МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Західноукраїнський національний університет
Навчально-науковий інститут новітніх освітніх технологій
Кафедра комп'ютерної інженерії

Винницька Тетяна Петрівна

Алгоритми детекції блоків тексту при оптичному розпізнаванні тексту / Algorithms for detection of text blocks in optical text recognition

спеціальність: 123 - Комп'ютерна інженерія
освітньо-професійна програма - Комп'ютерна інженерія
Кваліфікаційна робота

Виконав студент групи КІзм-21
Т.П. Винницька
_____

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

Кваліфікаційну роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____Л. О. Дубчак

Тернопіль – 2023

РЕЗЮМЕ

Кваліфікаційна робота на тему «Алгоритми детекції блоків тексту при оптичному розпізнаванні тексту» зі спеціальності 123 «Комп'ютерна інженерія» освітнього ступеня «магістр» написана обсягом 70 сторінок і містить 19 ілюстрацій, 6 таблиць, 3 додатки та 54 джерела за переліком посилань.

Метою кваліфікаційної роботи є розробка алгоритму детекції блоків тексту на зображеннях на основі текстурних ознак.

Методи дослідження: методи системного аналізу, математичного моделювання.

Наукова новизна одержаних результатів. Розроблено алгоритми детекції блоків тексту з різними шрифтами та області «не тексту» на основі локальних текстурних ознак, що дозволило підвищити точність оптичного розпізнавання тексту.

Практичне значення полягає в розробці та впровадженні покращеного алгоритму детекції блоків тексту для оптичного розпізнавання тексту, що відповідає сучасним вимогам щодо точності та ефективності обробки зображень із текстовим контентом. Здійснено програмну реалізацію та експериментальне дослідження розроблених алгоритмів. Досягнуто значне покращення точності розпізнавання тексту – від 71% до 173% в залежності від класу документу.

КЛЮЧОВІ СЛОВА: ОПТИЧНЕ РОЗПІЗНАВАННЯ СИМВОЛІВ, ШРИФТ, ТЕКСТУРА, КОНТУР.

RESUME

Qualification work on «Algorithms for detection of text blocks in optical text recognition» in the specialty 123 "Computer Engineering" educational degree "Master" is written in 70 pages and contains 19 illustrations, 6 tables, 3 appendices and 54 sources by reference.

The purpose of the qualification work is to develop an algorithm for detecting text blocks in images based on texture features.

Research methods: methods of system analysis, mathematical modeling.

Scientific novelty of the results. The algorithms for detecting text blocks with different fonts and "non-text" areas based on local texture features have been developed, which has improved the accuracy of optical text recognition.

The practical significance lies in the development and implementation of an improved text block detection algorithm for optical text recognition that meets modern requirements for the accuracy and efficiency of image processing with textual content. The software implementation and experimental study of the developed algorithms were carried out. A significant improvement in text recognition accuracy has been achieved - from 71% to 173%, depending on the document class.

KEYWORDS: OPTICAL CHARACTER RECOGNITION, FONT, TEXTURE, CONTOUR.

ЗМІСТ

ВСТУП

Оптичне розпізнавання символів (OCR) у складних документах, таких як книги, журнали, газети та формуляри, є надзвичайно складним завданням, яке включає ряд викликів. Ці документи часто мають неоднорідну структуру: колонки тексту, які можуть бути вирівняні горизонтально або вертикально, різноманітні шрифти та розміри тексту, вставлені зображення, рамки, відомості у вигляді таблиць та інші елементи, що вимагають специфічної обробки. Проблеми розпізнавання:

1. Різноманітність макетів. Наприклад, текст може бути розташований у кількох колонках, які потрібно правильно розпізнавати та читати в правильному порядку.

2. Різноманітність шрифтів, від курсиву до жирного тексту, може спотворити символи та ускладнити їх ідентифікацію.

3. Змішаний вміст. Наявність зображень, таблиць та графічних елементів може ввести в оману алгоритми OCR, які можуть помилково інтерпретувати ці елементи як текст.

4. Пошкоджені або застарілі документи можуть мати плями, затертості або зів'ялені частини, що ускладнює розпізнавання.

5. Документи можуть включати вертикальний текст або текст, розташований під кутом, що вимагає складного аналізу орієнтації.

Існує постійна потреба у вдосконаленні алгоритмів, щоб вирішувати вказані вище проблеми. Зокрема покращена сегментація. Сучасні алгоритми мають здатність більш точно розрізняти блоки тексту, зображення та інші елементи на сторінці, що дозволяє покращити точність розпізнавання.

Об'єкт дослідження - Процес виділення та розпізнавання блоків тексту на зображеннях.

Предмет дослідження - алгоритми текстурного та контурного аналізу зображенні.

Метою роботи є розробка алгоритму детекції блоків тексту на зображеннях на основі текстурних ознак. Головною метою є покращення точності та ефективності процесу детекції, зокрема для зображень із складною структурою та різними шрифтами.

Для досягнення вищенаведеної мети, в рамках дослідження передбачаються наступні завдання:

- здійснити аналітичний огляд методів детекції блоків тексту та алгоритмів текстурного аналізу зображень;
- розробити алгоритм обчислення текстурних ознак;
- розробити алгоритм детекції блоків тексту;
- здійснити програмну реалізацію розроблених алгоритмів;
- провести тестування розробленого програмного забезпечення.

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

Наукова новизна одержаних результатів. Розроблено алгоритми детекції блоків тексту з різними шрифтами та області «не тексту» на основі локальних текстурних ознак, що дозволило підвищити точність оптичного розпізнавання тексту.

Практичне значення полягає в розробці та впровадженні покращеного алгоритму детекції блоків тексту для оптичного розпізнавання тексту, що відповідає сучасним вимогам щодо точності та ефективності обробки зображень із текстовим контентом.

Апробація роботи. Отримані результати опубліковані в межах VIII Науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі», опубліковано тези доповіді [1,2].

Впровадження результатів ДР. Результати роботи рекомендовано до впровадження на підприємстві (додаток Б).

Кваліфікаційна робота складена із трьох розділів, висновків, списку використаних джерел та додатків.

1 ОСНОВИ ОПТИЧНОГО РОЗПІЗНАВАННЯ ТЕКСТУ

1.1 Поняття та цілі оптичного розпізнавання тексту

Оптичне розпізнавання тексту (OCR) - це процес автоматичного визначення та розпізнавання текстової інформації на зображеннях або сканах документів. Основною метою OCR є перетворення тексту із фізичного носія (наприклад, паперового документа або фотографії) у цифровий текст, який можна зберегти, редагувати та обробляти за допомогою комп'ютера.

OCR знайшло широке застосування в різних сферах:

1. У сфері бізнесу OCR дозволяє автоматизувати обробку документів, включаючи розпізнавання інформації з рахунків, контрактів, податкових декларацій і інших документів. Це зменшує ризик помилок та підвищує продуктивність;

2. В медичній галузі OCR використовується для розпізнавання рецептів, медичних звітів, лікарських карток і т.д. Це сприяє швидкій обробці медичної інформації та підвищує точність діагностики;

3. У сфері фінансів OCR використовується для розпізнавання інформації з фінансових документів, таких як чеки, рахунки та виписки з банку. Це допомагає клієнтам банків та компаній здійснювати швидкі та точні фінансові операції;

4. У сфері архівування та пошуку OCR дозволяє створювати цифрові копії паперових документів та виконувати текстовий пошук у великих архівах документів, що значно спрощує роботу з інформацією.

Попри великий успіх OCR, він також стикається з деякими викликами та проблемами:

1. Рукописний текст: OCR має проблеми з розпізнаванням рукописного тексту, оскільки рукопис може бути дуже різним та неоднорідним.

2. Складні макети документів: Деякі документи мають складні макети, наприклад, газети або брошури, що може призвести до неправильної детекції та розпізнавання тексту.

3. Низька якість зображень: Якість зображень може суттєво впливати на точність OCR. Шум, розмивання, низька роздільна здатність можуть призвести до помилок.

4. Розпізнавання мов: Деякі OCR системи можуть мати обмеження у розпізнаванні тексту на різних мовах, особливо на мовах з складною графікою.

Для реалізації OCR використовуються різні засоби:

1. Оптичні сканери: Вони перетворюють фізичний документ у цифровий зображення.

2. Програмне забезпечення: Для розпізнавання тексту на зображеннях використовуються спеціальні програми та бібліотеки, такі як Tesseract, ABBYY FineReader, Adobe Acrobat і багато інших.

3. Мобільні додатки: Деякі мобільні додатки дозволяють користувачам здійснювати OCR прямо зі смартфонів або планшетів.

Мобільні додатки для OCR стали дуже популярними серед користувачів (рисунок 1.1). Вони надають зручний спосіб для швидкого та простого розпізнавання тексту на зображеннях. Деякі з популярних мобільних додатків для OCR включають в себе:

1. Додаток CamScanner дозволяє користувачам створювати скани документів і розпізнавати текст.

2. Microsoft Office Lens дозволяє користувачам створювати скани документів, а також розпізнавати текст та інтегрувати його з Microsoft Office.

3. Додаток Google Keep має функцію розпізнавання тексту на зображеннях та зберігання його у вигляді нотаток.

4. ABBYY FineScanner використовує могутній OCR для створення цифрових копій документів та розпізнавання тексту.

OCR є важливою технологією, яка має великий потенціал у різних сферах. Ця технологія допомагає автоматизувати обробку текстової інформації, зробивши її доступною та оброблюваною для комп'ютерів. Хоча вона стикається з певними викликами, такими як рукописний текст та складні макети документів, постійні дослідження та розробки в галузі OCR допомагають подолати ці обмеження та покращувати точність та ефективність цієї технології.

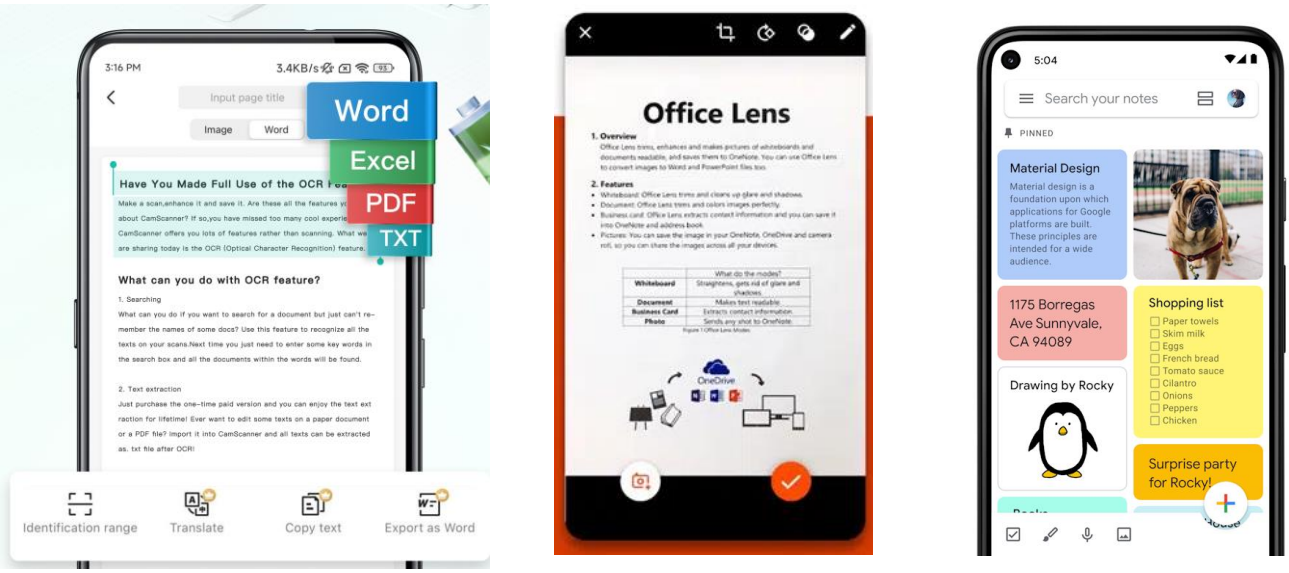


Рисунок 1.1 - Мобільні додатки для OCR

OCR (Optical Character Recognition) – це процес перетворення зображень написаного тексту (манускриптів, друкованих документів, інших) у машиночитаємий формат. Основні етапи роботи OCR системи:

1. Попередня обробка зображення. Зокрема, корекція спотворень, підвищення контрасту, видалення шуму.

2. Бінаризація зображення. Перетворення зображення у чорно-білий формат, де текст являється чорним на білому фоні або навпаки.

3. Сегментація зображення. Розділення зображення на окремі символи чи слова.

4. Розпізнавання символів. На цьому етапі, OCR система намагається ідентифікувати кожен символ на основі його форми.

5. Пост-обробка. Після розпізнавання тексту можливі виправлення, базуючись на синтаксисі і контексті.

6. Виведення результату: Останнім етапом є конвертація розпізнаного тексту в потрібний формат (наприклад, TXT, DOC, PDF).

1.2 Методи детекції блоків тексту

Дослідження [10] презентує новий метод детекції тексту в природних зображеннях. Головні внески дослідження включають:

1. Розробка швидкого і масштабованого механізму для генерації синтетичних зображень тексту на складних фонах: Цей механізм накладає синтетичний текст на існуючі зображення фонів, при цьому враховуючи локальну 3D геометрію сцени. Це дозволяє моделювати реалістичні сценарії, в яких текст візуально інтегрований у складні фони.

2. Тренування повністю конволюційної регресійної мережі (FCRN) на синтетичних зображеннях для детекції тексту: FCRN ефективно виконує детекцію тексту та регресію обмежувальних рамок на всіх локаціях і на декількох масштабах у зображенні. Цей підхід відзначається ефективністю в обробці зображень, дозволяючи одночасно виявляти текст і визначати його розташування.

Автори також обговорюють відношення FCRN до нещодавно запропонованого детектора YOLO, а також інших систем детекції об'єктів з використанням глибокого навчання, що працюють "кінець-в-кінець" (end-to-end). Основним фокусом є порівняння цих підходів та обговорення переваг і недоліків нового методу.

Результатом дослідження є мережа для детекції тексту, яка значно перевершує існуючі методи у виявленні тексту в природних зображеннях. За результатами, на стандартному бенчмарку ICDAR 2013 було досягнуто високого рівня точності (F-міри 84.2%). Крім того, система може обробляти до 15 зображень за секунду на GPU, що свідчить про її високу продуктивність.

Дослідження [12] фокусується на розвитку методу детекції тексту в сценах природного середовища. Хоча попередні підходи до виявлення тексту в сцені вже показали обнадійливі результати на різних бенчмарках, вони часто зазнавали невдач при роботі зі складними сценаріями, навіть при використанні моделей глибоких нейронних мереж. Основна проблема полягала у взаємодії різних етапів і компонентів у пайплайнах. У цій роботі автори пропонують простий, але потужний пайплайн, який забезпечує швидке і точне виявлення тексту в природних сценах. Відмінною рисою цього підходу є безпосереднє прогнозування слів або текстових ліній довільної орієнтації та квадратної форми на повних зображеннях, що усуває непотрібні проміжні кроки (наприклад, агрегування кандидатів і розділення слів), за допомогою єдиної нейронної мережі. Такий

простий пайплайн дозволяє зосередитися на розробці функцій втрати та архітектури нейронної мережі. Експерименти на стандартних наборах даних, включаючи ICDAR 2015, COCO-Text та MSRA-TD500, показують, що запропонований алгоритм значно перевершує існуючі методи з точки зору як точності, так і ефективності. На наборі даних ICDAR 2015 запропонований алгоритм досягає F-оцінки 0.7820 при швидкості 13.2 кадрів в секунду при роздільній здатності 720p. Загалом, це дослідження представляє значний прогрес у галузі детекції тексту в сценах природного середовища, демонструючи ефективність єдиної інтегрованої системи замість складних багатоетапних пайплайнів.

Дослідження [11] представляє "TextBoxes", що є швидким детектором тексту в сценах, який можна навчати "кінець-в-кінець". Основна перевага TextBoxes полягає у здатності виявляти текст у сценах з високою точністю та ефективністю за один прохід через мережу, без необхідності будь-якої післяпроцедури, окрім стандартного методу немаксимального придушення.

Однією з ключових особливостей TextBoxes є його висока швидкість обробки — лише 0.09 секунди на зображення у швидкісній реалізації. Це робить його значно швидшим за конкуруючі методи, а також вищим за них за точністю локалізації тексту. Коли TextBoxes комбінується з системою розпізнавання тексту, він значно перевершує сучасні підходи у завданнях пошуку слів та розпізнавання тексту "від початку до кінця" (end-to-end). У сукупності, це дослідження вносить значний внесок у розвиток швидких і точних технологій для детекції тексту в природних сценах, пропонуючи інтегроване рішення, яке ефективно поєднує виявлення тексту та його подальше розпізнавання.

Детекція текстових блоків на зображеннях є важливою задачею в галузі обробки зображень і комп'ютерного зору. Ця задача включає в себе виявлення і сегментацію областей зображення, що містять текст, і може бути виконана за допомогою різних методів. Основні:

- методи на основі морфологічних операцій;
- методи на основі машинного навчання;
- методи на основі нейронних мереж.

Розглянемо їх детальніше.

1. Методи на основі морфологічних операцій - це інструменти для обробки форми об'єктів на зображенні. Вони включають операції, такі як ерозія, диляція, відкриття, і закриття, які допомагають виділити або видалити певні структури на зображенні. Переваги: простота в реалізації, низька вимога до обчислювальних ресурсів.

Недоліки: невисока точність і надійність, особливо в умовах низької якості зображення або при наявності складних фонів.

Математичний апарат: основний математичний апарат включає теорію морфологічних трансформацій і булеву алгебру.

2. Методи на основі машинного навчання включають в себе використання класичних алгоритмів машинного навчання, таких як опорні векторні машини (SVM), рандомізовані дерева рішень, тощо. Вони навчаються на позначених даних, щоб класифікувати пікселі або області зображення як 'текст' або 'не-текст'. Переваги: більша точність і надійність порівняно з морфологічними методами, здатність адаптуватися до різних типів зображень.

Недоліки: висока залежність від якості та різноманітності тренувальних даних, відносно високі вимоги до обчислювальних ресурсів.

