Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Технології оптичного розпізнавання реквізитів банківських карт

Виконав(ла): студент(ка) VI курсу, групи СІм-61
спеціальності 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)

(підпис) Бурмістр В.О. (прізвище та ініціали)

Керівник (підпис) Осухівська Г.М. (прізвище та ініціали)
Нормоконтроль (підпис) Луцик Н.С. (прізвище та ініціали)
Завідувач кафедри (підпис) Осухівська Г.М. (прізвище та ініціали)
Рецензент (підпис) Крамар О.І. (прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Осухівська Г.М.
(підпис) (прізвище та ініціали)
« » 2020 р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)
за спеціальністю 123 «Комп'ютерна інженерія»
(шифр і назва спеціальності)
студенту Бурмістру Володимиру Олександровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Технології оптичного розпізнавання реквізитів банківських карт

Керівник роботи Кандидат технічних наук, доцент, Осухівська Галина Михайлівна
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
Затверджені наказом ректора від «*28*» *вересня* 2020 року № *4/7 - 687*
2. Термін подання студентом завершеної роботи 22.12.2020 р.
3. Вихідні дані до роботи Програмний модуль на мові C# у вигляді окремого класу «ImageProcessing» для попереднього опрацювання зображення

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз сучасних технологій оптичного розпізнавання та можливості їх використання для розпізнавання реквізитів банківської карти. Дослідження можливості покращення результату розпізнавання реквізитів банківської карти. Створення власного алгоритму попереднього опрацювання зображення з метою покращення точності розпізнавання реквізитів банківської карти. Розробка програмного модуля який буде працювати за розробленим алгоритмом попереднього опрацювання зображення.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Тема та актуальність роботи. 2. Мета і задачі 3. Алгоритм оптичного розпізнавання.
4. Алгоритм опрацювання зображення 5. Розроблене програмне забезпечення.
6. Вплив опрацювання зображення на результат розпізнавання.
7. Результат тестування алгоритму попереднього опрацювання зображення банківської карти.
8. Висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	Осухівська Г.М.		
Безпека життєдіяльності	Стадник І.Я.		

7. Дата видачі завдання 01.10.2020

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Затвердження теми кваліфікаційної роботи магістра	02.10.2020	
2	Аналіз літературних джерел	03.10.2020-13.10.2020	
3	Обґрунтування актуальності досліджень	14.10.2020-21.10.2020	
4	Аналіз предмета дослідження та предметної області	22.10.2020-03.11.2020	
5	Огляд існуючих технологій оптичного розпізнавання символів	04.11.2020-15.11.2020	
6	Оформлення розділу «Огляд технологій оптичного розпізнавання символів»	16.11.2020-25.11.2020	
7	Дослідження методів покращення якості зображення	26.11.2020-07.12.2020	
8	Оформлення розділу «Методи оптичного розпізнавання символів та опрацювання зображення банківської карти»	08.12.2020-09.12.2020	
9	Розробка алгоритму попереднього опрацювання зображення, та його програмного представлення	10.12.2020-11.12.2020	
10	Оформлення розділу «Програмна реалізація опрацювання зображення банківської карти»	12.12.2020-13.12.2020	
11	Оформлення розділу «Охорона праці та безпека в надзвичайних ситуаціях»	14.12.2020-15.12.2020	
12	Захист кваліфікаційної роботи магістра	22.12.2020	

Студент _____ Бурмістр В.О.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Осухівська Г.М.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Технології оптичного розпізнавання реквізитів банківських карт// Кваліфікаційна робота магістра // Бурмістр Володимир Олександрович // ТНТУ, Комп'ютерна інженерія, група СІм-61 // Тернопіль, 2020 // С. – 66, рис. – 21, табл. – 6, бібліогр. – 30.

Ключові слова: реквізити, банківська карта, опрацювання зображення, оптичне розпізнавання.

Кваліфікаційну роботу магістра присвячено дослідженню технологій оптичного розпізнавання реквізитів банківських карт з метою покращення якості оптичного розпізнавання їх реквізитів. Створено алгоритм попереднього опрацювання вхідного зображення та програмний модуль попереднього опрацювання зображення банківської карти.

Для покращення якості зображення запропоновано використання лінійних та нелінійних фільтрів, перетворення зображення у сірі відтінки, бінаризацію зображення з локальним максимумом та мінімумом. Модуль попереднього опрацювання зображення, в якому реалізований алгоритм покращення зображення, написано на мові c#, що дозволяє в майбутньому вбудовувати цей модуль у велику кількість систем з оптичним розпізнаванням символів реквізитів банківської карти, включаючи і мобільні додатки.

У роботі наведені конкретні результати розпізнавання реквізитів банківських карт із застосуванням алгоритму попереднього опрацювання зображення, і порівняно ці результати з розпізнаванням без попереднього опрацювання.

ANNOTATION

Technologies for recognizing bank card details // Qualification work of the master // Burmistr Volodymyr Oleksandrovych // TNTU, Computer Engineering, SIM-61 group // Ternopil, 2020 // P. - 66, fig. - 21, table. - 6, bibliogr. - 30.

Keywords: details, bank card, image processing, optical recognition.

The master's qualification work is devoted to the study of technologies of optical recognition of bank card details in order to improve the quality of optical recognition of their details. An algorithm for pre-processing the input image and a software module for pre-processing the bank card image have been created.

To improve the image quality, it is proposed to use linear and nonlinear filters, convert the image to grayscale, binarize the image with a local maximum and minimum. The image pre-processing module, which implements the image enhancement algorithm, is written in c#, which allows in the future to embed this module in a large number of systems for optical character recognition of bank card details, including mobile applications.

The paper presents specific results of recognition of bank card details using the image pre-processing algorithm, and compares these results with recognition without pre-processing.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ 8
ВСТУП 9
РОЗДІЛ 1 ОГЛЯД ТЕХНОЛОГІЙ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ 11
1.1 Розпізнавання реквізитів банківської карти 11
1.2 Оптичне розпізнавання символів 12
1.3 Комп'ютерний зір 14
1.4 Технології оптичного розпізнавання символів 15
1.5 Методи покращення роботи алгоритмів ОРС 18
1.6 Висновок до 1 розділу 18
РОЗДІЛ 2 МЕТОДИ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ ТА ОПРАЦЮВАННЯ ЗОБРАЖЕННЯ БАНКІВСЬКОЇ КАРТИ 20
2.1 Tesseract OCR і його алгоритм розпізнавання символів. 20
2.2 Нейронні мережі, рекурентні нейронні мережі та LSTM нейронні мережі . 23
2.3 Математичне забезпечення для роботи алгоритму Tesseract OCR 25
2.4 Алгоритми опрацювання зображення 26
2.5 Висновок до 2 розділу 31
РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ ОПРАЦЮВАННЯ ЗОБРАЖЕННЯ БАНКІВСЬКОЇ КАРТИ 33
3.1 Алгоритм попереднього опрацювання реквізитів БК 33
3.2 Реалізація методів попереднього опрацювання зображення БК 35
3.3 Тестування 49
3.4 Висновок до розділу 53
РОЗДІЛ 4 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ 55
4.1 Охорона праці 55

4.2 Фактори ризику і можливі порушення здоров’я користувачів комп’ютерної мережі 57
ВИСНОВКИ 61
ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ 63
Додаток А. Тези конференцій 65
Додаток Б. Лістинг коду 8

ПЕРЕЛІК СКОРОЧЕНЬ

OCR - Optical Character Recognition

API - Application Programming Interface

SDK - SoftWare Development Kit

БК - Банківська карта

ОРС - Оптичне розпізнавання символів

НДІ - Науково-дослідний інститут

ШНМ - Штучна нейронна мережа

ВСТУП

Актуальність. У повсякденному житті досить часто виникає потреба у здійсненні певних фінансових операцій з використанням банківської карти, тому зчитування реквізитів з банківських карт використовується у будь якій сфері, де виникає потреба у коректному їх введенні, наприклад, інтернет-банкінг, виставлення рахунків за різні послуги та інше.

Для проведення таких фінансових операцій необхідно вводити номер банківської карти. Для того, щоб уникнути помилок при вводі, збільшити швидкість вводу реквізитів, а також для загального покращення взаємодії з користувачем виникає необхідність в оптичному розпізнаванні реквізитів банківської карти.

Ряд компаній, зокрема, SmartEngine, LabsNeural та інші, займаються розробкою алгоритмів для розпізнавання інформації з банківських карт. Вирішенням проблеми розпізнавання номеру банківської карти із зображення, присвячені роботи Yu.Xin, Yu.Lin, P.Shi, S.Han, B.Tian їх рішення базувались на глибокому навчанні.

Але, з огляду на точність, дослідження технологій оптичного розпізнавання реквізитів банківської карти вимагає подальшого вивчення і є актуальною задачею.

Мета роботи - покращення результату оптичного розпізнавання реквізитів банківських карт.

Досягнення цієї мети передбачено вирішення таких задач:

- проаналізувати існуючі технології оптичного розпізнавання реквізитів банківської карти;
- проаналізувати існуюче програмного забезпечення, яке використовується для розпізнавання реквізитів банківських карт;
- дослідити методи покращення оптичного розпізнавання символів для збільшення точності розпізнавання реквізитів банківської карти;

- запропонувати метод покращення якості опрацювання зображення реквізитів банківської карти;

- розробити програмну реалізацію методу;

- провести тестування запропонованого модуля;

- дослідити точність отриманих даних.

Об'єкт дослідження: процес оптичного розпізнавання реквізитів банківської карти.

Предмет дослідження: технології оптичного розпізнавання реквізитів банківської карти.

Наукова новизна:

- Отримав подальшого розвитку метод підвищення точності розпізнавання реквізитів банківської карти.

- На основі аналізу існуючих технологій вперше запропоновано введення модуля попереднього опрацювання зображення банківських карт з метою розпізнавання їх реквізитів.

Практичне значення результатів: отримані результати дозволяють покращити якість зображення банківських карт для розпізнавання їх реквізитів і можуть бути використані в подальших розробках аналогічних програм, а також для вбудовування в існуючі програми для покращення результату оптичного розпізнавання символів.

Публікації: Результати дослідження апробовано на IX Міжнародній науково-технічній конференції молодих учених та студентів «Актуальні задачі сучасних технологій» м. Тернопіль 25-26 листопада 2020 р. та VIII науково-технічна конференція «Інформаційні моделі, системи та технології» м.Тернопіль. 9-10 грудня 2020 р.

Структура роботи: Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається із вступу, чотирьох розділів, висновків, списку використаних джерел та додатків – загальним обсягом 84 аркуші формату А4, графічна частина – 8 аркушів формату А1.

РОЗДІЛ 1
ОГЛЯД ТЕХНОЛОГІЙ ОПТИЧНОГО РОЗПІЗНАВАННЯ СИМВОЛІВ

1.1 Розпізнавання реквізитів банківської карти

Розпізнавання даних банківської карти(БК) пов’язане з вирішенням декількох проблем, по перше, це оптичне розпізнавання тексту, по друге, це опрацювання зображення. Оскільки йдеться про розпізнавання за фотографією, тому включається ряд факторів якості зображення: це і напрям світла(освітленість), кольори, чіткість контурів тексту на зображенні, інтенсивність освітлення. Тобто складність розпізнавання реквізитів БК залежить від багатьох факторів, але умовно їх можна розділити як типи завдань, які потрібно вирішити. Перше завдання - це опрацювання зображення. Для того, щоб прибрати непотрібну інтенсивність світла або, навпаки, збільшити інтенсивність, виділити контури тексту та забрати лишні шуми. Друге завдання складається саме з оптичного розпізнавання символів на картці.

Оптичне розпізнавання символів є досить розвинутою технологією і для ідеальних зображень, на які не впливають фактори яскравості, куту освітлення, потертості та інших візуальних ефектів, вона працює досить добре, але використання технології в такому випадку є досить обмеженим, оскільки не так багато пристроїв, які можуть робити настільки ідеальні зображення.

Ряд вчених займались розробкою рішень направлених саме на розпізнавання реквізитів БК. Наприклад, група дослідників таких як Yuanxue Xin, Yuhan Lin, Pengfei Shi, Song Han, Bin Tian розробила власний алгоритм, результативність якого досягає 95%(після навчання)[3], але, не дивлячись на такий результат, їх рішення є досить ресурсоємним, що і є приводом до розробки власних рішень, які можна буде вбудовувати у доступні пристрої, наприклад, смартфони.

1.2 Оптичне розпізнавання символів

Оптичне розпізнавання символів (ОРС) - це технологія, яка дозволяє отримати текстову інформацію з різного роду документів, наприклад, з відсканованих документів, фотографій та ін. з метою подальшого використання цих даних у вигляді текстових файлів. Існує багато розроблених технологій з своїми особливостями розпізнавання, але концептуально їх можна поділити на 2 основні типи, це алгоритм розпізнавання з матричною відповідністю, та метод інтелектуального розпізнавання символів.

Метод матричної відповідності має бібліотеку шаблонів з якими послідовно порівнюють зображення, які знаходить програмне забезпечення ОРС. Якщо якийсь відсоток зображення відповідає шаблону, зараховується розпізнавання символа, який відповідає цьому шаблону.

Метод інтелектуального розпізнавання символів є сучаснішим методом і базується він на використанні комп’ютерного інтелекту, завдяки чому технології, які працюють за цим алгоритмом можуть бути навчені, і не потребують сталого шаблону.

Найсучасніші ж технології ОРС мають в своєму складі як алгоритми матричної відповідності, так і методи інтелектуального розпізнавання, оскільки в кожного з цих методів є свої переваги та недоліки.

