Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Методи сегментації та шумопригнічення у комп’ютерній системі обробки сканованих документів

(тема)

Виконав:

студент II курсу, групи КСМм-20-1

Бас Є.О.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма

Комп’ютерні системи та мережі
(повна назва освітньої програми)

Керівник: доц.Барковська О.Ю.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2021 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)

Освітня програма Комп’ютерні системи та мережі
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Басу Євгенію Олеговичу
(прізвище, ім’я, по батькові)

1. Тема роботи Методи сегментації та шумопригнічення у комп’ютерній системі обробки сканованих документів

затверджена наказом по університету від “ 05 ” листопада 2021 р. № 1656 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 13 грудня 2020 р.

3. Вхідні дані до роботи
Навчальна вибірка графічних файлів, тестова вибірка графічних файлів, графічна карта

4. Перелік питань, що потрібно опрацювати у роботі
Розробка системи впорядкованого зберігання сканованих документів
Дослідження методів роботи із зображеннями
Дослідження методів роботи із текстом
Оцінка таймінгу роботи алгоритму обробки зображення
Оцінка таймінгу роботи етапу аналізу тексту
Оцінка якості роботи системи впорядкованого зберігання сканованих документів

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів) 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Дослідження загальних параметрів систем	09.11.21-14.11.21	
	розпізнавання сканованих документів		
2	Дослідження існуючих методів	15.11.21-21.11.21	
	шумопригнічення та сегментації зображень		
3	Розробка системи впорядкованого зберігання	22.11.21-30.11.21	
	зберігання сканованих документів		
4	Адаптація методів шумопригнічення	01.12.21-06.12.21	
	та сегментації під різні типи обчислювачів		
5	Оцінка таймінгу роботи розробленої системи	07.12.21-09.12.21	
	системи		
6	Оформлення пояснювальної записки	09.12.21-10.12.21	

Дата видачі завдання 8 листопада 2021 р.

Студент _____
(підпис)

Керівник роботи _____ доц..Барковська О.Ю.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 70 с., 31 рис., 6 табл., 1 дод., 13 джерел.

OCR, ОБРОБКА ЗОБРАЖЕНЬ, ШУМОПРИГНІЧЕННЯ, АНАЛІЗ ТЕКСТУ, ЛЕМАТИЗАЦІЯ, СТЕМІНГ OPENCV, TESSERACT, SPACY.

Метою кваліфікаційної роботи є роботи є дослідження впливу використання різних методів сегментації та шумопригнічення на точність та швидкість розпізнавання сканованих документів.

У ході виконання кваліфікаційної роботи були розглянуті існуючі методи обробки зображення для шумопригнічення та сегментації, проведено аніліз впливу цих методів на точність роботи системи розпізнавання тексту.

Також була розроблена система впорядкованого зберігання сканованих документів з виділенням ключових слів, яка складається з модуля обробки зображень, модуля розпізнавання тексту та модулів аналізу отриманого тексту. Окрім цього була проведена адаптація методів обробки зображення та тексту для різних типів обчислювачів.

ABSTRACT

Master’s thesis: 70 pages, 31 figures, 6 tables, 1 appendices, 13 sources.

OCR, IMAGE PROCESSING, NOISE REDUCTION, TEXT ANALISYS, LEMMATIZATION, STEMMING OPENCV, TESSERACT, SPACY

The major goal of this thesis is is to study the impact of the use of different methods of segmentation and noise reduction on the accuracy and speed of recognition of scanned documents.

In order to the qualification work, the existing methods of image processing for noise reduction and segmentation were considered, the analysis of the impact of these methods on the accuracy of the text recognition system was analyzed.

A system of orderly storage of scanned documents with keyword selection was also developed, which consists of an image processing module, a text recognition module and modules for text analysis. In addition, image and text processing methods have been adapted for different types of computers.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

OCR – оптичне розпізнавання символів

POS – розбір на частини мови (part-of-speech)

LSTM – довга короткочасна пам’ять (Long short-term memory)

ВСТУП

Великі обсяги інформації, накопичені до поширення комп’ютерів, досі залишаються доступними лише паперових носіях. Переведення цієї інформації в електронний вигляд може суттєво підвищити її доступність широкого кола користувачів. Незважаючи на те, що в даний час більшість документів складається на комп’ютерах, задача створення електронного документообігу ще далека до повної реалізації. Як правило, існуючі системи охоплюють діяльність окремих організацій, а обмін даними між організаціями здійснюється за допомогою традиційних паперових документів.

Виявлення та розпізнавання тексту з зображень може мати безліч функціональних додатків для аналізу документів, таких як допомога людям з ослабленим зором, розпізнавання номерних знаків транспортних засобів, оцінка статті, що містить таблиці, вуличні знаки, карти, діаграми, дослідження зображень на основі ключових слів, розпізнавання сканованих документів, а також індексування відео на основі тексту.

Текстові інформаційні системи стають все більш важливими майже в усіх сферах. У багатьох ситуаціях (наприклад, у звичайних газетах або старих друкованих книгах) джерелом вхідного тексту є документи в їх вихідній паперовій формі, які не можна редагувати. У деяких випадках системи обробки зображень можуть використовуватися для зберігання та вилучення цих документів за допомогою зазначених вручну ключових слів, але повнотекстовий доступ може бути більш ефективним, оскільки він дозволить автоматизувати процес зберігання, індексації та вилучення інформації з повним доступом до всіх ключових слів вмісту [1].

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

Останнім часом прогрес у розпізнаванні образів прискорився завдяки появі безлічі нових технологій, які не тільки складні, а й потребують більше обчислень, таких як оптичне розпізнавання символів (OCR), класифікація документів, комп'ютерний зір, інтелектуальний аналіз даних, розпізнавання форм та біометричні дані. OCR стає невід'ємною частиною сканерів документів і використовується в багатьох програмах, таких як поштова обробка, розпізнавання сценаріїв, банківська справа, безпека (наприклад, аутентифікація паспорта) та ідентифікація мови. Дослідження в цій галузі ведуться вже понад півстоліття, і результати вражають: рівень успішного розпізнавання друкованих символів перевищує 99%, а також значно покращено продуктивність розпізнавання рукописних символів, коли показники розпізнавання перевищили позначку 90%. В даний час багато організацій покладаються на системи оптичного розпізнавання тексту, щоб виключити людську взаємодію для підвищення продуктивності та ефективності.

Область розпізнавання образів – це мультидисциплінарна область, що становить основу інших областей, таких як, обробка зображень, машинний зір і штучний інтелект. Отже, OCR не може застосовуватись без допомоги обробки зображень або штучного інтелекту. Будь-яка система OCR проходить безліч етапів, включаючи збирання даних, попередню обробку, вилучення ознак, класифікацію та постобробку, де найбільш важливим аспектом є попередня обробка, яка необхідна для зміни даних або для виправлення недоліків у процесі збирання даних або для їх підготовки до наступних дій на етапі опису чи класифікації. Попередня обробка даних описує будь-який тип обробки, що виконується з необробленими даними, щоб підготувати їх до наступної процедури обробки. Отже, попередня

обробка – це попередній етап, на якому дані перетворюються на формат, який буде легше та ефективніше обробляти [2].

Отже, основне завдання попередньої обробки отриманих даних полягає в тому, щоб зменшити кількість факторів, які викликають зниження швидкості розпізнавання і збільшують складність. Таким чином, попередня обробка є важливим етапом перед вилученням ознак, оскільки вона контролює придатність результатів наступних етапів.

1.1 Оптичне розпізнавання символів

Оптичне розпізнавання символів (OCR) – це механічна або електронна інтерпретація, зчитування зображень рукописного, машинописного або друкованого тексту (зазвичай записаного сканером або планшетом) у текст, який можна редагувати машиною.

OCR – це ігрове поле для досліджень в галузі ідентифікації шаблонів, штучного інтелекту та машинного зору. Система OCR дозволяє взяти книгу або журнальну статтю, завантажити їх безпосередньо в файл електронного комп’ютера, а потім відредагувати файл за допомогою текстового процесора.

Усі системи OCR включають оптичний сканер для читання тексту та зручне програмне забезпечення для аналізу зображень. Більшість систем OCR використовують для розпізнавання символів суміш апаратних засобів (спеціалізованих плат) і програмного забезпечення, хоча деякі економічні системи роблять це повністю за допомогою програмного забезпечення. Розширені системи розпізнавання тексту на латинській мові можуть читати текст у великій різноманітності шрифтів, але вони все ще мають труднощі з рукописним текстом [3].

Системи оптичного розпізнавання символів перетворюють двовимірне зображення тексту, яке може містити машинно-надрукований або рукописний текст, в ідеалі в будь-якому сценарії, з його графічного подання в текст, що машинно читається. Системи OCR зазвичай працюють у конвеєрі, і

до фактичного розпізнавання тексту потрібно кілька кроків. Типова система OCR може включати попередню обробку, аналіз компонування, розпізнавання символів та мовне моделювання. Попередня обробка зазвичай включає бінаризацію, видалення шуму, корекцію перекосу і, необов’язково, визначення сценарію та орієнтації. Аналіз макету визначає текстові стовпці, текстові блоки, текстові рядки та порядок читання текстової сторінки. Розпізнавання символів відповідає за розпізнавання тексту, який міститься в текстовому рядку. Статистичне мовне моделювання покращує результати розпізнавання тексту за рахунок інтеграції попередніх знань про мову, лексику та предметну область документа[4]. На рисунку 1.1 показано блок-схему типової системи розпізнавання тексту.

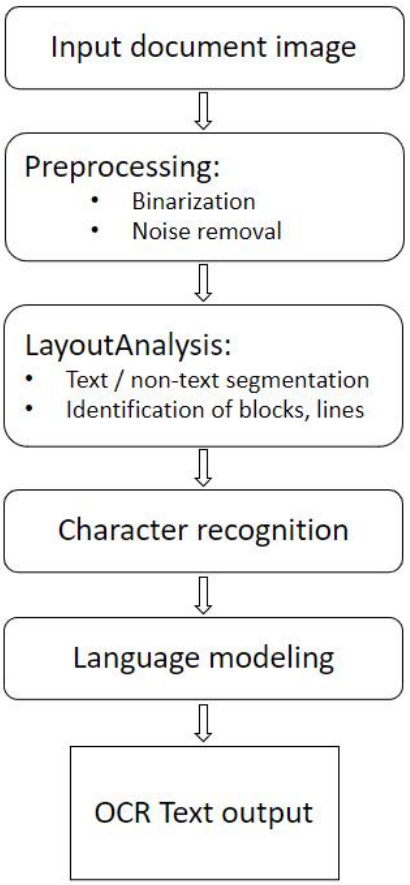

Рисунок 1.1 – Схема типової OCR

1.2 Огляд існуючих програмних рішень

Програмне забезпечення для розпізнавання символів має вирішальне значення для компаній, які прагнуть швидко розвиватися за рахунок використання цифрових робочих процесів і автоматизованих процесів. Програмне забезпечення OCR автоматизує збір даних зі сканованих документів/зображень і оцифровує дані в зручних, редагованих форматах, які вписуються в робочі процеси організації.

Сканування та обробка документів, таких як рахунки-фактури, квитанції та зображення для отримання цінних даних, традиційно було ручним процесом, який пов’язаний із помилками та затримками. Програмні рішення для OCR допомагають підприємствам заощадити час і ресурси, які в іншому випадку були б витрачені на введення даних та ручну перевірку/перевірку. Все більше організацій автоматизують робочі процеси обробки документів, щоб зробити їх безпаперовими, і використовують хмарні цифрові рішення, які покращують прибуток [5].

1.2.1 ABBYY Finereader

ABBYY FineReader PDF – це програма для розпізнавання текстів із підтримкою редагування файлів PDF. Програма дозволяє конвертувати графічні документи в електронні формати, які можна редагувати.

Переваги Finereader:

- зрозумілий інтерфейс;
- експортування в декілька форматів;
- унікальна функція порівняння документів.

Недоліки Finereader:

- потрібен час для навчання;
- ціни можуть бути непомірно високими;
- повільна обробка великих файлів.

1.2.2 ABBYY Flexicapture

FlexiCapture – це масштабоване програмне забезпечення для створення зображень та вилучення даних, яке автоматично перетворює документи будь-якої структури, мови чи вмісту в корисні та доступні для бізнесу дані.

До переваг Flexicapture можна віднести:

- точність розпізнавання зображень;
- легкість зберігання друкованого результату;
- автоматичне вилучення даних з документів (до певної міри).

Однак у Flexicapture є і мінуси:

- інсталяція може бути складною і довгою;
- низька точність із зображеннями/документами з низькою роздільною здатністю;
- важко налаштувати;
- висока ціна.

1.2.3 Adobe Acrobat DC

Adobe Acrobat DC – це потужне програмне забезпечення для перегляду, друку та анотування PDF-файлів, надає повний PDF-редактор із вбудованою функцією OCR. Програма має багато функцій, за допомогою яких можна працювати з електронною документацією, переводити її в потрібні формати і навіть спільно редагувати файли.

Маючи потужний функціонал, Adobe Acrobat викликає багато питань з приводу його застосування навіть у досвідчених користувачів через безліч компонентів, що використовуються.

Плюси:

- стабільність та сумісність;
- простота використання.

Мінуси:

- висока ціна;
- займає багато місця на жорсткому диску;
- потрібна ліцензія Adobe Creative Cloud;
- сильно навантажує систему.

1.3 Постановка задачі

Етапи в системі розпізнавання образів являються послідовними, що означає, що кожен етап залежить від успіху попереднього для отримання найкращих результатів. Однак очевидно, що найбільш відповідні вектори ознак для стадії класифікації будуть створені лише за сприяння етапу попередньої обробки. Основна мета етапу попередньої обробки – нормалізувати та видалити фактори, які в іншому випадку ускладнили б класифікацію та знизили б швидкість розпізнавання.

Існує ряд факторів, які впливають на точність тексту, розпізнаного за допомогою OCR. Ці фактори включають: якість сканера, роздільну здатність сканування, тип друкованих документів (лазерний принтер або фотокопіювання), якість паперу, шрифти, використані в тексті, мовну складність та використаний словник. Прикладом проблем, які впливають на точність розпізнавання розпізнавання, порівняно з чистим текстом на білому тлі, є «проглядання тексту», яке зустрічається в старих паперових документах, водяні знаки та нерівномірне освітлення. Інші фактори включають особливості друку, такі як однорідність, вирівнювання та розташування тексту на сторінці та вміст зображень.