Математичний апарат: статистичне навчання, теорія ймовірності, оптимізаційні методи.

3. Методи на основі нейронних мереж використовують глибокі нейронні мережі, такі як свердловинні (CNN) або рекурентні (RNN) мережі. Вони здатні виявляти складні взаємозв'язки в даних і ефективно виділяти текстові області на зображеннях. Переваги: Висока точність і надійність, гарна здатність до узагальнення, ефективність у великомасштабних застосуваннях.

Недоліки: Великі вимоги до обчислювальних ресурсів і тренувальних даних, складність у реалізації та налагодженні.

Математичний апарат**:** Теорія глибокого навчання, нелінійна оптимізація, алгоритмічні аспекти нейронних мереж.

Переваги і недоліки трьох описаних методів детекції тексту на зображеннях показані в Таблиці 1.1

Таблиця 1.1 - Методи детекції блоків тексту

Метод	Переваги	Недоліки
Морфологічні операції	Простота в реалізації, низька вимога до ресурсів	Невисока точність, проблеми при складних фонах
Машинне навчання	Висока адаптивність, краща точність порівняно з морфологічними методами	Залежність від тренувальних даних, високі вимоги до обчислювальних ресурсів
Нейронні мережі	Висока точність і надійність, ефективність у великомасштабних застосуваннях	Великі вимоги до обчислювальних ресурсів і даних, складність у реалізації

Для порівняння різних методів детекції текстових блоків на зображеннях, важливо врахувати наступні критерії:

1. Точність Детекції**:** Наскільки ефективно метод може виявляти текст на різних типах зображень.

2. Обчислювальна Вимогливість**:** Ресурси та час, необхідні для обробки зображень.

3. Універсальність**:** Здатність методу працювати з різними шрифтами та фонами.

4. Стійкість до помилок: Як метод справляється з шумом, спотвореннями та іншими зовнішніми впливами на зображення.

5. Масштабованість**:** Здатність методу адаптуватися до великих обсягів даних або різних розмірів зображень.

6. Навчання та налаштування**:** Складність налаштувань та потреба в навчанні моделі для методів на основі машинного навчання та нейронних мереж.

1.3 Аналіз зображень на основі текстурних ознак

У більшості випадків природні сцени на великих площах розріджені значущими деталями. В таких областях сцени можна часто описати як вияв повторюючої структури, аналогічної структурі тканини або візерунку на підлозі

з кахель. Безліч прикладів можна навести, коли необхідно визначити межі текстурної області та розмір зерна текстури всередині кожної області. Деякі вчені спробували надати якісне визначення поняття текстури. Наприклад, Пікет запропонував такий варіант: "Текстура використовується для опису двовимірних масивів змін яскравості. Елементи текстури та правила їх просторового розташування можуть бути довільно змінені, за умови, що залишаються незмінними характеристики повторюваності змін яскравості." Хоукінс надав більш детальний опис текстури: "Очевидно, текстура охоплює такі властивості зображення:

1) можливість виділення фрагмента, "рисунок" якого регулярно повторюється всередині області, значно більшої за розмір фрагмента;

2) цей "рисунок" утворюється з елементарних складових частин фрагмента, розміщених в деякому невипадковому порядку;

3) елементарні частини - це приблизно однорідні одиниці, які мають приблизно однакову форму в усій текстурній області."

Текстури можна поділити на два основних класи: штучні та природні. Штучні текстури - це структури, які складаються з графічних знаків, що розміщені на нейтральному фоні. Серед таких знаків можуть бути лінії, точки, зірки, букви та цифри. На рисунку 1.2 показані кілька прикладів штучних текстур.

Природні текстури, як із їхньої назви зрозуміло, представляють собою зображення природних сцен, що містять майже періодичні структури. Прикладами можуть служити фотографії цегляних стін, черепиці на дахах, піску, трави та інших природних об'єктів. На рисунку 1.2 показано приклади природних текстур.

Текстурні характеристики часто описуються за допомогою розміру текстурного зерна. Наприклад, шматок вовняної тканини може бути "грубшим" за текстурою, ніж шовкова тканина, при однакових умовах спостереження. Зрозуміло, що розмір текстурного зерна сам по собі недостатній для точного вимірювання текстури, але його можна використовувати для оцінки напрямку зміни текстурних ознак. Іншими словами, невеликі значення текстурних ознак вказують на дрібну текстуру, тоді як великі значення вказують на велику

текстуру. Важливо враховувати, що текстура є властивістю околиці точки на зображенні. Тому текстурні ознаки залежать від розміру околиці, на якій вони вимірюються. Оскільки текстура - це просторова характеристика, розміри її ознак повинні обмежуватися областями, які мають відносну однорідність. Таким чином, перед тим як проводити аналіз текстури, необхідно визначити межі областей однорідної текстури, спостерігаючи або використовуючи один із методів автоматичної сегментації зображення.

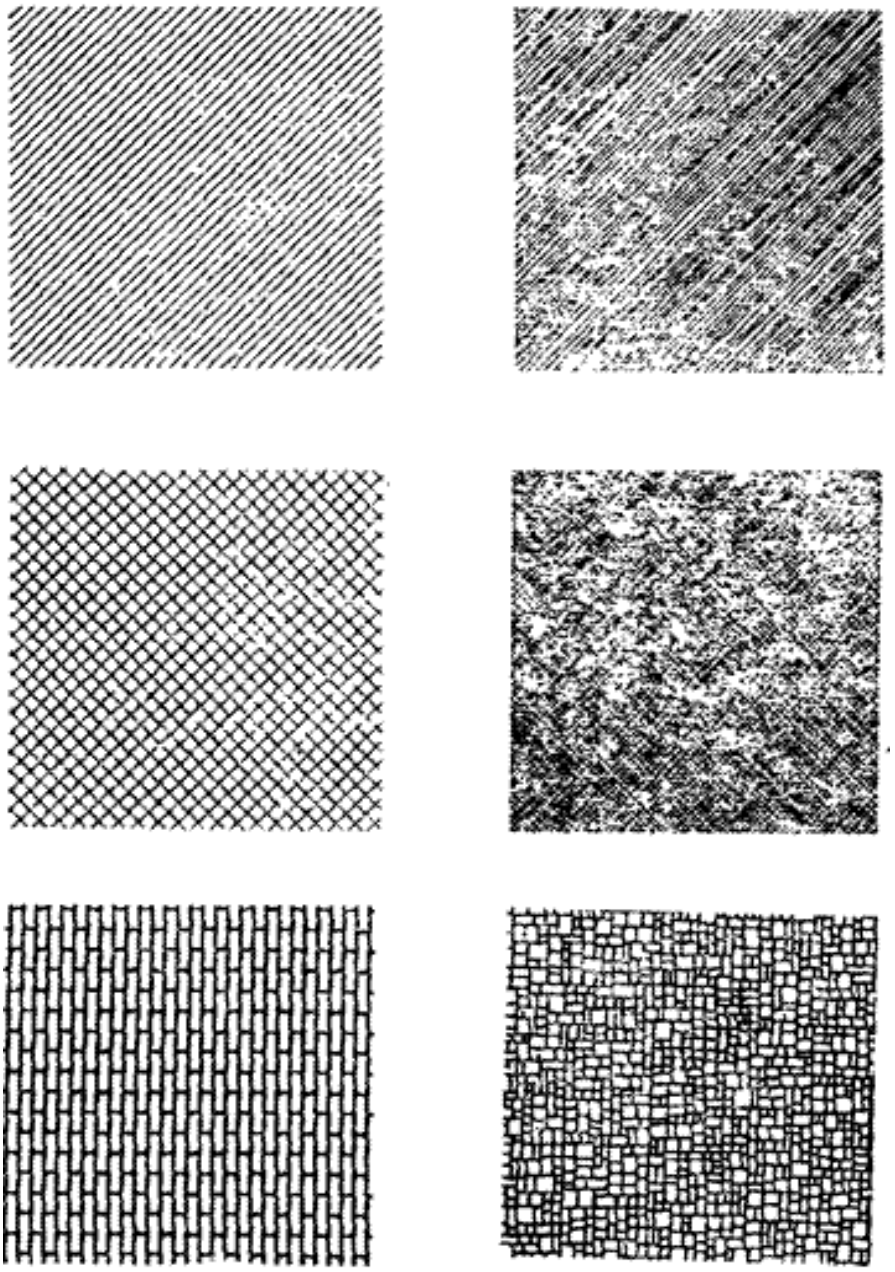


Рисунок 1.2 - Штучні текстури [1]

Під час обробки зображень синтез текстури часто виявляється корисним. Наприклад, якщо на фотографії відсутня або зіпсована певна область, то її можна відновити шляхом додавання області з штучною текстурою. Крім того, під час кодування зображень можна виділяти текстурні області, де важливі деталі відсутні, і виміряти параметри текстури. Під час декодування кожну таку область можна змінити на штучну текстуру. Такий підхід до кодування може бути більш ефективним порівняно з безпосереднім кодуванням зображення.

Синтез текстур базується на основному підходу, який передбачає створення початкового базисного образу та його подальше повторення на площині зображення з використанням визначених правил розміщення. Цей початковий образ може бути маленьким фрагментом, виділеним із природної текстури, або складатися з менших елементів, таких як точки чи відрізки прямих ліній. Розміщення початкових образів може відбуватися як систематично, так і випадково, або ж враховувати певну комбінацію обох підходів. Важливо також враховувати можливі "неправильні" контури на межах початкових образів, і виконати відповідне згладжування, щоб уникнути цих ефектів.

Огляд методів текстурного аналізу.

Значна частина зусиль у дослідженнях в цій області не спрямована на пошук нових текстурних ознак, а на використання вже відомих ознак для розпізнавання об'єктів та образів. Наприклад, в дослідженнях Haralick та Шанмугана використовувалися ознаки, що базуються на гістограмі яскравостей другого порядку, для класифікації спектрозональних аерофотознімків, тоді як Дайєр і Уешка досліджували розпізнавання місцевості з використанням різних типів текстурних ознак. Інші дослідники, такі як Крюгер, Томпсон і Тернер, використовували текстурні ознаки для виявлення та класифікації антракозу легенів на рентгенівських знімках грудної клітки. Зобрист і Томпсон використовували текстурні ознаки для створення функції, яка оцінює відмінність між різними текстурними областями.

На сьогоднішній день методи текстурного аналізу, розроблені в результаті інтенсивних досліджень, широко використовуються для розв'язання завдань

класифікації та сегментації зображень. Ці методи можна поділити на чотири категорії: геометричні, статистичні, методи, що базуються на моделях, та методи, що базуються на обробці сигналів. Серед традиційних методів особливо популярні статистичні підходи, які використовують матриці суміжності статистик другого порядку або статистику першого порядку для аналізу локальних значень (наприклад, різниця гістограм). Також використовуються методи обробки сигналів, зокрема, локальні лінійні перетворення, багатоканальна фільтрація Габора та вейвлет-аналіз, а також моделі, що базуються на марківських випадкових полях або фракталах.

Більшість із оглянутих методів, на жаль, не завжди ефективно справляються з аналізом реальних текстур та вимагають значних обчислювальних ресурсів для застосування в реальному часі у великій кількості сфер комп'ютерного зору. Проте в останні роки були розроблені методи, які здатні ефективно розрізняти текстури за допомогою знаходження локальних дескрипторів, таких як "Local binary patterns" (LBP), що призвело до значних досягнень у використанні текстурного аналізу в різних областях комп'ютерного зору. На даний момент область досліджень в цій галузі розширилася на 3D текстури та динамічні текстури, охоплюючи такі сучасні області, як розпізнавання облич, вираження обличчя, розпізнавання об'єктів, виділення фону, візуальне розпізнавання мови та розпізнавання дій та рухів.

Методи формування дескрипторних значень ґрунтуються на обробці різноманітних локальних областей зображень, що мають різні форми. Зазвичай використовуються блоки невеликих розмірів, які не перекриваються один з одним. Кожен такий блок може бути перетворений у вектор різними способами, наприклад, шляхом обходу пікселів блоку від центрального пікселя по колу або обчислення вектора на основі периметру локальної області та інших методів. Таким чином, для кожної локальної області зображення формується власний локальний дескриптор, який може бути у вигляді локальної гістограми яскравості текстури на зображенні або спеціально сформованого вектора-масиву.

Текстура на зображенні, подібно до кольору, є надзвичайно важливою для сприйняття людиною. Вона надає інформацію про структуру поверхонь та

об'єктів на зображенні. Хоча кожен з нас може визначити текстуру, її точне визначення є складнішим завданням. Текстура не визначається для окремого пікселя, а залежить від розподілу рівнів яскравості у зображенні. Властивості текстури включають періодичність, масштаб та можливість описувати її за допомогою основних напрямів, контрастності та різкості. Аналіз текстури відіграє важливу роль у порівнянні зображень та доповнює колірні характеристики. У науковій літературі існує безліч різних методів представлення інформації про текстуру на зображенні та порівняння зображень на основі цієї інформації. Текстурні ознаки можна поділити на кілька категорій, таких як статистичні, геометричні, модельні та спектральні [16,6]. (рисунок 1.3).

Перший клас методів, який ми розглянемо, включає статистичні ознаки. Ці ознаки використовуються для опису розподілу рівнів яскравості на зображенні, використовуючи різні статистичні параметри. Варто зазначити, що ці ознаки є одними з перших, що були запропоновані в літературі з машинного зору для опису текстур. Вони виявили широке застосування не лише в завданнях класифікації текстур, але також у задачах пошуку текстурних структур. Серед найвідоміших статистичних ознак можна виділити наступні:

1. Прості статистичні параметри, які обчислюються на основі значень яскравості кожного пікселя [10].

2. Параметри, які визначаються за допомогою матриць суміжності [22], і які відображають структуру та взаємне розташування пікселів у зображенні.

3. Текстурні гістограми, які конструюються на основі ознак Тамури (або Tamura features) [13], що включають в себе інформацію про текстурні властивості, такі як різкість та періодичність.

Ці статистичні методи є ефективними для аналізу текстур та знаходження текстурних характеристик на зображеннях. Вони стали важливим інструментом у багатьох областях, включаючи комп'ютерне зорове сприйняття та обробку зображень.

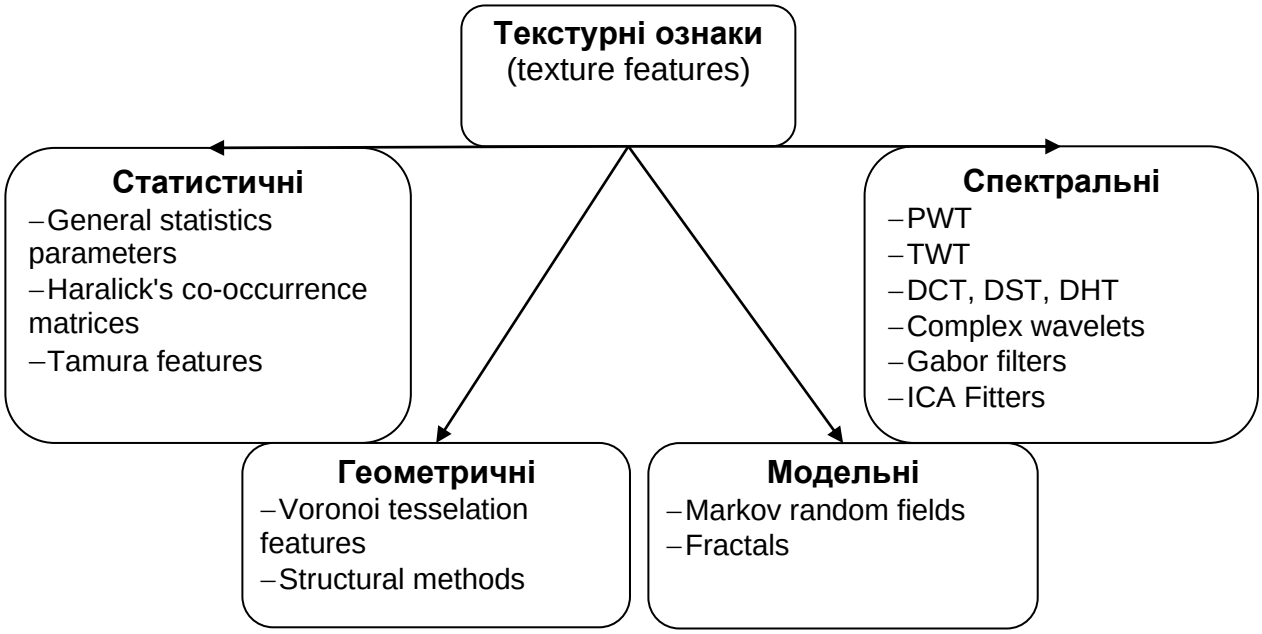


Рисунок 1.3 - Методи представлення текстури

Матриці суміжності

Методику аналізу текстурних властивостей зображення через використання матриць суміжності (Grey Level Co-occurrence Matrices - GLCM) вперше запропонували Haralick та його колеги [10,34]. GLCM представляє собою матриці, які описують спільний розподіл рівнів яскравості на зображенні. Вони описують, як часто пікселі з певним рівнем яскравості зустрічаються поряд у певній просторовій взаємодії на зображенні. У точних термінах, для зображення I розміром N x M, матриця суміжності GLCM визначається як збір частот зустрічей пар пікселів із заданими рівнями яскравості, що знаходяться в певному відношенні один до одного на цьому зображенні.

$$C(i,j)=\sum_{p=1}^{N}\sum_{q=1}^{M}\begin{cases}1, \textit{якщо } I(p,q)=i, \quad I(p+\Delta x, q+\Delta y)=j\\ 0, \textit{інакше}\end{cases}$$

Параметр зсуву (Δx, Δy) відіграє ключову роль, оскільки він визначає відносне розташування пар пікселів на зображенні. Так, $I(p,q)$ відноситься до рівня яскравості пікселя, який знаходиться на координатах (p,q). Процес

формування матриці суміжності для певної області зображення при заданому зсуві, наприклад (Δx, Δy) =(1,0), демонструється на рисунку 1.4.