Розробками алгоритмів для розпізнавання інформації з БК на сьогодні займаються ряд компаній, серед них LabsNeural, SmartEngine та багато інших, а також розроблено додатки, які, за власними алгоритмами, здійснюють «видобування» реквізитів БК. Але, зважаючи на точність розпізнавання інформації, дослідження технологій розпізнавання реквізитів БК залишається все ще актуальною задачею і вимагає подальшого вивчення.

Розвиток алгоритмів ОРС починається в 1929р. з першим патентом Густава Таушека. До 2000-х років найбільшої застосовності ці алгоритми набули в сканерах, наявність таких рішень в сканерах була великою перевагою для користувача, оскільки пошук інформації з відсканованого і оцифрованого

документа був набагато швидшим за стандартний пошук. Одним з піонерів використання таких алгоритмів в своїх принтерах і сканерах була компанія Hewlet Packard, яка створила таке рішення для оптичного розпізнавання як Tesseract OCR.

На даний момент оптичне розпізнавання символів є важливою складовою не тільки програмного забезпечення для принтерів та сканерів, а і в текстових редакторах, програмах, що аналізують зображення, в додатках для перекладу, і т.д. Тобто багато програм використовує ці алгоритми, що призводить до активного розвитку цієї технології.

Використання алгоритмів ОРС дає ряд переваг над традиційним зберіганням даних в паперовому вигляді, та дозволяє покращити рівень клієнтоорієнтованості компанії, яка буде використовувати цю технологію там, де це необхідно. Серед переваг можна виділити такі як:

- Велика продуктивність: перевагою використання її є швидке вилучення необхідних даних з документів, що дозволяє їм орієнтувати свої сили і кошти в більш правильному напрямку.

- Зниження витрат – ОРС вже зараз є набагато ефективнішим в фінансовому плані ніж людина, оскільки розробка і підтримка такої системи буде коштувати набагато менше, ніж заробітна плата людини, яка буде цим займатись.

- Точність розпізнавання – завдяки стрімкому розвитку та потребі в таких технологіях, точність розпізнавання є досить великою, але все ще людина необхідна для перевірки сумнівних моментів, які така система не може розпізнати.

- Зменшення приміщень для зберігання паперових документів – не дивлячись на те, що всі компанії рухаються в напрямку переходу на електронний облік, є ще ряд компаній, які повністю не перейшли на електронний облік, що накладає на такі компанії додаткові витрати на зберігання цих документів.

- Захист даних – захист даних є дуже важливим в компаніях. Завдяки тому що потрібні документи будуть зберігатись як оцифровані файли можна контролювати доступ до цих файлів, тобто зберегти файли від

несанкціонованого доступу, також є випадки втрати паперових документів через вплив на матеріал з якого вони зроблені, що призводить до знищення даних. З наявними системами зберігання даних в електронному вигляді, таких проблем немає.

- Можливість швидкого пошуку даних в документі – це один із ключових моментів чому ці технології є популярними, оскільки бувають ситуації коли потрібно знайти певну інформацію в книзі по ключовим словам. В традиційному форматі книга чи наукова стаття має в своєму складі тільки зміст, в якому коротко записане те що ти можеш знайти, але для більш детального пошуку, потрібно мати електронний(оцифрований) варіант цієї книги.

- Можливість редагування документів – це також одна із ключових цілей створення таких технологій, оскільки з часом інформація змінюється, доповнюється і деякі документи потребують оновлення.

Також слід зазначити, що такі технології використовуються в додатках-перекладачах, що дозволяє користувачу без вводу тексту розпізнавати його за допомогою камери та мати швидкий переклад. В банківській сфері такі технології використовуються для розпізнавання реквізитів БК, що знову ж таки економить час клієнта.

1.3 Комп’ютерний зір

Активна розробка алгоритмів ОРС сприяє розвитку алгоритмів розпізнавання образів (АРО) на зображенні, оскільки для коректного оцифрування специфічних документів, або фотокопій, потрібно спочатку знайти «область» з текстом, а тільки тоді застосувати до неї алгоритм ОРС.

Комп’ютерний зір (КЗ) є важливою технологією для смартфонів, автомобілів, відеокамер спостереження, медичного обладнання, робототехніки, автоматизації виробничих процесів, кінематографії.

Смартфони завдяки технологіям КЗ отримали біометричний захист (такий як FaceID від компанії Apple та аналогічні реалізації інших компаній), додатки,

що дозволяють сканувати приміщення для подальшого планування, та багато інших рішень. Автомобілі з КЗ вже зараз мають функцію автопілоту, яка, за статистикою 2019р., дозволила зменшити кількість аварій, в які потрапляє автомобіль в 6,5 разів[4].

Відеокамери спостереження, завдяки використанню КЗ, отримали функції розпізнавання обличчя, автомобільних номерів, що дає змогу швидше знаходити порушників. Оскільки КЗ може проаналізувати набагато більше даних ніж людина, його доцільно використовувати в медичній галузі, оскільки підготовка висококваліфікованого фахівця, який буде виявляти аномалії в результатах комп'ютерної томографії набагато дорожча та триваліша, ніж створення системи, яка буде аналізувати дані а, завдяки машинному навчанню – ще вчитись. Однією з галузей, де комп'ютерний зір отримав найширше використання є промислове виробництво, оскільки технологія дає змогу швидко і без вагань(які виникають у людини) приймати рішення, з великою точністю виконувати ту чи іншу роботу на конвеєрі, та відбраковувати дефектний продукт.

1.4 Технології оптичного розпізнавання символів

На даний момент технології OPC досить активно розвиваються. Існує ряд технологій і компаній, які в даній області є фаворитами, їх переваги та недоліки варто проаналізувати. Серед компаній, які займаються або займались розробкою власних рішень для OPC можна виділити такі як ABBYY, Hewlett-Packard, Google, Asprise, Nicomsoft. Програмні реалізації для OPC цих компаній розглянуто далі.

ABBYY FineReader Engine - багатофункціональний інструментарій розробника, який дозволяє вбудовувати в додатки інтелектуальні технології розпізнавання даних. За допомогою OPC на основі технологій штучного інтелекту можна створювати додатки з функціями якісного розпізнавання інформації з документів, зображень, фотографій, скріншотів, моніторів і

дисплеїв, визначення типу документа, конвертації сканованих документів у файли форматів Word, Excel і PDF з можливістю пошуку. ABBYY FineReader Engine є кросплатформним рішенням, доступним під Windows, Linux, Mac[5].

Перевагами ABBYY FineReader Engine з поміж конкурентів можна виділити такі моменти як можливість обрати режим розпізнавання символів та тип розпізнавання, повнотекстове або зональне. Режим розпізнавання потрібний, щоб налаштовувати швидкість і точність.

Користувачам доступні такі режими розпізнавання: Normal, Fast, Balanced режими OCR і ICR-розпізнавання. Типи розпізнавання потрібні для того, аби виділяти потрібні дані і відкидати непотрібні. Повнотекстове розпізнавання використовує технологію розпізнавання друкованого тексту (OCR) і застосовується в сценарії конвертації документів. Зональний, тобто розпізнавання на рівні областей, використовує не тільки OCR, але і технологію розпізнавання рукописного тексту (ICR), а також інші технології, які можуть бути затребувані для розпізнавання специфічних даних, розташованих в тій чи іншій області на документі. Також у ABBYY FineReader Engine є можливість зміни мови розпізнавання, та налаштування власного набору мов і правил, які можуть допомогти в розпізнаванні символів. У більшості випадків ABBYY FineReader Engine може успішно розпізнавати текст без попереднього навчання. Проте, щоб підвищити якість розпізнавання декоративних і контурних шрифтів, може знадобитися навчання призначеним для користувача стандартам[6].

Недоліками є цінова політика, яка є індивідуальною до кожної компанії, яка використовує ABBYY FineReader Engine.

Tesseract – це вільна та кросплатформенна система для OPC, компанії Hewlet-Packard. Після того, як компанія закінчила розробку даної технології було прийняте рішення передати її в інформаційний науково-дослідницький інститут при Невадському університеті в Лас Вегасі, для продовження розвитку технології. Участь у вдосконалені розробки та виправленні ряду недопрацювань приймали спеціалісти компанії Google. Тому в 2006р. було прийняте рішення, що технологія отримала достатню стабільність для повторного релізу. В тому ж

2006р. (серпень) році компанія Google придбала цю технологію і відкрила її вихідні тексти під ліцензією Apache2.0.

В 1995 році Tesseract займав третє місце за точністю розпізнавання.[7] В 2006р. Tesseract OCR був найточнішим рішенням для ОРС, з відкритим кодом.

Tesseract може розпізнавати тексти з приблизно 190мов, крім того, є можливість «вивчення» нових мов.

Але Tesseract буде давати досить поганий результат роботи, якщо зображення, яке подається на опрацювання буде мати низьку якість, тому найкращим шляхом вирішення цієї проблеми є попереднє опрацювання зображення.

Останньою версією Tesseract є версія 4.0, головним поліпшенням, якої є використання LSTM (довгої короткочасної пам'яті).

Серед переваг Tesseract 4.0 з-поміж конкурентів можна відмітити: безкоштовне використання через Apache-ліцензію, велику кількість мов та форматів. Недоліками ж даної технології є неякісне розпізнавання певних шрифтів та низька швидкість роботи при розпізнаванні досить великого об'єму тексту на одній сторінці.

Asprise OCR – це SDK бібліотека для ОРС і штрих-кодів. Asprise SDK розпізнає текст у форматі JPEG,TIFF,PDF,PNG, і конвертує їх в документи xml та PDF.

Asprise 15 версії підтримує всього близько 30 мов[8].

Головними перевагами Asprise OCR 15 є точне розпізнавання, висока швидкість та зрозуміле API. Головний недолік цієї бібліотеки - це велика вартість ліцензії.

NicomsoftOCR SDK завдяки власним унікальним рішенням дозволяє розпізнавати більше шрифтів, ніж інші конкуруючі технології. Дане рішення підтримує 26 мов для розпізнавання і для кращого результату розпізнавання використовуються словники. Підтримує велику кількість форматів і дозволяє їх переводити у більшість найпопулярніших текстових форматів. Для підвищення якості розпізнавання є можливість підключення словників. Має зрозумілий API.

Найбільша перевага Nicomsoft OCR те, що він є безкоштовним, а найважливішим недоліком використання Nicomsoft є відсутність технічної підтримки, оскільки розробники технології більше не займаються її покращенням.

1.5 Методи покращення роботи алгоритмів ОРС

Якість ОРС завжди найбільшою мірою залежить від якості вхідного зображення. Програмні реалізації деяких компаній, які спеціалізуються на ОРС мають в своєму складі власний набір функцій для покращення якості зображення. Серед таких програмних реалізацій можна виділити ABBYY FineReader Engine, Asprise та інші.

Серед функцій, які будуть впливати на якість розпізнавання тексту з зображення можна виділити ряд основних:

- виправлення текстових рядків;
- видалення шумів;
- збільшення контрастності;
- адаптивна бінаризація.

Для оптичного розпізнавання реквізитів БК варто детальніше дослідити існуючі методи та, враховуючи особливість досліджуваного зображення, запропонувати найоптимальніший.

1.6 Висновок до 1 розділу

Проаналізувавши тематику роботи встановлено, що дослідження технологій оптичного розпізнавання реквізитів БК є актуальним завданням.

Обґрунтовано доцільність використання для розпізнавання реквізитів БК ПП з відкритим кодом Tesseract OCR.

Встановлено, що для розпізнавання реквізитів БК, враховуючи специфіку нанесення на БК символів, існуючі методи є недостатньо точними для таких

завдань, оскільки на зображення є вплив багатьох чинників, таких як освітлення, матеріал картки, роздільна здатність пристрою, який робить знімки, а також завади у вигляді різнокольорових зображень на БК.

Обґрунтовано необхідність детальнішого дослідження методів покращення ОРС для збільшення точності розпізнавання реквізитів БК.

РОЗДІЛ 2
МЕТОДИ ОРС ТА ОПРАЦЮВАННЯ ЗОБРАЖЕННЯ БК

Завдання розпізнавання реквізитів БК можна умовно розділити на опрацювання вхідного зображення з метою покращення його якості та, безпосередньо, розпізнавання реквізитів БК. Є велика кількість алгоритмів опрацювання зображення, які служать для покращення якості зображення. Але поняття “якість” є досить специфічним для вирішення поставленого завдання. Алгоритми, які проводять оцифрування зображення, найкраще працюють з друкованими, чорно-білими документами, текст яких є чітко горизонтальним. Тому завдання викорінення номеру БК буде зводитися до перетворення зображення у вигляд, який є найзручнішим для існуючих алгоритмів оптичного розпізнавання символів.

2.1 Tesseract OCR і його алгоритм розпізнавання символів.

В Tesseract OCR розпізнавання відбувається за конвеєрною архітектурою, але з деякими «особливостями», які і на даний момент часу залишаються.

Перший етап розпізнавання пов’язаний з компонентним аналізом. Компонентний аналіз дозволяє визначити позицію того чи іншого об’єкта, його відношення до інших об’єктів. Завдяки такому компонентному аналізу Tesseract якийсь час був єдиним алгоритмом, який міг настільки тривіально проводити аналіз чорно-білого тексту. Далі, завдяки компонентному аналізу, структури, які були визначені збираються у «блоби». Блоби організовуються в текстові рядки, і проходять аналіз на наявність фіксованого тону та пропорційного тексту. Рядки тексту розбиваються на слова по-різному, залежно від типу інтервалів між символами. Текст із фіксованою висотою розбивається клітинками символів. Пропорційний текст розбивається на слова за допомогою певних пробілів та нечітких пробілів.