Важливість етапу попередньої обробки системи розпізнавання символів полягає в її здатності усунути деякі проблеми, які можуть виникнути через деякі фактори, наведені вище. Таким чином, використання методів попередньої обробки може покращити зображення документа, готуючи його до наступного етапу в системі розпізнавання символів. Тому

для досягнення більш високого рівня розпізнавання важливо мати ефективний етап попередньої обробки. Використання ефективних алгоритмів попередньої обробки робить систему OCR більш надійною, в основному за рахунок точного покращення зображення, видалення шуму, визначення порогу зображення, виявлення/виправлення перекосу, сегментації сторінок, сегментації символів, нормалізації символів і морфологічних прийомів.

Методи попередньої обробки необхідні для кольорових, сірих або двійкових зображень документів, що містять текст та/або графіку. У системах розпізнавання символів більшість додатків використовують сірі або двійкові зображення, оскільки обробка кольорових зображень є обчислювальною високою. Такі зображення також можуть містити неоднорідний фон та/або водяні знаки, що ускладнює вилучення тексту документа із зображення без виконання певної попередньої обробки; бажаним результатом попередньої обробки є двійкове зображення, що містить лише текст [6].

Таким чином, метою кваліфікаційної роботи є дослідження впливу використання різних методів сегментації та шумопригнічення на точність та швидкість розпізнавання сканованих документів.

Для досягнення поставленої мети мають бути вирішені наступні задачі:

- дослідження загальних параметрів систем розпізнавання сканованих документів;
- дослідження існуючих методів шумопригнічення та сегментації зображень;
- розробка системи впорядкованого зберігання сканованих документів;
- адаптація методів шумопригнічення сканованих документів під різні типи обчислювачів;
- адаптація методів сегментації сканованих документів під різні типи обчислювачів;
- оцінка таймінгу роботи розробленої системи;
- аналіз отриманих результатів.

2 АНАЛІЗ АПАРАТНОЇ ТА МЕТОДОЛОГІЧНОЇ БАЗИ ПРИ РОБОТІ ЗІ СКАНОВАНИМИ ДОКУМЕНТАМИ

Для отримання повнотекстового контенту з паперових документів використовується оптичне розпізнавання символів (OCR). Для сканованих документів методи OCR можуть розпізнавати слова з високим рівнем точності і тому можуть використовуватися для отримання та індексації інформації для подальшого пошуку інформації. Однак автоматичне видалення тексту з зображень документа, створеного цифровою камерою або мобільною камерою, є дуже складним завданням. Розпізнавання тексту з цих документів з високою точністю забезпечить технологічні можливості для ряду додатків, таких як покращене індексування, документи з можливістю онлайн-пошуку та виявлення тексту у природних зображеннях. Технологія OCR пропонує кращі функції пошуку та вилучення, ніж це було можливо раніше. На відміну від класичних сканованих документів, виявлення тексту у зображеннях документів, створених цифровими та мобільними камерами, є дуже складним завданням, далеко не вирішеним. Нерівномірне освітлення та наявність великої кількості текстових шрифтів, розмірів та орієнтації тексту створюють великі труднощі навіть для найсучасніших методів виявлення тексту [2].

2.1 Методи шумозаглушення

Шум – це випадкова варіація інтенсивності зображення, помітна на зображенні у вигляді зерен. У зображенні шум відноситься до пікселів, що показують різні значення інтенсивності замість дійсних значень пікселів. Він з’являється під час отримання чи передачі зображення. У двійковому зображенні ми називаємо шум чорним пікселем, який має бути білим або навпаки. Шум слід видаляти на ранніх етапах, якщо він поширюється на

подальші етапи обробки, то може вплинути на роботу системи. Є два основні типи шуму. Сольовий шум, коли піксель має бути чорним, але представлений білим пікселем. Перечний шум, коли піксель має бути білим, але представлений чорним пікселем. Ми можемо використовувати різні типи фільтрів для видалення шуму із зображень, такі як середній фільтр, медіанний фільтр, фільтр Гауса, тощо.

2.1.1 Середній фільтр

Середній фільтр, найпростіший лінійний фільтр нижніх частот, простий та зручний у використанні для згладжування зображення. При такій фільтрації сума всіх сусідніх пікселів у вікні поділяється на загальну кількість пікселів. Значення, яке ми отримуємо після обчислення, потім замінює значення пікселя, що обробляється. Згладжування зображення відбувається за рахунок зменшення кількості варіацій інтенсивності між сусідніми пікселями. Середній фільтр працює, переміщаючись по зображенню піксель за пікселем, замінюючи кожне значення середнім значенням сусідніх пікселів, включаючи його самого. Є деякі потенційні проблеми: один піксель з дуже нерепрезентативним значенням може значно вплинути на середнє значення всіх пікселів у його околиці. Коли околиця фільтра охоплює край, фільтр інтерполуватиме нові значення пікселів на краю і таким чином розмити цей край. Одна з основних проблем такої фільтрації – розмиття краю.

Вікно або ядро зазвичай квадратне, але може мати будь-яку форму. Приклад середньої фільтрації одного вікна значень розміром 3x3 показано нижче.

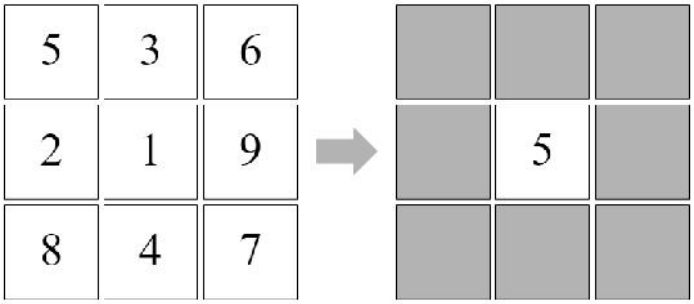

Рисунок 2.1 – Схема роботи середнього фільтру

2.1.2 Медіанний фільтр

Медіанний фільтр – це нелінійний метод, який використовується для видалення шуму з зображень. Він широко використовується, оскільки дуже ефективний при видаленні шуму із збереженням країв. Він особливо ефективний при видаленні шуму типу сіль і перець. Медіана обчислюється шляхом сортування всіх значень пікселів з вікна в числовому порядку, а потім заміняючи піксель, що розглядається, середнім (медіанним) значенням пікселя.

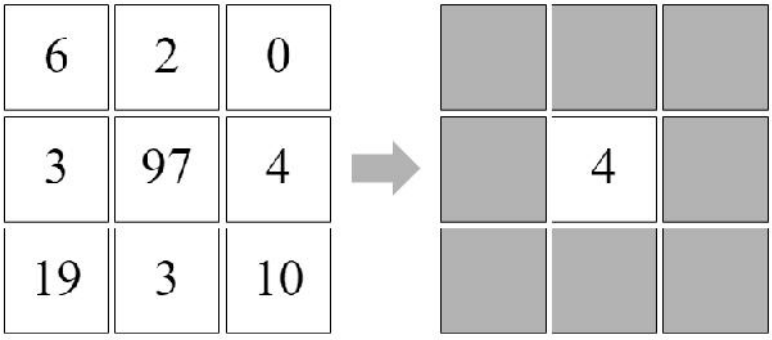

Рисунок 2.2 – Схема роботи медіанного фільтру

Варто звернути увагу, що для попереднього прикладу (рисунок 2.1) медіанний фільтр також поверне значення 5, оскільки впорядковані значення дорівнюють 1, 2, 3, 4, 5, 6, 7, 8, 9. Наприклад, середній фільтр повертає значення 16, оскільки сума дев’яти значень у вікні дорівнює 144 і 144/9 = 16. Це ілюструє одну із особливостей медіанного фільтра: здатність видаляти

імпульсний шум. Медіанний фільтр також широко відомий як «зберігаючий краї», оскільки теоретично він зберігає краї переходів без розмиття. Однак за наявності шуму він трохи розмиває краї зображень.

2.1.3 Фільтр Гауса

Зображення можуть містити багато шуму або випадкових змін яскравості або відтінку між пікселями. Пікселі цих зображеннях мають високе стандартне відхилення, що означає, що є багато варіацій всередині груп пікселів. Оскільки фотографія двовимірна, розмиття Гауса використовує дві математичні функції (одну для осі x і одну для осі y), щоб створити третю функцію, також відому як згортка. При роботі із зображеннями необхідно використовувати двовимірну функцію Гаусса. Це просто добуток двох одновимірних функцій Гауса, по одній для кожного напряму (2.1).

$$G(x,y) = \frac{1}{2®\pm^{2}} e^{-\frac{x^2+y^2}{2\pm^{2}}}. \qquad (2.1)$$

Фільтрація Гауса проходить шляхом розмиття (згладжування) зображення з використанням функції Гауса для зниження рівня шуму на зображенні. Його можна розглядати як неоднорідний фільтр нижніх частот, який зберігає низьку просторову частоту та зменшує шум зображення та незначні деталі у зображенні. Графічне представлення двовимірного розподілу Гауса із середнім значенням (0,0) та σ = 1 показано на рисунку 2.3.

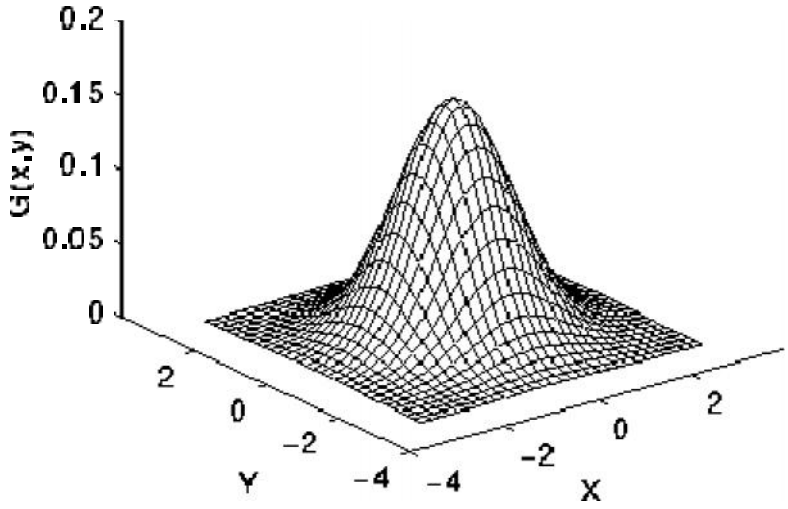

Рисунок 2.3 – Довимірний розподіл Гауса

2.2 Методи бінаризації

Порогова обробка – це метод, який являє собою присвоєння значень пікселів по відношенню до наданого порогового значення. При граничній обробці значення кожного пікселя порівнюється з граничним значенням. Якщо значення пікселя менше порогового значення, воно встановлюється на 0, інакше встановлюється максимальне значення (зазвичай 255). Порогова обробка – дуже популярний метод сегментації, що використовується для відокремлення об’єкта, що розглядається як передній план від його фону. Поріг – це значення, яке має дві області з кожного боку, тобто нижче за поріг або вище за поріг.

2.2.1 Адаптивна порогова обробка

В звичайному методі знаходження порогових значень використовується глобальне значення як порогове значення. Але воно може бути підходящим не за всіх умов, наприклад, коли зображення має різне освітлення у різних областях. У цьому випадку краще використовувати адаптивну граничну обробку, в якій алгоритм обчислює поріг для невеликих

ділянок зображення. Таким чином, будуть отримані різні пороги для різних областей того самого зображення, що дає найкращі результати для зображень з різним освітленням.

2.2.2 Метод Оцу

Перетворення зображення у відтінках сірого на монохромне – поширене завдання обробки зображень. Метод Оцу, названий на честь його винахідника Нобуюкі Оцу, одна із багатьох алгоритмів бінаризації.

Алгоритм ітеративно шукає поріг, який мінімізує дисперсію всередині класу, що визначається як зважена сума дисперсій двох класів (фоновий та передній план). Мета полягає в тому, щоб знайти граничне значення, при якому сума різниці переднього та заднього плану мінімальна[10].

Формула для знаходження внутрішньокласової дисперсії при будь-якому порозі t визначається так:

$$\pm^2(t) = \mu_{bg}(t)\pm^2_{bg}(t) + \mu_{fg}(t)\pm^2_{fg}(t), \qquad (2.2)$$

де $\mu_{bg}(t)$ і $\mu_{fg}(t)$ – ймовірність кількості пікселів кожного класу;

$\pm^2$ – дисперсія значень кольору.

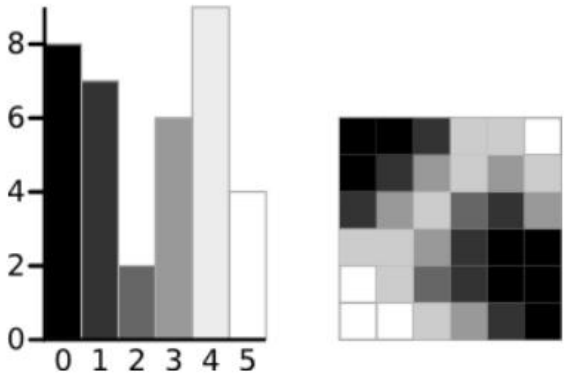

Рисунок 2.4 – Зображення у відтінках сірого та його гістограма

Шляхом невеликих маніпуляцій можна обчислити так зване міжкласове відхилення, яке обчислити набагато швидше. Поріг з максимальною дисперсією між класами має мінімум дисперсії всередині класу, таким чином, його також можна використовувати для пошуку порогу.

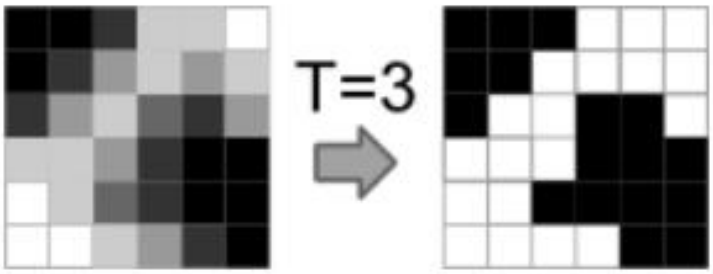

Рисунок 2.5 – Результат методу Оцу

2.3 Морфологічні операції

Морфологічна обробка зображення – це сукупність нелінійних операцій, пов’язаних із формою чи морфологією об’єктів зображення. Морфологічні операції покладаються лише на відносне впорядкування значень пікселів, а не на їх числові значення, і тому особливо підходять для обробки двійкових зображень. Морфологічні операції також можна застосувати до зображень у відтінках сірого, так що їх функції передачі світла невідомі, і тому їх абсолютні значення пікселів не представляють інтересу або не мають значення[11].

Морфологічні методи досліджують зображення за допомогою невеликої форми або шаблону, який називається структурним елементом. Елемент структуризації розташовується в усіх можливих місцях на зображенні і порівнюється з відповідним оточенням пікселів. Деякі операції перевіряють, чи елемент «вписується» в околиці, тоді як інші перевіряють, чи він «входить» чи перетинає околиці. Морфологічна операція над двійковим зображенням створює нове двійкове зображення, в якому піксель має

ненульове значення, тільки якщо тест успішно пройшов у цьому місці вхідного зображення[12].