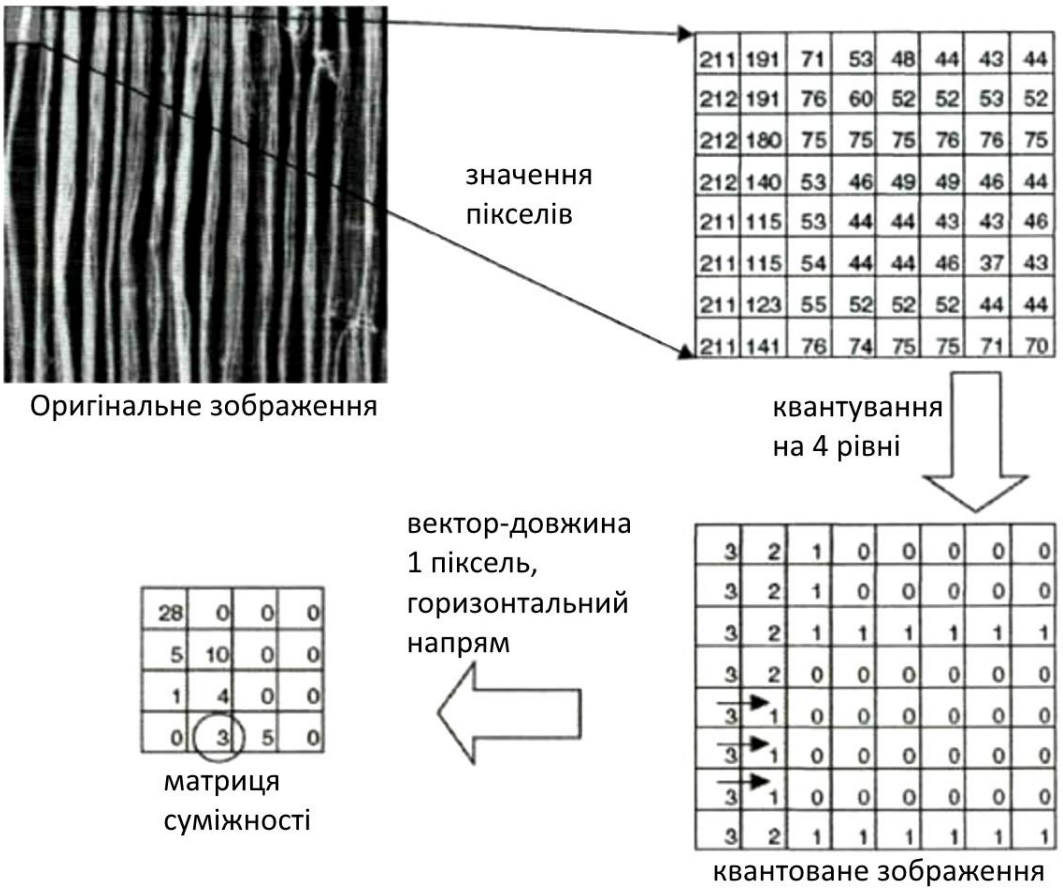


Рисунок 1.4 – Матриці суміжності

Цікаво, що такий параметр зсуву робить характеристику текстурних особливостей чутливою до повороту зображення, який часто є небажаним у контексті систем CBIR (Content-Based Image Retrieval). Один із підходів до вирішення цієї проблеми полягає у створенні декількох матриць суміжності з різними параметрами зсуву, що відповідають повороту зображення на 0°, 45°, 90°, 135°, як це було запропоновано в дослідженнях [12].

На основі матриць суміжності можна розрахувати різні статистичні показники, за допомогою яких порівнюються зображення. Haralick та його колеги пропонували 14 таких показників [20], включаючи другий кутовий момент, контрастність, кореляцію та інші. Кожен із цих показників відображає певну характеристику текстури. Наприклад, другий кутовий момент відображає схожість розподілу у матриці суміжності до діагональної матриці, що вказує на

однорідність текстури. Кореляція вказує на частоту зміни яскравості, відображаючи таким чином різнобарвність зображення. Однак використання всіх чотирнадцяти показників, разом із необхідністю створення декількох матриць суміжності для одного і того ж зображення, могло б зробити будь-яку систему пошуку або класифікації зображень надмірно повільною через значну кількість обчислень. Тому, багато досліджень [33,45] було спрямовано на виділення тих статистичних показників, які найкраще здатні описати особливості текстури. Серед них були визначені деякі ключові показники, які знайшли широке застосування:

1) показник однорідності «Енергія»

$$Energy = \sum_i \sum_j C^2(i, j).$$

2), показник неоднорідності «Ентропія»

$$Entropy = -\sum_i \sum_j C(i, j) \log_2 C(i, j).$$

3) показник контрастності - момент другого порядку різниці елементів

$$Contrast = \sum_i \sum_j (i - j)^2 C(i, j).$$

4) показник «гладкості» -зворотний момент різниці другого порядку,

$$Inverse \quad Different \quad Moment = \sum_i \sum_j \frac{C(i, j)}{1 + (i - j)^2}.$$

Методи автоматичного анотування зображень.

На сьогоднішній день існує розмаїття різних методів автоматичного анотування зображень, які можна умовно розділити на дві основні категорії: алгоритми анотування для всього зображення та алгоритми анотування для окремих об'єктів на зображенні. У випадку алгоритмів анотування за об'єктами, терміном "зображення" ми маємо на увазі певну область даного зображення, що містить шуканий об'єкт, а також саме зображення цього об'єкта, яке використовується для навчання алгоритму. Багато з алгоритмів анотування виконують свою роботу на основі загальної схеми:

1. Знаходження набору ознак для зображення або окремого об'єкта.
2. Підбір датасету зображень і навчання класифікатора.
3. Класифікація колекції зображень, що призводить до призначення набору міток для кожного зображення.

Існують також методи пошуку об'єктів на зображеннях, включаючи методи витягнення текстових міток, які вивчаються в рамках різних завдань, таких як пошук нечітких дублікатів або визначення об'єктів на зразок. Методи пошуку нечітких дублікатів та визначення об'єктів на зразок зазвичай мають найкращі показники повноти та точності серед інших методів. Також, існують методи побудови текстових міток для зображень, які базуються на пошуку нечітких дублікатів для даного зображення в заздалегідь анотованій колекції великого об'єму.

Для алгоритмів детекції характерні проблеми неправильних спрацьовувань, коли певна область зображення, що не є об'єктом інтересу, проте містить деякі локальні ознаки, помилково визнається класифікатором як об'єкт. Це може спричинити зниження точності результатів. Велика кількість деталей і шумів на зображенні також підвищує ймовірність неправильного спрацьовування. Замінюючи параметри алгоритму, також можна впливати на ймовірність детекції об'єкта або неправильного спрацьовування. Однак у рамках одного алгоритму важко підвищити обидва показники одночасно. Для вирішення цієї проблеми може бути використана додаткова класифікація та сортування результатів детектування з максимальною повнотою.

Загальна схема процесу ручної сегментації може використовуватися у випадках, коли відсутня база даних об’єктів, необхідна для автоматичного розпізнавання об’єктів на зображеннях. У таких умовах процес сегментації може покладатися на експерта. Проте при наявності додаткових засобів можна автоматизувати цей процес. Розглянемо метод анотування, який включає такі кроки:

1. Вибір об'єктів.

2. Розрахунок попарних відстаней між об'єктами. Враховуючи, що у цього процесу складність виділення пам'яті $O(N^2)$, можна зберігати відстані між кількома найближчими об'єктами, наприклад, 10.

3. Анотування. Цей процес ітеративний і виконується до тих пір, поки всі знайдені об'єкти не будуть позначені. Кожна ітерація включає в себе наступні дії:

а) Експерт вибирає наступний непозначений об'єкт, і програма наступним пропонує обрати один зі схожих об'єктів у порядку зростання відстані до поточного.

б) Якщо експерт відзначає подібність між об'єктами, програма створює асоціацію між ними.

в) Якщо поточний об'єкт та асоційовані з ним об'єкти не мають міток, тоді експерт позначає ці об'єкти. Після цього програма надає введену мітку всім пов'язаним об'єктам, і вони розглядаються як позначені.

1.4 Постановка задач дослідження

Проблеми детекції блоків тексту:

1. Комплексний фон або шум. Якщо фон документа містить зображення, штрихи чи інші елементи, визначення тексту може бути ускладненим.

2. Спотворення тексту. Якщо текст нахилився або викривився, це може ускладнити його виявлення.

3. Близьке розташування тексту і не-текстових елементів. Це може викликати помилки у визначенні меж тексту.

4. Різноманітний розмір і стиль шрифтів. Це може змусити OCR систему не впізнавати деякі символи.

5. Вертикальний текст. Багато OCR систем оптимізовані для горизонтального читання тексту.

Долати проблеми детекції блоків тексту на зображеннях можна за допомогою ряду методів і підходів, які спрямовані на покращення точності і надійності систем OCR (оптичного розпізнавання символів). Кілька стратегій для кожної з описаних проблем.

1. Комплексний фон або шум

а. Покращення попередньої обробки: Використання фільтрів і методів зниження шуму для підвищення контрасту між текстом і фоном.

б. Сегментація: Застосування алгоритмів сегментації для відокремлення тексту від складного фону.

в. Використання глибокого навчання: Нейронні мережі, навчені на різноманітних даних, можуть краще справлятися зі складними фонами.

2. Спотворення тексту

а. Геометричні перетворення: Застосування методів корекції перспективи та вирівнювання тексту.

б. Адаптивність OCR: Використання OCR систем, які адаптовані до розпізнавання спотвореного тексту.

3. Близьке розташування тексту і не-текстових елементів

а. Методи точної сегментації: Використання високоточних методів сегментації для розрізнення текстових і не-текстових елементів.

б. Глибоке навчання: Навчання нейронних мереж для розрізнення тексту від інших об'єктів.

4. Різноманітний розмір і стиль шрифтів

а. Розширення тренувальних даних: навчання на великій і різноманітній вибірці текстів з різними шрифтами та розмірами;

б. використання OCR систем, які можуть адаптуватися до різних стилів шрифту.

5. Вертикальний текст

а. Орієнтація тексту: розробка алгоритмів для виявлення орієнтації тексту і його відповідного обертання;

б. Навчання на різних орієнтаціях: включення вертикального тексту в тренувальні дані для нейронних мереж.

Підсумуємо стратегії для подолання різних проблем детекції блоків тексту на зображеннях в таблиці.

Таблиця 1.2 - Стратегії подолання проблем детекції блоків тексту

Проблема	Стратегії вирішення
Комплексний фон або шум	- Покращення попередньої обробки - Сегментація - Глибоке навчання
Спотворення тексту	- Геометричні перетворення - Адаптивність OCR
Близьке розташування тексту і не-текстових елементів	- Методи точної сегментації - Глибоке навчання
Різноманітний розмір і стиль шрифтів	- Розширення тренувальних даних - Гнучкі алгоритми OCR
Вертикальний текст	- Орієнтація тексту - Навчання на різних орієнтаціях

Практичне значення досліджень це відтворення структури оригінального документа:

1) Покращення Доступності Інформації: Цей підхід дозволяє перетворювати відскановані документи та зображення тексту в редаговані та пошукові формати, роблячи інформацію більш доступною для обробки та аналізу.

2) Збереження Форматування та Структури: Виділення структурних елементів, таких як заголовки, абзаци, списки, таблиці тощо, дозволяє зберегти оригінальне форматування документа, що є критично важливим для юридичних, наукових та освітніх документів.

3) Автоматизація Обробки Документів: Автоматичне виділення та класифікація структурних елементів сприяє більш ефективній автоматизації процесів, пов'язаних з обробкою великих обсягів документації, наприклад, в бухгалтерії, юридичній сфері, управлінні документообігом.

4) Підвищення Точності OCR: Виділення та розуміння структури документа поліпшує загальну точність OCR, оскільки система може краще розпізнавати текст відповідно до його контексту.

5) Інтеграція з Іншими Системами**:** Структуровані дані легше інтегрувати з іншими цифровими системами, такими як бази даних, системи управління вмістом (CMS) та інші автоматизовані процеси обробки даних.

1.5 Висновки до розділу

Розглянуто поняття та цілі оптичного розпізнавання тексту. Визначено що Основними методами детекції блоків тексту є методи на основі морфологічних операцій, на основі машинного навчання та на основі нейронних мереж. Підсумовано критерії порівняння їх якості. Проаналізовано текстурні ознаки зображень. Систематизовано проблеми детекції блоків тексту та методи їх подолання.

2 РОЗРОБКА АЛГОРИТМУ ДЕТЕКЦІЇ БЛОКІВ ТЕКСТУ

2.1 Текстурні ознаки зображень текстів

Термін "текстура" в контексті зображень не має єдиного визначення, але загальна думка дослідників полягає в тому, що текстурні зображення характеризуються просторовою однорідністю та наявністю повторюваних, іноді випадково змінюваних структур. Текстура віддзеркалює фізичні атрибути поверхні об'єкта та має характеристики, такі як гранулярність, контраст, орієнтація, лінійність та шорсткість.

Різноманітність текстур поділяється за критеріями, такими як взірці розміщення (регулярні, квазірегулярні, нерегулярні, квазістохастичні, стохастичні) та походженням (натуральні/фотографічні, штучні).

У текстурному аналізі зображень існують два основні напрямки: сегментація текстури та класифікація текстури. Для сегментації текстури розрізняють область-орієнтований підхід, який фокусується на ідентифікації областей з однорідною текстурою, та контурно-орієнтований підхід, який базується на виявленні відмінностей у текстурі, щоб визначити границі між різними текстурними областями.

Класифікація текстури залучає віднесення зображення до певної категорії текстур, ґрунтуючись на апріорних знаннях про класи текстур. Цей процес застосовується у таких сферах, як аналіз аерокосмічних фотографій, контроль якості, біомедичне зображення.

При оцінці текстури зображення розглядаються такі атрибути, як однорідність, інтенсивність, грубість/гранулярність, шорсткість, регулярність, лінійність, орієнтація. Це розмаїття характеристик призвело до появи широкого спектру моделей текстури, перший крок в створенні яких полягає в описі цих атрибутів. Методи опису текстури варіюються від статистичних до геометричних, від теорії випадкових процесів до фрактальних і методів обробки сигналів.

Статистичні методи, наприклад, описують просторовий розподіл рівнів інтенсивності на зображенні. Матриці розподілу інтенсивності (MPI) зображають

статистику другого порядку, де кожен елемент матриці відображає частоту певних пар значень інтенсивності, розташованих на заданій відстані *d*. Кожна точка зображення (*x*, *y*) відповідає унікальній матриці суміжності Pd, яка відображає розподіл інтенсивності у визначеному регіоні розміром $W \times W$ *і*з центром в точці (*x*, *y*) . Кожен елемент матриці P_d обчислюється:

$$P_d(i, j) = \sum_{(m,n)\in D} f_{i,j}(x_{m,n}; x_{m+d,n+d}), \quad (2.1)$$

де

$$f_{i,j}(x_{m,n}; x_{m+d,n+d}) = \begin{cases} 1, (x_{m,n} = i \text{ та } x_{m+d,n+d} = j) \text{ або } (x_{m,n} = j \text{ та } x_{m+d,n+d} = i) \\ 0, \text{інакше} \end{cases}, \quad (2.2)$$

де D – регіон розміром $W \times W$,

i, j = 0..255 – значення яскравості кожної точки,

$x_{m,n}$ – яскравість точки з заданими координатами (m,n) .

Функція $f_{i,j}(x_{m,n}; x_{m+d,n+d})$ служить як показник того, що точки, розташовані на певній відстані одна від одної, мають конкретні рівні яскравості. Параметр *d* в цьому контексті вказує на просторову відстань, на якій здійснюється аналіз для визначення взаємозв'язку між сусідніми точками. Використовуючи матриці суміжності, можна ідентифікувати та аналізувати різні текстурні характеристики, включаючи такі показники, як енергія, ентропія, контраст, однорідність, кореляція, та властивість затінення. Ці матриці знаходять своє основне застосування в задачах класифікації текстур.

Дана функція визначається наступним чином:

$$f_{i,j}(x_{m,n}; x_{m+d,n+d}) = \begin{cases} 1, & (x_{m,n} = i \quad та \quad x_{m+d,n+d} = j) \\ & або\ (x_{m,n} = j \quad та \quad x_{m+d,n+d} = i) \\ 0, & інакше \end{cases}$$

Ця функція служить маркером для ідентифікації точок, розміщених на певному віддаленому інтервалі, які характеризуються визначеними рівнями яскравості. Параметр d встановлює специфічну дистанцію для аналізу взаємодії між сусідніми точками.

Використовуючи Матриці Розподілу Яскравості P_d (2.2), які відображають розподіл яскравості в області навколо точки (x, y), розраховуються різні характеристики текстури. Для кожної характеристики створюється окрема матриця, яка зберігає відповідні значення для усіх аналізованих точок, формуючи таким чином простір текстурних характеристик. Нижче наведено методи розрахунку різних характеристик:

1) Загальне середнє значення:

$$F_1 = \sum_{i=1}^{W} m_i p_i \ ,$$

2) Інерція (або міра розсіяння):

$$F_2 = \sum_{i=1}^{W}\sum_{j=1}^{W}(i-j)^2 P(i,j)\,,$$

3) Другий кутовий момент (міра однорідності):

$$F_3 = \sum_{i=1}^{W}\sum_{j=1}^{W}P^2(i,j) \ ,$$

4) Загальна кореляція (міра залежності між яскравістю точок):

$$F_4 = \sum_{i=1}^{W}\sum_{j=1}^{W}m_i m_j \ ,$$

5) Ентропія (міра невизначеності або варіативності):

$$F_5 = -\sum_{i=1}^{W}\sum_{j=1}^{W}\ln(P(i,j))P(i,j) \ ,$$

6) Локальна кореляція:

$$F_6 = \sum_{i=1}^{W}\sum_{j=1}^{W}(i-M)(j-M)P(i,j) \ ,$$

7) Затінення (міра асиметрії розподілу яскравості):

$$F_7 = \sum_{i=1}^{W}\sum_{j=1}^{W}(i+j-2M)^3 P(i,j) \ ,$$

8) Контраст (міра різниці між яскравістю сусідніх точок):

$$F_8 = \sum_{i=1}^{W}\sum_{j=1}^{W}|i-j|P(i,j) \ ,$$

9) Загальна ентропія (сумарна міра невизначеності):

$$F_9 = \sum_{i=1}^{W}\ln(p_i)\,p_i$$

де $m_i = \sum_{j=1}^{W} jP(i,j)$, $p_i = \sum_{j=1}^{W} P(i,j)$, $M = \sum_{i=1}^{W} ip_i$.