Розпізнавання самого тексту відбувається в два проходи, під час першого проходу проводиться спроба розпізнати кожне слово по черзі. Кожне слово додається до адаптивного класифікатора як навчальні дані, які є інформацією, яка допомагає в розпізнаванні наступних слів, таким чином до кінця сторінки розпізнані дані будуть більш точними, ніж на початку. Через те, що алгоритм під час першого проходу «навчився» розпізнавати слова краще, виникає потреба в другому проході, під час якого розпізнаються слова, які мали низький результат розпізнавання під час першого проходу. Заключним етапом розпізнавання є перевірка нечітких пробілів, та перевірка альтернативних гіпотез з розміщенням малого тексту[9-10].

Послідовність основних етапів роботи Tesseract відображено на рисунку 2.1.

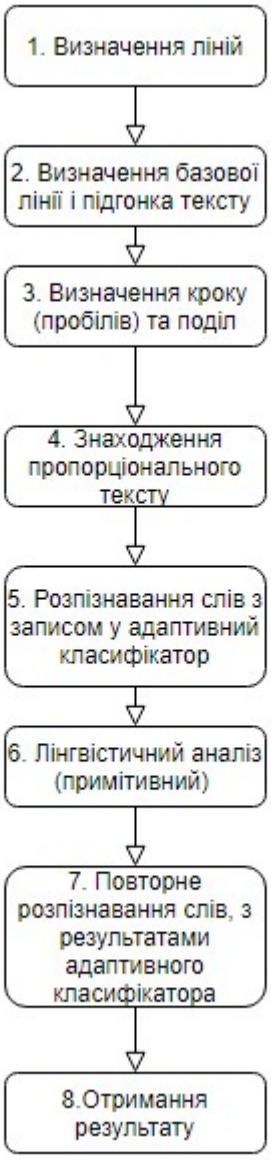

Рис. 2.1 – етапи розпізнавання Tesseract OCR

Принциповою відмінністю Tesseract 4.0 від попередніх версій є використання LSTM-мереж для визначення макета сторінки (layout analysis), а також для виділення меж рядків і окремих слів паралельно зі стандартним алгоритмом.

Впровадження LSTM-мереж в Tesseract призвело до значного поліпшення результатів розпізнавання. У таблиці 2.1 представлено порівняння результатів розпізнавання тексту українською мовою для стандартної версії OCR (base Tesseract). Тестувалася версія 3.04, версії, які використовують LSTM-моделі, і

версії, що використовує LSTM і словникові дані. Одна колонка представляє відсоток невірно розпізнаних символів, а друга - невірно розпізнаних слів. Значення в другій колонці більше, так як ймовірність помилки в слові вище і складається з ймовірності помилки в кожному з символів, з яких це слово складається.

Таблиця 2.1

Порівняння результатів розпізнавання тексту для різних версій Tesseract OCR

Тип OCR	Відсоток помилки при розпізнаванні окремого символа	Відсоток помилки при розпізнаванні слова
Tesseract	19.06 %	30.83 %
Tesseract + LSTM	4 %	15.34 %
Tesseract + LSTM + словникові дані	4.10 %	14.04 %

2.2 Нейронні мережі, рекурентні нейронні мережі та LSTM нейронні мережі

Штучна нейронна мережа (ШНМ) – це змодельоване представлення біологічної нейронної мережі. Нейронна мережа представляє з себе шари нейронів, які певним способом пов’язані між собою. Основними складовими нейрона є:

1. адаптивний суматор;
2. нелінійний перетворювач.

Нейрони, як структура, впливають одне на одного в залежності від типу нейронної мережі.

Поведінка нейрона в більшості архітектур ШНМ визначається сумарним впливом на нього попередніх шарів нейронів.

Завдяки розвитку найпростіших ШНМ, наука про нейронні мережі за проблематикою вирішення задач була поділена на такі види:

1. методи навчання мереж;
2. топології архітектури мереж.

Суть роботи нейронної мережі - це отримання на вхід одного вектора з даними, і, під час роботи - дати на виході інший вектор, який і буде результатом[11].

Рекурентні нейронні мережі (РНС), підвидом яких є LSTM (Long short-term memory), - тип нейронних мереж, що задіюють зворотний зв'язок. На відміну від нейромереж прямого поширення, в яких інформація передається послідовно від шару до шару, в РНС - кожен шар нейронів, крім вхідних даних, отримує деяку інформацію про попередній стан мережі. Це робить можливим аналіз послідовностей даних. Тобто нейрони рекурентної нейронної мережі отримують інформацію як від попереднього шару нейронів, так і від самих себе. Вони запам’ятовують свій стан. Найбільшою складністю РНМ є проблема «зникаючого градієнта», суть проблеми в особливості мережі, яка має бути доволі потужною(в плані обчислень), а кожна наступна ітерація отримує більш впорядковані дані, ніж попередня. З часом інформація може втрачатись(якісно).

Головною ж особливістю і перевагою LSTM-мереж є вміння працювати з довгостроковими залежностями: простіше кажучи, вони, на відміну від звичайних РНМ, вміють працювати з інформацією, обробленою багато циклів до того. Саме тому використання LSTM зараз є одним з головних трендів в розпізнаванні рукописного тексту.

Нейронні мережі з довгою короткочасною пам’яттю(LSTM) націлені якраз на вирішення проблем рекурентних нейронних мереж, оскільки використовують фільтри та явно задані “клітки” з пам’яттю. Якщо конкретніше, то у кожного нейрона є три фільтри, вхідний, вихідний і забуваючий. Ціль таких фільтрів – захист інформації.

Вхідний фільтр визначає скільки інформації буде зберігатись з попереднього шару. Вихідний фільтр визначає кількість інформації, яку будуть отримувати наступні шари. Забуваючий фільтр виконує також важливу функцію, оскільки при зміні умов(наприклад, перехід до іншої теми, або інша глава книги) потрібно частину інформації, яка не потрібна - видалити. Такі мережі потребують немало ресурсів[12].

2.3 Математичне забезпечення для роботи алгоритму Tesseract OCR

Алгоритм Tesseract OCR для успішного розпізнавання тексту з зображення виконує ряд задач, таких як визначення ліній тексту, знаходження базових ліній (ліній, на які будуть орієнтуватись деякі наступні кроки), визначення середнього розміру відстані між символами.

Визначення ліній тексту відбувається за допомогою інтерполяції квадратичними сплайнами.

Інтерполяція - це знаходження невідомих значень деякої функції за наявним дискретним набором її відомих значень. На зображенні функціями, які потрібно знайти, будуть виступати обмежуючі лінії тексту, а значеннями, за якими буде відбуватись інтерполяція, виступає текст[13].

Сплайн – це функція яка поєднує 2 або декілька точок відрізками, які в свою чергу відповідають певному поліному, таким чином задовольняючи умову безперервності[14].

Квадратичний сплайн – підвид сплайну, в якому відрізки що поєднують точки відповідають квадратному поліному[14].

На кожному відрізку від $[x_{i-1},x_i]$ квадратичний сплайн - це многочлен другого степеня функції $S_i(x)$.

$$S_i(x) = a_i + b_i x + c_i^2 x^2 \qquad (2.1)$$

Визначення базових ліній, ліній з найбільшою «популяцією» символів визначається за допомогою інтерполяції кубічним сплайном. Для визначення базових ліній можна використовувати і інтерполяцію квадратичним сплайном, але, не дивлячись на те, що квадратичний сплайн є стабільнішим за кубічний, для виявлення декількох базових ліній з текстом краще використовувати саме інтерполяцію кубічним сплайном.

Кубічний сплайн – підвид сплайну, в якому відрізки що поєднують точки відповідають кубічному поліному. [13]

На кожному відрізку від $[x_{i-1},x_i]$ кубічний сплайн - це многочлен третього степеня функції $S_i(x)$.

$$S_i(x) = a_i + b_i x + c_i^2 x^2 + d_i x^3 \qquad (2.2)$$

Результат визначення ліній тексту і базових ліній, показано на рисунку 2.2[8].

Рис. 2.2 – знаходження ліній тексту та базових ліній

Після визначення базових ліній починається процес ОРС.

2.4 Алгоритми опрацювання зображення

Алгоритмів для опрацювання зображення з метою покращення якості розпізнавання символів є велика кількість. Але потрібно виходити із специфіки виду зображень, для яких ці алгоритми будуть використані.

При використанні алгоритмів для опрацювання зображення, необхідно врахувати ряд особливостей та проблем: якість самого зображення, можливість нерівномірного освітлення різних областей і таке інше. Оскільки зображення кольорове, тому потрібно його бінаризувати, тобто зробити чорно білим. З таким зображенням алгоритми ОРС працюють найкраще, і це призводить до зменшення кількості шумів.

Для покращення якості розпізнавання попереднє опрацювання зображення буде складатись з таких етапів як:

- збільшення розмірів зображення;
- видалення фону;
- виправлення шумів;
- підвищення різкості;

Типовими алгоритмами, які можуть бути використані для покращення якості оптичного розпізнавання є локальна/глобальна бінаризація, лінійна/нелінійна фільтрація [15-21].

Бінаризація – це процес перетворення кольорового зображення в чорно-біле із збереженням потрібної інформації. Головним параметром, завдяки якому і відбувається таке перетворення, є поріг бінаризації. За результатами порівняння значення пікселів із пороговим значенням їм присвоюються відповідні значення нуля або одиниці. Є різні методи бінаризації. Умовно їх можна розділити на 2 групи: локальні та глобальні.

Глобальні методи бінаризації – порівнюють кожен піксель вхідного зображення з пороговим значенням.

Локальні ж методи бінаризації мають складніший алгоритм. По-перше, вони застосовуються до певної області, по-друге, - в цій області, за певною формулою, обраховується локальне порогове значення, яке буде використовуватись для бінаризації саме цієї області.

Ціль бінаризації – зменшення кількості інформації, яку потрібно обробити алгоритму розпізнавання символів.

В цілому операція бінаризації досить нескладна, оскільки суть її в проведенні порівняння двох значень та в присвоєнні значення пікселю. Складність процесу бінаризації у випадку опрацювання зображення БК представляє пошук локального порогу для порівняння, оскільки зображення є нерівномірним. Існує багато методів, які дозволяють знайти локальний поріг, найпопулярнішим є алгоритми Niblack`a.

Суть алгоритму в тому, щоб знайти уявний поріг для пікселя в межах області, де він знаходиться і виконати порівняння як при звичайній бінаризації. Розраховується локальний поріг за допомогою інтегрального зображення, яке допомагає швидко розрахувати суму пікселів(яскравість) в певній області.

Значення елемента інтегрального зображення розраховується за формулою:

$$S(x,\ y) = I(x,\ y) + S(x-1,\ y) + S(x,\ y-1) - S(x-1,\ y-1), \qquad (2.3)$$

де S – результат попередніх ітерацій для даної позиції пікселя, I – яскравість пікселя в початковому зображенні, x,y – координати.

Після того, як всі елементи інтегрального зображення обраховані, розраховується середній поріг і порівнюється з тим, що задано користувачем.

Наступним етапом опрацювання зображення є зменшення шумів. Для цієї операції можна використати різні типи фільтрів, роботу деяких з розглянемо детальніше.

Суть роботи лінійних(або згладжуючих) фільтрів полягає у зведенні значень пікселів до певного середнього значення. Найпопулярнішими з лінійних фільтрів є фільтри з обмеженою прямокутною апертурою, зважаючи на відносну швидкість роботи та простоту їх реалізації. Для роботи таких фільтрів потрібно підібрати розміри апертури.

Формула, яка уособлює «відгук» фільтра може бути записана у вигляді [16]:

$$S(x,y) = \sum_{n=1}^{N} \sum_{m=1}^{M} h(n,m)\, K(x-n+1, y-m+1), \qquad (2.4)$$

де h(n,m) – матриця коефіцієнтів імпульсної характеристики області в апертурі, N,M – розміри апертури фільтра.

Процес фільтрації здійснюється послідовним переміщенням вікна фільтрації(апертури) на один елемент зображення[16]. При кожному переміщенні вікна фільтра, перемножуються всі вагові множники з відповідними значеннями яскравості зображення та додавання отриманих результатів. За розміри апертури фільтра беруться непарні числа, для точного визначення центрального елемента, найчастіше розміри апертури в межах 3x3, 5x5 пікселів. Поєднання значень коефіцієнтів фільтру визначає властивості результуючого зображення. Якщо всі коефіцієнти у вікні будуть додатніми, то виконується згладжування перепадів яскравості – тобто результуюче зображення буде розфокусовуватись. Результати роботи такого фільтра при опрацюванні БК зображено на рисунку 2.3.

Рис. 2.3 – результат роботи лінійного фільтра(зверху вхідне зображення, знизу вихідне зображення)

Лінійний фільтр знаходить аномальні місця і просто їх «розмиває», таким чином страждають і корисні дані. Найпростішим з лінійних фільтрів є фільтр ковзного середнього, в якому всі коефіцієнти – рівні (див.формулу 2.5).

$$[h(n,m)] = \begin{bmatrix} 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \\ 1/9 & 1/9 & 1/9 \end{bmatrix} \quad (2.5)$$

Фільтр, завдяки якому можна з відстанню зменшувати вплив пікселів одне на одного - фільтр Гауса. Ядро такого фільтра можна записати формулою:

$$F_{gauss}(x,y) = \frac{1}{2\pi\sigma^2}\ exp(-\frac{x^2+y^2}{2\sigma^2}), \quad (2.6)$$

де y,x – координати пікселя, σ – шум, на вхідному зображенні.
Використовуючи фільтр Гауса можна розмивати зашумлені ділянки з мінімальним впливом на контури потрібного зображення.

Нелінійні фільтри використовують в основному для видалення імпульсних шумів. Видалення проблемних пікселів відбувається шляхом їх заміни на відповідні значення які закладені в фільтр. Нелінійні фільтри побудовані на основі рангової статистики і використовують маски.