Елементом структуризації є невелике двійкове зображення, тобто невелика матриця пікселів, кожен з яких має значення нуль або одиницю:

- розміри матриці визначають розмір елемента структури.
- шаблон одиниць і нулів задає форму структурного елемента.
- початком координат елемента структуризації зазвичай є один з його пікселів, хоча загалом початок може бути поза структурним елементом [13].

2.3.1 Ерозія

Основна ідея ерозії подібна лише ерозії ґрунту, вона розмиває межі об’єкта переднього плану. Ядро ковзає по зображенню (як у 2D згортки). Піксель у вихідному зображенні (або 1, або 0) буде вважатися 1, лише якщо всі пікселі під ядром дорівнюють 1, інакше він розмивається (зводиться до нуля).

Ерозія бінарного зображення A структуруючим елементом B визначається як:

$$A \ominus B = \{z \in E \mid B_z \subseteq A\} \tag{2.3}$$

Отже, всі пікселі поблизу кордону будуть відкинуті залежно від розміру ядра. Таким чином, зменшується товщина або розмір об’єкта переднього плану або просто зменшується біла область на зображенні. Це корисно для видалення невеликих білих шумів, від’єднання двох пов’язаних об’єктів тощо.

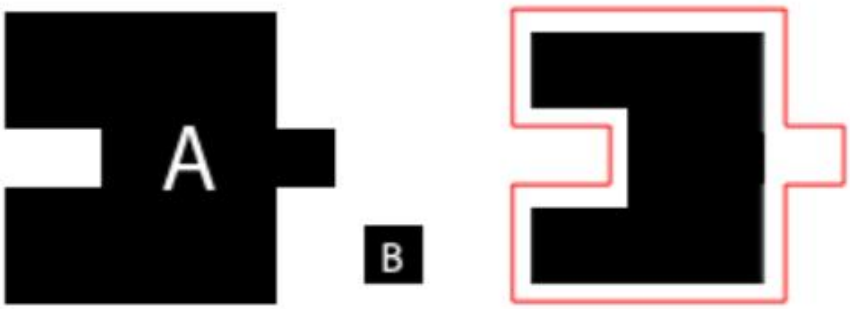

Рисунок 2.6 – Результат операції ерозії

2.3.2 Операція розширення

Це повна протилежність ерозії. Тут елемент пікселя дорівнює "1", якщо хоча б один піксель під ядром дорівнює "1". Таким чином, біла область збільшується на зображенні або збільшується розмір об’єкта переднього плану. Зазвичай у таких випадках, як видалення шуму, за ерозією слідує розширення. Тому що ерозія забирає білі шуми, але вона також стискає наш об’єкт.. Оскільки шум зник, то він не повернеться, але площа нашого об’єкта збільшується. Це також корисно для з’єднання зламаних частин об’єкта.

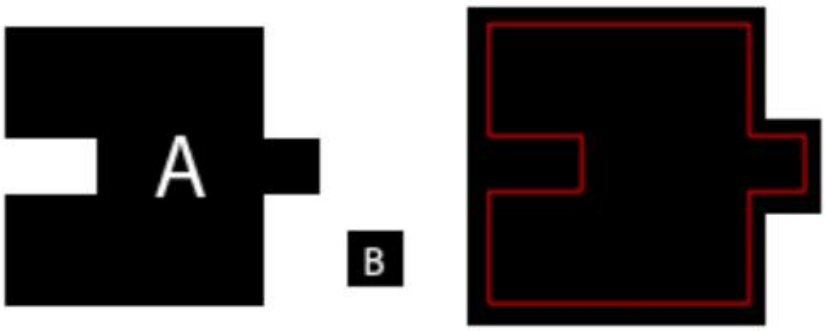

Рисунок 2.7 – Результат операції розширення

Розширення бінарного зображення A структуруючим елементом B визначається таким чином:

$$A \oplus B\{z \in E \mid (B^s)_z \cap A \neq \varnothing\} \tag{2.4}$$

2.3.3 Операція відкриття

Відкриття зазвичай згладжує контури, розриває вузькі ділянки та усуває виступи. Частини, які залишаться в зображенні, - це місця, в які структурний елемент потрапляє належним чином, інакше – контент буде видалено. Відкриття A на B виходить розмиттям A на B з подальшим розширенням результуючого зображення B.

Відкриття двійкового зображення A структуруючим елементом B визначається таким чином:

$$A \circ B = (A \ominus B) \oplus B. \tag{2.5}$$

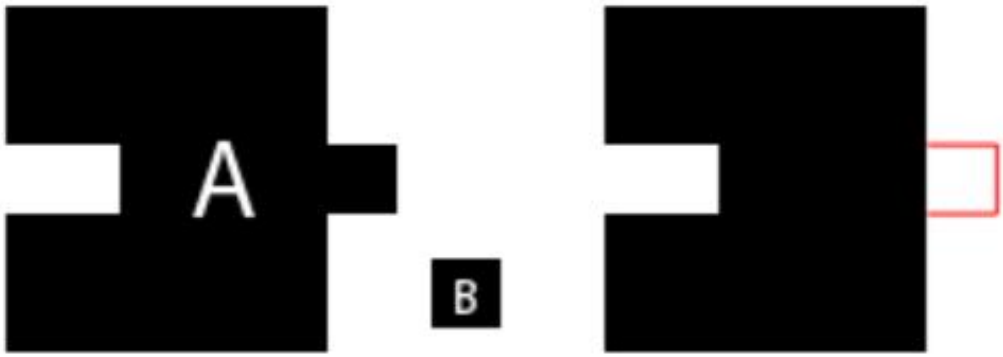

Рисунок 2.8 – Результат операції відкриття

2.3.4 Операція закриття

Закриття також має тенденцію згладжувати контури, але на відміну від відкриття, воно зазвичай поєднує вузькі розриви та довгі тонкі прірви, усуває невеликі дірки та заповнює прогалини в контурі. Закриття двійкового зображення A структуруючим елементом B виконується спочатку розширенням набору A B, а потім розмиванням результату B.

Закриття двійкового зображення A структуруючим елементом B визначається таким чином:

$$A \bullet B = (A \oplus B) \ominus B . \quad (2.6)$$

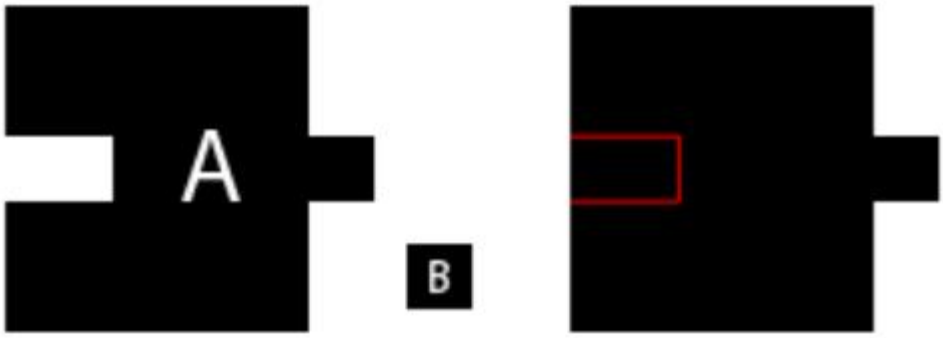

Рисунок 2.9 – Результат операції закриття

2.4 Огляд технологій для вирішення поставленої задачі

2.4.1 OpenCV

OpenCV (Open Source Computer Vision Library) – це програмний периферійний інтерфейс (API), розроблений компанією Intel, який можна використовувати для багатьох програм обробки зображень і комп’ютерного зору. OpenCV було офіційно запущено в 1999 році, і спочатку цей проект був ініціативою Intel Research для просування додатків із інтенсивним процесором, частиною серії проектів, включаючи трасування променів у реальному часі та 3D-дисплеї.

Бібліотека OpenCV – це набір алгоритмів і функцій, а також кілька класів, які реалізують деякі алгоритми обробки зображень і комп’ютерного зору. OpenCV був розроблений для обчислювальної ефективності та з сильним акцентом на додатках реального часу. OpenCV написаний на оптимізованому C і може використовувати переваги багатоядерних процесорів.

OpenCV містить понад 500 функцій, які охоплюють багато областей зору, включаючи заводську перевірку продукції, медичне зображення, безпеку, інтерфейс користувача, калібрування камери, стереобачення та

робототехніку. Принципи створення бібліотеки полягають у тому, щоб сприяти комерційному використанню комп’ютерного зору в інтерфейсі «людина-комп’ютер», робототехніці, моніторингу, біометрії та безпеці, забезпечуючи безкоштовну та відкриту інфраструктуру. Підтримка OpenCV включає звичайну підтримку для введення, відображення та зберігання фільмів та окремих зображень. OpenCV використовує DirectX, який являє собою набір API, розроблений Microsoft для створення мультимедійних програм та ігор.

OpenCV був розроблений як портативний. Спочатку він був написаний для компіляції на Borland C++, Microsoft Visual Studio C++ та компіляторах Intel. Код C і C++ мав бути досить стандартним, щоб спростити міжплатформну підтримку. Бібліотека OpenCV є багатоплатформною і працює як в операційних системах Windows, так і в Linux. OpenCV швидко набирає популярності для розробки програм комп’ютерного зору в режимі реального часу. Функціонал бібліотеки включає засоби розпізнавання облич, розпізнавання об’єктів і трекери руху, і це лише деякі з них. Бібліотека особливо набула популярності в дослідницькому середовищі комп’ютерного зору. Це дозволяє дослідникам швидко запускати демонстраційні або дослідницькі проекти, а також користуватися перевагами великої колекції алгоритмів, які вже доступні.

Використання термінів комп’ютерного зору та обробки зображень зазвичай чергується. Навпаки, існує розрив між комп’ютерним зором та обробкою зображень. Обробка зображення – це низькорівнева обробка зображення , тоді як комп’ютерний зір – це. OpenCV спеціально розроблений для початку розвитку комп’ютерного зору. OpenCV спрямований на надання основних інструментів, необхідних для вирішення проблеми комп’ютерного зору. Бібліотеки високого рівня буде достатньо для вирішення більш складних завдань комп’ютерного зору. Базові компоненти бібліотеки достатньо повні, щоб створити повне рішення практично будь-якої проблеми комп’ютерного зору.

Обробка зображень належить до низькорівневих процесів, які включають примітивні операції, такі як попередня обробка зображення для зменшення шуму. Дані обробки зображень низького рівня складаються з оригінальних зображень, представлених матрицями, що складаються із значень яскравості. Методи низького рівня зазвичай використовують дуже мало знань про зміст зображень. Низький рівень характеризується тим, що на його входах і виходах є зображення. Послідовність обробки зображень низького рівня передбачала попередню обробку та сегментацію зображення.

Обробка зображень зазвичай намагається виконати одну з трьох речей:

- відновлення зображень;
- покращення зображень;
- розуміння зображень.

З іншого боку, комп’ютерний зір – це високорівнева обробка. Обробка високого рівня включає в себе розуміння або «осмислення» групи розпізнаних об’єктів, аналіз зображень і, нарешті, виконання когнітивних функцій, пов’язаних з людським баченням. Комп’ютерний зір високого рівня намагається імітувати людське сприйняття і його здатність приймати рішення відповідно до інформації, що міститься на зображенні. Наприклад, розмір, форма та взаємозв’язки між об’єктами на зображенні. Дані високого рівня зазвичай виражаються в символічній формі [7].

OpenCV підтримує наступні основні функції (рисунок 2.10):

- загальні алгоритми комп’ютерного зору та обробки зображень (API середнього та низького рівня);
- модулі комп’ютерного зору високого рівня;
- ШІ та методи машинного навчання;
- методи створення та аналізу бінарних зображень;
- математичні процедури для обробки зображень, комп’ютерного зору та інтерпретації зображень.

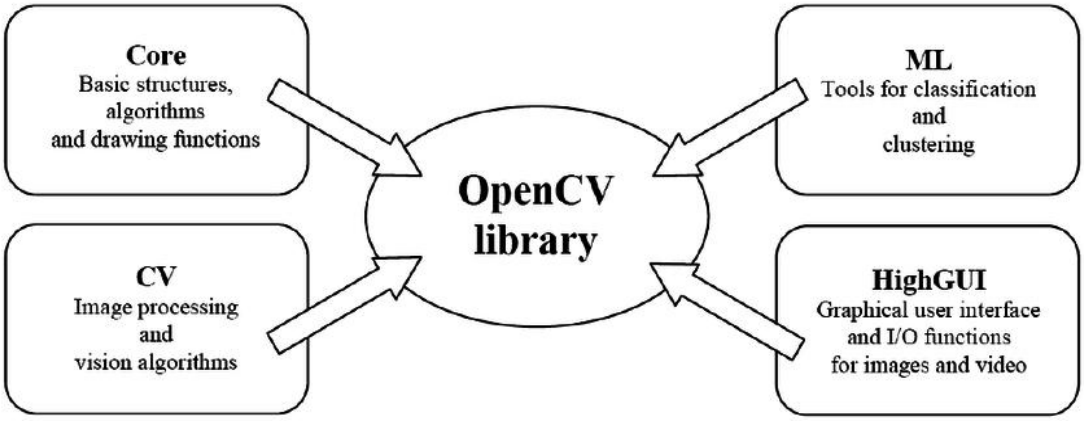

Рисунок 2.10 – Основна структура бібліотеки OpenCV

2.4.2 Tesseract

Tesseract – це механізм OCR з відкритим кодом, який був розроблений в HP між 1984 і 1994 роками. Tesseract розпочався як дослідницький проект у HPLabs, БрістольМотивація була забезпечена тим фактом, що комерційні механізми OCR того часу були в зародковому стані і точність їхнього розпізнавання була на низикому рівні .

Розпізнавання проводиться двічі: під час першого запуску розпізнавання використовується статичний класифікатор, а для другого – адаптивний класифікатор. Tesseract розроблено для розпізнавання тексту, навіть якщо він має невеликий перекіс, без необхідності виправляти зображення, хоча для кращого розпізнавання бажано, щоб текст був горизонтальним. Першою частиною процесу розпізнавання є аналіз зв’язаних компонентів. Він включає пошук рядків, підгонку базової лінії, сегментацію символів і слів. Потім кожне слово, передане в статичному класифікаторі, подається в адаптивний класифікатор для навчання. На той час це було дороге в обчислювальному відношенні рішення, але воно мало значну перевагу: шляхом перевірки вкладеності контурів і кількості дочірніх легко виявити зворотний текст і розпізнати його так само легко, як і чорно-білий текст. Tesseract був, ймовірно, першим механізмом розпізнавання текстів, здатним так тривіально обробляти білий текст на чорному фоні. На цьому

етапі контури збираються разом, просто вбудовуючись у великі двійкові об’єкти (Blob). Ці об’єкти організовані в текстові ряди, а рядки та області аналізуються на предмет пропорційного тексту.