Як вбачаємо з обчислення матриці імовірнісного розподілу та текстурних ознак, згідно з вищезазначеною визначеністю, обчислення для кожної точки на зображенні вимагає значного обсягу часу, тому рекомендується обчислювати матрицю та ознаки за допомогою рекурсивного методу. Крім того, нормалізацію матриці краще застосовувати лише для обчислення певних ознак (F_5, F_6, F_7, F_9).

Для проведення рекурсивних обчислень спочатку необхідно повністю обчислити матрицю і текстурні ознаки для першої точки (x = L, y = L), де $L = \frac{W-1}{2}$. Вікно, на основі якого обчислюється матриця для наступної точки (x+1, y), зсувається вправо, що еквівалентно наступній операції перетворення.

$$P_y^{x+1}(i,j) = P_y^x(i,j) - \sum_{(m,n)\in D1} f_{i,y}(x_{m,n}; x_{m+d,n+d}) +$$

$$\sum_{(m,n)\in D2} f_{i,y}(x_{m,n}; x_{m+d,n+d}),$$

де $D1 = \{(m,n) \mid m \in [x-W, x+W], n = y-W\}$,

$D1 = \{(m,n) \mid m \in [x-W, x+W], n = y+W\}$.

Переміщення до наступного рядка зображення відбувається за подібним принципом: у цьому випадку вікно, що було використане для аналізу першої точки попереднього рядка, переміщується вниз, а матриця обчислень адаптується для нової позиції цього вікна. Визначення характеристик зображення переглядається наступним образом: кожного разу, коли до елемента P(i, j) додається або віднімається одиниця, параметри текстури модифікуються відповідно до приведених нижче рівнянь:

$$m_i = m_j \pm j$$

$$p_i = p_i \pm 1$$

$$F_2 = F_2 \pm (i-j)^2$$

$$F_5 = F_5 + \ln\left(\frac{P(i,j)}{N}\right)P(i,j) + \ln\left(\frac{P(i,j)\pm 1}{N}\right)\left(P(i,j)\pm 1\right)$$

$$F_8 = F_8 \pm |i-j|,\ Mx = Mx \pm \frac{i}{N},\ N = 2S^2.$$

Якщо $i = j$, то вийде $F_3 = F_3 \pm 4P(i,j) + 4$, у іншому випадку $F_3 = F_3 \pm 2P(i,j) + 1$. У наступному етапі аналізу матриць текстурних ознак виконується пошук найвищих і найнижчих значень. Після цього проводиться

нормалізація до стандартного діапазону градацій сірого кольору, який коливається від 0 до 255, за допомогою наступної формули:

$$g = 255\frac{f - \min}{\max - \min},$$

де f - це елемент матриці;

g - значення яскравості;

$\max$ і $\min$ - елементи максимальний і мінімальні.

Після цього, матриці текстурних ознак візуалізуються у вигляді зображення текстурного поля, що дозволяє отримати візуальне відображення аналізованих даних. Важливо зазначити, що розмір вікна, в межах якого проводиться обчислення текстурних ознак, впливає на якість та чіткість зображення об'єктів. При збільшенні розміру вікна (позначеного як S), область аналізу стає більшою, але водночас текстурні характеристики можуть стати менш чіткими. Тому вибір параметру S варто проводити з огляду на розмір вихідного зображення та розмір об'єктів, які необхідно виділити.

Параметр d вказує на відстань між точками, які враховуються при розрахунку яскравості та їх імовірного розподілу у матриці текстурних ознак.

На рисунку 2.1 наведено блок-схему алгоритму обчислення матриць імовірного розподілу рівнів яскравості.

Функція автокореляції зображення.

Важливою характеристикою численних текстур є їх регулярна структура та спосіб розташування елементів на зображенні. Для визначення рівня регулярності та зернистості текстури часто використовується функція автокореляції зображення. Ця функція вимірює ступінь взаємозв'язку між різними частинами зображення та дає уявлення про те, наскільки часто повторюються образи у текстурі. Зазвичай, зернистість текстури корелює з розсіюванням значень функції автокореляції.

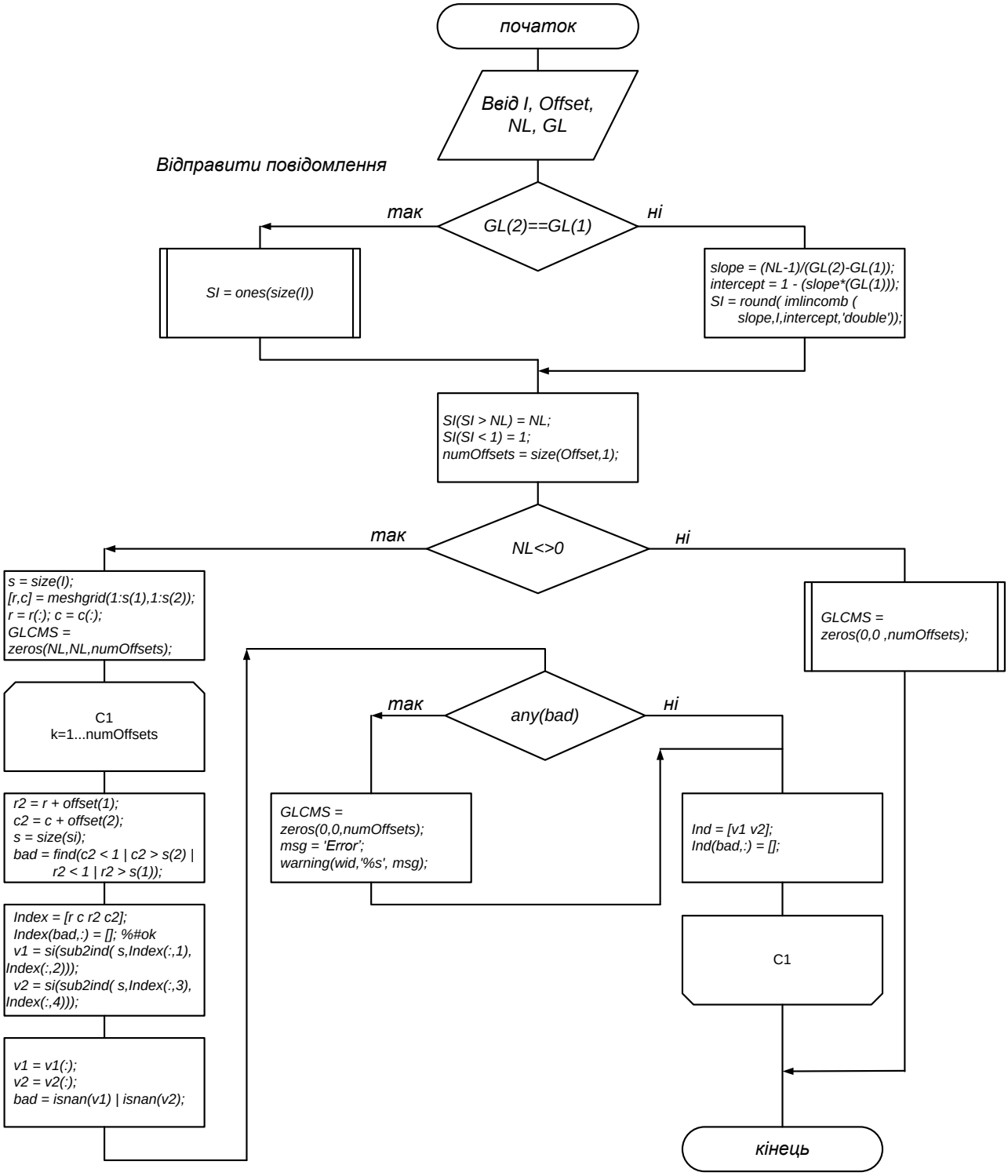


Рисунок 2.1 - Блок-схема алгоритму обчислення матриць імовірного розподілу рівнів яскравості

Формула для обчислення функції автокореляції має наступний вигляд:

$$R(m,n) = \sum_{x=0}^{N-1} \sum_{y=0}^{M-1} I(x,y) I(x+m, y+n)$$

де $I(x,y)$ - яскравість пікселя в координатах (x, y),

m і n - зсуви.

Існують різні методи аналізу текстур, які використовують геометричні та структурні підходи. Геометричні методи визначають текстури як сукупності "текстурних елементів" та досліджують їх статистичні характеристики як ознаки текстури. Вони також можуть вивчати правила розташування цих елементів. Для визначення текстурних елементів можна застосовувати різні методи, наприклад, розбиття поверхні на багатокутники Вороного, що служать текстурними ознаками.

Структурні методи побудови текстурних моделей передбачають, що текстури складаються з примітивів, які розташовані відповідно до певних правил. Такі методи зазвичай обмежені регулярними текстурами і включають визначення елементів текстури та правил їх розташування.

Також існують методи, що базуються на моделях текстур, які можуть бути використані як для опису, так і для синтезу текстур. Параметри таких моделей можуть відтворювати візуальні характеристики текстури.

Крім того, важливими є методи, які використовують випадкові процеси, такі як Випадкові Марківські процеси, для аналізу та синтезу текстур. Також фрактальна геометрія може бути використана для оцінки ступеня самоподібності та шершавості текстур.

Розрахунок Фрактального виміру (метод підрахунку квадратів):

1. Розглянемо зображення I, яке представляє текстуру.
2. Розділімо I на неперекривні квадрати розміром s×s.
3. Підрахуємо кількість квадратів N(s), що містять частину текстури (тобто квадрати, які не є повністю порожніми).
4. Фрактальний вимір D оцінюється шляхом знаходження нахилу лінії на логарифмічному графіку N(s) проти 1/s. Математично D може бути наближено як:

$$D = \lim_{s \to 0} \frac{\log N(s)}{\log(1/s)}.$$

5. На практиці *D* оцінюється на ряді масштабів *s* і усереднюється.

Наступний скрипт:

– конвертує зображення в бінарний формат;

– застосовує метод підрахунку квадратів для різних розмірів квадратів;

– оцінює фрактальний вимір за допомогою лінійної регресії на логарифмічному графіку.

Імпорт бібліотек

```
import numpy as np
import cv2
from matplotlib import pyplot as plt
```

Крок 2: Визначення функції для розрахунку фрактального виміру

```
def box_count(img, box_size):
    count = 0
    for i in range(0, img.shape[0], box_size):
        for j in range(0, img.shape[1], box_size):
            if np.any(img[i:i+box_size, j:j+box_size]):
                count += 1
    return count

def fractal_dimension(img):
    img = (img > 128).astype(np.uint8)
    sizes = np.arange(4, 50, 2)
    counts = [box_count(img, size) for size in sizes]
    coeffs = np.polyfit(np.log(1.0/sizes), np.log(counts), 1)
    return coeffs[0]
```

Крок 3: Створення Тестового Зображення

```
# Створити біле зображення
test_image = np.ones((400, 400), dtype=np.uint8) * 255
# накласти текст на зображення
cv2.putText(test_image,          "Текст",          (50,          200),
cv2.FONT_HERSHEY_SIMPLEX, 4, (0, 0, 0), 2, cv2.LINE_AA)

# Відобразити зображення
plt.imshow(test_image, cmap='gray')
plt.title("Тестове Зображення")
plt.show()

# функція для визначення фрактального виміру
```

```
fd = fractal_dimension(test_image)
print("Фрактальний Вимір:", fd)
```

Фрактальний Вимір: 1.9457537868820112

Фрактальний Вимір: 1.9456241331634487

Рисунок 2.2 – Обчислення фрактальних ознак для зображення тексту

Цей код спочатку створює біле зображення з чорним текстом, потім відображає його за допомогою matplotlib, і нарешті використовує функції для визначення фрактального виміру зображення.

Кожен з цих підходів має свої переваги та застосування в залежності від конкретного завдання аналізу текстур.

Техніки аналізу сигналів

Початкові дослідження в галузі використання просторових фільтрів зосереджувались на вимірюванні кількості країв на одиницю площі. Використання фільтрів дозволяє ефективно виявляти базові шаблони завдяки їхній чутливості до елементів, схожих на сам фільтр. Це дозволяє представити текстурні характеристики зображення через відгуки на різні фільтри. Для цього створюється колекція фільтрів, що включає узори різних форм, таких як крапки та лінії, в декількох масштабах. Ці фільтри здатні розрізняти неорієнтовані структури (крапки) та орієнтовані (смуги).

Один із методів створення потрібних фільтрів полягає у використанні взаємної різниці між Гаусовими фільтрами різних масштабів. Кількість масштабів і напрямків визначають експериментально, варіюючи від 4 до 11 масштабів та від 2 до 18 напрямків. Сукупність оброблених зображень не дає повного опису текстури, тому додатково збирають статистичні дані вихідних сигналів різних фільтрів для класифікації текстур. Це дозволяє створити характеристичний вектор на основі середнього значення та стандартного відхилення вихідних сигналів фільтрів.

Аналіз текстур також здійснюється через спектральний аналіз Фур'є, який відображає залежність між просторовими частотами та зернистістю текстури. Дослідження показують, що людське сприйняття аналізує окремо просторову частоту та орієнтацію текстури, використовуючи орієнтовані фільтри для детального аналізу.

Вейвлет-метод, заснований на дискретному вейвлетному перетворенні (ДВП), є альтернативою Фур'є перетворенню, використовуючи "хвилі" змінної частоти та обмеженої тривалості. Цей метод корисний для класифікації текстур, пропонуючи інваріантність текстурних характеристик до повороту та масштабування.

2.2 Локальні бінарні шаблони

Стаття [13] зосереджується на проблемі відокремлення нетекстових компонентів від текстових у рукописних документах, що є важливою, але маловивченою дослідницькою областю. У статті представлено емпіричне дослідження щодо застосування різних текстурних ознак на основі локальних бінарних шаблонів (LBP) для вирішення цієї проблеми. Також пропонується незначна модифікація одного з варіантів оператора LBP для покращення результатів у задачі класифікації тексту/нетексту.

Описувачі ознак потім оцінюються на базі даних, що складається з зображень з 104 рукописних лабораторних робіт та конспектів лекцій з різних інженерних та наукових галузей, використовуючи п'ять відомих класифікаторів. Результати класифікації підтверджують ефективність описувачів ознак на основі LBP у відокремленні тексту від нетексту.

Отже розроблення алгоритму на основі LBP є актуальним.

Метод локальних бінарних шаблонів (LBP) представляє собою ефективний інструмент для ідентифікації різноманітних компонентів на зображенні. Цей метод базується на створенні гістограм LBP кодів, які слугують в якості простору ознак. Особливістю LBP є його висока швидкість обчислень, завдяки чому він набув популярності у сфері комп'ютерного зору та аналізу зображень.

Фільтр LBP, позначений як $LBP_{R,P}(x, y)$, працює шляхом визначення коду для кожної точки зображення, ґрунтуючись на значеннях навколишніх точок. У цій формулі P означає кількість точок, а R - радіус області навколо точки. Визначаючи точки в області як g_i (для $i = \overline{0, P-1}$), координати кожної точки визначаються:

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

Позначимо зображення $f(x, y)$ як $f(g)$, де g представляє собою точку. Розглянемо таку ситуацію:

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

де g_c - точка в якої координати (x, y).

Отже

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

Двійкова послідовність коду LBP це послідовність $s_i(x, y)$, при $i = \overline{0, P-1}$,. Тому

$$LBP_{P,R}(x,y) \in \left[0, 2^P - 1\right].$$

Для порівняння даних двох зображень, розглядаючи їх як вектори ознак, ми використовуємо гістограми на основі LBP кодів. Для кожного зображення ми створюємо гістограму $H(l)$ для значень фільтру $LBP_{P,R}(x, y)$, де LBPP - це один з кодів LBP. Існують різні методи обчислення відстані між цими гістограмами, і одним з них є використання відстані 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 - кількість кодів.

Існують коди, які містять більше інформації, ніж решта. Коди, які у двійковому циклічному представленні мають не більше двох переходів між послідовностями "1" і "0", позначаються як «uniform» або «рівномірні». Для даного значення P існує $P*(P-1)+2$ «рівномірних» кодів. У випадку модифікованого фільтра $LBP_{P,R}^{u2}$ повертаються коди рівномірних значень, і додається лише один код для нерівномірних значень.

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

Етап 4. Ураховуючи, що периферія представляє собою круг, можна виявити групи LBP-кодів, які не змінюються при обертанні. Кожен LBP-код має P варіантів, стійких до ротації, отримуваних через циклічне зміщення P-бітової послідовності. В межах кожної групи для фільтру обирається найменший код.

Визначення кількості обертово-інваріантних кодів є складною задачею. Відповідний фільтр позначається $LBP_{P,R}^{ri}$.

Етап 5. З урахуванням обох попередніх характеристик, також встановлюється уніфікований фільтр, стійкий до обертання. Існує всього P + 2 LBP-кодів, які водночас мають обидві характеристики, відрізняючись кількістю бітів, встановлених у положення «1». Даний фільтр $LBP_{P,R}^{riu2}$ визначається наступним чином:

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

В третьому розділі буде реалізовано алгоритм на основі ЛБШ.

2.3 Алгоритм виділення блоків тексту на основі текстурних ознак

Алгоритм виділення блоків тексту на зображенні на основі текстурних ознак може включати кілька кроків, які використовують методи комп'ютерного зору та обробки зображень. Ось загальний підхід:

Крок 1: Підготовка Зображення

1. Зчитування зображення: Завантажитт зображення, на якому потрібно виділити блоки тексту.

2. Перетворення в сірі тони: Конвертувати зображення у градації сірого для спрощення аналізу текстури.

Крок 2: Визначення текстурних ознак

1. Розрахунок текстурних ознак: алгоритми розрахунку текстурних ознак, наприклад, метод матриць розподілу рівнів сірого (GLCM), для отримання характеристик текстури кожного пікселя або області зображення.

2. Використання фільтрів**:** Застосувати фільтри (наприклад, фільтр Собеля) для виділення горизонтальних та вертикальних ліній, які часто присутні в тексті.