Наступним типом фільтра, який можна використати для опрацювання БК є медіанний фільтр. Це нелінійний фільтр, завдяки якому можна отримати добрі результати із збереженням перепадів відтінків, контурів об’єктів, але, при цьому, вилучити із зображення аномальні області, спричинені імпульсними завадами. Нелінійність такого фільтра проявляється в методі пошуку значення для вихідного зображення. На відміну від лінійних фільтрів, в яких значення буде розраховане, і буде мати певне середнє значення, медіанний фільтр бере медіанне значення в заданій апертурі. Знаходження медіанного значення в апертурі схематично зображено на рисунку 2.4.

223	186	114
204	161	106
219	194	138

106 114 138 161 **186** 194 204 219 223

Рис. 2.4 – знаходження медіанного значення в апертурі розміром 3 на 3 пікселя

Результат роботи фільтру з тестовим зображенням можна побачити на рисунку 2.5.

Рис. 2.5 – результат роботи медіанного фільтра

Завдяки такому підходу алгоритм медіанної фільтрації не створює додаткових значень, що позитивно впливає на швидкодію, та сусідні пікселі не впливають на значення пікселя, яке розраховується. Тому такий фільтр зберігає контури зображень такими які вони, але може з’явитись ефект «заокруглених кутів».

2.5 Висновок до 2 розділу

Проаналізувавши алгоритм роботи Tesseract OCR, з'ясовано, що він найкраще працює із зображеннями, наближеними до ідеальних, а власного

алгоритму для суттєвого покращення зображення шляхом попереднього опрацювання у нього немає.

Запропоновано для оптичного розпізнавання “неідеальних” зображень, таких як зображення реквізитів на БК, здійснювати попереднє опрацювання, а саме:

- видалення імпульсних шумів медіанним фільтром;
- виділення контурів з допомогою збільшення різкості зображення;
- “бінаризацію” з локальними максимумами та мінімумами.

РОЗДІЛ 3
ПРОГРАМНА РЕАЛІЗАЦІЯ ОПРАЦЮВАННЯ ЗОБРАЖЕННЯ БК

Враховуючи необхідність покращення зображення БК з метою якісного розпізнавання реквізитів на них, запропоновано здійснювати їх попереднє опрацювання. Для цього необхідно розробити модуль попереднього опрацювання зображення, що дозволить підвищити точність роботи Tesseract`у саме для таких зображень.

Розглянувши відомі математичні методи для опрацювання зображення запропоновано алгоритм попереднього опрацювання БК, суть якого полягає у послідовному використанні таких методів: масштабування зображення, видалення шумів медіанним фільтром, підвищення різкості згорткою, перетворення всіх кольорів у спектр сірого кольору, застосування бінаризації з локальним максимумом та мінімумом.

Оскільки одним із завдань кваліфікаційної роботи є дослідження впливу попереднього опрацювання зображення на кінцевий результат оцифрування, тому доцільно розглянути як роботу запропонованих модулів, які відповідають безпосередньо за опрацювання зображення, так і оцінити їх вплив на результат розпізнавання реквізитів БК.

3.1 Алгоритм попереднього опрацювання реквізитів БК

Для отримання максимальної результативності від оптичного розпізнавання варто реалізувати повний алгоритм, за яким буде відбуватись опрацювання вхідного зображення БК. Блок-схему алгоритму попереднього опрацювання зображення показано на рис.3.1.

Рис. 3.1. Блок-схема повного алгоритму опрацювання зображення БК

Оскільки окремо всі методи були реалізовані, реалізація повномасштабного алгоритму опрацювання зображення перед оптичним розпізнавання, буде містити “виклики” методів, описаних в розділі 2 (підрозділ 2.4) в порядку, визначеному повним алгоритмом опрацювання зображення.

На базі запропонованого алгоритму опрацювання зображення БК розроблено його програмну реалізацію на мові C# у вигляді окремого класу, який в подальшому можна буде вбудовувати в аналогічні програмні реалізації або системи, призначені для розпізнавання реквізитів БК.

Оскільки C# на сьогоднішній день займає досить високі позиції на ринку мов для автоматизації бізнес-процесів, додатки на цій мові можна розробляти під будь-яку операційну систему, що і забезпечить легке вбудовування розробленого модуля для опрацювання зображення БК в існуючі рішення, або в рішення на стадії розробки.

Лістинг коду реалізованого алгоритму наведено на рис 3.2.

```
public void applyAllFilters(Bitmap image) {

    this.image = image;
    imageScaling(1.2);
    noiseCorrectionMedian();
    noiseCorrectionConvolution();
    greyScale();
    binarisationWithLocalMaxMin();
}
```

Рис. 3.2. Лістинг коду з повною реалізацією алгоритму опрацювання зображення

3.2 Реалізація методів попереднього опрацювання зображення БК

Для опрацювання зображення було створено програмне забезпечення у вигляді окремого класу ImageProcessing.

Клас «ImageProcessing» складається з всього одного поля «private Bitmap image», в цьому полі зберігається зображення, для якого будуть застосовуватись різноманітні алгоритми попередньої опрацювання.

Алгоритми попередньої опрацювання зображення БК представлені у вигляді методів:

- greyScale(перетворення в сірі відтінки);
- noiseCorrectionMedian(медіанний фільтр);
- imageScaling(масштабування зображення);
- noiseCorrectionConvolution(фільтрація згорткою).
- binarisationGlobal(глобальна бінаризація зображення)

−binarisationWithLocalMaxMin(бінаризація з локальними максимумами та мінімумами)

Окремо розглянемо кожен метод і його вплив на результат опрацювання зображення БК.

Масштабування зображення БК має покращити якість розпізнавання, оскільки в Tesseract OCR, як і у всіх алгоритмів ОРС, є залежність від розмірів символів, які необхідно розпізнавати. Таким чином мінімальне значення висоти символа має складати не менш як 25 пікселів, а товщина символів має складати не менш як 3 пікселя, через що і виникає необхідність в алгоритмі збільшення розмірів зображення.

Для дослідження якості розпізнавання Tesseract OCR з масштабуванням зображення, проведено тестування на 10 зображеннях БК .

Тести проводились при 2 фіксованих значеннях коефіцієнту масштабування, а саме: 1.2 та 1.4. Результати проведених тестувань для різних коефіцієнтів відображено в таблиці 3.1-3.2.

Таблиця 3.1

Результати тестувань алгоритму ОРС при збільшенні зображення у 1.2 рази

Tesseract OCR + масштаб x1.2	Точне значення	Кількість помилок при масштабі x1.2
5168 75 3763 5284	5168 7553 3763 5284	2
5168 7553 0621 0689	5168 7553 0621 0689	0
5168 7553 4097 1	5168 7553 4097 1833	3
-	5168 7555 1455 0934	Не розпізнано
4149 4978 2769 4540	4149 4978 2769 4540	0
-	4149 4978 7045 7399	Не розпізнано
5168 7573 1471 1062	5168 7573 1471 1062	0
4874 1200 2153	4874 1200 2153 9632	4
4790 9601 2188 2333	4790 9601 2166 2333	2
5211 5373	5211 5373 0518 6731	8

Кількість помилок опрацювання зображення БК при збільшенні на коефіцієнт масштабування 1.2 рівна:

$$\frac{51}{160} * 100\% = 31.875\% \quad (3.1)$$

Відповідно відсоток правильно розпізнаних символів на тестованих БК:

$$100\% - 31.875\% = 68.125\% \quad (3.2)$$

Таким чином можна зробити висновок, що кількість помилок при масштабуванні на 1.2 - зменшилась, але найбільшого ефекту таке масштабування дало для 2 зображень. Якщо оцінити розміри цих зображень, то можна побачити, що на одному із зображень БК було зроблено з досить великої відстані, а інше - має низьку роздільну здатність. Оскільки таке невелике збільшення масштабу зображення БК зменшило кількість хибних розпізнавань їх реквізитів, тому є сенс провести тестування розпізнавання при більшому коефіцієнті масштабування. Результат розпізнавання реквізитів БК з масштабом зображення 1.4 наведено в таблиці 3.2.

Таблиця 3.2

Результати тестувань алгоритму оптичного розпізнавання при збільшенні зображення у 1.4 рази

Tesseract OCR + масштаб x1.4	Точне значення	Кількість помилок масштаб x1.4
5168 75 3763 5284	5168 7553 3763 5284	2
5168 7553 0621 0689	5168 7553 0621 0689	0
5168 7553 4097	5168 7553 4097 1833	4
-	5168 7555 1455 0934	Не розпізнано
4149 4978 2769 4540	4149 4978 2769 4540	0
-	4149 4978 7045 7399	Не розпізнано
5168 7573 1471 1062	5168 7573 1471 1062	0
4874 1200 2153	4874 1200 2153 9632	4
4799 9601 2188 2333	4790 9601 2166 2333	3
5211 5373	5211 5373 0518 6731	8

Розрахована кількість помилок розпізнавання БК при масштабуванні 1.4 буде дорівнювати:

$$\frac{53}{160} * 100\% = 33.125\% \quad (3.3)$$

Відсоток правильно розпізнаних символів на БК:

$$100\% - 33.125\% = 66.875\% \quad (3.4)$$

Отже, порівнявши відсоток правильно розпізнаних символів БК при здійсненні масштабування з коефіцієнтами 1.2 та 1.4, можна підсумувати, що подальше збільшення розміру зображення може погіршити якість розпізнавання, тому оптимальним значенням коефіцієнту масштабування є 1.2.

Для зменшення шумів на вхідному зображенні було використано медіанний фільтр. Для тестування його здатності зменшувати імпульсні шуми на

зображеннях БК було нанесено чорні мітки, які і будуть вважатися імпульсними шумами.

Порівняння вихідних зображень після застосування медіанних фільтрів показано на рис. 3.3.

Рис. 3.3 – застосування медіанної фільтрації до зображення

Медіанний фільтр був застосований до всіх попередніх зображень БК. Змін в результатах не відбулось, оскільки зображення реквізитів на БК не мають імпульсних шумів. Але додатково введена можливість їх виправлення не буде завадою для розпізнавання.

Оскільки зображення після медіанного фільтру буде мати ефект розмиття. Тому доцільно застосувати алгоритм збільшення різкості зображення. Для вирішення такої задачі чудово підходить фільтрація згорткою.

Фільтрація згорткою дозволить збільшити різкість зображення, що дозволить виділити області з символами, що має позитивно вплинути на результат розпізнавання. Для збільшення різкості використано стандартну матрицю 3 на 3 з коефіцієнтами, наведеними на рис. 3.4.

-1	-1	-1
-1	9	-1
-1	-1	-1

Рис. 3.4 – матриця коефіцієнтів ядра згортки

Використавши для зображення БК фільтрацію згорткою разом із медіанною, вдалось зменшити імпульсні шуми без втрати чіткості символів на зображенні БК. Результат цього етапу опрацювання БК показано на рис. 3.5.

Рис. 3.5 – результат застосування медіанного фільтра разом із фільтрацією методом згортки

Додавання відтінків сірого дозволяє швидше проводити операцію бінаризації зображення, також важливо те, що за допомогою перетворення зображення у його представлення у відтінках сірого, буде яскраво відділятись фон від контенту. Причому неважливо чи буде фон чорним, а контент білим, чи навпаки, оскільки алгоритм Tesseract OCR однаково добре працює в обох випадках. Лістинг програми «greyScale», яка використовується для перетворення кольорового зображення показано на рис. 3.6

```
public void greyScale(){

    Bitmap bmap = (Bitmap)image.Clone();
    Color c;
    for (int i = 0; i < bmap.Width; i++)
    {
        for (int j = 0; j < bmap.Height; j++)
        {
            c = bmap.GetPixel(i, j);
            byte gray = (byte)(.299 * c.R + .587 * c.G + .114 * c.B);

            bmap.SetPixel(i, j, Color.FromArgb(gray, gray, gray));
        }
    }
    image = bmap;
}
```

Рис. 3.6. Лістинг програми «greyScale»

В лістингу 3.4. можна побачити сталі коефіцієнти 0.299, 0.587, 0.114 – це стандартні коефіцієнти, які слугують для того, аби вираховувати яскравість того чи іншого кольору в кольоровій моделі YUV (тобто в кольоровій моделі, яка складається з 3 кольорів).

Результат тестування методу «greyScale» показано на рис. 3.7.

Рис. 3.7 – тестування методу «greyScale»

Як видно з зображення БК на рисунку, відтінки сірого добре відділяють темну частину фонового зображення від символів, який потрібно буде розпізнавати. Важливо також те, що, хоча для людського зору зображення на БК почало містити менше інформації (оскільки насичена кольорова гама була вилучена), зображення все ще зберігає інформацію про те, з яких кольорів той чи інший відтінок сірого створений.

Останнім етапом покращення якості зображення БК для розпізнавання реквізитів, буде бінаризація. Оскільки вона «завершує» роботу над виділенням основного контенту, який і треба буде розпізнавати алгоритму ОРС. Для тестування алгоритмів бінаризації було створено 2 методи, один з яких бінаризує зображення глобально, не зважаючи на загальний стан яскравості в області, яку він змінює, а інший працює, визначаючи локальні максимуми і мінімуми, вираховуючи оптимальний поріг бінаризації.

Алгоритм роботи методу глобальної бінаризації зображення БК досить простий. Користувач сам обирає порогове значення, від якого і буде залежати

результат бінаризації. Найчастіше це порогове значення виставляється як 162. Тому, для тестування глобальної бінаризації при опрацюванні зображення на БК, було обрано це порогове значення. Лістинг коду глобальної бінаризації зображення реквізитів на БК показано на рис 3.8.

```
public void binarisationLinear()
{
    for (var x = 0; x < image.Width; x++)
    {
        for (var y = 0; y < image.Height; y++)
        {
            var pixel = image.GetPixel(x, y);
            if (pixel.R < 162 && pixel.G < 162 && pixel.B < 162)
                image.SetPixel(x, y, Color.Black);
            else if (pixel.R > 162 && pixel.G > 162 && pixel.B > 162)
                image.SetPixel(x, y, Color.White);
        }
    }
}
```

Рис. 3.8. Лістинг коду глобальної бінаризації

Порівняння вхідного зображення і вихідного можна побачити на рис 3.9.

