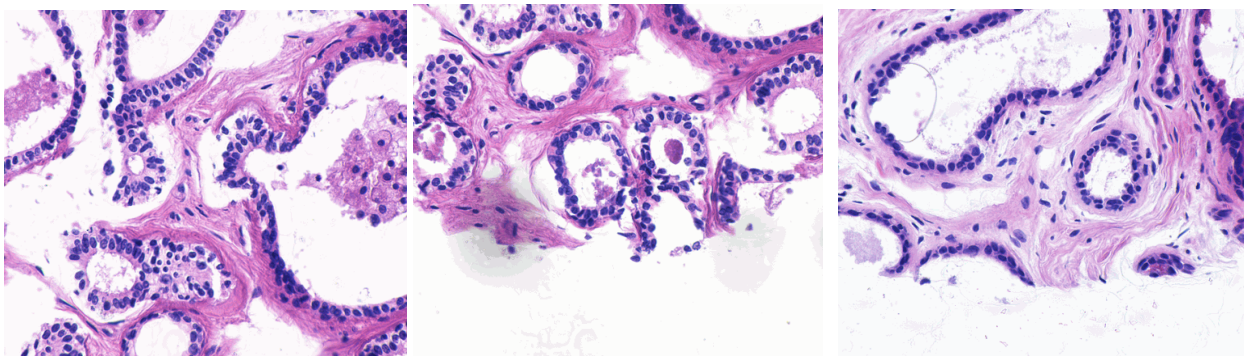


Рисунок 3.10 – Тестовий запуск проектованої системи

3.3 Експериментальні результати

Для перевірки розроблених алгоритмів проведемо наступний експеримент. У якості експериментальної бази зображень візьмемо публічно доступну базу зображень раку молочної залози Breast Cancer Dataset. Відібрані зображення можуть розглядаються як навчальна множина двома класами (кластерами) семантично близьких зображень: доброякісні новоутворення (benign), злоякісні новоутворення (malignant).

Для експерименту було відібрано п'ять значень параметра α: 0, 0.25, 0.5, 0.75 і 1, що дає п'ять різних змішаних метрик (2.5).

Експериментальний стенд являє собою інтерактивну систему, ціль роботи якої - з'ясувати суб'єктивну думку спеціалістів мендиків про подобу зображень. Ліворуч відображається випадковим чином обране зображення- зразок, а основну частину екрана займає результат пошуку, отриманий із застосуванням змішаної метрики з випадково обраним параметром α (з п'яти можливих значень).

Учасникові експерименту пропонується оцінити подібність зразка й кожного з отриманих зображень по трибальній шкалі: «зовсім не схожі», «трохи схожі», «схожі». У якості результату оцінки зберігається пара порівнюваних зображень і сама оцінка (відповідно, 0, 0.5, 1). Після того, як кластери були сформовані, був здійснений пошук із застосуванням змішаних метрик, використовуючи кожне із зображень кластерів як запитів.

Результати обчислення метрик наведені в таблиці 3.1. Для кожного кластеру зображень.

Таблиця 3.1 – Результати обчислення змішаної метрики подібності

	Значення α				
Кластери зображень	0	0.25	0.5	0.75	1
Злоякісні новоутворення	0,15	0,6	0,48	0,28	0,15
Доброякісні новоутворення	0,135	0,68	0,33	0,21	0,13

Залежності середньої повноти від значення параметра α, що визначає змішану метрику, представлені для кожного із кластерів на рисунку 3.11