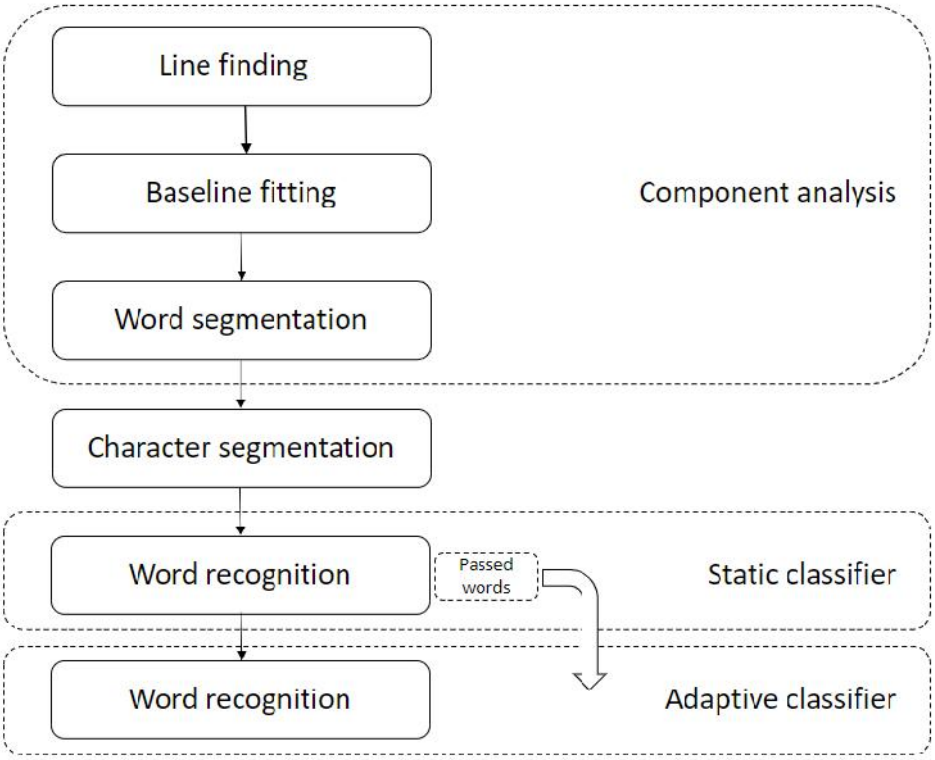

Рисунок 2.11 – Компонентна архітектура Tesseract

Текстові рядки розбиваються на слова по-різному залежно від міжсимвольного інтервалу. Пропорційний текст розбивається на слова з використанням певних прогалин, потім розпізнавання відбувається як двохетапний процес. На першому етапі робиться спроба розпізнати кожне слово по черзі. Кожне задовільне слово передається адаптивному класифікатору як навчальні дані. Потім адаптивний класифікатор отримує можливість точніше розпізнавати текст знизу сторінки. Оскільки адаптивний класифікатор міг занадто пізно засвоїти щось корисне, щоб зробити свій внесок у верхній частині сторінки, по сторінці виконується другий прохід, в якому слова, які не були впізнані, розпізнаються знову [8]. Компонентна архітектура Tesseract зображена на рисунку 2.11.

2.5 Алгоритми пошуку рядків і слів

Алгоритм пошуку ліній розроблений таким чином, що перекіс сторінки можна розпізнати без необхідності виправлення перекосу, що дозволяє уникнути втрати якості зображення. Ключовими частинами процесу є фільтрація великих двійкових об’єктів та побудова ліній. Припускаючи, що аналіз макета сторінки вже надав текстові області приблизно однакового розміру, простий фільтр висоти видаляє буквиці та символи, що накладаються вертикально. Середня висота наближається до розміру тексту в області, тому можна безпечно відфільтрувати об’єкти, розмір яких менше деякої частини середньої висоти, швидше за все, це розділові знаки, діакритичні знаки і шум. Сортування та обробка великих двійкових об’єктів за координатою x дозволяє призначати їх дл унікального текстового ряду, відстежуючи при цьому нахил по сторінці, що значно знижує небезпеку присвоєння неправильному текстовому ряду за наявності перекосу. Після того, як відфільтровані об’єкти були призначені рядам, для оцінки базових ліній використовується найменше медіа значення квадратів, а відфільтровані вихідні Blob знову вставляються у відповідні рядки. Останній крок процесу створення лінії поєднує блоки, які перекриваються на мінімум наполовину по горизонталі, поставивши діакритичні знаки разом із правильною основою та правильно зіставивши частини деяких зламаних знаків.

2.6 Знаходження базових ліній

Ключовим компонентом пошуку базової лінії є blob фільтрація та побудова рядків. Blob – це слово, символ або будь-який вміст, не пов’язаний з рештою зображення. Середня висота blob є наближеною, що допомагає двигуну відфільтрувати та видалити всі невеликі краплі, як правило, пунктуацію або можливий шум. Коли лінії blob були знайдені, Тессеракт розглядає їх трохи ближче. Базові лінії більш точно підходять до

квадратичного сплайна, тобто чотирьох паралельних ліній, які аналізують blob. Ця функція дуже корисна, щоб допомогти Tesseract обробляти криві слова; наприклад, відскановані книги, де слова найчастіше вигнуті (рисунок 2.12) в центрі біля палітурки [9].

Рисунок 2.12 – Приклад вигнутої базової лінії

2.7 Сегментація слів

Слова, символи яких мають однакову ширину (фіксований крок), розглядаються як особливий випадок, слова нарізаються однаково на основі висоти та позначаються для розпізнавання. Однак найчастіше символи в словах мають різну висоту і їх потрібно розглядати окремо. Tesseract обробляє різні кроки, вимірюючи розриви в обмеженому вертикальному діапазоні між базовою і середньою лінією. Коли він знаходить проміжки, які знаходяться занадто близько до порогу, він позначає простір як нечіткий. Потім передає це для рішення, яке буде прийнято пізніше під час другого розпізнавання, коли адаптивний класифікатор зможе дізнатися більше корисної інформації.

GPU – це потужний багатоядерний процесор, який підтримує паралельну обробку даних і високоточні обчислення. Він включає в себе набір потокових мультипроцесорів (SM) для однієї команди і кількох даних (SIMD). SM одночасно виконують одну і ту ж операцію з кількома даними. Кожен SM складається з кількох процесорів (ядер). Графічні процесори включають текстурну, константну, глобальну, локальну, спільну та регістрову пам’ять, як показано на рисунку 2.14. Це дає нам можливість запускати графіку HD (високої чіткості), яка є вимогою сучасних обчислень.

Обчислення на графічному процесорі забезпечили величезну перевагу над процесором з повагою до швидкості обчислень

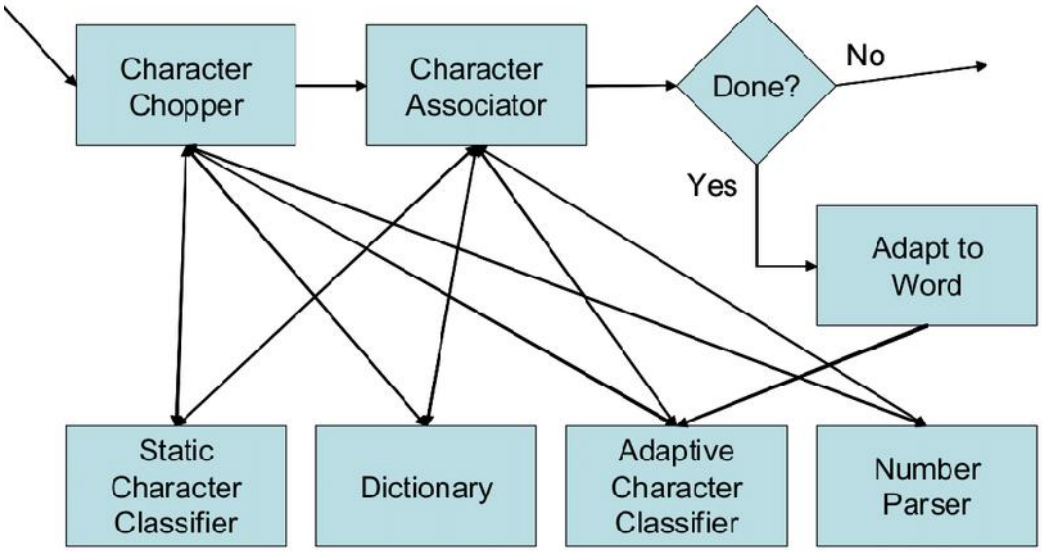

Рисунок 2.13 – Схема алгоритму розпізнавання слів

2.8 Платформа паралельних обчислень CUDA

Основна відмінність між CPU та GPU полягає в тому, що CPU є послідовним процесором, тоді як GPU є потоковим процесором. ЦП оптимізовано для високої продуктивності при послідовних операціях. Він використовує класну логіку керування для виконання багатьох потоків, зберігаючи ресурси послідовного виконання. Велика кеш-пам’ять використовується для зменшення затримки доступу, а повільна пропускна здатність пам’яті також сприяє розриву продуктивності.

Паралельні обчислення – це одночасне використання кількох обчислювальних ресурсів для вирішення обчислювальної задачі. Щоб максимізувати переваги паралельних обчислень, завдання має складатися з кількох незалежних підзадач. У процесі видалення шуму обчислення нових пікселів незалежні один від одного, що дозволяє виконувати ці обчислення одночасно. Створення паралельних потоків ЦП на графічному процесорі вимагає певної підготовки. Оскільки графічні процесори не мають прямого

доступу до пам’яті комп’ютера, спочатку слід вибрати область достатнього розміру у відеопам’яті, а потім скопіювати дані з оперативної пам’яті (ОЗП). Після завершення розрахунків необхідно перемістити дані назад в оперативну пам’ять і очистити раніше виділену область відеопам’яті.

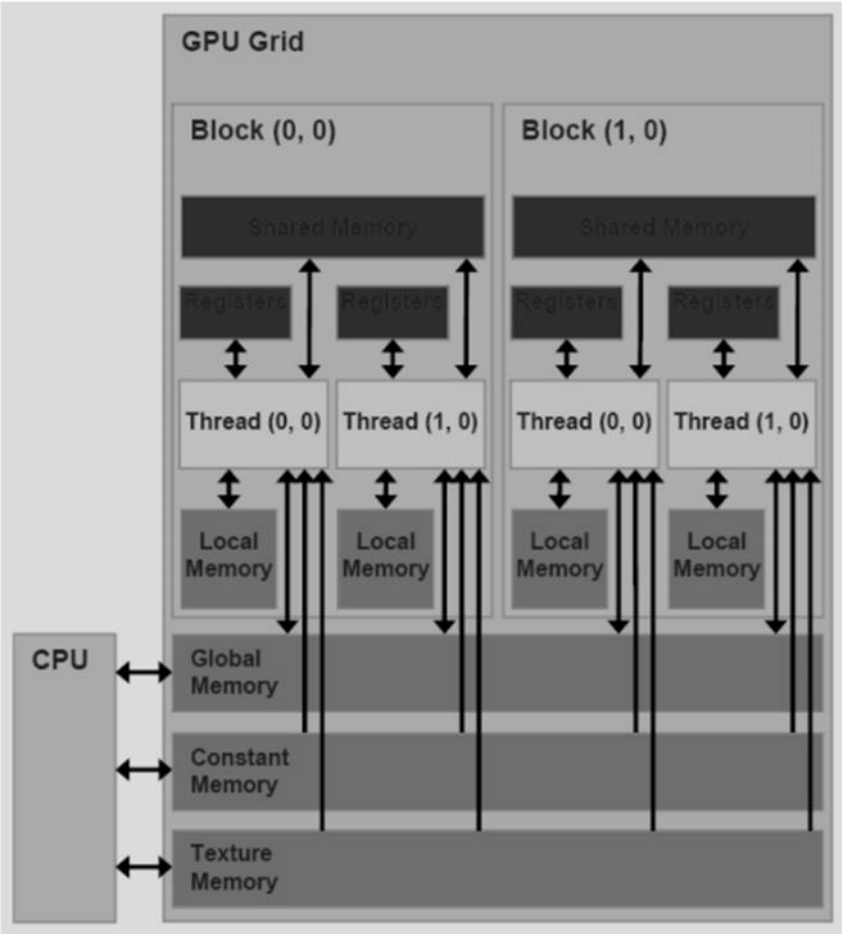

Рисунок 2.14 – Архітектура графічного процесора в середовищі CUDA

Графічні процесори мають спеціальне обладнання для текстур, крім того, графічні процесори включають постійну пам’ять. Це швидка пам’ять, яку можуть читати всі ядра GPU. Зауважимо, що константна та текстурна пам’ять доступні лише для читання, що є серйозним недоліком, якщо потрібно записати дані. Основна пам’ять у графічних процесорах є глобальною. До нього можна отримати доступ з усіх ядер пристрою. Залежно від моделі графічного процесора, він може вмістити один гігабайт (ГБ) або більше, що дозволяє обробляти велику кількість пікселів. Глобальна пам’ять

оптимізована для одночасної обробки великої кількості запитів на пам’ять, забезпечуючи значну пропускну здатність. Доступ до цієї пам’яті кешується, що робить його дуже швидким. Крім того, локальна пам’ять є абстракцією глобальної. Доступ до неї здійснюється лише через блоки потоків. Крім того, він знаходиться поза чіпом, що робить його доступ таким же дорогим. Компілятор використовує локальну пам’ять, коли визначає, що в регістрі недостатньо місця для зберігання змінних. З іншого боку, графічний процесор включає спільну пам’ять, яка в 100 разів швидше, ніж глобальна. Його можна використовувати як кеш L1, керований користувачем, щоб зменшити кількість повільних доступів до глобальної пам’яті. Цей тип пам’яті є загальним для групи ядер і може одночасно отримати доступ кільком ядер. У такому випадку ядра можуть співпрацювати в певному завданні або просто зберігати тимчасові дані та зменшувати тиск на реєстрацію. Зрештою, найшвидший доступ до пам’яті в GPU забезпечується регістрами.