Рис. 3.9. Результат роботи методу глобальної бінаризації

Через те, що зображення БК має свої локальні особливості (може містити зображення різних картинок чи фото), тому можна побачити дефект в центрі зображення, який не дозволить адекватно розпізнати реквізити БК. В такому випадку і людині важко чітко зрозуміти, яке число 3 чи 5.

Для позбавлення від таких недоліків було протестовано локальний метод бінаризації, а саме - бінаризацію Бредлі. Для того, аби застосувати бінаризацію алгоритмом Бредлі використано бібліотеку «AForge», в якій передбачено відповідний фільтр. Лістинг коду локальної бінаризації можна побачити на рис 3.10.

```
public void binarisationWithLocalMaxMin()
{
    BradleyLocalThresholding filter = new BradleyLocalThresholding();
    image = filter.Apply(Grayscale.CommonAlgorithms.BT709.Apply(new Bitmap(image)));
}
```

Рис. 3.10. Лістинг коду локальної бінаризації

Алгоритм Бредлі покращує результат опрацювання зображення БК, оскільки дозволяє враховувати локальні особливості зображення. Для перевірки - протестуємо роботу реалізованого методу та порівняємо результати локальної і глобальної бінаризації. Результат роботи локальної бінаризації Бредлі зображень БК показано на рис 3.11.

Рис.3.11. Результат роботи методу локальної бінаризації

Результати порівняння обох методів бінаризації при опрацюванні зображення однієї і тієї ж БК показано на рис 3.12.

а) б)

Рис.3.12. порівняння методів бінаризації одного зображення БК (а) лінійна бінаризація, б) бінаризація з локальними максимумами і мінімумам)

Таким чином, проаналізувавши отримані результати опрацювання зображення БК, можна бути впевненим, що метод, який працює за алгоритмом визначення локальних максимумів і мінімумів працює краще, тому, відповідно, і помилок в розпізнаванні реквізитів БК буде менше.

Для проведення бінаризації також було використано метод перетворення зображення в сірі відтінки, оскільки алгоритми бінаризації працюють саме з такими зображеннями. А також ще була використана інвертація зображення.

Розпізнавання реквізитів БК з проведеною бінаризацією було протестоване на 10 зображеннях. Результати тестування наведено в таблиці 3.3.

Таблиця 3.3

Результати тестувань алгоритму оптичного розпізнавання з попередньою бінаризацією

Tesseract OCR + бінаризація	Точне значення	Кількість помилок
5168 75 3763 5284	5168 7553 3763 5284	2
5168 7553 0621 0689	5168 7553 0621 0689	0
6188 7555 4897 1833	5168 7553 4097 1833	4
5168 7555 1455	5168 7555 1455 0934	4
4149 4978 2769 4540	4149 4978 2769 4540	0
4148 4878 7043 7388	4149 4978 7045 7399	5
5168 7573 1471 1062	5168 7573 1471 1062	0
4874 1200 2153	4874 1200 2153 9632	4
4790 9601 2188 2888	4790 9601 2166 2333	5
5211 5373	5211 5373 0518 6731	8

У відсотках кількість помилок при розпізнаванні при попередній бінаризації зображення БК буде дорівнювати:

$$\frac{32}{160} * 100\% = 20\% \quad (3.5)$$

Відсоток правильно розпізнаних символів на БК:

$$100\% - 20\% = 80\% \quad (3.6)$$

Отже, проаналізувавши результати тестування, можна побачити, що повністю нерозпізнаних номерів на досліджуваних БК не залишилось, але якість все ще досить погана.

Цей алгоритм є “останнім штрихом” в розробці програмного продукту. Загальний вигляд розробленого програмного забезпечення для розпізнавання реквізитів БК показано на рис 3.13.

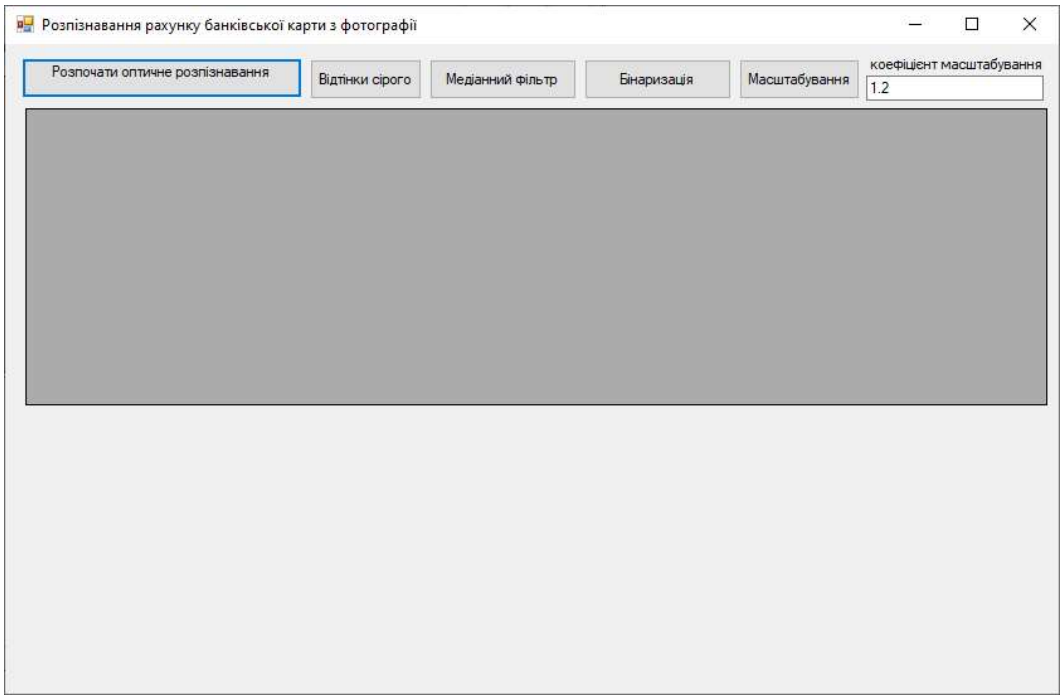

Рис. 3.13. Інтерфейс програми для ОРС з попереднім опрацюванням зображення

На рис.3.7 можна побачити ряд елементів управління програмою, а саме кнопки, які дозволяють застосовувати кожний метод для опрацювання зображення БК окремо, та кнопка «Розпочати оптичне розпізнавання», яка автоматично застосовує повномасштабний алгоритм опрацювання зображення та його розпізнавання. Вибір користувачем потрібного зображення БК в діалоговому вікні здійснюється натисканням на відповідну кнопку.

Діалогове вікно програми попереднього опрацювання БК показано на рис.3.14.

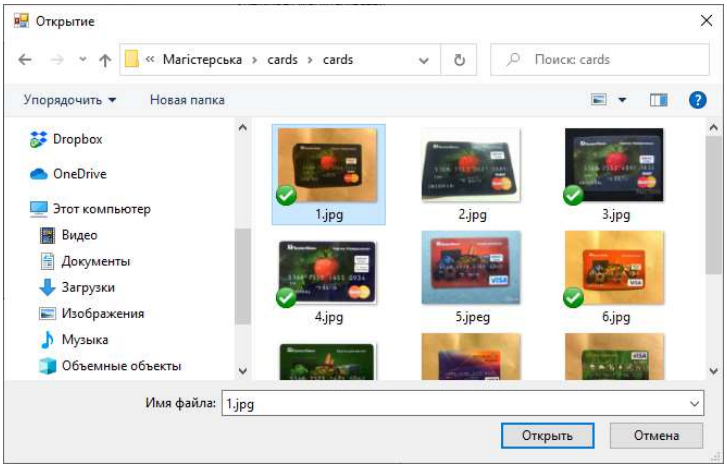

Рис 3.14 діалогове вікно вибору зображення для попереднього опрацювання зображення і розпізнавання реквізитів БК

Після етапу вибору зображення БК, програма застосовує алгоритм опрацювання зображення. Далі, після підвищення якості зображення, розпочинається процес ОРС. Як тільки цей процес розпізнавання закінчується, у вікні програми появляться результати розпізнавання символів БК. Вхідне та вихідне зображення БК є результатом застосування повномасштабного алгоритму підвищення якості. Вікно програми після виконання ОРС відображено на рис. 3.15.

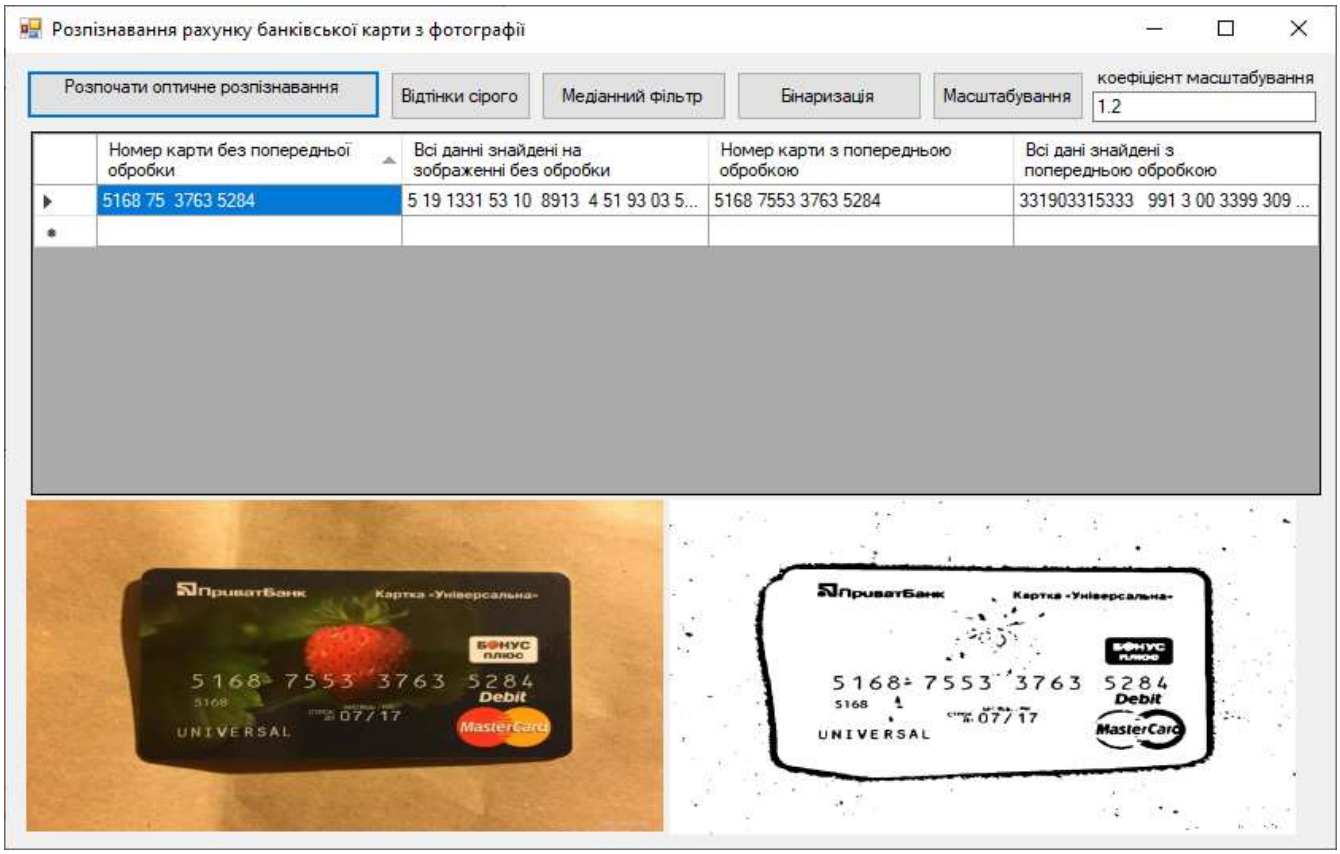

Рис. 3.15 Результат попереднього опрацювання зображення БК і розпізнавання реквізитів

У вікні розробленої програми (рис.3.15) в нижньому лівому кутку розміщується вхідне зображення, в правому нижньому - отриманий результат. Отримана інформація про реквізити БК виводиться у вигляді таблиці. Зокрема, в 1 колонці виводиться результати, які отримані без попереднього опрацювання зображення, а в 2 колонці - отримані результати роботи програми без попереднього опрацювання. Реквізити, виокремлені з зображення, яке пройшло процедуру покращення якості виводяться в 3 колонці, і, відповідно, всі дані, які вдалось знайти на зображенні з попереднім опрацюванням БК - представляються в 4 колонці. Навіть після опрацювання першого зображення БК, можна помітити покращення результату розпізнавання реквізитів БК.

3.3 Тестування

Для визначення ефективності розпізнавання реквізитів БК проведемо тестування роботи алгоритму OPC з використанням запропонованого модуля попереднього опрацювання зображення в порівнянні із розпізнаванням без такого модуля.

Tesseract OCR може виконувати розпізнавання тексту із зображення і без його попереднього опрацювання, але якість такого розпізнавання буде досить низькою, оскільки залежність від якості вхідного зображення дуже висока.
Оскільки Tesseract OCR має найкращу якість розпізнавання в ідеальних умовах, а саме якщо текст є надрукованим, з правильними горизонтальними лініями, та без завад. Якість розпізнавання такого тексту буде досягати 99% правильно розпізнаних символів. В ідеалі потрібно наближати зображення саме до такої якості., а також, відповідно, і результат, оскільки фотографії БК мають багато лишнього інформаційного шуму у вигляді зображень.