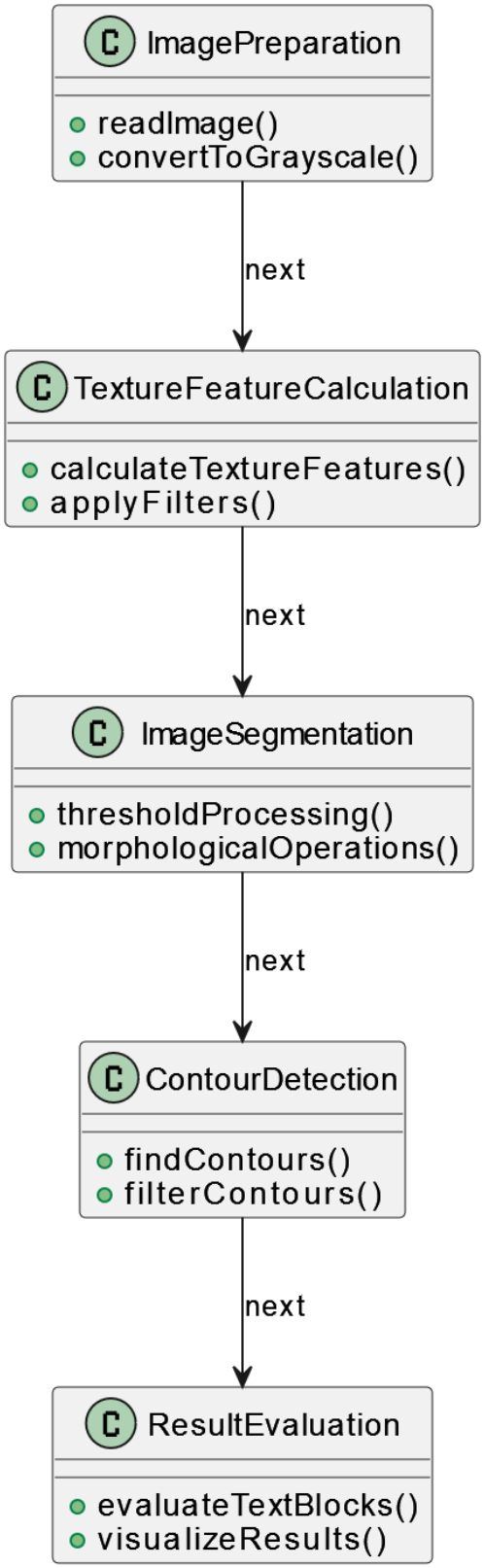


Рисунок 2.3 - UML Діаграма класів програмного засобу

Крок 3. Сегментація зображення:

1. Порогова обробка: Застосуйте порогову обробку для виділення областей із високою текстурною відмінністю, які можуть відповідати тексту.

2. Морфологічні операції: Використовуйте морфологічні операції, такі як дилатація та ерозія, для поліпшення сегментації тексту.

Крок 4: Виділення контурів

1. Знаходження контурів: Використовуйте алгоритми знаходження контурів для ідентифікації потенційних блоків тексту.

2. Фільтрація контурів: Фільтруйте контури на основі розмірів, пропорцій та інших характеристик, щоб відокремити імовірні блоки тексту від інших елементів зображення.

Рисунок 2.4 – UML діаграма активності

Крок 5: Оцінка Результатів

1. Оцінка блоків тексту: Перевірити, чи відповідають виділені області тексту ваших критеріїв (наприклад, за розміром, формою, орієнтацією).

2. Візуалізація результатів: Відобразіть виділені блоки тексту на зображенні для візуальної оцінки.

Алгоритм може потребувати налаштування та оптимізації параметрів для кращої роботи з конкретними типами зображень.

Ця діаграма класів показує п'ять основних компонентів процесу обробки зображень, кожен з яких містить відповідні методи.

Діаграма активності для процесу обробки зображень на рисунку.

Ця діаграма активності представляє послідовність кроків, виконуваних під час обробки зображень для виділення текстових блоків.

Алгоритм детекції тексту на зображенні з використанням морфологічних операцій можна реалізувати за допомогою Python та бібліотеки OpenCV. Ось базовий приклад такого алгоритму:

```
import cv2
import numpy as np

def detect_text(image_path):
    # Завантаження зображення
    image = cv2.imread(image_path)
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    # Застосування порогової обробки для виділення тексту
    threshold = cv2.adaptiveThreshold(gray, 255,
cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
    # Використання морфологічних операцій
    kernel = np.ones((5,5), np.uint8)
    dilate = cv2.dilate(threshold, kernel, iterations=1)
    erode = cv2.erode(dilate, kernel, iterations=1)
```

Знаходження контурів та Виділення блоків тексту на зображенні

```
    contours, _ = cv2.findContours(erode, cv2.RETR_EXTERNAL,
cv2.CHAIN_APPROX_SIMPLE)
    # Виділення блоків тексту на зображенні
    for contour in contours:
```

```
            x, y, w, h = cv2.boundingRect(contour)
            cv2.rectangle(image, (x, y), (x + w, y + h), (0, 255, 0), 2)
    # Відображення результату
    cv2.imshow('Detected Text', image)
    cv2.waitKey(0)
    cv2.destroyAllWindows()

# Виклик функції зі шляхом до зображення
detect_text('path_to_image.jpg')
```

У цьому коді спочатку застосовується порогова обробка для виділення можливих областей з текстом. Потім використовуються морфологічні операції (дилатація та ерозія) для покращення детекції текстових блоків. Нарешті, контури, що відповідають текстовим блокам, виділяються на оригінальному зображенні.

Опис діаграми:

1. Початок - Початок процесу детекції тексту.
2. Завантаження зображення - Завантаження вхідного зображення.
3. Конвертування в відтінки сірого - Перетворення зображення в шкалу сірого для спрощення обробки.
4. Застосування порогової обробки - Використання порогової обробки для виділення тексту.
5. Дилатація - Застосування морфологічної операції дилатації.
6. Ерозія - Застосування морфологічної операції ерозії.
7. Визначення контурів - Пошук контурів для визначення потенційних блоків тексту.
8. Виділення блоків тексту на зображенні - Маркування знайдених блоків тексту на зображенні.
9. Кінець - Закінчення процесу детекції тексту.

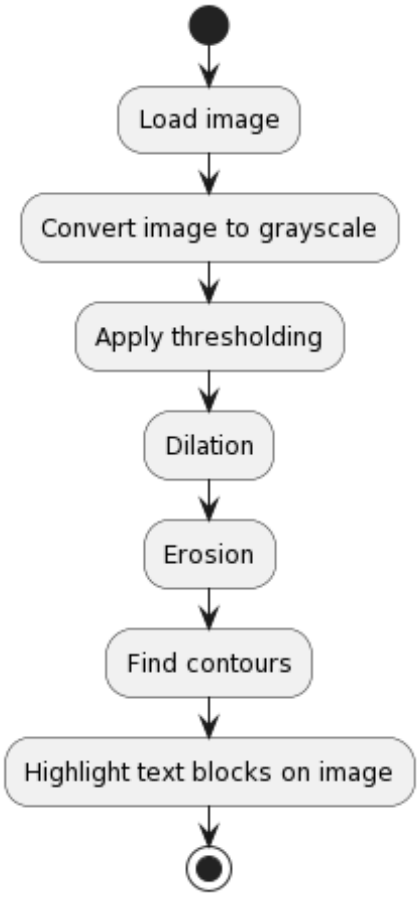


Рисунок 2.5 – Діаграма активності алгоритму на основі морфологічних операцій

2.4 Висновки до розділу

Розроблено блок-схему алгоритму обчислення матриць імовірного розподілу рівнів яскравості, алгоритми виділення блоків тексту на основі обчислення ознак локальних бінарних шаблонів. Розроблено об’єкту модель програмного забезпечення.

3 ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТИ

3.1 Засоби розробки

Створення OCR (Optical Character Recognition) додатків у Python вимагає обрання певного набору бібліотек і технологій. Структура OCR додатка зазвичай включає обробку зображень, розпізнавання тексту, обробку тексту та збереження даних. Бібліотеки обробки зображень:

1. OpenCV (Open Source Computer Vision Library) - це бібліотека з відкритим кодом для обробки зображень та комп'ютерного зору. Вона включає інструменти для обробки та підготовки зображень до розпізнавання тексту.

2. Pillow - це вдосконалена версія бібліотеки PIL (Python Imaging Library) і забезпечує зручні інструменти для маніпулювання зображеннями в Python.

Розпізнавання тексту

1. Tesseract - одна з найпопулярніших бібліотек OCR, підтримувана Google. Pytesseract - це обгортка Python, яка дозволяє легко інтегрувати Tesseract в Python-проекти.

2. EasyOCR - це більш сучасна бібліотека, яка підтримує багато мов та забезпечує хороші результати розпізнавання.

Після розпізнавання тексту може знадобитися його додаткова обробка:

1. NLTK (Natural Language Toolkit) - для обробки тексту на природній мові, наприклад, для видалення стоп-слів, стемінгу тощо.

2. Regular Expressions (re) - для роботи з регулярними виразами, які можуть бути корисні для витягу специфічної інформації з тексту.

Збереження даних

1. SQLite - це легка база даних, яка не вимагає окремого сервера. Ідеально підходить для додатків, де потрібна проста інтеграція з базою даних.

2. PostgreSQL - більш потужна система управління базами даних, яка підходить для складніших застосувань.

3. Pandas - для аналізу та обробки датасетів, особливо у вигляді таблиць.

Швидкодія і оптимізація

1. NumPy - основна бібліотека для наукових обчислень у Python, яка дозволяє ефективно працювати з масивами та матрицями.

2. CUDA - технологія від NVIDIA, яка дозволяє значно прискорити обчислення, використовуючи GPU.

3. Cython - може використовуватися для написання C-подібного коду, який компілюється в Python-модуль для підвищення швидкодії.

Таблиця 3.1 – Систематизація інструментів розроблення

Категорія	Бібліотека/Технологія	Опис
Обробка зображень	OpenCV, Pillow	Обробка та підготовка зображень
Розпізнавання тексту	Tesseract, EasyOCR	OCR для розпізнавання тексту
Обробка тексту	NLTK, re	Аналіз та обробка розпізнаного тексту
Збереження даних	SQLite, PostgreSQL	Бази даних для зберігання результатів
Аналіз даних	Pandas	Обробка та аналіз датасетів
Швидкодія і оптимізація	NumPy, CUDA, Cython	Підвищення швидкодії обчислень

OpenCV (Open Source Computer Vision Library) - це велика бібліотека програмного забезпечення для обробки зображень та комп'ютерного зору. Вона надає інструменти для різноманітних операцій, включно з аналізом зображень, обробкою відео, калібруванням камери, машинним навчанням та багатьма іншими. OpenCV підтримує різні мови програмування, включно з Python, і є ідеальним вибором для реалізації рішень у сфері комп'ютерного зору, від простої обробки зображень до складних алгоритмів глибокого навчання. Її широкий спектр функціоналу та легка інтеграція з іншими бібліотеками роблять її однією з найпопулярніших у цій сфері.

Pillow - це проект Python, який надає зручні можливості для роботи з зображеннями. Він є нащадком Python Imaging Library (PIL) і забезпечує розширений інтерфейс для створення, обробки, та маніпулювання зображеннями.

Pillow підтримує широкий діапазон форматів файлів, надає інструменти для редагування зображень, такі як обертання, масштабування, обрізання, а також фільтри та ефекти. Це ідеальний інструмент для розробників, яким потрібно легко втручатися в зображення в Python, що робить його популярним вибором для веб-розробки, наукових досліджень, автоматизації та багатьох інших сфер.

Tesseract є однією з найпопулярніших бібліотек для розпізнавання тексту (OCR) у світі, підтримуваною Google. Вона може розпізнавати текст на понад 100 мовах і широко використовується в індустрії та академічних дослідженнях. Tesseract була спочатку розроблена Hewlett-Packard у 1980-х роках, а потім оптимізована Google для роботи з машинним навчанням та глибокими нейронними мережами. Її Python обгортка, pytesseract, дозволяє легко інтегрувати її можливості в Python проекти. Tesseract ефективна для різних видів документів, від сканованих сторінок до фотографій з текстом, роблячи її незамінним інструментом у багатьох областях, включно з автоматизацією офісу, управлінням документами та цифровим зберіганням.

EasyOCR - це сучасна інноваційна бібліотека для розпізнавання тексту, яка відрізняється своєю здатністю працювати з більш ніж 80 мовами та підтримкою комплексних символів і шрифтів. Вона використовує глибоке навчання для підвищення точності та швидкості розпізнавання тексту і є ідеальним вибором для проектів, де потрібна висока точність OCR, особливо в умовах, де зображення можуть бути неякісними або містити складний макет. EasyOCR легко інтегрується з Python і є відмінним інструментом для розробників, які шукають ефективні рішення для розпізнавання тексту в різних додатках.

NLTK (Natural Language Toolkit) - це потужна бібліотека для роботи з людською мовою в Python. Вона надає легкий доступ до більше ніж 50 корпусів та лексичних ресурсів, таких як WordNet, а також набори інструментів для класифікації, токенізації, стемінгу, тегування, синтаксичного та семантичного аналізу. NLTK широко використовується в академічних дослідженнях та комерційних проектах для рішення завдань обробки природньої мови, включно з аналізом настроїв, класифікацією тексту, машинним перекладом та іншими. Ця

бібліотека є незамінною для розробників, які хочуть досліджувати та використовувати мовні дані в своїх проектах.

re (Regular Expressions) - це модуль в Python, який надає повний набір інструментів для роботи з регулярними виразами. Регулярні вирази - це потужний інструмент для здійснення складних пошуків та маніпуляцій з текстом. Модуль re дозволяє швидко знаходити патерни, розбивати рядки на частини, замінювати фрагменти тексту та багато іншого. Він є незамінним у задачах валідації даних, обробки лог-файлів, аналізу текстів та інших сценаріях, де потрібно працювати зі складними шаблонами тексту.

SQLite - це легка, самодостатня база даних SQL, яка не вимагає окремого сервера та мінімально використовує ресурси системи. Її можна вбудувати прямо в додатки, роблячи її ідеальною для мобільних застосунків, вбудованих систем і невеликих додатків. SQLite підтримує більшість основних функцій SQL і є дуже надійною - всі дані зберігаються в одному файлі, що спрощує управління даними. Ця база даних широко використовується у всьому світі і є відмінним вибором для проектів, де необхідна простота і ефективність.

PostgreSQL є потужною, відкритою системою управління реляційними базами даних. Вона відрізняється високою надійністю, гнучкістю та підтримкою розширених функцій SQL, включаючи складні запити, транзакції, зовнішні ключі, підтримку JSON, а також можливості для розширення. PostgreSQL широко використовується у великих системах та проектах, де потрібна висока продуктивність та надійність. Її архітектура дозволяє легко масштабуватися та управляти великими обсягами даних, роблячи її популярним вибором для веб-додатків, комерційних програм та даних, що швидко змінюються.

Pandas є основною бібліотекою для аналізу даних у Python, надзвичайно популярною в галузях даних, фінансів, наукових досліджень та багатьох інших. Вона надає швидкі, гнучкі та виразні структури даних, призначені для роботи з "реляційними" або "мітковими" даними, легко інтегруючись з іншими бібліотеками. Pandas значно спрощує завдання очищення, трансформації, аналізу та візуалізації даних. Вона використовує дві основні структури даних: DataFrame та Series, які дозволяють ефективно обробляти великі набори даних, здійснювати

складні обчислення та забезпечують інтуїтивно зрозумілі методи для обробки даних.

NumPy є фундаментальною бібліотекою для наукових обчислень у Python, надаючи потужну підтримку для великих багатовимірних масивів і матриць, разом з великою колекцією математичних функцій для роботи з цими масивами. Важливою особливістю NumPy є її здатність виконувати швидкі векторизовані операції, що робить її ідеальною для широкого спектра наукових обчислень. NumPy є ключовою бібліотекою в екосистемі Python для обробки даних, лежачи в основі більшості операцій з даними, здійснених іншими високорівневими бібліотеками, такими як Pandas, SciPy, Matplotlib.

CUDA від NVIDIA є паралельною обчислювальною платформою та програмним інтерфейсом (API), який дозволяє значно прискорити обчислення, використовуючи потужність графічних процесорів (GPU). CUDA дозволяє розробникам використовувати C/C++ для написання програмного забезпечення, яке виконується паралельно на GPU, значно прискорюючи обробку даних та обчислення. Це особливо корисно в областях, що вимагають великої обчислювальної потужності, таких як глибоке навчання, наукові обчислення, обробка зображень/відео та інші. CUDA стала стандартом у галузі для прискорення вимогливих завдань, зокрема для алгоритмів машинного навчання та обробки великих даних.

Cython - це оптимізатор та компілятор для Python, який дозволяє писати C-подібний код для Python, використовуючи статичну типізацію та компіляцію для підвищення швидкодії. Це інструмент, який робить можливим ефективне використання обчислювальних ресурсів, особливо у випадках, коли потрібно оптимізувати швидкість виконання критичних частин коду. Cython можна використовувати для створення розширень Python, інтеграції з C/C++ бібліотеками та підвищення продуктивності Python-програм. Це чудовий вибір для проектів, де необхідна висока продуктивність, і де Python використовується разом з іншими мовами програмування.

3.2 Програмна реалізація алгоритму виділення блоків тексту

Алгоритм використовується для обробки зображень у різних форматах, таких як jpg, gif, bmp, png, і визначення класів для окремих частин зображень. Основна ідея полягає в застосуванні методу співставлення (matching) текстур на базі локальних двійкових шаблонів.

Для початку, кольорове зображення перетворюється на зображення у відтінках сірого, оскільки метод LBP працює з рівнями сірого. Далі, для кожного пікселя у зображенні сірого обирається його сусіднє оточення, і обчислюється значення LBP для даного пікселя, використовуючи інформацію про його сусідів. Отримавши значення LBP для поточного пікселя, ми внесемо зміни в відповідному місці на масці LBP. Розмір цієї маски такий самий, як у вихідному зображенні, і в ній будуть зберігатися обчислені значення LBP для кожного пікселя.

На зображенні (див. рисунок 3.1), можна помітити, що ми розглянемо 8 пікселів сусідів для обчислення значення LBP.