Для використання платформи GPU можна використовувати кілька технологій. Найпопулярнішими є мова Open Computing Language (OpenCL) і Compute Unified Device Architecture (CUDA). З одного боку, OpenCL є відкритим стандартом, який пропонує міжплатформні можливості. Найважливішим недоліком OpenCL є висока складність інтерфейсу прикладного програмування (API) і моделі виконання. З іншого боку, CUDA – це API, який дозволяє програмувати високий рівень C і C++ на графічних процесорах NVIDIA. Крім того, середовище CUDA є більш дружнім і здатне переносити програмний код набагато більш послідовно. Як показано на рисунку 2.14, CUDA відзначає SM і ядра відповідно блоками і потоками. Контейнер усіх блоків CUDA називається сіткою. Відповідно до середовища CUDA, вважається, що віртуальна архітектура визначає модель виконання, як описано на рисунку 2.15. Тоді модель виконання добре підходить для паралельної обробки зображень, оскільки блоки групують тисячі незалежних потоків, готових паралельно обробляти величезну кількість пікселів.

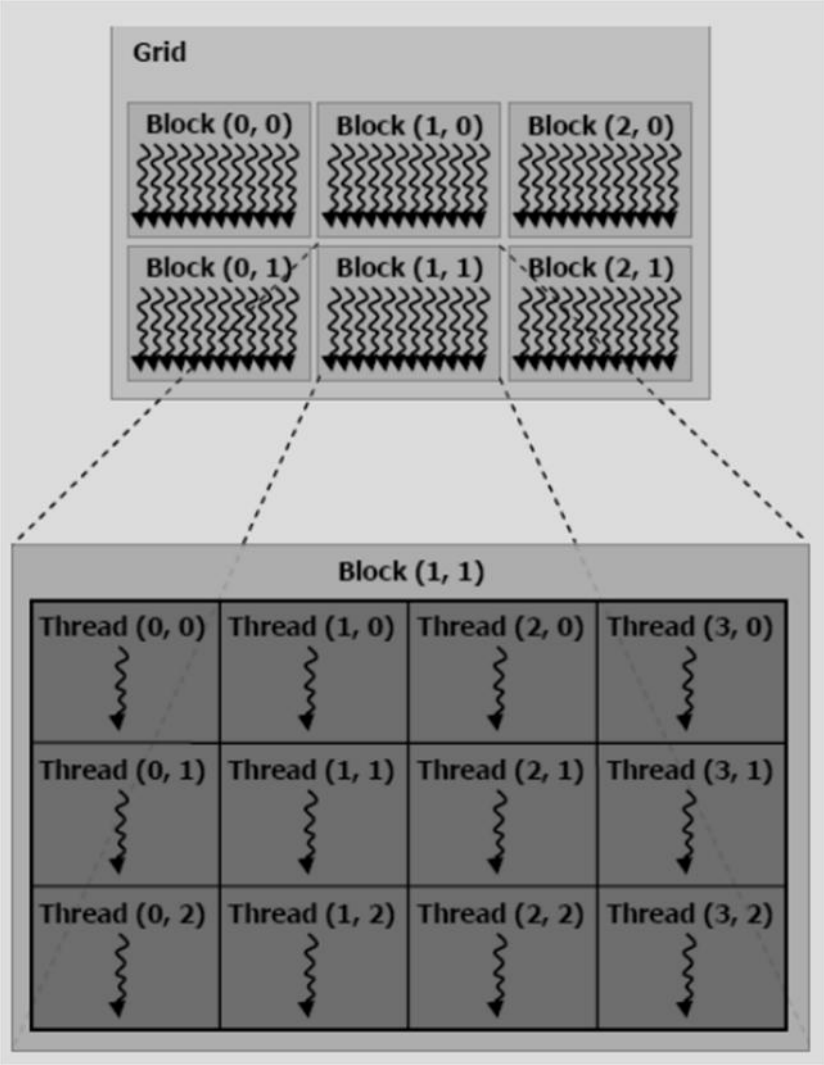

Рисунок 2.15 – Модель виконання GPU в середовищі CUDA

Ієрархія пам’яті введена в ряді рівнів (рисунок 2.16). На нижньому рівні є невеликі, швидкі, не загальні регістри. Тоді кожен SM містить спільну пам’ять, як згадувалося вище. Ця пам’ять також швидка і мала; він може забезпечити зв’язок між потоками в блоці. Інший важливий простір пам’яті - це постійна пам’ять, яка доступна лише для читання. Оскільки він кешується, він зазвичай служить місцем швидкого зберігання параметрів функцій і констант. Пам’ять текстур також доступна лише для читання та кешується, але вона також включає кілька корисних апаратних функцій, таких як ряд режимів адресації, інтерполяція та місцевість. На найвищому рівні є великий простір глобальної пам’яті з великою затримкою.

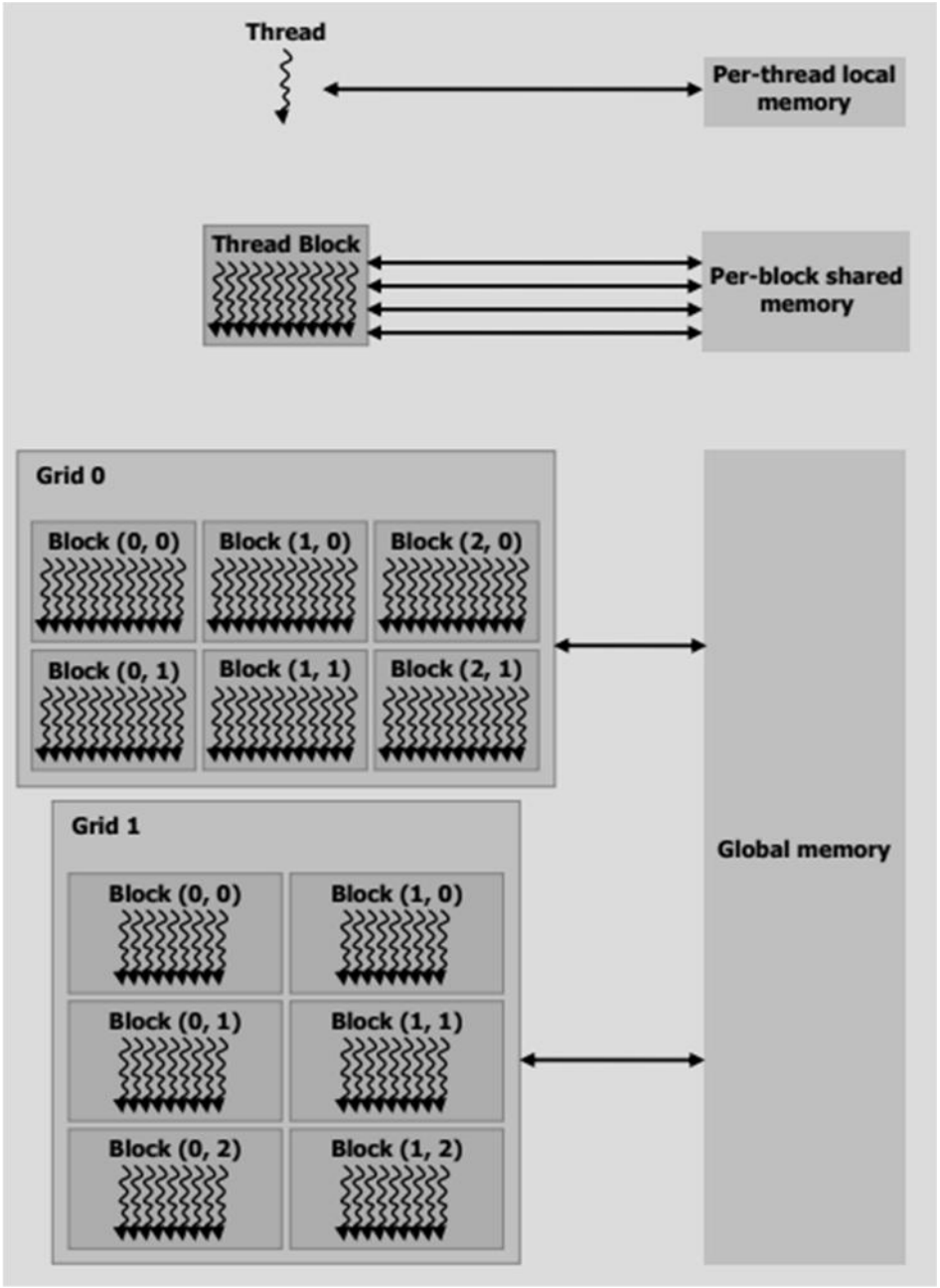

Рисунок 2.16 – Ієрархія пам’яті в середовищі CUDA

2.9 Аналіз тексту

Тегування частин мови (POS) є однією з найбільш добре вивчених проблем у галузі обробки природної мови (NLP). Для автоматизації завдання для англійської та інших західних мов уже випробували різні підходи.

Тегування частин мови (POS) означає призначення граматичних класів, тобто відповідних тегів частин мови, кожному слову в реченні природної

мови. Призначення POS-тега кожному слову тексту вручну займає дуже багато часу, що призводить до існування різних підходів до автоматизації роботи. Таким чином, автоматизоване тегування POS є технікою для автоматизації процесу анотації лексичних категорій. Процес приймає слово або речення як вхідні дані та призначає POS-тег слову або кожному слову в реченні і створює текст із тегами як вихідний.

POS-теги також відомі як класи слів, морфологічні класи або лексичні теги. Значення їх полягає у великій кількості інформації, яку вони дають про слово та його сусідів. POS-теги можна використовувати в програмах перетворення тексту в мовлення (TTS), пошуку інформації, синтаксичного аналізу, вилучення інформації, лінгвістичних досліджень для корпусів, а також може використовуватися як проміжний крок для завдань

Тегування POS має важливе значення в наступній обробці тексту, як-от синтаксичний аналіз, оскільки воно полегшує обробку, приєднуючи клас до слова, а також широко використовується для лінгвістичного аналізу тексту. Тегування POS використовується як ранній етап аналізу тексту в багатьох програмах, таких як отримання підкатегорій, синтез тексту в мовлення та вирівнювання паралельних корпусів.

Теги POS корисні для синтаксичного аналізу, оскільки теги зменшують неоднозначність у вхідному речення синтаксичного аналізатора, що робить синтаксичний аналіз швидшим, зменшуючи обчислювальну проблему, а результат менш неоднозначним. Це також усуває деякі неоднозначності, які не вирішуються мовною моделлю синтаксичного аналізатора. За допомогою тегів на основі правил вихід синтаксичного аналізатора стає менш неоднозначним без значного зниження швидкості розпізнавання, і в той же час швидкість розбору до певної міри збільшується.

3 ІНТЕГРАЦІЯ МЕТОДІВ СЕГМЕНТАЦІЇ ТА ПРИГНІЧЕННЯ ШУМУ У КОМПЮТЕРНІЙ СИСТЕМІ

3.1 Розробка системи впорядкованого зберігання сканованих документів із характерними ознаками

Зростаюча потреба в автоматизації систем значною мірою вплинула на розвиток виявлення та розпізнавання тексту з зображень. Розпізнавання тексту має широкий спектр застосувань, кожне з яких залежить від сценарію проблем і ускладнень. Типи зображень документів, для розпізнавання, включають загальні документи, такі як газети, книги та журнали, бланки, наукові документи.

Для дослідження впливу використання різних методів сегментації та шумопригнічення на точність та швидкість розпізнавання сканованих документів, у системі запропонована система впорядкованого зберігання сканованих документів із характерними ознаками, а саме із виділеними після розпізнавання тексту ключовими словами.

Запропонована система складається з наступних модулів:

- модуль шумопригнічення;
- модуль сегментаціїї тексту;
- модуль аналізу тексту.

На вхід системи подається сканований документ, перед відправкою зображення в алгоритм розпізнавання тексту використовуються кілька алгоритмів попередньої обробки, щоб допомогти алгоритму правильно прочитати текст на зображенні, а саме:

- методи нормалізації зображення;
- методи шумопригнічення;
- методи бінаризації;
- морфологічні операції.

В сукупності використовувані методи попередньої обробки сканованого документу дозволяють покращити вихідне зображення, що приводить до вищої точності розпізнавання тексту. Попередня обробка – це загальний термін для великої кількості різних технік і навіть більшої кількості їх варіантів. Оскільки в центрі уваги OCR протягом тривалого часу були відскановані документи, велика частина попередньої роботи щодо попередньої обробки проводилася з урахуванням цього типу введення.

Щоб мати можливість забезпечити кращий і послідовний результат від попередньої обробки системи OCR, необхідна попередня обробка. Механізми OCR зазвичай мають дуже обмежену внутрішню обробку зображень і потребують допомоги. Обробка зображень може допомогти, виділяючи важливі елементи зображення та забезпечуючи кращі результати. Це корисно для пошуку та виділення важливого вмісту зображень, який пізніше може бути окремо розпізнаний механізмом OCR. Це також корисно для виявлення та виправлення можливого перекосу обертання. Загалом обробка зображень може використовуватися для спостереження за об’єктами, які не видно неозброєним оком, підвищення різкості або відновлення кращої якості зображення, вимірювання візерунків у різних об’єктах або розрізнення об’єктів на зображенні.

Після процедури розпізнавання отриманий текст треба попередньо обробити для вилучення непотрібних слів та закінчень. Процес обробки тексту складється з наступних етапів:

- токенізація;
- Pos тегування;
- видалення стоп-слів;
- стемінг;
- пошук ключових слів.

Непідготовлений текст містить слова, які не повинні враховуватися для виділення ключових слів з тексту. Також з граматичних міркувань у документах будуть використовуватися різні форми слова. Крім того, існують

сімейства деривaційно споріднених слів зі схожими значеннями. У багатьох ситуаціях здається, що було б корисно для пошуку за одним із цих слів повернути документи, які містять інше слово в наборі.

Однак два слова відрізняються за сенсом. Стемінг зазвичай відноситься до грубого евристичного процесу, який обриває кінці слів в надії на правильне досягнення цієї мети більшу частину часу, і часто включає видалення деривaційних афіксів. Лематизація зазвичай відноситься до правильного виконання з використанням словникового запасу та морфологічного аналізу слів, як правило, з метою видалення лише закінчень і повернення основи або словникової форми слова, яка відома як лема. Якщо зіткнутися з лексемою saw, корекція може повернути лише s, тоді як лемматизація спробує повернути або see, або saw залежно від того, чи використовувалася лексема як дієслово чи іменник. Ці два можуть також відрізнятися тим, що корекція найчастіше згортає деривaційно споріднені слова, тоді як лемматизація згортає лише різні флективні форми леми.

Алгоритми стемінгу залежать від мови. Таким чином, знаходячи спільну для всіх граматичних форм слова – основу, відсікаючи суфікси і закінчення, стеммінг здійснює морфологічний розбір слова. Стемінг має недолік: різні форми слова можуть мати різні основи. Тому після стеммінга результати обробки цих слів будуть різні. У зв’язку з цим застосовується інша техніка – лематизації.