Для дослідження якості такого розпізнавання реквізитів БК розроблена програма, яка виконує оптичне розпізнавання вхідного зображення БК без проведення попереднього опрацювання. Результати проведення такого тестування можна зведені в таблицю 3.4.

Таблиця 3.4

Результат тестування програми без попереднього опрацювання зображення

Tesseract OCR	Точне значення	Кількість помилок
5168 75 3763 5284	5168 7553 3763 5284	2
5168 7553 0621 0689	5168 7553 0621 0689	0
-	5168 7553 4097 1833	Не розпізнано
-	5168 7555 1455 0934	Не розпізнано
4149 4978 2769 4540	4149 4978 2769 4540	0
-	4149 4978 7045 7399	Не розпізнано
5168 7573 1471 1062	5168 7573 1471 1062	0
4874 1200 2153	4874 1200 2153 9632	4
-	4790 9601 2166 2333	Не розпізнано
5211 5373	5211 5373 0518 6731	8

В колонці «Tesseract OCR» збережені результати ОРС без попереднього опрацювання зображення, в колонці «точні значення» розташовані вручну розпізнані(правильні) номери БК, «Кількість помилок» відображає кількість помилок які допустив алгоритм розпізнавання. З отриманих результатів можна вирахувати відсоток помилок поділивши кількість помилок на загальну кількість символів які можливо розпізнати. Оскільки було протестовано 10 БК то кількість символів які можна розпізнати – 10*16 = 160, серед цих 160 символів було допущено в сумі 78 помилок. Таким чином відсоток помилок буде дорівнювати:

$$\frac{78}{160} * 100\% = 48.75\% \qquad (3.7)$$

Відсоток правильно розпізнаних символів:

$$100\% - 48.75\% = 51.25\% \quad (3.8)$$

З огляду на отримані результати тестування ОРС без попереднього опрацювання можна зробити висновок про низьку точність розпізнавання.

Для того щоб впевнитись в покращенні результату ОРС з модулем попереднього опрацювання зображення потрібно застосувати створений алгоритм зображень та виконати розпізнавання реквізитів БК, розрахувати відсоток помилок та правильних розпізнавань, і порівняти результати з результатами розпізнавання без покращення якості зображення. Після проведення 10 тестових розпізнавань зображень вікно програми буде містити інформацію про розпізнавання даних кожної картки, вікно програми відображено на рис 3.16.

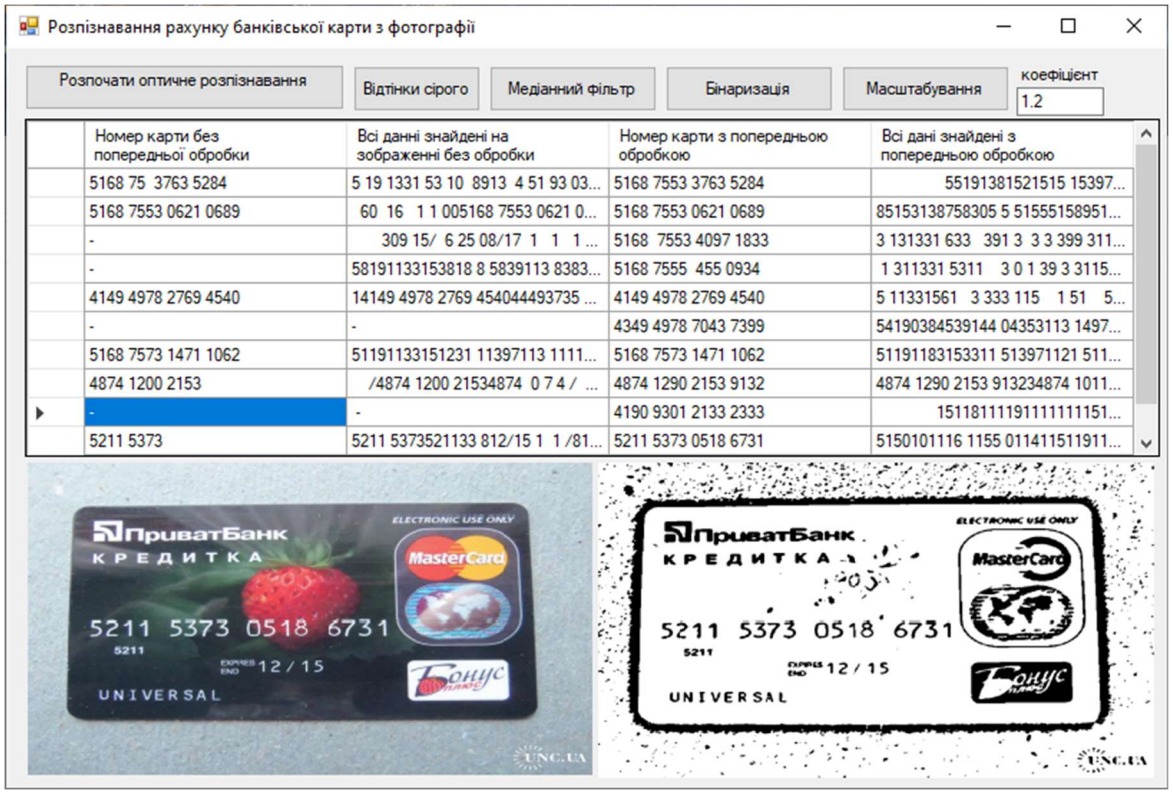

Рис. 3.16. Вікно програми після проведення 10 тестових розпізнавань

Таким чином, маючи результати розпізнавання без попереднього опрацювання зображення, та маючи дані розпізнавання після покращення якості зображення потрібно розрахувати результативність даного алгоритму. Для того щоб було простіше вирахувати відсоткові відношення потрібно систематизувати дані в нову більш інформативну таблицю, і виділити кількість помилок яка була допущена в тому чи іншому способі ОРС. Таблицю з такими даними можна побачити на рис. 3.5.

Таблиця 3.5

Зведена таблиця з результатами розпізнавань з попереднім опрацюванням зображення та без нього

Точне значення реквізитів банківських карт	Tesseract OCR		Tesseract OCR з попереднім опрацюванням зображення	
	Результат опрацювання	Кількість помилок	Результат опрацювання	Кількість помилок
5168 7553 3763 5284	5168 75 3763 5284	2	5168 7553 3763 5284	0
5168 7553 0621 0689	5168 7553 0621 0689	0	5168 7553 0621 0689	0
5168 7553 4097 1833	-	Не розпізнано	5168 7553 4097 1833	0
5168 7555 1455 0934	-	Не розпізнано	5168 7555 455 0934	1
4149 4978 2769 4540	4149 4978 2769 4540	0	4149 4978 2769 4540	0
4149 4978 7045 7399	-	Не розпізнано	4349 4978 7043 7399	2
5168 7573 1471 1062	5168 7573 1471 1062	0	5168 7573 1471 1062	0
4874 1200 2153 9632	4874 1200 2153	4	4874 1290 2153 9132	2
4790 9601 2166 2333	-	Не розпізнано	4190 9301 2133 2333	3
5211 5373 0518 6731	5211 5373	8	5211 5373 0518 6731	0

Відсоток помилок алгоритму без покращення зображення дорівнює 48.75, правильних розпізнавань відповідно дорівнює 51.25% (див. підрозділ 3.1).

Для розрахунку відсотка помилки при ОРС з алгоритмом попереднього опрацювання потрібно розрахувати кількість помилок. Кількість помилок з покращенням якості дорівнює 8. Знаючи максимальну кількість правильно

розпізнаних символів та кількість допущених помилок можна розрахувати відсоток помилок. Відсоток помилок буде дорівнювати:

$$\frac{8}{160} * 100\% = 5\% \quad (3.9)$$

Відсоток правильно розпізнаних символів:

$$100\% - 5\% = 95\% \quad (3.10)$$

Маючи відсоток правильно розпізнаних символів без попереднього опрацювання зображення БК та відсоток правильно розпізнаних символів з попереднім опрацюванням зображення БК можна розрахувати відсоток символів, які були розпізнані завдяки роботі алгоритму покращення якості зображення. Покращення якості розпізнавання у відсотках буде дорівнювати:

$$95\% - 51.25\% = 43.75\% \quad (3.11)$$

Отже, враховуючи отримані результати тестування, можна зробити висновок, що використання попереднього опрацювання зображення БК дозволяє покращити розпізнавання реквізитів на 43.75% - до 95%. Тобто підтверджується доцільність використання запропонованого алгоритму.

3.4 Висновок до розділу

На основі запропонованих рішень для опрацювання зображення БК з метою розпізнавання їх реквізитів створено програмне забезпечення у вигляді окремого класу ImageProcessing.

В результаті тестування встановлено, що використання запропонованого методу дозволяє збільшити точність розпізнавання реквізитів БК до 95%.

Розроблений модуль для опрацювання зображення може бути використаний в подальших розробках подібних програм, а також для вбудовування в існуючі програми для покращення результату ОРС.

РОЗДІЛ 4
ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Оскільки у даному випадку процес праці являє собою дослідницьку діяльність, яка проводиться за персональним комп’ютером, то аналіз небезпечних або шкідливих факторів в умовах праці буде розглядатись саме з цієї позиції.

4.1 Охорона праці

При дослідженні технологій ОРС враховані всі законодавчі та нормативні документи з охорони праці, техніки безпеки та протипожежної безпеки. Оскільки запропоновано в роботі використання програмного засобу для покращення попереднього опрацювання зображень БК тому необхідно враховувати усі вимоги, які стосуються безпечного використання комп'ютерної техніки[25-27].

Робочі місця обладнані персональними комп’ютерами, повинні відповідати вимогам «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», затверджених Наказом Міністерства соціальної політики від 14.02.2018 року №207 та «Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно – обчислювальних машин», затверджених постановою Головного державного санітарного лікаря України від 10.12.1998 року №7 (ДСанПіН 3.3.2-007-98).[29]

В основному для розпізнавання реквізитів БК в цілому достатньо використання персональний комп’ютер в робочій кімнаті тільки один, деякі пункти нормативно правових актів можна не враховувати, такі як відстані між бічними і тильними сторонами персональних комп’ютерів.

Щодня перед початком роботи користувач повинен:

−оглянути своє робоче місце: про виявлення ознак пошкодження обладнання інформувати свого безпосереднього керівника;

−відрегулювати освітленість на робочому місці, переконатися в відсутності відблисків на екрані комп’ютера, відсутності зустрічного світла;

−перевірити правильність підключення обладнання ЕОМ до електромережі;

−очистити екран комп’ютера від пилу та інших забруднень;

−перевірити правильність організації робочого місця й за необхідності провести відповідні коригування.

Вимоги щодо розміщення і планування приміщень для роботи з комп’ютером:

−Робочі місця, обладнані персональними комп’ютерами, заборонено облаштовувати у підвальних або цокольних приміщеннях будівель;

−забороняється використання полімерних матеріалів, що виділяють шкідливі хімічні речовини;

−достатнім для здійснення роботи рівнем освітлення (природного та штучного – у темну пору доби) та звукоізоляції;

−щодня має здійснюватися вологе прибирання з метою недопущення запиленості підлоги та меблів;

−лінії електромережі мають бути захищені від виникнення короткого замикання, а також від перепадів мережевої напруги, що може спричинити збої в роботі електронно–обчислювальної техніки;

При виконанні робіт оператор персонального комп’ютера повинен:

−витримувати відстань від очей до екрана комп’ютером в межах 60 - 70см;

−дотримуватися внутрішньозмінного режиму праці та відпочинку, регламентованих перерв у роботі;

−перерва для розробників програм - 15 хвилин через кожну годину роботи;

−перерва для операторів комп'ютерного набору - тривалістю 10 хвилин, після кожної години роботи.

При роботі на персональному комп’ютері не допускається:

−виконання ремонту та налагодження комп’ютерої техніки безпосередньо на робочому місці оператора;

−зберігання біля комп’ютера паперу, дисків, інших носіїв інформації, запасних блоків, деталей тощо, якщо вони не використовуються для поточної роботи;

−відключення захисних пристроїв, самочинні зміни в конструкції комп’ютера;

−використання комп’ютерів, на екранах яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані тощо;

−доторкання до задньої панелі системного блоку при включеному живленні;

−вимикання живлення під час виконання активного завдання;

−попадання вологи на поверхню системного блоку, монітора, клавіатури, дисководів, принтерів та інших пристроїв.

Після закінчення роботи з використанням персонального комп’ютеру необхідно дотримуватися такої послідовності вимикання обладнання:

−закрити всі активні завдання;

−переконатися у відсутності дискет та дисків у дисководах;

−використавши опцію "Завершення роботи" у меню "Пуск", вимкнути живлення системного блоку;

−вимкнути блок аварійного живлення (за наявності).

Таким чином, використання запропонованого модуля для розпізнавання реквізитів БК є безпечним з точки зору охорони праці та техніки безпеки.

4.2 Фактори ризику і можливі порушення здоров’я користувачів комп’ютерної мережі

В сучасному житті величезного значення набуває комп’ютеризація усіх сфер життя. Без неї не може функціонувати жодна галузь людської діяльності. Усе наше життя тісно пов’язане з комп’ютером та інтернетом. Комп’ютер прискорює процес навчання та підвищує його якість, набагато розширює творчі

можливості людини, допомагає опанувати складні виробничі процеси та взагалі полегшує працю й побут.[30]

Під час роботи за комп'ютером найбільшому ризику підлягають:

- органи зору;
- м'язово-скелетна система;
- нервово-психічна діяльність з можливим формуванням стресу і депресій;
- шкіра;
- імунна система.