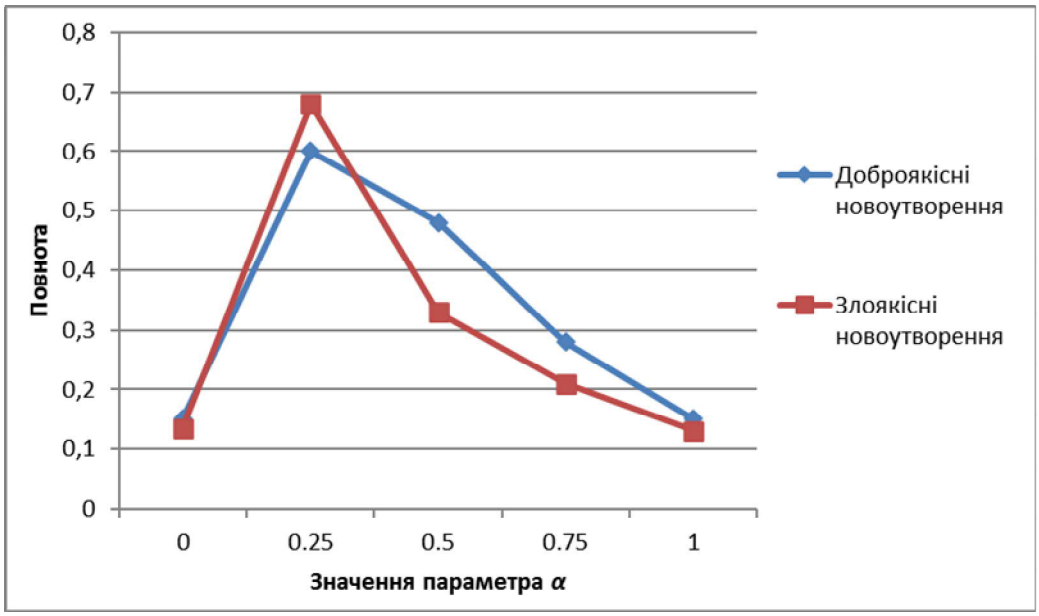


Рисунок 3.11 - Графік залежності середньої повноти від значення коефіцієнта α

Для кожного зображення з тестової множини були обчислені колірні й текстурні ознаки. Для кожної пари зображень були обчислені відстані в обох просторах ознак з використанням відповідних метрик. Далі отримані відстані нормовані.

3.4 Висновки до розділу

Експерименти показали, що для різних класів зображень можна визначити відносну значимість колірної ознаки та дескриптора структурних змін при здійсненні пошуку по елементах класу. У цілому, як і слід було сподіватися, дескриптор структурних змін виявляється більш значимим для більшості зображень.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи одержані наступні результати:

1. Проаналізовано характеристики біомедичних зображень та їх ознаки.Проаналізовано напрямки досліджень в області пошуку зображень. Розглянуто підходи до побудови векторів ознак. Проведено аналіз числових ознак гістологічних зображень.

2. Розроблено алгоритм обчислення колірних і текстурних дескрипторів Запропоновано колірну ознаку, котра є розширенням класичних колірних гістограм, що дозволило врахувати просторове розташування кольорів. Для порівняння дескрипторів була запропонована функція подібності.

3. Розроблено алгоритм пошуку зображень по вмісту на основі обчислення оцінки подібності зображень запиту. Оцінка подібності є лінійною комбінацією колірної метрики і функцій відстані текстурних ознак з певними вагами.

4. Здійснено програмну реалізацію компонентів системи пошуку по вмісту на мові Python.

5. Здійснено експериментальне дослідження розроблених алгоритмів, що дало змогу підвищити ефективність пошуку гістологічних зображень по вмісту і досягти рівня повноти пошуку не менше 0,6.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Євтушок М.А., Любий Ю.І. Алгоритми пошуку біомедичних зображень по вмісту. III Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 26 листопада 2020, Тернопіль, Україна. Тернопіль: ЗУНУ, 2020. с.57

2. Любий Ю.І., Євтушок М.А Процедура сегментації в системах пошуку зображень. III Науково-практична конференція молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі». 26 листопада 2020, Тернопіль, Україна. Тернопіль: ЗУНУ, 2020. с.58

3. Kingma, D. P. Adam: a Method for Stochastic Optimization / Kingma, D. P., Ba, J.L. // International Conference on Learning Representations, May 7-9, 2015, San-Diego, USA.

4. Lofti M., Solimani A., Dargazany A., et al. Combining wavelet transforms and neural networks for image classification // Proc. of the 41st Southeastern symp. on system theory. Tennessee (USA), Mar. 15-17, 2009. IEEE SSST, 2009. P. 44-48.