Лематизація – це більш тонкий процес, який використовує словник і морфологічний аналіз, щоб в результаті привести слово до його канонічної формі - лемми. На відміну від стеммінга вона працює на основі словника, де і зберігаються дані про слова і їх початкових формах. Це дозволяє уникати помилок, описаних вище. У разі якщо слова немає в словнику, то відбувається побудова гіпотези про спосіб зміни слова.

Отриманий після виконання лематизації набір слів уже може використовуватися для проведення машинного навчання і вирішення конкретних завдань класифікації, маршрутизації і т.д.

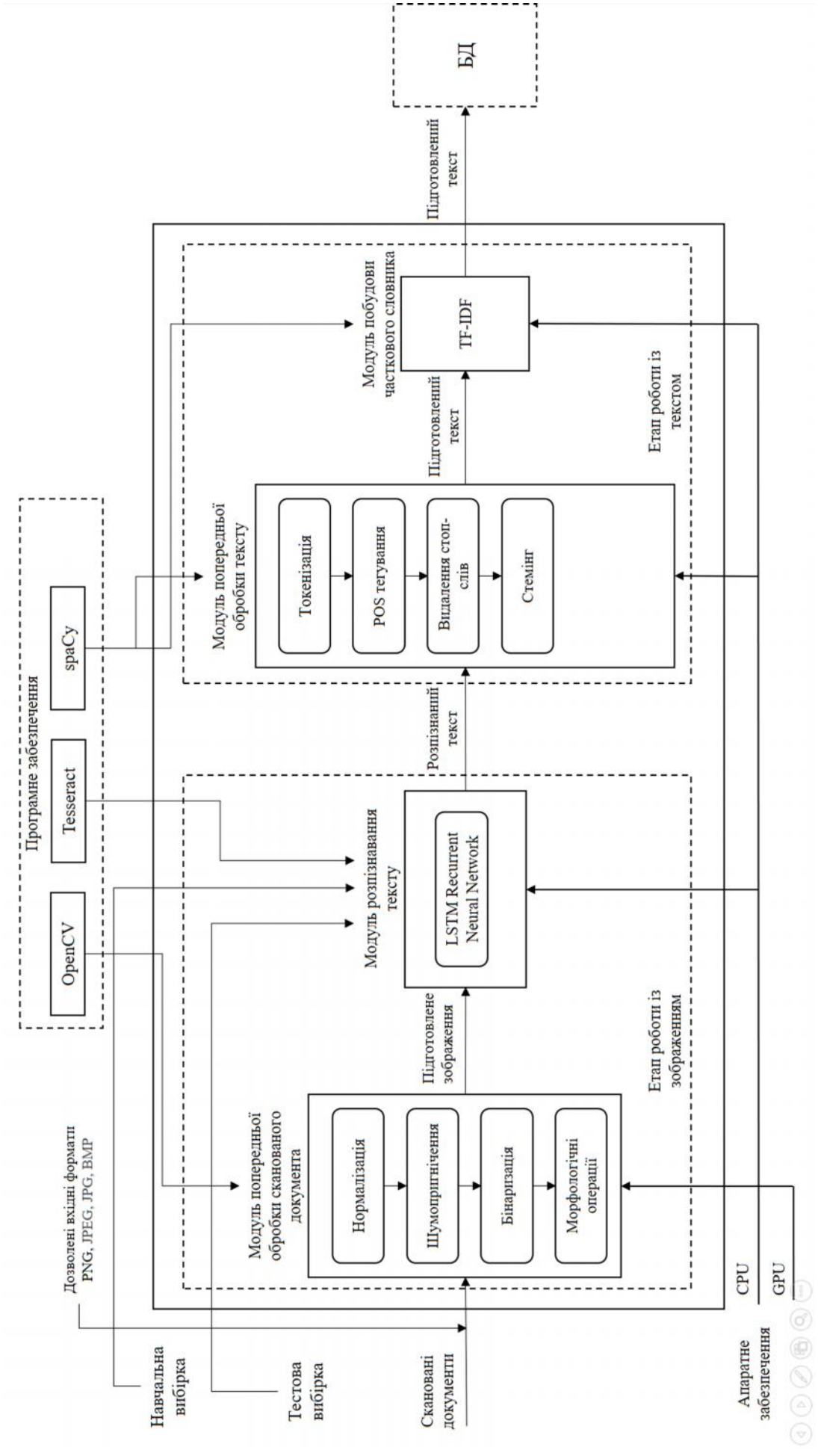

Рисунок 3.1 – Система впорядкованого зберігання сканованих документів

3.2 Дослідження методів роботи із зображеннями

Отримане зображення сканованого документа (рисунок 4.4) потрібно згладити, щоб зменшити можливі шуми. Зниження шуму можливо досягнути застосувавши фільтрацію Гауса чи медіанний фільтр. Приклади роботи фільтрів зображено на рисунку 4.5 та 4.6 відповідно. Як можна побачити з результатів, медіанний фільтр справляється краще.

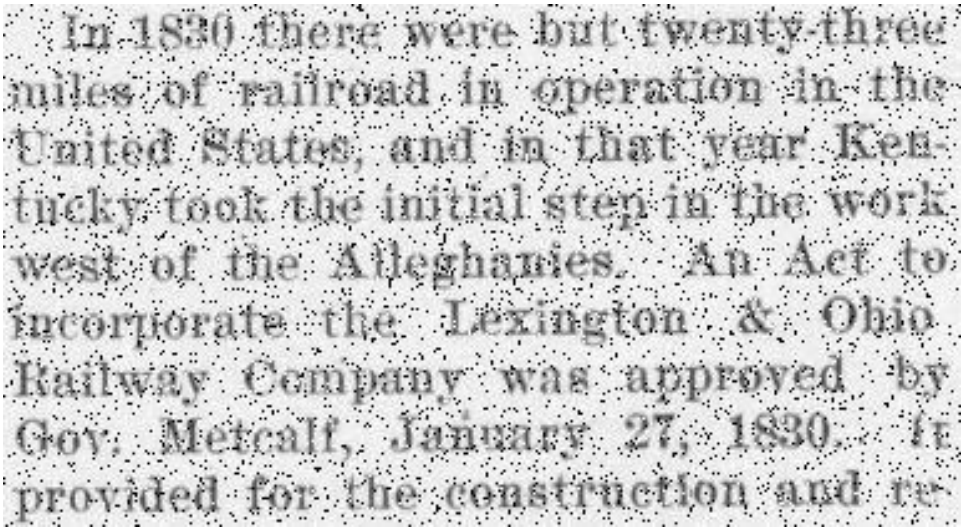
In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830. It
provided for the construction and re-

Рисунок 4.4 – Вхідне зображення

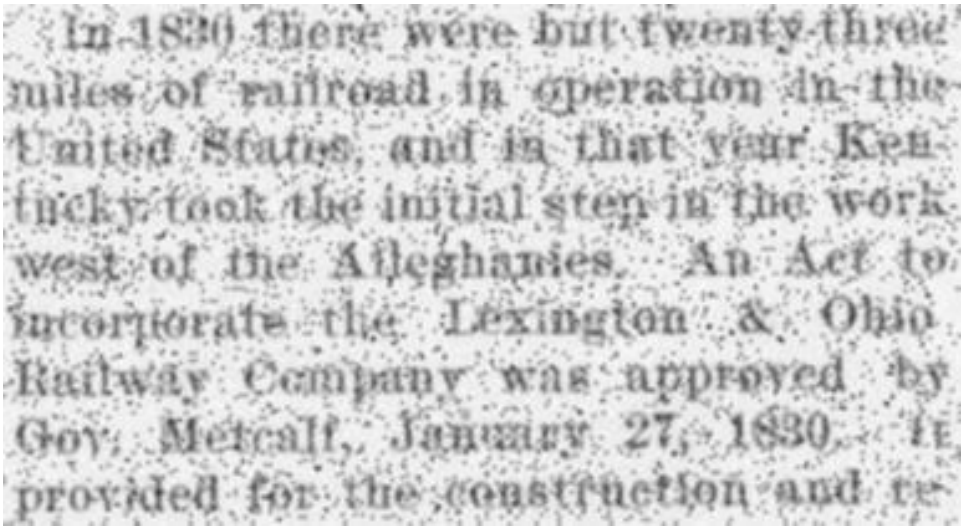
In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830. It
provided for the construction and re-

Рисунок 4.5 – Результат застосування фільтру Гауса

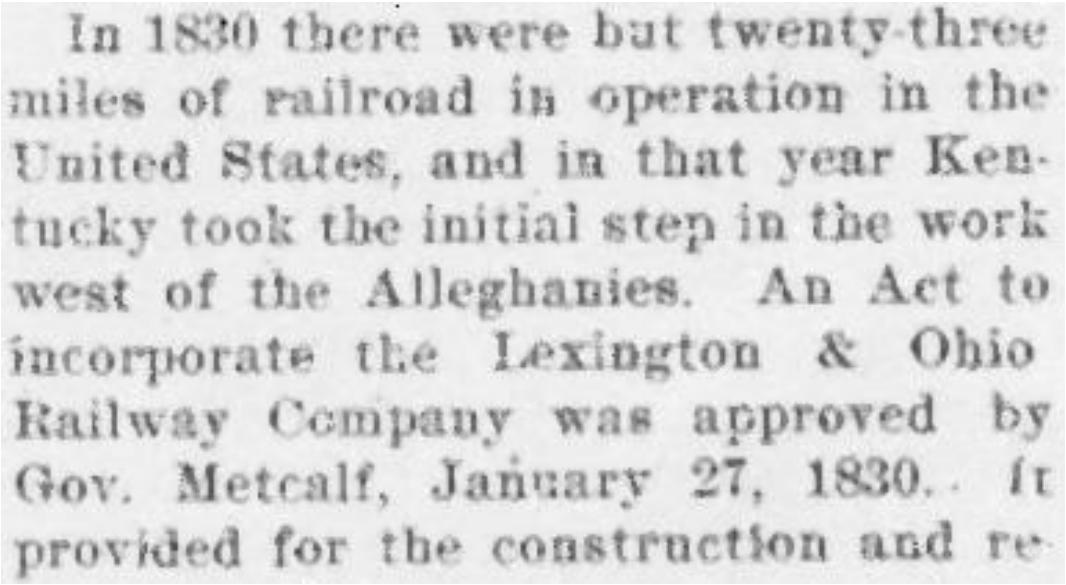
In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830. It
provided for the construction and re-

Рисунок 4.6 – Результат застосування фільтру Гауса медіанного фільтру

Далі фільтроване зображення треба бінаризувати, для цього застосовується метод Оцу. За допомогою нього пікселі розділяються на два класи, за розрахованим порогом, при якому досягається мінімальне значення дисперсії (рисунок 4.7).

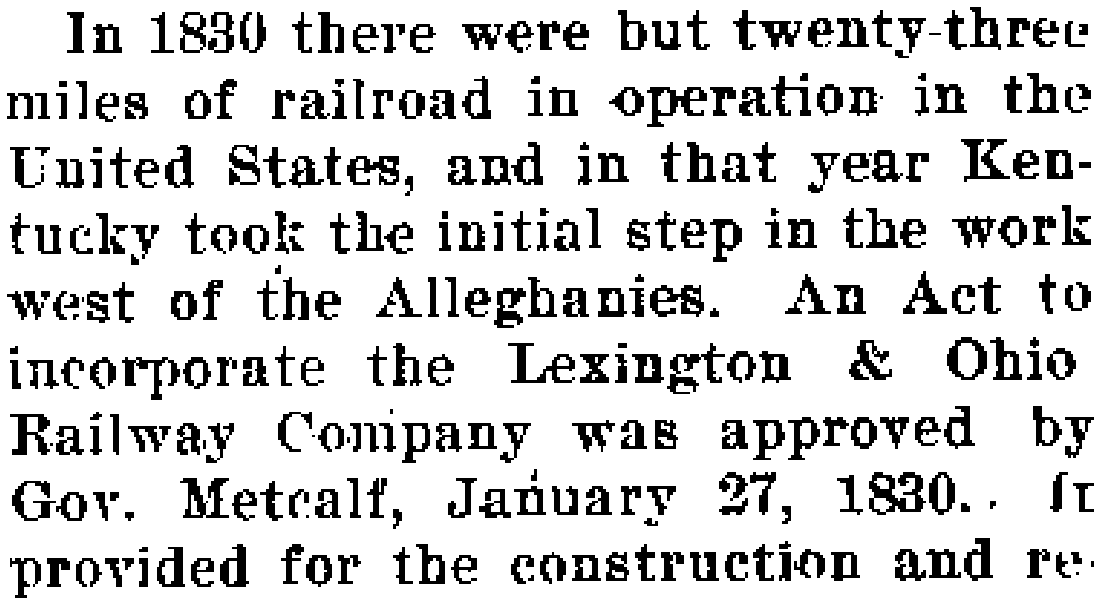
In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830. It
provided for the construction and re-

Рисунок 4.7 – Результат бінаризації

Для зменшення кількості можливих артефактів та збільшенню чіткості тексту до зображення застосовуються морфологічні операції (рисунок 4.6).