Близько 70% користувачів комп'ютерів скаржаться на порушення функції органу зору. Зір є важливою складовою багатогранної діяльності людини. А робота за комп'ютером, особливо безперервна, може привести до розвинення короткозорості, зниження гостроти зору, різні кон'юнктивіти та безліч інших очних хвороб. Сльозотеча, головний біль, мерехтіння в очах, головокружіння також супроводжують користувачів комп'ютерів. У найтяжчих випадках можуть виникнути глаукома, катаракта та дистрофія сітківки, які ведуть до повної сліпоти. Причина цих проблем із зором полягає у миготінні монітору, а це подразнююче діє на зоровий аналізатор. А ще недостатня чіткість символів, їх спотворення на екрані, низька зручність читання безпосередньо впливають на продуктивність праці та на очі.

Скелетно-м'язова система також страждає через сидяче положення користувачів. У людей, які багато працюють за комп'ютером, можуть виникнути больові відчуття в м'язах та суглобах, потилиці, попереку, пальцях рук. Виникненню цих захворювань сприяє неправильне положення тіла щодо клавіатури, відхилення ліктів від тулуба, нераціональне розміщення передпліччя та кисті рук. Робота із клавіатурою є інтенсивною та динамічною роботою кисті, яка супроводжується одночасним напруженням м'язів передпліччя й плеча. Це приводить до швидкої втоми, до розвитку нейроміозитів, болей, оніміння, тремору та повільної рухливості пальців. Навіть до нападів судом, болі в руках і плечах уночі.

Велике зорове та нервово-емоційне напруження викликають порушення функціонального стану нервової системи. Це проявляється погіршенням психологічного стану й працездатності. У медичній літературі широко описані психічні розлади, які діагностуються у користувачів. Це агресивність, нервозність, фрустрація, тривога, депресія, пригніченість, порушення сну, стресові ситуації. Стрес може бути корисним: він тренує організм і підвищує його можливості, наприклад, захисту. Проте стрес може досягнути такого рівня напруження, що виснажує захисні сили організму. Це може спровокувати різноманітні захворювання і навіть смерть. До таких захворювань належать гіпертонічна хвороба, інфаркти міокарда та інсульти, виразкові хвороби травного тракту, навіть цукровий діабет. Людина піддається дії сильних стрес-факторів, до яких належать гігієнічні умови праці в робочому приміщенні, самого трудового процесу та мікроклімату в колективі.

Стосовно стану шкіри найчастіше виявляються зміни на шкірі обличчя. За характером вони різноманітні: еритеми, папульозні висипи, рожеві вугрі, дерматити – себорейний, атиповий, різні телеангіектазії. У жінок-користувачів найчастіше стає суха шкіра та відбувається передчасне старіння шкіри. Ці зміни пов’язані із сухим повітрям у приміщенні, частотою виникнення електростатичних зарядів. Також різноманітні шкірні реакції у чутливих людей залежать від природи аерозольних частинок, які відкладаються на тілі операторів.

Існує негативний вплив комп’ютерів на імунологічну активність організму. Порушуються та пригнічуються процеси імуногенезу. Виникає аутоімунітет – це реакція організму, спрямована проти власних тканин і органів.

Комп’ютер приносить нам багато задоволення, розширює межі пізнання світу, однак користуватися ним слід розумно, дотримуючись нормативно правових актів з організації робочого місця.

Отже під час дослідження шкідливих чинників на організм людини при роботі за персональним комп’ютером було виділено декілька найвразливіших органів, а саме: органи зору, м’язово-скелетна система, нервово-психічна

діяльність, шкіра, імунна система. Також були досліджені причини виникнення такого шкідливого впливу.

ВИСНОВКИ

У кваліфікаційній роботі вирішене актуальне завдання покращення якості оптичного розпізнавання реквізитів БК та отримано такі основні результати:

1) В процесі аналізу існуючих технологій розпізнавання реквізитів БК обґрунтовано актуальність завдання підвищення якості оптичного розпізнавання їх реквізитів.
2) Встановлено, що для розпізнавання реквізитів БК, враховуючи специфіку нанесення на БК символів, існуючі методи є недостатньо точними, та обгрунтовано необхідність детальнішого дослідження методів покращення оптичного розпізнавання реквізитів БК.
3) Обґрунтовано доцільність використання для ОРС на БК Tesseract OCR та проаналізовано алгоритм його роботи, в результаті чого з'ясовано, що він найкраще працює із зображеннями, наближеними до ідеальних, а власного алгоритму для суттєвого покращення зображення шляхом попереднього опрацювання у нього немає.
4) Запропоновано для оптичного розпізнавання “неідеальних” зображень, таких як зображення реквізитів на БК, здійснювати попереднє опрацювання, зокрема, видалення імпульсних шумів медіанним фільтром, виділення контурів з допомогою збільшення різкості зображення, “бінаризацію” з локальними максимумами та мінімумами.
5) На основі запропонованих рішень для опрацювання зображення БК з метою розпізнавання їх реквізитів створено програмне забезпечення у вигляді окремого класу ImageProcessing.
6) Проведено тестування розробленого ПЗ, в результаті чого визначено, що використання запропонованого методу дозволяє збільшити точність розпізнавання реквізитів БК до 95%.
7) Підтверджено працездатність та доцільність використання запропонованих рішень для підвищення якості оптичного розпізнавання реквізитів БК

8) Запропоновано використання розробленого модуля в подальших розробках аналогічних програм і для вбудовування в існуючі програми для покращення результату ОРС.

ПЕРЕЛІК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ

1. В.О. Бурмістр, Г.М. Осухівська. Технології розпізнавання реквізитів банківських карт / Матеріали IX Міжнародної науково-технічної конференції молодих учених та студентів. Актуальні задачі сучасних технологій – Тернопіль, 25-26 листопада 2020. – Т.II. – С.8

2. В.О. Бурмістр, Г.М. Осухівська. Покращення якості зображення реквізитів банківських карт для їх оптичного розпізнавання / Матеріали VIII науково-технічної конференції ТНТУ ім. І. Пулюя, 9-10 грудня 2020 року – С.96.

3. Cai S., Wen J., Xu H., Chen S., Ming Z. (2017) Bank Card and ID Card Number Recognition in Android Financial APP. In: Qiu M. (eds) Smart Computing and Communication. SmartCom 2016. Lecture Notes in Computer Science, vol 10135. Springer, Cham. - Режим доступу: https://doi.org/10.1007/978-3-319-52015-5_21.

4. Tesla с автопилотом попадает в аварии в 6,5 раз реже обычного автомобиля [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://hightech.plus/2019/07/18/tesla-s-avtopilotom-popadaet-v-avarii-v-65-raz-rezhe-obichnogo-avtomobilya.

5. ABBYY FineReader Engine [Електронний ресурс] – Режим доступу до ресурсу: https://www.abbyy.com/ru/ocr-sdk.

6. ABBYY FineReader Engine - Полный набор технологий распознавания [Електронний ресурс] – Режим доступу до ресурсу: https://www.abbyy.com/ru/ocr-sdk/key-features/ocr/.

7. GitHub Tesseract Documentation Tesseract Open Source Engine [Електронний ресурс] - Режим доступу: https://github.com/tesseract-ocr/tesseract

8. Asprise Documentation [Електронний ресурс] - Режим доступу: https://asprise.com/royalty-free-library/c%23-sharp.net-ocr-api-overview.html

9. Ray Smith. An Overview of the Tesseract OCR Engine [Електронний ресурс] /Google Inc.. – 2007. – Режим доступу до ресурсу: https://static.googleusercontent.com/media/research.google.com/ru//pubs/archive/33418.pdf.

10. Improving the Efficiency of Tesseract OCR through Super Resolution [Електронний ресурс]. – 2017. – Режим доступу до ресурсу: https://edu.authorcafe.com/academies/7609/improving-the-efficiency-of-tesseract-ocr-throughsuperresolution.

11. Aditi Mittal. Understanding RNN and LSTM [Електронний ресурс] / Mittal Aditi. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/understanding-rnn-and-lstm-f7cdf6dfc14e.

12. Разнообразие нейронных сетей. Часть вторая. Продвинутые конфигурации [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://tproger.ru/translations/neural-network-zoo-2/.

13. Селиванова И. А. Интерполяционные сплайны / Методические материалы – Екатеринбург, УГТУ-УПИ, 2009. – 16 с.

14. Костенко Л. С. Методы и алгоритмы сглаживания фона изображений в системах распознавания образов [Електронний ресурс] / Открытые информационные и компьютерные интегрированные технологии № 64. – 2014. – Режим доступу до ресурсу: http://www.irbis-nbuv.gov.ua/cgi-bin/irbis_nbuv/cgiirbis_64.exe?C21COM=2&I21DBN=UJRN&P21DBN=UJRN&IMAGE_FILE_DOWNLOAD=1&Image_file_name=PDF/vikt_2014_64_21.pdf.

15. Методы фильтрации цифровых изображений [Електронний ресурс]. – 2015. – Режим доступу до ресурсу: https://studfile.net/preview/3497261/.

16. Функция Свертка (Convolution) [Електронний ресурс] – Режим доступу до ресурсу: https://desktop.arcgis.com/ru/arcmap/10.3/manage-data/raster-and-images/convolution-function.htm.

17. Pre-Processing in OCR [Електронний ресурс]. – 2019. – Режим доступу до ресурсу: https://towardsdatascience.com/pre-processing-in-ocr-fc231c6035a7.

18. Boiangiu C. Improving the Accuracy of Tesseract 4.0 OCR Engine Using Convolution-Based Preprocessing [Електронний ресурс] / C. Boiangiu, E. Cușnir, D. Sporici. – 2020. – Режим доступу до ресурсу: https://www.mdpi.com/2073-8994/12/5/715/htm.

19. Helland T. Seven grayscale conversion algorithms [Електронний ресурс] / Tanner Helland. – 2011. – Режим доступу до ресурсу: http://www.tannerhelland.com/3643/grayscale-image-algorithm-vb6/.

20. Improve OCR Accuracy With Advanced Image Preprocessing [Електронний ресурс] – Режим доступу до ресурсу: https://docparser.com/blog/improve-ocr-accuracy/.

21. Bank Card Number Intelligent Identification Based on Digital Image Processing Technology [Електронний ресурс] // Journal of Image and Signal Processing. – 2016. – Режим доступу до ресурсу: https://www.researchgate.net/publication/305691055_Bank_Card_Number_Intelligent_Identification_Based_on_Digital_Image_Processing_Technology.

22. Median Filters [Електронний ресурс] // 1995 – Режим доступу до ресурсу: https://www.sciencedirect.com/topics/engineering/median-filters.

23. Naresh Kumar Garg. Binarization Techniques used for Grey Scale Images [Електронний ресурс] / Naresh Kumar Garg // 2013 – Режим доступу до ресурсу: https://pdfs.semanticscholar.org/c58f/8ab48711f0d5596be78c7f403027030ecc5b.pdf?_ga=2.221549976.1290872424.1608215307-326916643.1608215307

24. Resize Image In C# [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://www.c-sharpcorner.com/article/resize-image-in-c-sharp/.

25. Методичні вказівки до виконання підрозділу "Охорона праці" в кваліфікаційних роботах магістрів спеціальності 123 «Комп’ютерна інженерія» /Укл.: Осухівська Г.М. – Тернопіль: ТНТУ імені Івана Пулюя, 2020. – 22 с.

26. Методичні вказівки до виконання лабораторних робіт з дисципліни: «Цифрове опрацювання сигналів» для студентів денної форми навчання за спеціальністю 123 «Комп’ютерна інженерія» / Осухівська Г.М., Луцик Н.С. – Тернопіль: ТНТУ імені Івана Пулюя, 2020. – 72 с.

27. Вимоги охорони праці при виконанні робіт на персональному комп’ютері [Електронний ресурс]. – 2016. – Режим доступу до ресурсу: https://selidovo-rada.gov.ua/novini/vimogi-okhoroni-pratsi-pri-vikonanni-robit-na-personalnomu-komp-yuteri.

28. Методичний посібник для виконання практичних робіт з дисципліни «Методологія та організація наукових досліджень» для студентів спеціальності 123 «Комп'ютерна інженерія» / Уклад. Осухівська Г.М. – Тернопіль: ТНТУ імені Івана Пулюя, 2018. – 36 с.

29. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». Київ. 2018. URL: http://sop.zp.ua/norm_npaop_0_00-7_15-18_01_ua.php

30. Смирнов В.А., Дикань С.А. Безпека життєдіяльності: навч. посібник. К. : Кафедра. 2012. 304 с.

Додаток А. Тези конференцій

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний технічний університет імені Івана Пулюя (Україна)
Національна академія наук України
Університет імені П'єра і Марії Кюрі (Франція)
Маріборский університет (Словения)
Технічний університет у Кошице (Словаччина)
Вільнюський технічний університет ім. Гедимінаса (Литва)
Шяуляйська державна колегія (Литва)
Жешувський політехнічний університет ім. Лукасевича (Польща)
Білоруський національний технічний університет (Республіка Білорусь)
Міжнародний університет цивільної авіації (Марокко)
Національний університет біоресурсів і природокористування України (Україна)
Наукове товариство ім. Шевченка
ГО «Асоціація випускників Тернопільського національного технічного університету імені Івана Пулюя»

АКТУАЛЬНІ ЗАДАЧІ СУЧАСНИХ ТЕХНОЛОГІЙ

Збірник

тез доповідей

Том ІІ

ІХ Міжнародної науково-технічної конференції молодих учених та студентів

25-26 листопада 2020 року

УКРАЇНА
ТЕРНОПІЛЬ – 2020

УДК 004.93
В.О. Бурмістр, Г.М. Осухівська канд. техн. наук, доц.
Тернопільський національний технічний університет імені Івана Пулюя, Україна

ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ РЕКВІЗИТІВ БАНКІВСЬКИХ КАРТ

V.O. Burmistr, H.M. Osukhivska Ph.D, Assoc. Prof.