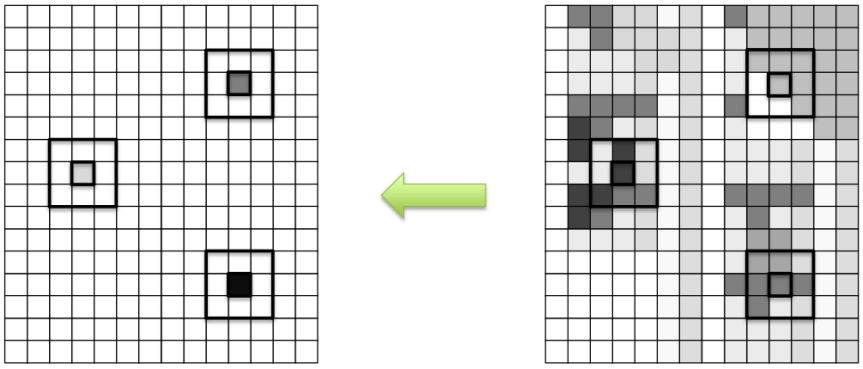


Рисунок 3.1 - Відображення перетворення зображення у відтінках сірого в LBP-маску

Для розрахунку значення LBP для кожного пікселя на зображенні у відтінках сірого, ми порівнюємо центральне значення пікселя зі значеннями його

сусідніх пікселів. Починаючи з будь-якого пікселя сусіднього, ми можемо обрати напрямок обходу пікселів за годинниковою стрілкою або проти неї, але важливо дотримуватися того ж напрямку для всіх пікселів. З огляду на наявність 8 сусідніх пікселів, ми виконуємо 8 порівнянь.

Результати обчислень зберігаються в 8-розрядному двійковому масиві. Якщо значення поточного пікселя рівне або більше значенню сусіднього пікселя, відповідний біт у бінарному масиві встановлюється в 1, а якщо поточне значення менше за значення пікселя сусіда, то відповідний двійковий біт встановлюється в 0.

Цей процес можна побачити на малюнку нижче (Малюнок 2). Для прикладу, центральний (поточний) піксель має значення 7. Ми розпочинаємо порівняння з сусіднього пікселя, який має мітку 0. Значення цього сусіднього пікселя становить 2, що менше, ніж поточне значення пікселя 7, тому ми встановлюємо 0-й біт у 8-розрядному масиві на 0. Потім ми продовжуємо обходити пікселі проти годинникової стрілки. Наступний сусідній піксель має мітку 1 та значення 7, що рівне значенню пікселя поточного. Отже ми встановимо перший розряд в двійковому 8-розрядному масиві в 1. Ми продовжуємо цей процес до досягнення 8-го сусіднього пікселя. Нарешті, 8-розрядний шаблон перетворюється в десяткове число, яке зберігається у відповідній частині LBP-маски (рисунок 3.2).

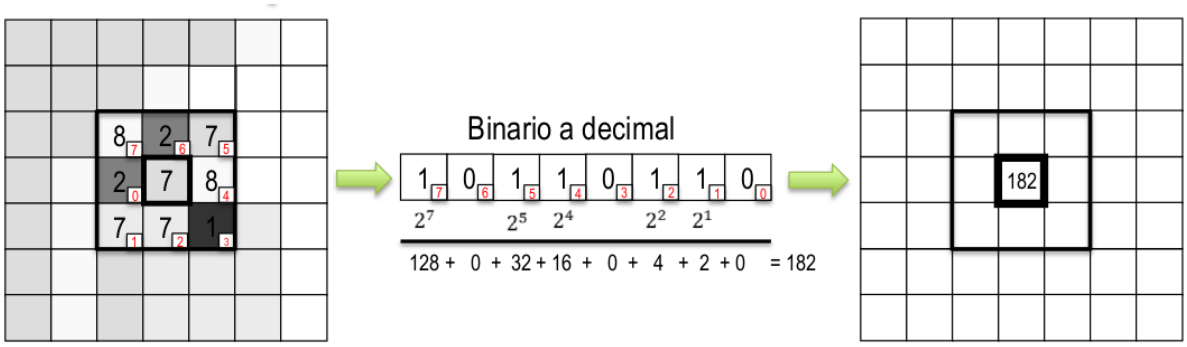


Рисунок 3.2 - Розрахунок значень LBP

Після визначення LBP-маски переходимо до обчислення LBP-гістограми. Значення LBP-маски належать діапазону від 0 до 255, тому розмір LBP-

дескриптора становить 1x256. Після цього ми проводимо нормалізацію LBP-гістограми. Узагальнена схема алгоритму наведена на зображенні нижче:

1. Завантажити кольорове зображення.
2. Перетворити його в зображення у відтінках сірого.
3. Обчислити LBP-маску.
4. Обчислити LBP-гістограму і здійснити нормалізацію.

Однією з головних переваг методу LBP є його інваріантність до освітлення та зсуву. На початку ми обрали 8-зв'язне сусідство, але багато реалізацій використовують сусідство по колу, як показано на малюнку нижче. Ми реалізуємо алгоритм з використанням кругового сусідства.

Перший крок - імпортувати модулі. Ми будемо використовувати модуль cv2 для завантаження зображень, зміни кольорового простору та інших операцій, а також модуль os для взаємодії з файловою системою (зокрема, роботи зі шляхами). Ми також модифікуємо функцію local_binary_pattern з модуля skimage.feature для обчислення LBP-маски.

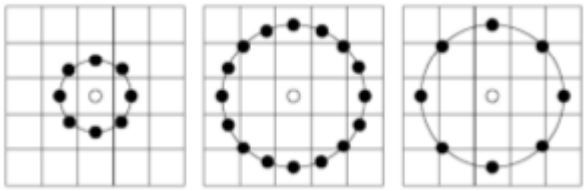
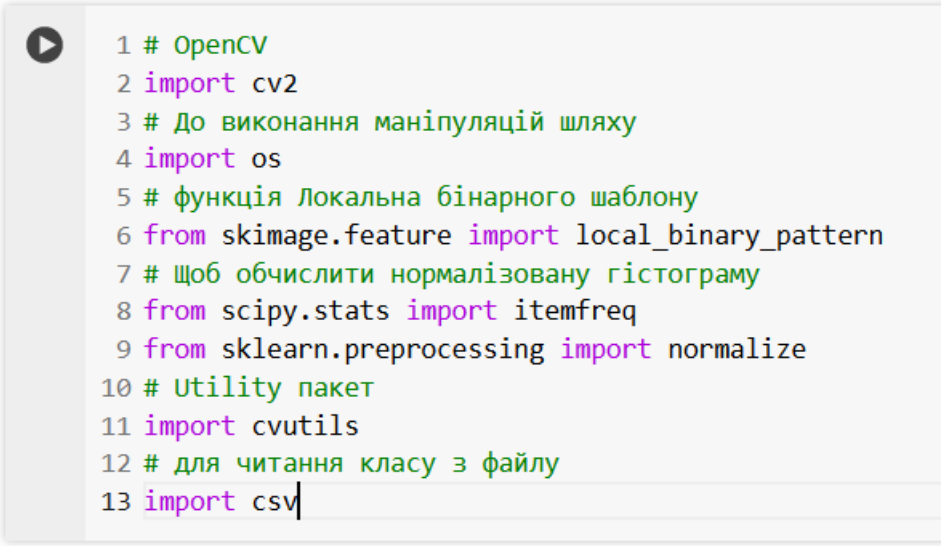


Рисунок 3.3 – Сусідство пікселів по колу

На основі Локальних Бінарних Шаблонів ми проведемо обчислення гістограми, використовуючи функцію scipy.stats.itemfreq, і після цього ми виберемо процедуру sklearn.preprocessing.normalize для нормалізації цієї гістограми. Пакет cvutils є незамінним для роботи з обчислювальною системою машинного зору та пакетами обробки зображень. Нарешті, модуль csv надає можливість ефективно розбирати текстові файли.

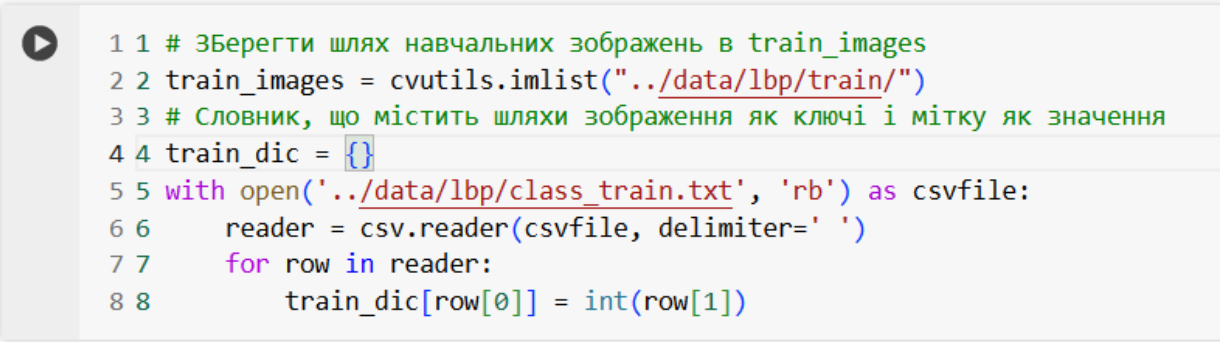


```
# OpenCV
import cv2
# До виконання маніпуляцій шляху
import os
# функція Локальна бінарного шаблону
from skimage.feature import local_binary_pattern
# Щоб обчислити нормалізовану гістограму
from scipy.stats import itemfreq
from sklearn.preprocessing import normalize
# Utility пакет
import cvutils
# для читання класу з файлу
import csv
```

Рисунок 3.4 – Імпорт бібліотек

Підготовка навчального датасету.

Спроектуємо програмний код, який буде автоматично записувати і зберігати шляхи до всіх зображень, що належать до навчального набору.



```
# ЗБерегти шлях навчальних зображень в train_images
train_images = cvutils.imlist("../data/lbp/train/")
# Словник, що містить шляхи зображення як ключі і мітку як значення
train_dic = {}
with open('../data/lbp/class_train.txt', 'rb') as csvfile:
    reader = csv.reader(csvfile, delimiter=' ')
    for row in reader:
        train_dic[row[0]] = int(row[1])
```

Рисунок 3.5 – Індексація датасету

У другому рядку коду, ми здійснюємо процес зберігання усіх шляхів до зображень, які входять до навчального набору, і зберігаємо їх у вигляді списку під назвою "train_images".

Далі, в четвертому рядку, ми створюємо словник під назвою "train_dic", який міститиме інформацію про ім'я зображення та відповідну мітку класу.

З 5-го по 8-й рядок коду, ми зчитуємо рядки з файлу "class_train.txt", який був описаний раніше, і додаємо запис «ключ-значення» (назва зображення та відповідна мітка класу) до словника "train_dic".

Отже, об'єкт "train_images" включає у себе всі шляхи до зображень, які були згадані.

```
train_images  = ['../data/lbp/train/text-1.jpg', '../data/lbp/train/text-2.jpg',
 '../data/lbp/train/text-3.jpg', '../data/lbp/train/text-4.jpg',
'../data/lbp/train/text-5.jpg', '../data/lbp/train/text-6.jpg']
```

Рисунок 3.6 - Список метаінформації

Провести розрахунок гістограм LBP (локальних бінарних шаблонів) - це перший етап. Подалі, наступний крок полягає в обчисленні нормалізованих гістограм LBP для навчальних зображень.

```
# Скласти список для зберігання Гістограм LBP, адреси зображень і відповідної мітки
X_test = []
X_name = []
y_test = []

```

Рисунок 3.7 - Списки для збпереження шляхів

Початок циклу обробки тренувальних зображень

```
# Для кожного зображення в навчальній множині обчислити гістограму LBP
# і модифікують випробування X_, ім'я X_i у_випробування
for train_image in train_images:
    # Read the image
    im = cv2.imread(train_image)
    # Convert до шкали яскравості, оскільки LBP працює над рівнів сірого зображенням
    im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
    radius = 3
    # Число точок, які розглядаються як сусіди
    no_points = 8 * radius
    # Однорідний LBP використано
    lbp = local_binary_pattern(im_gray, no_points, radius, method='uniform')
```

Рисунок 3.8 – Цикл по всім тренувальним зображенням

Обчислення гітограми для ознаки.

```
18 18     # Обчислити гістограму
19 19     x = itemfreq(lbp.ravel())
20 20     # Нормалізуйте гістограму
21 21     hist = x[:, 1]/sum(x[:, 1])
22 22     # Приєднати шлях зображення в X_name
23 23     X_name.append(train_image)
24 24     # Приєднайте гістограму до X_name
25 25     X_test.append(hist)
26 26     # Приєднайте класу мітку в y_випробуванні
27 27     y_test.append(train_dic[os.path.split(train_image)[1]])
```

Рисунок 3.9 - Обчислення гітограми

Візуалізація

```
29 29 # Відобразити навчальні зображення
30 30 nrows = 2
31 31 ncols = 3
32 32 fig, axes = plt.subplots(nrows,ncols)
33 33 for row in range(nrows):
34 34     for col in range(ncols):
35 35         axes[row][col].imshow(cv2.cvtColor(cv2.imread(X_name[row*ncols+col]), cv2.COLOR_BGR2RGB))
36 36         axes[row][col].axis('off')
37 37         axes[row][col].set_title("{}".format(os.path.split(X_name[row*ncols+col])[1]))
```

Рисунок 3.10 - Візуалізація

Дігарама послідовності на рисунку 3. На діаграмі послідовності ми представимо взаємодію між різними об'єктами чи компонентами в процесі обробки зображення.

1. Клієнтський код, який ініціює процес.
2. Об'єкт обробки зображень, який виконує основні операції, такі як завантаження та обробка зображення.
3. Бібліотека OpenCV (cv2), що використовується для обробки зображень.
4. Функції для обчислення LBP та гістограми.
5. Об'єкт візуалізації для відображення зображень.

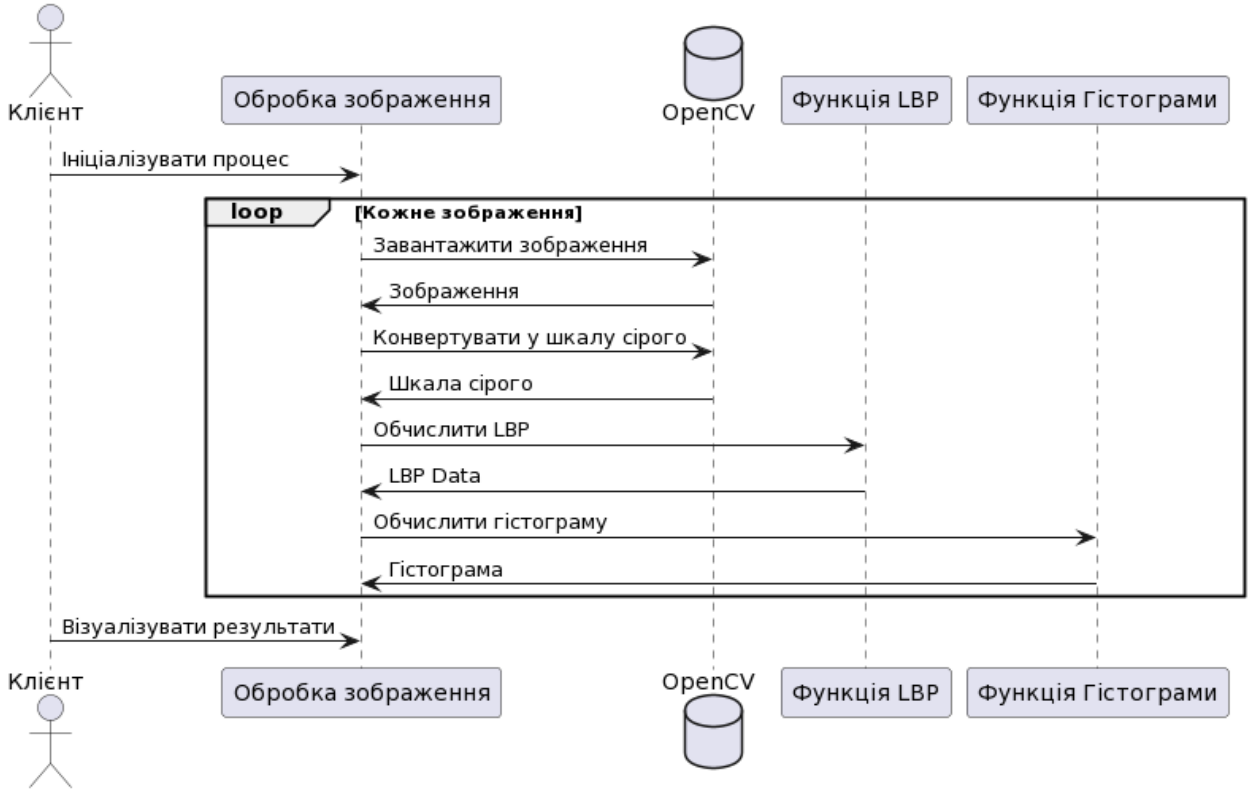


Рисунок 3.11 - Дігарама послідовності алгоритму

На діаграмі активності ми будемо представляти основні етапи обробки зображень та зберігання результатів.

1. Ініціалізація списків для зберігання даних (X_test, X_name, y_test).
2. Цикл по зображенням (for train_image in train_images).
   - Завантаження зображення.
   - Конвертація зображення у шкалу сірого.
   - Обчислення LBP.
   - Обчислення гістограми LBP.
   - Нормалізація гістограми.
   - Зберігання гістограми та мітки класу.
3. Візуалізація зображень.

Діаграма послідовності на рисунку 3.12.

Рисунок 3.12 - Діаграма послідовності

Розкриємо код для більшого розуміння. У рядках 2-4 створюються три списки:

1. x_test містить нормалізовані гістограми LBP.
2. x_name містить адреси зображень.
3. y_test містить відповідні мітки класу.

Кожен індекс у цих трьох списків відповідає одному і тому ж зображенню. В рядку з 8 по 27 знаходиться цикл, який обчислює нормалізовані гістограми ЛБШ для тренувальних зображень. Далі в рядку 10, читаємо поточне зображення за допомогою функції cv2.imread. Потім зображення перетворюється у відтінки сірого, оскільки LBP працює з зображеннями у цьому кольоровому просторі.