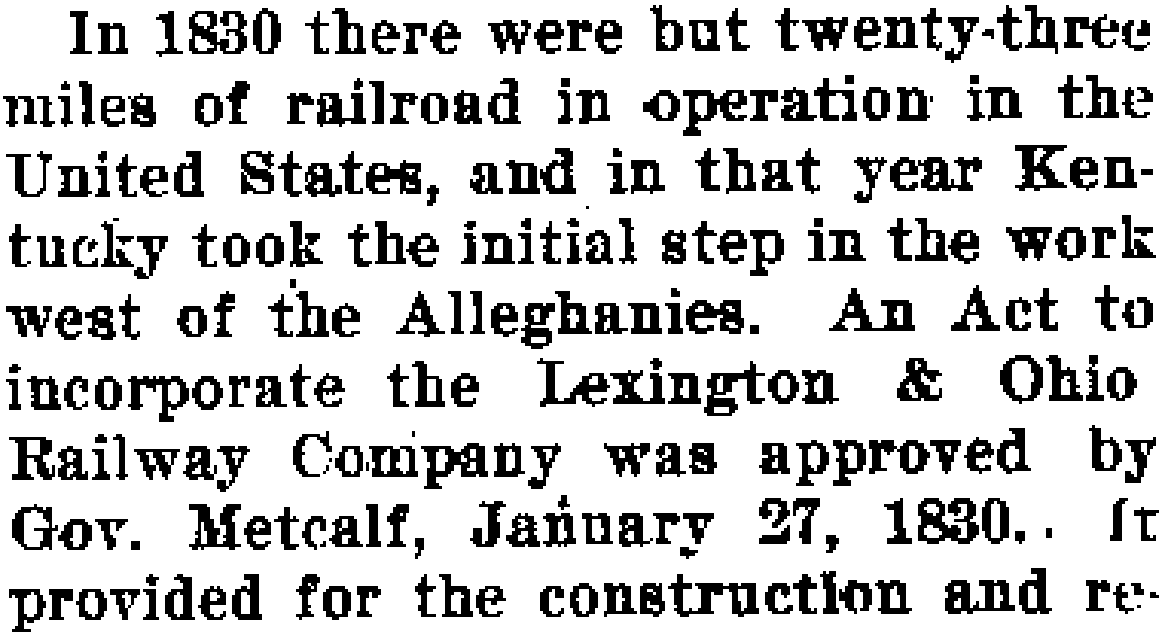
In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830. It
provided for the construction and re-

Рисунок 4.8 – Результат морфологічних операцій

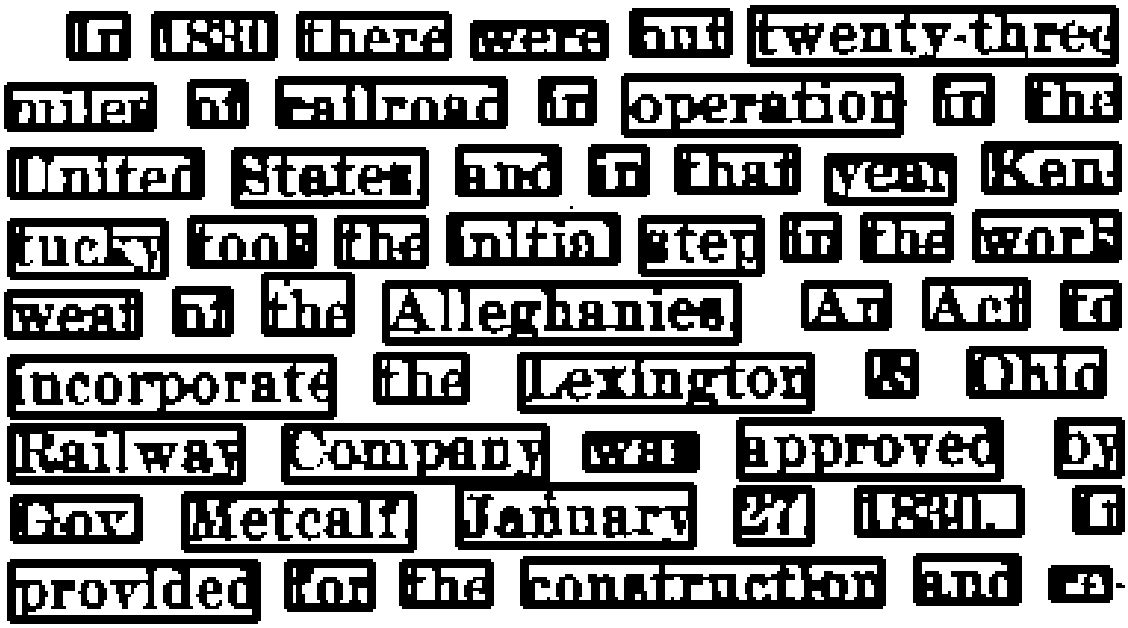
In 1830 there were but twenty-three
miles of railroad in operation in the
United States, and in that year Ken-
tucky took the initial step in the work
west of the Alleghanies. An Act to
incorporate the Lexington & Ohio
Railway Company was approved by
Gov. Metcalf, January 27, 1830. It
provided for the construction and re-

Рисунок 4.9 – Результат сегментації

Метою паралельних обчислень є вирішення складної задачі шляхом її розподілу на підмножини та розв’язання кожного завдання як окремого потоку, який може виконуватися паралельно. Деякі процеси алгоритму можуть виконуватись одночасно паралельно, а деякі повинні виконуватися послідовно один за одним. Паралельним алгоритмом називається алгоритм, у якому завдання можна виконувати одночасно завдяки незалежності данних.

3.3 Дослідження методів роботи із текстом

Для аналізу розпізнаного тексту була застосована бібліотека spaCy. Це безкоштовна бібліотека з відкритим вихідним кодом для розширеної обробки природної мови (NLP) на Python. ЇЇ можна використовувати для створення систем вилучення інформації або розуміння природної мови або для попередньої обробки тексту для глибокого навчання.

Хоча деякі функції spaCy працюють незалежно, інші вимагають завантаження навчених конвеєрів, які дозволяють spaCy передбачати мовні анотації – наприклад, чи є слово дієсловом чи іменником. Навчений конвеєр може складатися з кількох компонентів, які використовують статистичну модель, навчену на позначених даних. Наразі spaCy пропонує навчені конвеєри для різних мов, які можна встановити як окремі модулі Python. Пакети конвеєрів можуть відрізнятися за розміром, швидкістю, використанням пам’яті, точністю та даними, які вони містять. Зазвичай вони включають такі компоненти:

- двійкові вагові коефіцієнти для теггера частини мови, аналізатора залежностей і розпізнавання іменованих об’єктів, щоб передбачити ці анотації в контексті.
- лексичні записи у словнику, тобто слова та їхні незалежні від контексту атрибути, як-от форма чи написання.
- файли даних, такі як правила лемматизації та таблиці пошуку.

- вектори слів, тобто багатовимірні представлення значень слів, які дозволяють визначити, наскільки вони схожі один на одного.

- параметри конфігурації, як-от параметри мови та обробки конвеєру та реалізація моделі, які потрібно використовувати, щоб привести spaCy у правильний стан під час завантаження конвеєра.

3.3.1 Конвеєр обробки

Конвеєр обробки складається з одного або кількох компонентів конвеєра, які викликаються по порядку. Токенізатор запускається перед компонентами. Вони можуть містити статистичну модель і навчені коефіцієнти або лише вносити зміни до документу на основі правил. spaCy надає ряд вбудованих компонентів для різних завдань обробки мови, а також дозволяє додавати власні компоненти.

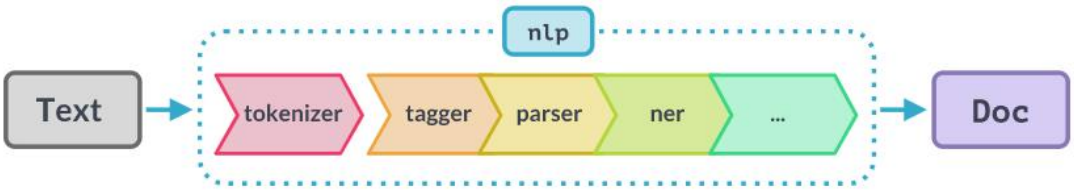

Рисунок 3.7 Схема конвеєра

3.3.2 Токенізація

Під час обробки тексту спочатку потрібно його токенізувати, тобто сегментувати на слова, розділові знаки тощо. Це робиться шляхом застосування правил, характерних для кожної мови. Наприклад, розділові знаки в кінці речення слід відокремити. Кожен документ складається з окремих маркерів, і їх можна визначити.

Проведемо токенізацію для вхідного тексту : «"Elon Musk built his electric car company, Tesla, around the promise that it represented the future of driving".

```
Elon
Musk
built
his
electric
car
company
,
Tesla
,
around
the
promise
that
it
represented
the
future
of
driving
```

Рисунок 3.8 – Результат токенізації тексту

3.3.3 POS тегування

Після токенізації можна провести тегування документу. Саме тут на допомогу приходять навчений конвеєр та його статистичні моделі, які дають змогу прогнозувати, який тег або мітка, швидше за все, застосовуються в цьому контексті. Навчений компонент включає двійкові дані, які виробляються шляхом показу системі достатньої кількості прикладів, щоб вона могла робити прогнози, які узагальнюються по всій мові – наприклад, слово після «the» в англійській мові, швидше за все, є іменником.

Лінгвістичні анотації доступні як атрибути Token. Всі рядки кодуються в хеш-значення, щоб зменшити використання пам’яті та підвищити ефективність.

Аналіз включає наступні атрибути:

- Text: оригінальний текст слова.
- Lemma: Основна форма слова.
- POS: простий тег частини мови UPOS.
- Tag: детальний тег частини мови.
- Dep: синтаксична залежність, тобто відношення між лексемами.

TEXT	LEMMA	POS	TAG	DEP		
Elon	Elon	PROPN	NNP	compound		
Musk	Musk	PROPN	NNP	nsubj		
built	build	VERB	VBD	ROOT		
his	-PRON-	DET	PRP$	poss		
electric		electric		ADJ	JJ	amod
car	car	NOUN	NN	compound		
company	company		NOUN	NN	dobj	
,	,	PUNCT	,	punct		
Tesla	Tesla	PROPN	NNP	appos		
,	,	PUNCT	,	punct		
around	around	ADP	IN	prep		
the	the	DET	DT	det		
promise	promise		NOUN	NN	pobj	
that	that	SCONJ	IN	mark		
it	-PRON-	PRON	PRP	nsubj		
represented		represent		VERB	VBD	acl
the	the	DET	DT	det		
future	future	NOUN	NN	dobj		
of	of	ADP	IN	prep		
driving	driving		NOUN	NN	pcomp	

Рисунок 3.9 – Результат токенізації тексту

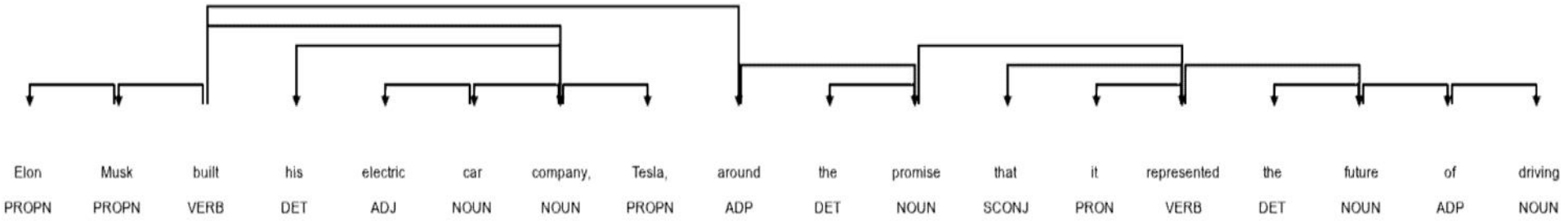

Рисунок 3.10 – Візуалізація аналізу залежності

3.3.4 Розпізнавання іменованих об’єктів

SpaCy має надзвичайно швидку статистичну систему розпізнавання об’єктів, яка призначає мітки суміжним діапазонам маркерів. За замовчуванням навчені конвеєри можуть ідентифікувати різноманітні іменовані та числові об’єкти, включаючи компанії, місцеположення, організації та продукти.

Іменований об’єкт – це «об’єкт реального світу», якому присвоєно ім’я, наприклад, особа, країна, продукт чи назва книги. Розпізнавати різні типи іменованих сутностей в документі можна завдяки моделі для передбачення. Оскільки моделі є статистичними і сильно залежать від прикладів, на яких їх навчали, це не завжди працює ідеально і може знадобитися додаткове налаштування, залежно від випадку використання.

Отриманий результат представлений у таблиці 3.1 та на рисунку 3.11 у вигляді виділених іменованих об’єктів.

Таблиця 3.1 – Іменовані об’єкти

Текст	Початковий індекс	Кінцевий індекс	Сутність
Elon Musk	0	9	PERSON
Tesla	42	47	ORG

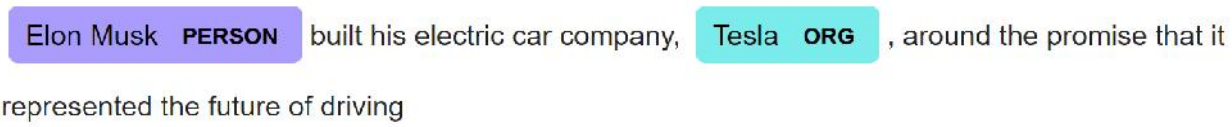

Рисунок 3.11 – Результат розпізнавання іменованих об’єктів

3.3.5 Видалення стоп-слів

Стоп-слово – це часто вживане слово (наприклад, «the», «a», «an», «in»), яке пошукова система була запрограмована на ігнорування, як під час

індексування записів для пошуку, так і під час їх отримання. як результат пошукового запиту.

Тому, щоб ці слова не займали місце в базі даних або забирали дорогоцінний час обробки їх потрібно видалити.

Лістинг 3.1 – Видалення стоп-слів

```
from spacy.lang.en.stop_words import STOP_WORDS

for word in token_list:
    lexeme = nlp.vocab[word]
    if lexeme.is_stop == False:
        filtered_sentence.append(word)
```

Лістинг 3.2 – Результат видалення стоп-слів

```
Source
['Elon', 'Musk', 'built', 'his', 'electric', 'car', 'company',
'Tesla', 'around', 'the', 'promise', 'that', 'it',
'represented', 'the', 'future', 'of', 'driving']
Without stop words
['Elon', 'Musk', 'built', 'electric', 'car', 'company', 'Tesla',
'promise', 'represented', 'future', 'driving']
```

3.3.6 TF-IDF

Наступним кроком буде знаходження частоти вживаності слів. Для цього будується матриця частотності. Код для отримання матриці частотності наведений в лістингу

Лістинг 3.3 – Метод для побудови матриці частотності

```
def get_frequency_matrix(sentences):
    frequency_matrix = {}
    ps = PorterStemmer()
    stopWords = set(stopwords.words("english"))

    for sent in sentences:
```

```
        table = {}
        words = nltk.word_tokenize(sent)

        for word in words:
            word = word.lower()
            word = ps.stem(word)
     if word in stopWords:
                continue

    if word in table:
                table[word] += 1
            else:
                able[word] = 1

        frequency_matrix[sent[:15]] = table

    return frequency_matrix
```

Лістинг 3.4 – Матриця частотності

```
{'Elon': {'elon': 1},
'Musk': {'musk': 1},
'built': {'built': 1},
'electric': {'electr': 1},
'car': {'car': 1},
'company': {'compani': 1},
'Tesla': {'tesla': 1},
'promise': {'promis': 1},
'represented': {'repres': 1},
'future': {'futur': 1},
'driving': {'drive': 1}}
```

3.3.7 Прискорення аналізу

Обробка текстів у вигляді потоку зазвичай ефективніша, ніж обробка їх один за одним. Це робиться за допомогою виклику мовного каналу, який внутрішньо розділяє дані на пакети, щоб зменшити кількість викликів функцій чистого Python. Це означає, що чим більше даних, тим кращий приріст продуктивності може отримати nlp.pipe.

Щоб використовувати мовний канал для потокової передачі текстів, визначено новий метод лемматизатора, який безпосередньо працює з об’єктом Doc. Потім цей метод викликається пакетами для роботи з послідовністю об’єктів Doc, які передаються через канал.