5. Niblack W., Barber R., Equitz W., et al. The QBIC project: querying images by content using color, texture, and shape // Proc. of the Intern, conf. on storage and retrieval for image and video databases. Bellingham, Washington, USA, Febr. 1993. IS&T/SPIE, SPIE, 1993. P. 173-187.

6. Андерсон К., Минаси М. Локальные сети. -М.: Век+, 2007.

7. Девянин П.Н., Михальский О.О., Правиков Д.И., ЩербаковА.Ю. Теоретические основи компьютерной безопасности. М.:Радио исвязь,2006.192с.

8. Высокопроизводительные сети. Энциклопедия пользователя. Марк А. Спортак и др.; перев. с англ. - Киев, ДиаСофт, 2004.

9. Жидецький В.Ц. Охорона праці користувачів комп’ютерів. Навчальний посібник. Вид. 2-ге., доп. Львів.: Афіша, 2000. 176с.

10. Средства связи для «последней мили». Денисьев и Мирошников, -Эко-Трендз, 2000.

11. Основы построения сетей. Учебное руководство для специалистов MCSE (+CD-ROM). Дж. Челлис, Ч. Перкинс, М. Стриб; перевод с англ. - Лори, 2002.

12. Bourzac K. Software: The computer will see you now / K. Bouczak / - Nature, 2013, 502(7473): S92-S94.

13. Гістологічні методи дослідження URL - http://ukrbukva.net/page,3,94380-Gistologicheskie-metody-issledovaniya.html.

14. Предмет гістології [Електронний ресурс]. - Режим доступу- https://studfiles.net/preview/5258227.

15. Методы гистологического анализа URL - http://worldofscience.ru/biologija/6481-metody-gistologicheskogo-analiza.html.

16. Автандилов Г.Г. Основы количественной паталлогической анатомии / Г.Г. Автандилов М.: Медицина, 2002. 240 с.

17. Теорія розпізнавання образів URL - https ://uk.wikipedia.org/wiki/Теорія_розпізнавання_образів

18. Вапник В.Н. Теория распознавания образов / В.Н. Вапник, А.Я. Червоненкис. М.: Наука, 1974. 416 с.

19. Васильев В.И. Распознающие системы. Справочник. 2-е изд. К.: Наукова думка, 1983. — 424 с.

20. Классификация данных методом опорных векторов URL - https://habrahabr.ru/post/105220/

21. Метод опорних векторів URL - https ://uk. wikipedia. o^Мікі/метод_опорних_векторів.

22. Алгоритм AdaBoost URL - http://www.machinelearning.ru/wiki/index.php?title=AdaBoost

23. Топ-10 data mining-алгоритмов простым языком URL - https://habrahabr.ru/company/itinvest/blog/262155/

24. Breiman, Leo (2001). Random Forests. Machine Learning 45 (1). с. 5-32.

25. Каллан Р. Основные концепции нейронных сетей. The Essence of Neural Networks First Edition. М.: Вильямс, 2001. 288 с.

26. Что такое искусственные нейронные сети? URL: https://habrahabr.ru/post/134998.

27. Элементы нейронных сетей. URL: https://www. intuit. ru/studies/courses/6/6/lecture/ 178?page=2

28. Kerlirzin, P. Robustness in multilayer perceptrons / P. Kerlirzin, F. Vallet / Neural Computation, 1993, vo1. 5, p. 473 - 482.

29. Mead C.A. Analog VLSI and Neural Systems, Reading / C.A. Mead / - MA: Addison-Wesley, 1989.

30. Black I.B. Information in the Brain: A Molecular Perspective / I.B. Black / - Cambridge, MA: MIT Press, 1991.

31. McCulloch W.S. A logical calculus of the ideas immanent in nervous activity / W.S. McCulloch, W. Pitts / Bulletin of Mathematical Biophysics, 1943.-№ 5.-P.115- 133.