У рядку 17 обчислюємо маску ЛБШ, встановлюючи радіус сусідства на 3 і кількість точок (пікселів) на 24. Після створення маски, ми обчислимо гістограму ДБШ в рядку 19 і нормалізуємо її в рядку 21. Потім додаємо гістограму до списку

x_test в рядку 25. Також додаємо назву зображення в список x_name і мітку класу зображення в y_test.

Далі, в рядках з 30 по 37, виводяться навчальні зображення, що були згенеровані. Результат відображений на рисунку 3.13.

Навчальні зображення анотовані вручну (див розділ 1).

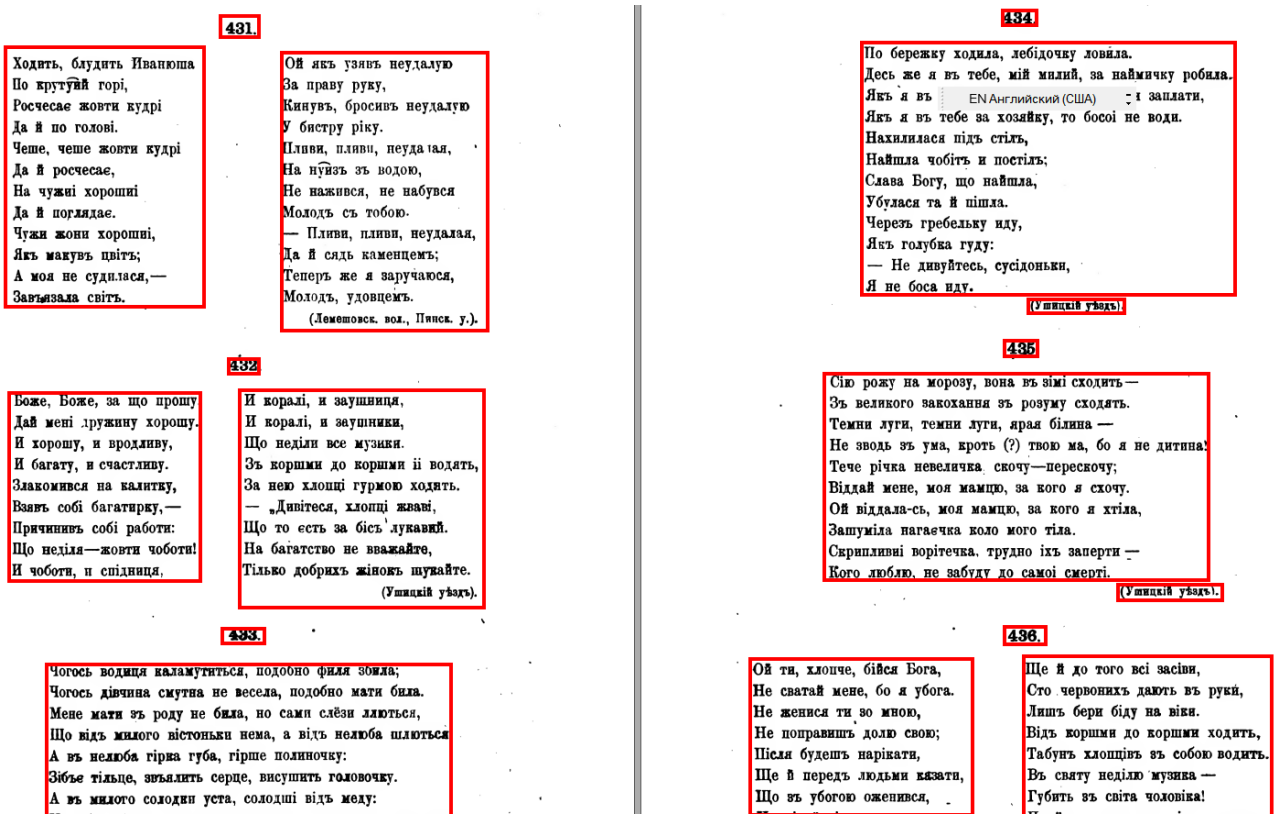


Рисунок 3.13 – Тренувальні зображення

Отримання тестових зображень важливо для перевірки правильності функціонування алгоритму LBP. Для цього я взяв три зображення з кожного класу, які не були включені до навчальної вибірки. Процес полягає в зчитуванні шляхів до цих трьох зображень і їх додаванні до списку `test_images`. Також, аналогічно до словника `train_dic`, я створюю словник `test_dic`, що містить імена зображень та відповідні мітки класів, аналогічно до попереднього етапу.

```
1 1 # Зберегти шлях тестових зображень у випробувальних_зображеннях
2 2 test_images = cvutils.imlist("../data/lbp/test/")
3 3 # Словник, що містить шляхи зображення як ключі і відповідну мітку як значення
4 4 test_dic = {}
5 5 with open('../data/lbp/class_test.txt', 'rb') as csvfile:
6 6     reader = csv.reader(csvfile, delimiter=' ')
7 7     for row in reader:
8 8         test_dic[row[0]] = int(row[1])
```

Рисунок 3.14 - Початок циклу

Вміст test_dic і test_images

```
1 # test_images
2 ['../data/lbp/test/text-1.png', '../data/lbp/test/text-1.jpg', '../data/lbp/test/text-1.jpg']
3 # test_dic
4 {'text-1.png': 0, 'textg-1.jpg': 1, 'textcheckered-1.jpg': 2}
5
```

Рисунок 3.15 - Список шляхів датасету

Проведемо обчислення відстані Хі-квадрат для кожного тестового зображення. Для цього, спочатку, ми обчислюємо нормалізовані гістограми LBP для кожного зображення в тестовій вибірці. Потім ми порівнюємо гістограми ЛБШ що були нормалізовані для тренувальних зображень з тестовими, використовуючи значення відстані Хі-квадрат. Результати сортуємо за значенням цієї відстані і відображаємо їх у відсортованому порядку. Чим менше значення відстані Хі-квадрат, тим краща відповідність між зображеннями.

```
1  for test_image in test_images:
2       # Читайти зображення
3      im = cv2.imread(test_image)
4      # Convert , оскільки LBP працює з зображенням рівнів сірого
5      im_gray = cv2.cvtColor(im, cv2.COLOR_BGR2GRAY)
6      radius = 3
7      # Число точок, які розглядаються як neighbourers
8      no_points = 8 * radius
9      # використано Однорідний LBP
10     lbp = local_binary_pattern(im_gray, no_points, radius, method='uniform')
```

Рисунок 3.16 - Цикл по кожному зображенню

У перших 14 рядках коду ми виконуємо обчислення нормалізованих гістограм LBP для тестових зображень, аналогічно до попереднього етапу, коли ми робили це для навчальних зображень.

```
11      # Обчислити гістограму
12      x = itemfreq(lbp.ravel())
13      # Нормалізуйте гістограму
14      hist = x[:, 1]/sum(x[:, 1])
15      # Display the query image
16      cvutils.imshow("** Query Image -> {}**".format(test_image), im)
17      results = []
```

Рисунок 3.17 - Обчислення гістограм

На 21-м рядку обчислюємо Хі-квадрат відстань між поточним тестовим зображенням і всіма навчальними зображеннями послідовно, використовуючи функцію `cv2.compareHist`.

```
18      # For each image in the training dataset
19      # Calculate the chi-squared distance and the sort the values
20      for index, x in enumerate(X_test):
21          score = cv2.compareHist(np.array(x, dtype=np.float32), np.array(hist, dtype=np.float32), cv2.cv.CV
22          results.append((X_name[index], round(score, 3)))
23      results = sorted(results, key=lambda score: score[1])
24      # Display the results
25      nrows = 2
26      ncols = 3
27      fig, axes = plt.subplots(nrows,ncols)
28      fig.suptitle("** Scores for -> {}**".format(test_image))
29      for row in range(nrows):
30          for col in range(ncols):
31              axes[row][col].imshow(cv2.cvtColor(cv2.imread(results[row*ncols+col][0]), cv2.COLOR_BGR2RGB))
32              axes[row][col].axis('off')
33              axes[row][col].set_title("Score {}".format(results[row*ncols+col][1]))
```

Рисунок 3.18 - Обчислення відстані між ознаками

Після обчислень на 22-м рядку отримані значення відстаней сортуються. Рядки з 25 по 33 відображають тренувальні зображення, супроводжуючи їх відповідними значеннями відстаней. Це дозволяє нам візуально оцінити, які навчальні зображення мають найбільшу відповідність з тестовим зображенням за значенням відстані Хі-квадрат.

Діаграма послідовності.

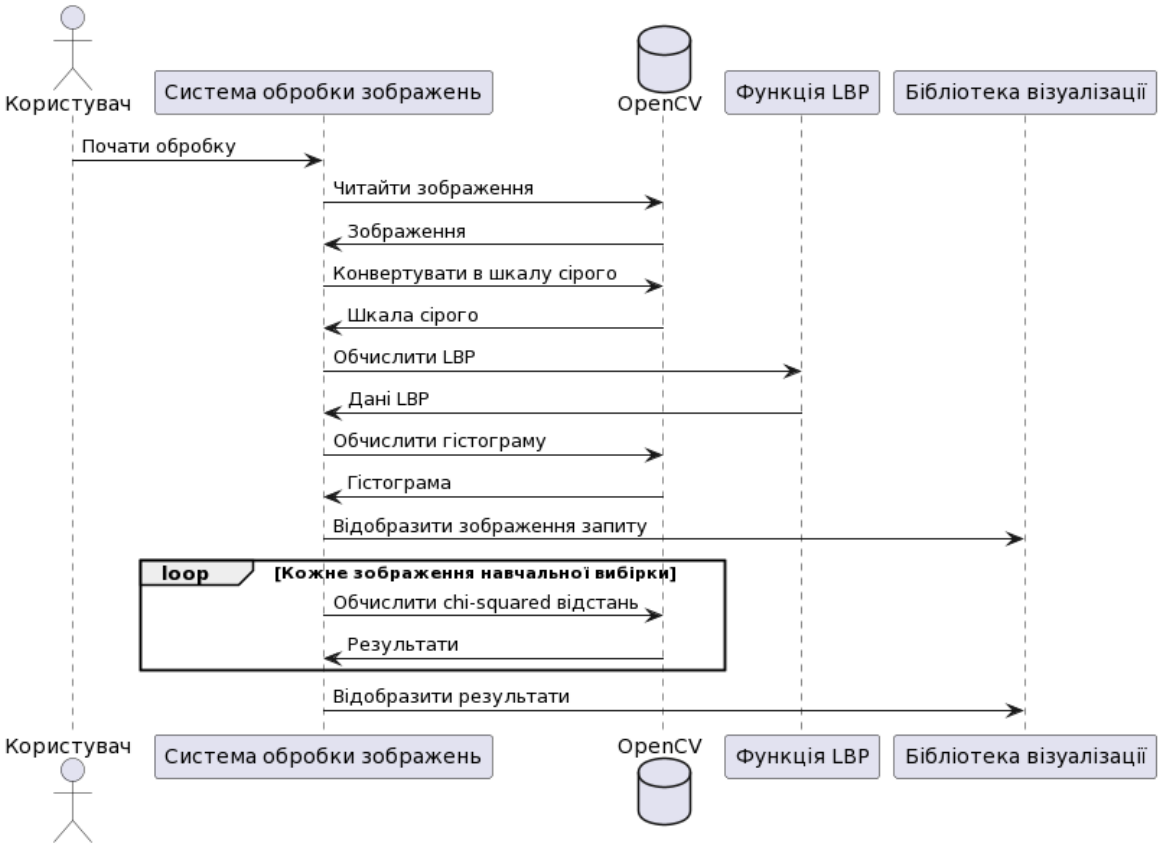


Рисунок 3.19 - Діаграма послідовності алгоритму

3.3 Експериментальні дослідження алгоритмів

Для створення бази даних, що зберігає метаінформацію про тренувальні зображення для експериментів з OCR (оптичного розпізнавання символів), ми можемо розробити модель, яка включає наступні основні елементи:

1. Images: Таблиця для зберігання інформації про кожне зображення.
   - ID: Унікальний ідентифікатор зображення.
   - FileName: Назва файлу зображення.
   - FilePath: Шлях до файлу зображення.
   - SourceType: Тип джерела (книга, журнал).
   - UploadDate: Дата завантаження зображення.
2. TextRegions: Таблиця для зберігання інформації про області тексту в зображеннях.

– RegionID: Унікальний ідентифікатор області.

– ImageID: Ідентифікатор зображення, до якого належить область.

– Coordinates: Координати області на зображенні.

– TextContent: Текст, витягнутий з цієї області.

3. Books: Таблиця для додаткової інформації про книги.

– BookID: Унікальний ідентифікатор книги.

– Title: Назва книги.

– Author: Автор книги.

– PublicationYear: Рік публікації.

4. Journals: Таблиця для додаткової інформації про журнали.

– JournalID: Унікальний ідентифікатор журналу.

– Title: Назва журналу.

– IssueNumber: Номер випуску.

– PublicationDate: Дата публікації.

Відносини між таблицями:

– кожне зображення може містити кілька текстових областей.

– зображення може бути пов'язане з книгою або журналом, в залежності від джерела.

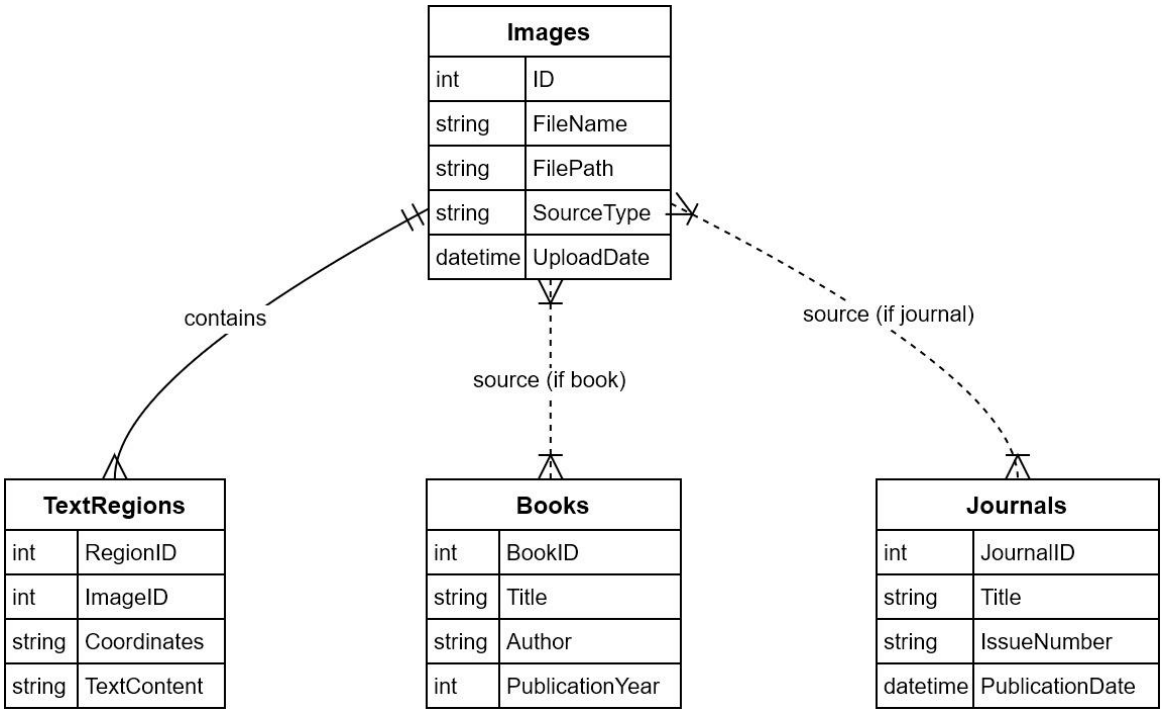


Рисунок 3.20 – Модель бази даних

Ця модель бази даних дозволяє зберігати детальну інформацію про кожне зображення, області тексту в межах цих зображень, а також про джерела цих зображень (книги та журнали).

Для ручної анотації текстових блоків на зображеннях, можна розробити клас TextBlockAnnotator. Цей клас буде взаємодіяти з раніше розробленою базою даних метаінформації, дозволяючи зберігати інформацію про анотовані області. Ось можливий опис цього класу:

Клас TextBlockAnnotator. Властивості:

– image_id: Ідентифікатор зображення, яке анотується.

– db_connection: Об'єкт для підключення до бази даних.

– annotated_regions: Список, що містить анотовані області на зображенні.

Методи:

– __init__(self, image_id, db_connection): Конструктор класу, що ініціалізує основні властивості.

– load_image(self): Завантажує зображення за його ідентифікатором з бази даних.

– annotate_region(self, coordinates, text_content): Додає анотацію для заданої області. coordinates - це кортеж, що визначає координати області, а text_content - текст, витягнутий з цієї області.

– save_annotations(self): Зберігає всі анотації для поточного зображення у базі даних.

– display_image(self): Показує зображення з анотаціями для перевірки та редагування.

– remove_annotation(self, region_id): Видаляє анотацію за заданим ідентифікатором області.

Приклад використання:

```
db_connection = DatabaseConnection(config)
annotator = TextBlockAnnotator(image_id=123,
db_connection=db_connection)

# Завантаження зображення
annotator.load_image()
```

```
# Додавання анотацій
annotator.annotate_region((x1, y1, x2, y2), "Текст з області")
annotator.annotate_region((x3, y3, x4, y4), "Інший текст")

# Збереження анотацій
annotator.save_annotations()

# Відображення зображення для перевірки
annotator.display_image()
```

У цьому класі передбачена можливість взаємодії з базою даних для збереження анотацій. Також є методи для видалення або зміни анотацій, якщо це необхідно. Важливо, що клас повинен володіти функціональністю для роботи з графічним інтерфейсом, щоб користувач міг зручно виконувати анотацію зображень.

Створимо діаграму класів та діаграму активності для системи ручної анотації текстових блоків на зображеннях, яка взаємодіє з базою даних метаінформації.