Попередні робочі процеси послідовно обробляли кожен документ новин, щоб створити списки лем, які потім були додані до DataFrame як новий стовпець. Оскільки вихід кожного рядка повністю незалежний від іншого, це паралельна проблема, що робить його ідеальним для використання кількох ядер. SpaCy рекомендує бібліотеку joblib для паралельної обробки блоків конвеєра NLP. Щоб розпаралелювати робочий процес, необхідно визначити ще кілька допоміжних методів.

Поділ на фрагменти: вміст статті новин – це список (довгих) рядків, де кожен документ представляє текст окремої статті. Ці дані повинні подаватися частинами до кожного робочого процесу, запущеного joblib. Кожен виклик методу chunker повертає генератор, який містить лише текст цього конкретного фрагмента у вигляді списку рядків. Під час лемматизації кожен новий фрагмент витягується на основі індексу ітератора (при цьому попередні фрагменти «забуваються»).

Згладжування: як тільки joblib створює набір робочих процесів, які працюють над кожним фрагментом, кожен працівник повертає «список списків», що містить леми для кожного документа. Ці списки потім об’єднуються виконавцем, щоб отримати 3-рівневий вкладений остаточний список. Щоб довжина виводу від виконавця була такою ж, як і фактична кількість статей, визначено метод «зрівняння», щоб об’єднати результат у список списків, що містять леми.

На додаток до вищезазначених методів, для кожного фрагмента текстів використовується подібний метод nlp.pipe.. Кожен із цих методів обгорнутий у метод preprocess_parallel, який визначає кількість робочих процесів, які будуть використані, розбиває вхідні дані на шматки та повертає результат.

4 АНАЛІЗ ОТРИМАНИХ РЕЗУЛЬТАТІВ

Дослідження проводилося на компьютері з характеристиками, які зазначені у таблиці 4.1.

Таблиця 4.1 – Характеристики комп'ютера

CPU	Intel Core i5-4210U (1.7 - 2.7 ГГц)
GPU	nVidia GeForce 840M
GPU (характеристики)	384 ядра CUDA, 1029 МГц, 2 ГБ DDR3
RAM	12 Gb
OS	Windows 10

Потенційна перевага паралельних обчислень зазвичай вимірюється часом, необхідним для виконання завдання на одному процесорі, порівняно з часом, необхідним для виконання того ж завдання на паралельних процесорах.

$$Speedup = \frac{T_s}{T_p} \tag{4.1}$$

де T_s – час виконання послідовного алгоритму;

T_p – час виконання паралельного алгоритму.

4.1 Оцінка таймінгу роботи алгоритму обробки зображення

Прискорення алгоритму попередньої обробки зображень відбовалося завдяки розпаралеленню методів фільтрації на GPU. Результати прискорення на зображеннях різної роздільної здатності завдяки паралельним обчисленням зображені в таблиці 4.2.

Таблиця 4.2 – Результат виконання

Розмір зображення	Час виконання		Прискорення
	CPU	GPU	
2560x1600	1176,00	173,45	6,780051888
1200x800	955,33	167,67	5,697813121
767x528	844,00	153,76	5,489073881
640x480	456,67	190,43	2,398081535
440x292	315,33	234,67	1,34375

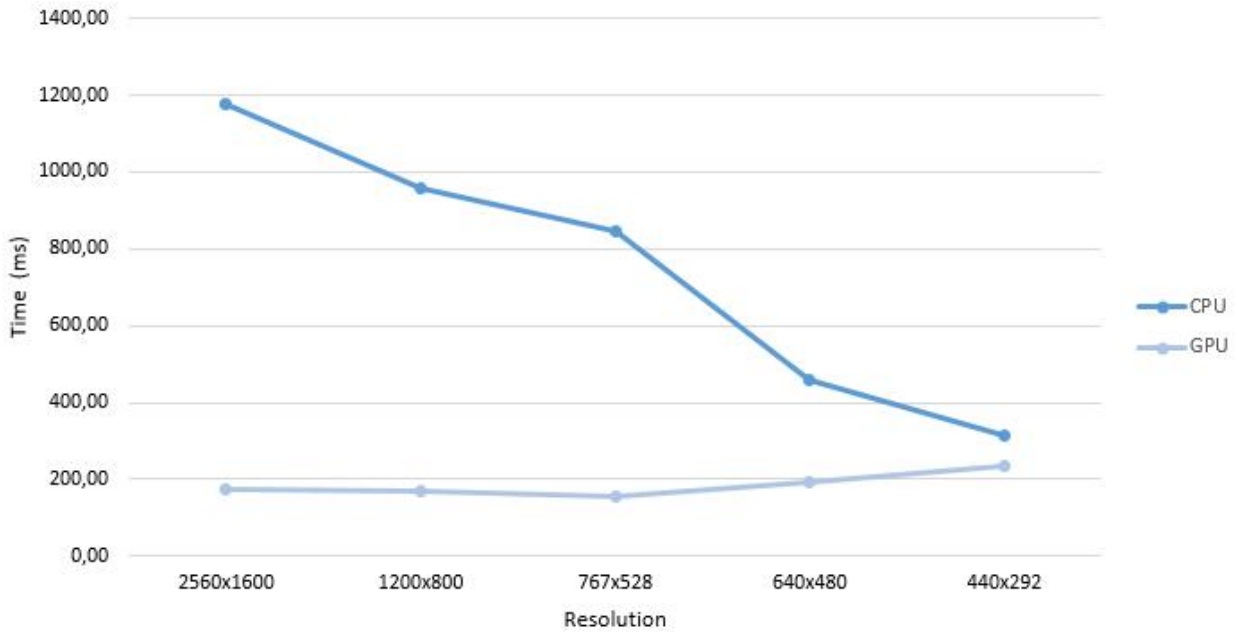

Рисунок 4.1 – Графік прискорення

Як видно по графіку (рисунок 4.1) прискорення зростає при збільшенні кількості пікселей зображення. Таким чином, для великої кількості зображень з високою роздільною здатністю краще виконувати обчислення на графічному процессорі.

4.2 Оцінка таймінгу при аналізі тексту

Прискорення аналізу відбувалося завдяки розділенню даних на пакети, використовуючи nlp.pipe та розпаралеленню робочого процесу на декілька потоків. Результати виконання подані в таблиці 4.4.

Таблиця 4.4 – Результати методів прискорення для аналізу тексту

Метод		Кількість документів		
		1	10	25
Без прискорення		3.5 с	39 с	107 с
nlp.pipe		3.6 с	28 с	72 с
Розпаралелення	2 потоки	4.5 с	21 с	49 с
	4 потоки	5.2 с	11 с	27 с

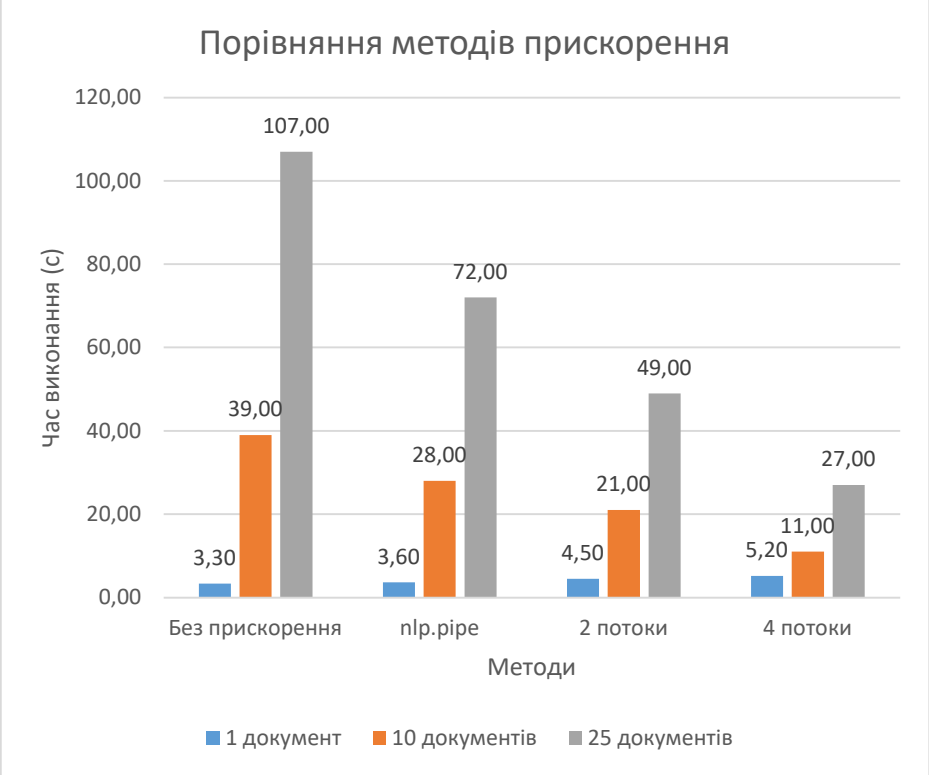

Рисунок 4.2 – Гістограма прискорення

Згідно результатам, зображених на гістограмі (рисунок 4.2), видно тенденцію прискорення. Розпаралелений алгоритм на декілька потоків має перевагу лише при великій кількості документів і чим більша кількість потоків тим більше прискорення. Якщо потрібно проаналізувати невелику кількість тексту краще розділяти дані на частини і передавати в nlp.pipe.

4.3 Оцінка якості роботи системи впорядкованого зберігання сканованих документів із характерними ознаками

Аналіз тексту проводився на попередньо навчених конвеєрах для англійської мови. Характеристики конвеєрів наведені в таблиці 4.2.

Таблиця 4.2 – Характеристики розглянутих контейнерів

Характеристика	Pipeline			
	en_core_web_sm	en_core_web_md	en_core_web_lg	en_core_web_trf
Тип	Vocabulary syntax entities	Vocabularysyntax entities vectors	Vocabulary syntax entities vectors	Vocabularysyntax entities
Розмір	13 MB	43 MB	741 MB	438 MB
Кількість векторів	0	20 тис.	685 тис.	0

Згідно результатам оцінки точності наведеними в таблиці 4.3, можна зазначити, що незважаючи на невеликий розмір конвеєра en_core_web_sm, точність аналізу тексту достатньо висока. Якщо ж потрібна максимальна точність та обмеження в пам’яті відсутні, то найкращим вибором буде конвеєр en_core_web_trf.

Таблиця 4.3 – Точність аналізу попередньо навчених контейнерів

Параметр	Точніть pipeline			
	en_core_web_sm	en_core_web_md	en_core_web_lg	en_core_web_trf
Токенізація	1.00	1.00	1.00	1.00
Pos тегування	0.97	0.97	0.97	0.98
Сегментація речення	0.92	0.91	0.92	0.96
Пошук залежностей	0.9	0.9	0.9	0.94
Розпізнавання іменованих об’єктів	0.85	0.85	0.86	0.9

Таким чином, запропонована у роботі система відповідає висунутим до неї вимогам та вирішує одразу декілька задач - обробку тексту для подальшого нейромережевого розпізнавання, а також попередню обробку та підготовку тексту для подальшого його аналізу із метою впорядкованого зберігання в електронному сховищі великої кількості сканованих документів.

ВИСНОВКИ

Оптичне розпізнавання символів являється придатним і надійним інструментом для розпізнавання тексту на зображеннях. Але для того, щоб підвищити якість і точність потрібна попередня обробка сканованого документу.

В процесі виконання кваліфікаційної роботи було проведено дослідження параметрів, які повинна задовольняти система розпізнавання сканованих документів. Також були розглянуті існуючі методи обробки зображення для шумопригнічення та сегментації.

На підставі проведеного дослідження була запропонована та розроблена система впорядкованого зберігання сканованих документів з виділенням ключових слів. В процесі розробки системи була проведена адаптація методів шумопригнічення та сегментації сканованих документів під різні типи обчислювачів. Також для вирішення завдання аналізу тексту було запропоновано методи прискорення обробки тексту.

В ході дослідження були проведені оцінка таймінгу роботи розробленої системи впорядкованого зберігання сканованих документів та аналіз отриманих результатів.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Багрова, И. А. Выбор признаков для распознавания печатных кириллических символов [Електронний ресурс] / И. А. Багрова, А. А. Грицай, С. А. Пономарев. – 2010. – Режим доступу до ресурсу: https://core.ac.uk/download/pdf/74268943.pdf.

2. Harraj A. E. OCR ACCURACY IMPROVEMENT ON DOCUMENT IMAGES THROUGH A NOVEL PRE-PROCESSING APPROACH / A. E. Harraj, N. Raissouni. – 2015.

3. Optical Character Recognition (OCR) [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://ukdiss.com/examples/optical-character-recognition.php.

4. Optical Character Recognition – A Combined ANN/HMM Approach : дис. докт. / Sheikh Faisal Rashid, 2014. – 161 с.

5. Best OCR Software of 2021 [Електронний ресурс]. – 2021. – Режим доступу до ресурсу: https://nanonets.com/blog/ocr-software-best-ocr-software/.

6. Shrestha P. OPTICAL CHARACTER RECOGNITION [Електронний ресурс] / Prakash Shrestha. – 2018. – Режим доступу до ресурсу: https://www.theseus.fi/bitstream/handle/10024/151264/Shrestha_Pramoj.pdf?sequence=1&isAllowed=y.

7. Mumtazimah M. A Review on OpenCV [Електронний ресурс] / M. Mumtazimah, M. Yazid, S. H. Muhammad. – 2015. – Режим доступу до ресурсу: https://www.researchgate.net/publication/280977983_A_Review_on_OpenCV.

8. Ohlsson V. Optical Character and Symbol Recognition using Tesseract [Електронний ресурс] / Victor Ohlsson. – 2016. – Режим доступу до ресурсу: http://www.diva-portal.org/smash/get/diva2:1019846/FULLTEXT02.pdf.

9. Smith R. An Overview of the Tesseract OCR Engine [Електронний ресурс] / Ray Smith – Режим доступу до ресурсу: https://static.googleusercontent.com/media/research.google.com/en//pubs/archive/33418.pdf.

10. Otsu Thresholding [Електронний ресурс]. – 2010. – Режим доступу до ресурсу: http://www.labbookpages.co.uk/software/imgProc/otsuThreshold.html.

11. Morphological Image Processing [Електронний ресурс] – 2000. – Режим доступу до ресурсу: https://www.cs.auckland.ac.nz/courses/compsci773s1c/lectures/ImageProcessing-html/topic4.htm.

12. Morphological Transformations [Електронний ресурс] – 2021. – Режим доступу до ресурсу: https://docs.opencv.org/3.4.15/db/df6/tutorial_erosion_dilatation.html.

13. Morphology Filters [Електронний ресурс] – Режим доступу до ресурсу: http://www.theobjects.com/dragonfly/dfhelp/4-0/Content/05_Image%20Processing/Morphology%20Filters.htm.