32. MATLAB - The Language of Technical Computing URL: http://www.mathworks.com/products/matlab.

33. MATLAB - Wikipedia URL: https ://rn.wikipedia. org/wiki/MATLAB.

34. Neural Network Toolbox - Математика URL: http ://matlab. exponenta. ru/neuralnetwork/

35. Введение в Neural Network Toolbox URL: http ://matlab. exponenta. ru/neuralnetwork/book1/

36. The Premier Neural Network Software URL: http://www. neurosolutions.com/

37. Deep Learning for Java URL: https ://deeplearning4j .org/

38. Convolutional Neural Networks (LeNet) URL: http://deeplearning.net/tutorial/lenet.html

39. van den Oord Deep content-based music recommendation / van den Oord Deep, Aaron; Dieleman, Sander; Schrauwen, B. Burges, L. Bottou, M. Welling, Z. Ghahramani, K. Weinberger / Curran Associates, Inc. c. 2643-2651.

40. Korekado A Convolutional Neural Network VLSI for Image Recognition Using Merged/Mixed Analog-Digital Architecture. Knowledge-Based Intelligent Information and Engineering Systems. p. 169-176.

41. Krizhevsky Alex. ImageNet Classification with Deep Convolutional Neural Networks / A. Krizhevsky, I. Sutskever, E. Hinton / Advances in Neural Information Processing Systems, 2012. №25. P.120-129.

42. Применение нейросетей в распознавании изображений URL: https://geektimes.ru/post/74326/

43. Классификатор изображений на основе свёрточной сети URL: http://mechanoid.kiev.ua/ml-lenet.html

44. Convolutional Neural Networks (CNNs / ConvNets) URL: http://cs231n.github.io/convolutional-networks

45. Y. LeCun Efficient BackProp / Y. LeCun, L. Bottou, G. Orr and K. Muller / Orr, G. and Muller K. (Eds), Neural Networks: Tricks of the trade, Springer, 1998

46. Алгоритм обратного распространения ошибки URL: http ://www. aiportal.ru/articles/neural-networks/back-propagation.html

47. Deep learning for complete beginners: convolutional neural networks with keras URL: https://cambridgespark.com/content /tutorials/convolutional-neural-networks-with-keras/index.html

48. The 9 Deep Learning Papers You Need To Know About URL: https://adeshpande3.github.io/adeshpande3.github.io/The- 9-Deep-Learning-Papers-You-Need-To-Know-About.html

49. Matthew D. Zeiler Visualizing and Understanding Convolutional Networks / D. Zeiler, R. Fergus / arXiv:1311.2901v3. - 2013

50. Christian Szegedy Going Deeper with Convolutions / Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, Andrew Rabinovich / THE COMPUTER VISION FOUNDATION. - 2015, 10p.

51. Ross Girshick Rich feature hierarchies for accurate object detection and semantic segmentation / Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik / arXiv:1311.2524v5. - 2014, 21p.

52. Ross Girshick Fast R-CNN / Ross Girshick // arXiv:1504.08083v2. - 2015, 9p.

53. Shaoqing Ren Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks / Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun // arXiv:1506.01497v3 [cs.CV]. - 2016, 14p.

54. Обучение нейронной сети URL: http://neuronus.com/theory/240-algoritmy-obucheniya-iskusstvennykh-nejronnykh-setej

55. Алгоритм обучения многослойной нейронной сети методом обратного распространения ошибки (Backpropagation) URL: https: //habrahabr. ru/post/198268.

56. База даних цитологічних та гістологічних зображень ауто- та ксеногенних тканин / Березький О.М., Мельник Г.М., Дацко Т.В., Вербовий С.О. // Науковий вісник національного лісотехнічного університету України: збірник науково-технічних праць. - Львів: РВВ НЛТУ України. - 2014.- Вип. 24.10. - С.338-345.

57. 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"/ Під ред. О.М. Березького. Тернопіль:ЗУНУ,2020.32 с.

58. Гураль І.В., Дубчак Л.О. Методичні вказівки до оформлення курсових проектів, звітів про проходження практики, випускних кваліфікаційних робіт для студентів спеціальності «Комп’ютерна інженерія»/Під ред. О.М. Березького. Тернопіль: ТНЕУ, 2019. 33 с.