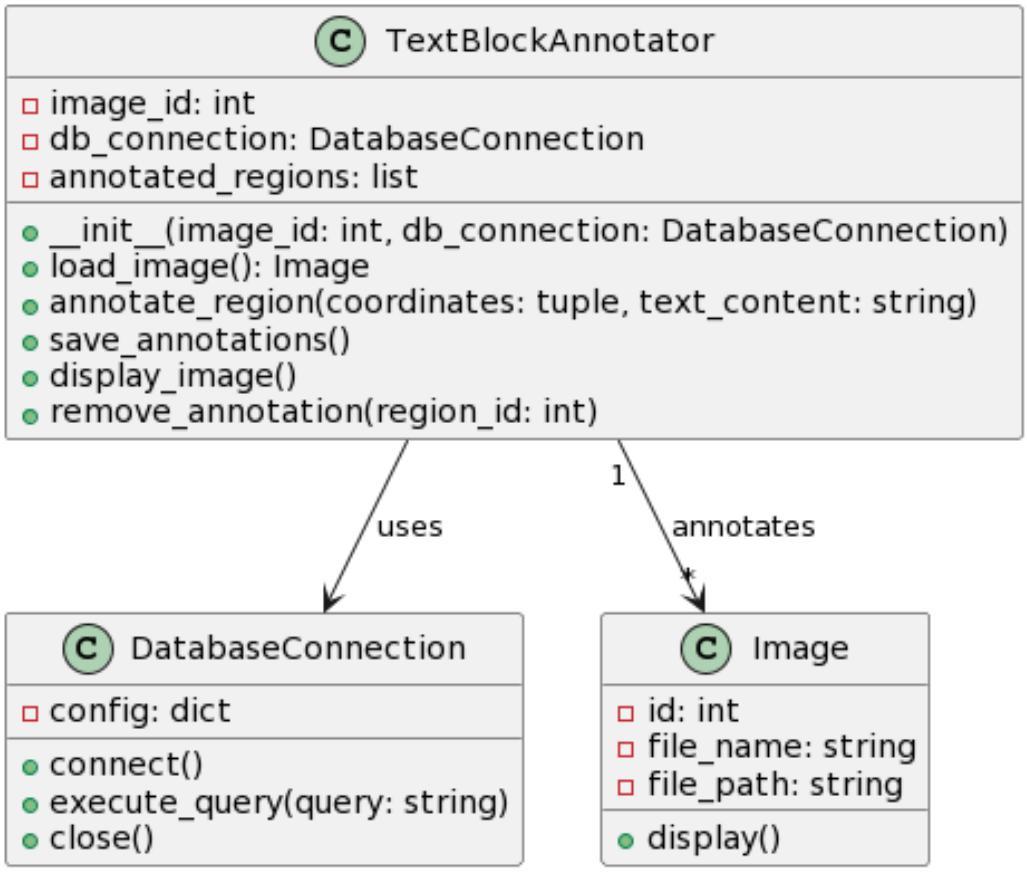


Рисунок 3.21 -Діаграма класів для системи ручної анотації текстових блоків

Ця діаграма класів показує структуру та взаємозв'язки між класами в системі анотації. Клас TextBlockAnnotator взаємодіє з DatabaseConnection для доступу до бази даних і з класом Image для відображення та анотації зображень.

Діаграма активності відображає процес анотації зображень в системі. Починається з створення об'єкта TextBlockAnnotator, потім ідуть кроки завантаження зображення, анотації різних областей, збереження анотацій в базі даних і відображення анотованого зображення для перевірки.

Рисунок 3.22- Діаграма активності для системи ручної анотації текстових блоків

Неправильна детекції блоків тексту , тобто параграфів та колонок тексту, веде до фрагментації речень тексту. Ми можемо використати міру фрагментації (змішування) тексту як міру яка покаже покращення точності при застосуванні розроблених алгоритмів. Для проведення експериментів обрано датасет із 100 зображень попередньо анотованих зображень книг та газет. В якості системи

розпізнавання обрано Tesseract OCR. На діаграмі показано значення точності для двох видів зображень книг та газет.

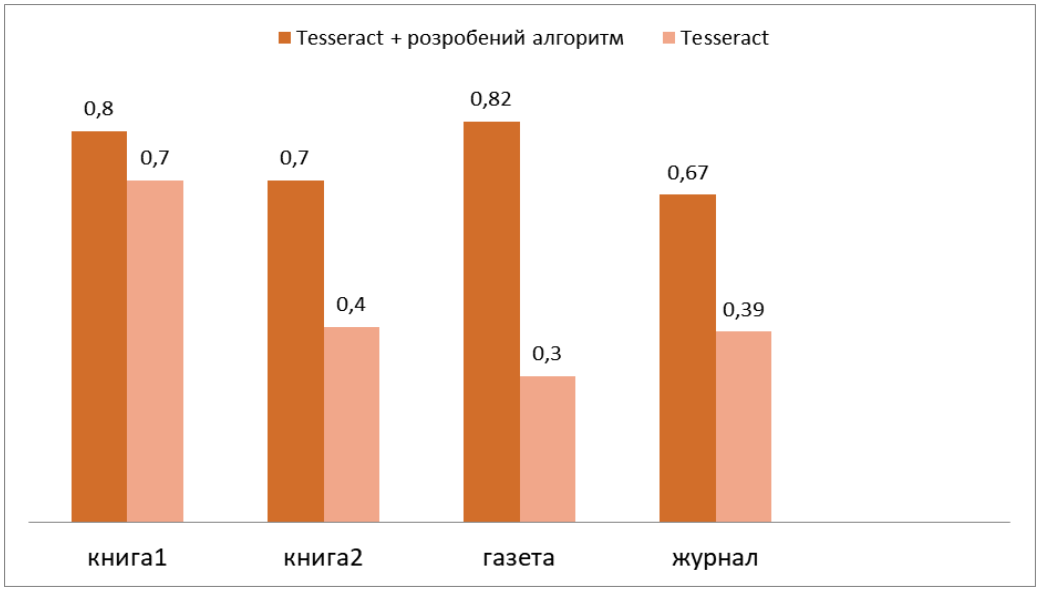


Рисунок 3.3 - Графік точності розпізнавання для системи із розробленим алгоритмом

Загалом, розроблений алгоритм показує кращу точність для всіх класів, з найвищим показником для категорії "газета" (0,82), і найнижчим у "книга2" (0,7). Простий "Tesseract" має найвищий показник точності у "книга1" (0,7) та найнижчий у "журнал" (0,39). Досягнуто значне покращення точності розпізнавання для класів "Книга2", "Газета", та "Журнал". Для цих категорій документів розроблений алгоритм значно покращив ефективність розпізнавання тексту – від 71% до 173%.

3.4 Висновки до розділу

Розроблено алгоритм детекції блоків тексту з різними шрифтами та області «не тексту», що дозволило підвищити точність оптичного розпізнавання тексту загалом. Здійснено програмну реалізацію розроблених алгоритмів на мові Python з використанням бібліотек.

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи одержані наступні результати:

1. Здійснено аналітичний огляд методів детекції блоків тексту та алгоритмів текстурного аналізу зображень. Розглянуто підходи до побудови векторів текстурних ознак.

2. Розроблено алгоритми обчислення текстурних ознак на основі локальних бінарних шаблонів та матриць розподілу рівнів сірого що дозволили обчислювати ознаки для блоків різних шрифтів. Класифікація блоків різних шрифтів виконується за допомого відстані Хі-квадрат в просторі текстурних ознак.

3. Розроблено алгоритм детекції блоків тексту з різними шрифтами та області «не тексту», що дозволило підвищити точність оптичного розпізнавання тексту загалом.

4. Здійснено програмну реалізацію розроблених алгоритмів на мові Python з використанням бібліотек

5. Здійснено програмну реалізацію та експериментальне дослідження розроблених алгоритмів. Досягнуто значне покращення точності розпізнавання для класів "Книга2", "Газета", та "Журнал". Для цих категорій документів розроблений алгоритм значно покращив ефективність розпізнавання тексту – від 71% до 173%.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Винницька Т.П., Костенюк А.В. Аналіз структури тексту при оптичному розпізнаванні документів. Матеріали науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ-2023)., м. Тернопіль, 5 груд. 2023 р. С. 24.

2. Костенюк А.В., Винницька Т.П. Реконструкція шрифтів при оптичному розпізнаванні тексту. Матеріали науково-практичної конференції молодих вчених і студентів «Інтелектуальні комп'ютерні системи та мережі» (ІКСМ-2023)., м. Тернопіль, 5 груд. 2023 р. С. 49.

3. Березький О.М., Дубчак Л.О., Мельник Г.М. Методичні рекомендації до виконання кваліфікаційної роботи освітнього ступеня «Магістр». Спеціальність: комп'ютерна інженерія. Магістерська програма – «Комп'ютерна інженерія» / Під ред. О.М. Березького. Тернопіль: ЗУНУ, 2020. 32 с.

4. 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.

5. Zhao N., Cao Y., Lau R. Modeling fonts in context: Font prediction on web designs. *Computer Graphics Forum*. 2018. P. 385--395.

6. Synthtiger: Synthetic text image generator towards better text recognition models / M. Yim et al. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*. 2021. P. 109--124.

7. Deepfont: Identify your font from an image / Z. Wang et al. *Proceedings of the International Conference on Multimedia*. 2015. P. 451--459.

8. Tibshirani R., Walther G., Hastie T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 2001. Vol. 63, no. 2. P. 411--423.

9. Takeshita K., Shioyama J., Uchida S. Label or message: A large-scale experimental survey of texts and objects co-occurrence. *Proceedings of the International Conference on Pattern Recognition (ICPR)*. 2021. P. 6227--6234.

10. Gupta A., Vedaldi A., Zisserman A. Synthetic Data for Text Localisation in Natural Images. CoRR. 2016. Abs/1604.06646. URL: http://arxiv.org/abs/1604.06646.

11. TextBoxes: A Fast Text Detector with a Single Deep Neural Network / M. Liao et al. CoRR. 2016. Abs/1611.06779. URL: http://arxiv.org/abs/1611.06779.

12. EAST: An Efficient and Accurate Scene Text Detector / X. Zhou et al. URL: 10.48550/ARXIV.1704.03155.

13. Text/Non-Text Separation from Handwritten Document Images Using LBP Based Features: An Empirical Study / S. Ghosh et al. Journal of Imaging. 2018. Vol. 4, no. 4. URL: 10.3390/jimaging4040057.

14. Dropout: a simple way to prevent neural networks from overfitting / N. Srivastava et al. *The Journal of Machine Learning Research*. 2014. Vol. 15, no. 1. P. 1929--1958.

15. Serif or sans: Visual font analytics on book covers and online advertisements / Y. Shinahara et al. *Proceedings of the International Conference on Document Analysis and Recognition (ICDAR)*. 2019. P. 1041--1046.

16. Exploratory font selection using crowdsourced attributes / P. O'Donovan et al. *ACM Transactions on Graphics*. 2014. Vol. 33, no. 4. P. 1--9.

17. Distributed representations of words and phrases and their compositionality / T. Mikolov et al. *Proceedings of the Advances in Neural Information Processing Systems (NIPS)*. 2013.

18. Efficient estimation of word representations in vector space / T. Mikolov et al. *International Conference on Learning Representations*. 2013.

19. Matsumura S., Choi S., Aizawa K. Font search across various languages based on multimodal learning. *Proceedings of the Conference on Multimedia Information Processing and Retrieval (MIPR)*. 2020. P. 173--176.

20. Kulahcioglu T., Melo G. D. Fonts like this but happier: A new way to discover fonts. *Proceedings of the International Conference on Multimedia*. 2020. P. 2973--2981.

21. Kulahcioglu T., Melo G. D. Paralinguistic recommendations for affective word clouds. *Proceedings of the International Conference on Intelligent User Interfaces*. 2019. P. 132--143.

22. Kulahcioglu T., Melo G. D. Predicting semantic signatures of fonts. *Proceedings of the International Conference on Semantic Computing (ICSC)*. 2018. P. 115--122.

23. Openimages: A public dataset for large-scale multi-label and multi-class image classification / I. Krasin et al. 2017.

24. Kingma D., Ba J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*. 2014.

25. Visual font pairing / S. Jiang et al. *IEEE Transactions on Multimedia*. 2019. Vol. 22, no. 8. P. 2086--2097.

26. Ikoma M., Iwana B., Uchida S. Effect of text color on word embeddings. *Proceedings of the International Workshop on Document Analysis Systems (DAS)*. 2020. P. 341--355.

27. Deep residual learning for image recognition / K. He et al. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016. P. 770--778.

28. BERT: pre-training of deep bidirectional transformers for language understanding / J. Devlin et al. *Proceedings of the Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 2019. P. 4171--4186.

29. Choi S., Matsumura S., Aizawa K. Assist users' interactions in font search with unexpected but useful concepts generated by multimodal learning. *Proceedings of the International Conference on Multimedia Retrieval*. 2019. P. 235--243.

30. Large-scale visual font recognition / G. Chen et al. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014. P. 3598--3605.

31. Character region awareness for text detection / Y. Baek et al. *Proceedings of the Conference on Computer Vision and Pattern Recognition (CVPR)*. 2019. P. 9365--9374.

32. Analyzing Font Style Usage and Contextual Factors in Real Images / N. Yasukochi et al. *Document Analysis and Recognition - ICDAR 2023 - 17th International Conference, San Jose, CA, USA, August 21-26, 2023, Proceedings, Part III*. 2023. P. 331--347. URL: 10.1007/978-3-031-41682-8_21.

33. Sundermeyer M., Schlüter R., Ney H. LSTM neural networks for language modeling. *Proc. Interspeech 2012*. 2012. P. 194--197. URL: 10.21437/Interspeech.2012-65.

34. Papers with Code - SVT Dataset. The latest in Machine Learning | Papers With Code. URL: https://paperswithcode.com/dataset/svt.

35. IAM Handwriting Database. Research Group on Computer Vision and Artificial Intelligence – Computer Vision and Artificial Intelligence. URL: https://fki.tic.heia-fr.ch/databases/iam-handwriting-database.

36. Europarl Parallel Corpus. Statistical and Neural Machine Translation. URL: https://www.statmt.org/europarl/.

37. The Analysis of Performance for Priority Distinction Double-queue and Double-server Communication Network / J. L. Xiong et al. *International Journal of Communication*. 2014. Vol. 3. URL: https://api.semanticscholar.org/CorpusID:62382654.

38. Du J.-q. The strategy research and method realization for the computer network flow control. *International Conference on Graphic and Image Processing*. 2013. URL: https://api.semanticscholar.org/CorpusID:62720079.

39. Gao F., Liu Q., Zhan H. Research of SIP DoS Defense Mechanism Based on Queue Theory. *International Conference on Computer Science, Environment, Ecoinformatics, and Education*. 2011. URL: https://api.semanticscholar.org/CorpusID:14887144.

40. Minkevius S., Sakalauskas L. On the law of iterated logarithm for extreme queue length in an open queueing network. *International Journal of Computer Mathematics: Computer Systems Theory*. 2021. Vol. 6. P. 220 - 235. URL: https://api.semanticscholar.org/CorpusID:237548367.

41. Stuckey N. Stochastic Estimation and Control of Queues within a Computer Network. 2012. URL: https://api.semanticscholar.org/CorpusID:59897689.

42. RouteNet-Erlang: A Graph Neural Network for Network Performance Evaluation / M. F. Galmes et al. *IEEE INFOCOM 2022 - IEEE Conference on Computer Communications*. 2022. P. 2018-2027. URL: https://api.semanticscholar.org/CorpusID:247159041.

43. The Analysis of Performance for Priority Distinction Double-queue and Double-server Communication Network / J. L. Xiong et al. *International Journal of Communication*. 2014. Vol. 3. URL: https://api.semanticscholar.org/CorpusID:62382654.

44. Andreji M. Analysis of the influence of queue type on packet delay in computer networks. 2015. URL: https://api.semanticscholar.org/CorpusID:195971842.

45. Queue Control Model in a Clustered Computer Network using M/M/m Approach / A. Ejem et al. *International Journal of Computer Trends and Technology*. 2016. Vol. 35. P. 12-20. URL: https://api.semanticscholar.org/CorpusID:38081651.

46. Розрахунок ефективності використання обчислювальних ресурсів самовідновлювальної комп'ютерної системи / N. Kuchuk та ін. *Системи управління, навігації та зв'язку. Збірник наукових праць*. 2021. Т. 3, № 65. С. 92–95. URL: 10.26906/sunz.2021.3.092.

47. Poslaiko N. I. Дослідження стаціонарного режиму в системі масового обслуговування типу gi /m /1 зі слабкою післядією. *Problems of applied mathematics and mathematic modeling*. 2022. URL: 10.15421/322119.

48. Порівняльна ефективність класифікаторів зображень під час розпізнавання зон інтересу при лапароскопічних втручаннях / М. Р. Баязітов та ін. Medical Informatics and Engineering. 2020. № 2. С. 62–69. URL: 10.11603/mie.1996-1960.2020.2.11175.

49. Франчук Н. П. Стан та перспективи технологій машинного перекладу тексту. Theory and methods of e-learning. 2014. Т. 3. С. 319–325. URL: 10.55056/e-learn.v3i1.356.

50. Ярошенко О. Огляд інструментів ocr для завдання розпізнавання таблиць і графіків у документах. Information, Computing and Intelligent systems. 2022. № 3. URL: 10.20535/2708-4930.3.2022.265200.

51. Денисенко О. Дослідження та розробка системи розпізнавання тексту на зображенні за допомогою згорткової нейронної мережі. LES TENDANCES ACTUELLES DE LA MONDIALISATION DE LA SCIENCE MONDIALE - VOLUME 1. 2020. URL: 10.36074/03.04.2020.v1.30.

52. Польшакова О., Мальченко Є. Представлення ефективного методу розпізнавання тексту на зображенні, заснованого на критерії схожості структурних моделей символів. Адаптивні системи автоматичного управління. 2021. Т. 1, № 38. С. 50–56. URL: 10.20535/1560-8956.38.2021.233184.

53. Tibshirani R., Walther G., Hastie T. Estimating the number of clusters in a data set via the gap statistic. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*. 2001. Vol. 63, no. 2. P. 411--423.

54. Takeshita K., Shioyama J., Uchida S. Label or message: A large-scale experimental survey of texts and objects co-occurrence. *Proceedings of the International Conference on Pattern Recognition (ICPR)*. 2021. P. 6227--6234.