TECHNOLOGIES FOR RECOGNIZING BANK CARD DETAILS

У сучасному світі досить часто здійснюється проведення різного роду фінансових онлайн операцій, тому зчитування даних з банківських карт актуально в будь-якій сфері, де користувачеві потрібно ввести свої платіжні реквізити (при замовленні товарів в інтернет-магазині, придбанні квитків, інтернет-банкінгу, тощо). Для здійснення таких онлайн операцій необхідно вводити номер банківської картки та іншу інформацію. З метою уникнення помилок при вводі, зменшення затрат часу для внесення даних, а також для покращення взаємодії з користувачем виникає необхідність в розробці комп'ютерних систем оптичного розпізнавання реквізитів банківських карт.

Впровадження технологій оптичного розпізнавання реквізитів банківських карт (які по суті є символами) має ряд переваг над ручним розпізнаванням. Основними з яких є точність, швидкість та вартість впровадження такої технології. Хоча зовсім обійтися без залучення людини не вдасться, але фактор її участі буде мінімальним.

Оптичне розпізнавання символів - це процес, спрямований на оцифровування текстової інформації, яка автоматично ідентифікується із зображенням, а потім зберігається як дані. Для розпізнавання символів використовують різні методи, серед яких варто відмітити такі як: матрична відповідність (є простішим та поширенішим) та інтелектуальне розпізнавання символів.

Матрична відповідність (або порівняння з шаблоном) – це метод, суть якого полягає в порівнянні виявленого програмою OCR символу з бібліотекою шаблонів. Програмне забезпечення OCR зіставляє це зображення з відповідним символом ASCII.

Метод інтелектуального розпізнавання символів - це метод оптичного розпізнавання тексту, який використовує комп'ютерний інтелект для пошуку характерних ознак, таких як відкриті області, замкнуті фігури, діагональні лінії, перетин ліній і т.д.

Розробками алгоритмів для розпізнавання інформації з банківських карт на сьогоднішній день займаються ряд компаній, зокрема LabsNeural, SmartEngine та інші, а також розроблено додатки, які, за власними алгоритмами, здійснюють «видобування» реквізитів банківських карт [1-3]. Але, зважаючи на точність розпізнавання інформації, дослідження технологій оптичного розпізнавання реквізитів банківських карт залишається все ще актуальною задачею і вимагає подальшого вивчення.

Література

1. Cai S., Wen J., Xu H., Chen S., Ming Z. (2017) Bank Card and ID Card Number Recognition in Android Financial APP. In: Qiu M. (eds) Smart Computing and Communication. SmartCom 2016. Lecture Notes in Computer Science, vol 10135. Springer, Cham. - Режим доступу: https://doi.org/10.1007/978-3-319-52015-5_21.

2. LabsNeural. Custom OCR software [Електронний ресурс] - Режим доступу: https://labsneural.com/custom-ocr-software/.

3. SmartEngines. Безопасное распознавание без передачи персональных данных сторонним сервисам и третьим лицам [Електронний ресурс] - Режим доступу: https://smartengines.ru/smart-cardreader/#.

ЗМІСТ

Секція: КОМПЮТЕРНО-ІНФОРМАЦІЙНІ ТЕХНОЛОГІЇ ТА СИСТЕМИ ЗВ'ЯЗКУ

1. **П.Б. Балькан, І.Д. Винник, В.В. Ковальчук, Ю.С. Чміль, В.С. Деревянко** 5
ДОСЛІДЖЕННЯ СИСТЕМИ АВТОМАТИЗОВАНОГО РЕГУЛЮВАННЯ НАПРУГИ ЗВАРЮВАЛЬНОЇ ДУГИ

2. **І.О. Баран, В.С. Воронін** 6
ДО ПИТАННЯ РОЗРОБКИ СИСТЕМИ ЗБЕРЕЖЕННЯ ДАНИХ ДЛЯ ХМАРНОЇ ПЛАТФОРМИ OPENSTACK

3. **В.В. Бронецька** 7
ПЕРСПЕКТИВИ ВИКОРИСТАННЯ СИСТЕМИ ВІРТУАЛЬНОЇ РЕАЛЬНОСТІ ЗІ ЗВОРОТНІМ ЗВ'ЯЗКОМ ДЛЯ ЗНЯТТЯ СТРЕСУ

4. **В.О. Бурмістр, Г.М. Осухівська** 8
ТЕХНОЛОГІЇ РОЗПІЗНАВАННЯ РЕКВІЗИТІВ БАНКІВСЬКИХ КАРТ

5. **Р.А. Буцій, С.А. Лупенко** 9
АНАЛІЗ ОСНОВНИХ ХАРАКТЕРИСТИК КОМЕРЦІЙНИХ НЕЙРОІНТЕРФЕЙСІВ

6. **В. І. Лизун, А. Я. Баран, В. Я. Гураль, В. В. Бабовал, М. І. Яворська** 11
S-МОДЕЛІ ДЛЯ ОЦІНКИ НАДІЙНОСТІ ІНФОРМАЦІЙНИХ СИСТЕМ

7. **Д.В. Величко, А.В. Прунчак** 13
АКТУАЛЬНІСТЬ ДЕТЕКТУВАННЯ СИГНАЛІВ НА ФОНІ ЗАВАД У КОМП'ЮТЕРНИХ СИСТЕМАХ

8. **Р. В. Владика, С.А. Галайчук, Віт. Я. Галевіч, Вол. Я. Галевіч** 14
ОПТИМІЗАЦІЯ ПРОЦЕСУ ПЕРВИННОЇ ПЕРЕРОБКИ НАФТИ

9. **А.О. Волоха, Л.П. Дмитроца** 15
МОНІТОРИНГ ТА АВТОМАТИЗАЦІЯ КЕРУВАННЯ СЕРВЕРАМИ В ВИСОКОНАВАНТАЖЕНИХ СИСТЕМАХ

10. **А. М. Луцків, М. П.Голубовський** 16
КРИТЕРІЇ ВИБОРУ ІНСТРУМЕНТІВ ІАС

11. **Н. В. Громадський, Ю. П. Гуцалюк, І. М. Лесів, С. Я. Козловський** 18
ОПТИМІЗАЦІЯ ПРОЦЕСУ КЕРУВАННЯ ПІЧЧЮ ДЛЯ ФОРМУВАННЯ КОКСУ

12. **В.О. Дармограй, С.А. Лупенко** 19
АНАЛІЗ RFID ТЕГІВ ДЛЯ РЕАЛІЗАЦІЇ BLOCKCHAIN В IOT-ІНФРАСТРУКТУРІ

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ТЕРНОПІЛЬСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ
ІМЕНІ ІВАНА ПУЛЮЯ

М А Т Е Р І А Л И

VIII НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ

«ІНФОРМАЦІЙНІ МОДЕЛІ, СИСТЕМИ ТА ТЕХНОЛОГІЇ»

9–10 грудня 2020 року

ТЕРНОПІЛЬ
2020

СЕКЦІЯ 3. КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ

УДК 004.932.4
В.О. Бурмістр, Г.М. Осухівська канд. техн. наук, доцент
(Тернопільський національний технічний університет імені Івана Пулюя)

ПОКРАЩЕННЯ ЯКОСТІ ЗОБРАЖЕННЯ РЕКВІЗИТІВ БАНКІВСЬКИХ КАРТ ДЛЯ ЇХ ОПТИЧНОГО РОЗПІЗНАВАННЯ

UDC 004.932.4
V.O. Burmistr, H.M. Osukhivska Ph.D, Assoc. Prof.

IMPROVING THE IMAGE QUALITY OF BANK CARD DETAILS FOR THEIR OPTICAL RECOGNITION

Покращення якості розпізнавання зображень для різних задач все ще залишається актуальним завданням на сьогоднішній день. При використанні програмного забезпечення для оптичного розпізнавання реквізитів банківських карт [1] важливим є якість вхідного зображення. Оскільки символи на картках мають далеко не ідеальний вигляд, а подальше отримання необхідної інформації залежить від точності вхідних даних, тому пропонується здійснювати попередню обробку зображення.

Алгоритмів обробки зображення з метою покращення якості розпізнавання символів є велика кількість [2], але, враховуючи специфіку нанесення реквізитів на банківських картках, існуючі методи розпізнавання не зовсім підходять. Для попередньої обробки зображення реквізитів банківських карт запропоновано використати медіанний фільтр, метод згортки для збільшення різкості, масштабування, додавання відтінків сірого та бінаризацію. На основі розробленого методу створено програмне забезпечення у вигляді класу на мові C#, який може бути вбудований у інші програмні продукти для покращення якості зображення.

Для перевірки ефективності запропонованого методу проведено порівняння якості розпізнавання реквізитів банківських карт з використанням програмного забезпечення з відкритим кодом Tesseract OCR та Tesseract OCR із попередньою обробкою запропонованим методом. Отримані результати зведені в таблицю.

Точне значення	Tesseract OCR		Tesseract OCR з попередньою обробкою	
	Результат	Кількість помилок	Результат	Кількість помилок
5168 7553 3763 5284	5168 75 3763 5284	2	5168 7553 3763 5284	0
5168 7553 0621 0689	5168 7553 0621 0689	0	5168 7553 0621 0689	0
5168 7553 4097 1833	-	Не розпізнано	5168 7553 4097 1833	0
5168 7555 1455 0934	-	Не розпізнано	5168 7555 455 0934	1
4149 4978 2769 4540	4149 4978 2769 4540	0	4149 4978 2769 4540	0
4149 4978 7045 7399	-	Не розпізнано	4349 4978 7043 7399	2
5168 7573 1471 1062	5168 7573 1471 1062	0	5168 7573 1471 1062	0
4874 1200 2153 9632	4874 1200 2153	4	4874 1290 2153 9132	2
4790 9601 2166 2333	-	Не розпізнано	4190 9301 2133 2333	3
5211 5373 0518 6731	5211 5373	8	5211 5373 0518 6731	0

Проаналізувавши отримані результати, можна зробити висновок про покращення якості розпізнавання реквізитів банківських карт після використання Tesseract OCR із попередньою обробкою запропонованим методом.

Література.

1. В.О. Бурмістр, Г.М. Осухівська. Технології розпізнавання реквізитів банківських карт / Матеріали IX Міжнародної науково-технічної конференції молодих учених та студентів. Актуальні задачі сучасних технологій – Тернопіль, 25–26 листопада 2020. Т. II. С.8.
2. Improve OCR Accuracy With Advanced Image Preprocessing. URL: https://docparser.com/blog/improve-ocr-accuracy/.

ЗМІСТ

СЕКЦІЯ 1. **МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ**

Р. Буцій, С. Лупенко
АНАЛІЗ МЕТОДІВ ДЛЯ ЗАДАЧ ОПРАЦЮВАННЯ СИГНАЛІВ НЕЙРОІНТЕРФЕЙСНИХ СИСТЕМ
R. Butsiy, S. Lupenko
ANALYSIS OF METHODS FOR TASKS OF PROCESSING SIGNALS FOR NEUROINTERFEISE SYSTEMS 3

Д. Вівчар, І. Дедів
ЗАСТОСУВАННЯ МЕТОДУ ДЕЛЬТА-МОДУЛЯЦІЇ ДЛЯ КОДУВАННЯ СИГНАЛІВ
D. Vivchar, I. Dediv
APPLICATION OF THE DELTA-MODULATION METHOD FOR SIGNAL CODING 4

В. Влащук, Б. Яворський, М. Влащук, Б. Яворський, В. Лесів, А. Марценюк
МЕТОД АДАПТИВНОЇ ФІЛЬТРАЦІЇ ЦИФРОВОЇ ОБРОБКИ СКЛАДНИХ РАДІОЛОКАЦІЙНИХ СИГНАЛІВ
V. Vlashchuk, B. Yavorskyi, V. Lesiv, A. Marcenjuk
ADAPTIVE FILTERING METHOD OF DIGITAL PROCESSING OF COMPLEX RADAR SIGNALS 5

Л. Данильченко, П. Дудар
КІНЦЕВО-ЕЛЕМЕНТНЕ МАТЕМАТИЧНЕ МОДЕЛЮВАННЯ ПРОЦЕСІВ РІЗАННЯ
L. Danylchenko, P. Dudar
FINITE-ELEMENT SIMULATION OF THE CUTTING PROCESSES 7

Л. Данильченко, Д. Радик
АНАЛІЗ МАТЕМАТИЧНИХ МОДЕЛЕЙ ТЕРТЯ В ОБОРОБЛЕННІ ТИСКОМ
L. Danylchenko, D. Radyk
ANALYSIS OF FRICTION MATHEMATICAL MODELS IN METAL FORMING 8

Л. Мельник
ОСОБЛИВОСТІ УПРАВЛІННЯ БІЗНЕС-ПРОЦЕСАМИ БУДІВЕЛЬНОЇ ОРГАНІЗАЦІЇ
L. Melnyk
THE FEATURES OF BUSINESS PROCESS MANAGEMENT OF A BUILDING ORGANIZATION 9

С. Новосад, Б. Яворський, В. Лесів, А. Марценюк
МЕТОД АДАПТИВНОЇ ФІЛЬТРАЦІЇ ЦИФРОВОЇ ОБРОБКИ СКЛАДНИХ РАДІОЛОКАЦІЙНИХ СИГНАЛІВ
S. Novosad, B. Yavorskyi, V. Lesiv, A. Marcenjuk
ADAPTIVE FILTERING METHOD OF DIGITAL PROCESSING OF COMPLEX RADAR SIGNALS 10

Р. Кривий, П. Баран
ДОСЛІДЖЕННЯ РИНКУ ТРАНСПОРТНИХ ПОСЛУГ
Kryvyj R., Baran P.
TRANSPORT SERVICES MARKET RESEARCH 12

Д. Резнік
ПОРІВНЯЛЬНИЙ АНАЛІЗ СТЕГАНОГРАФІЧНИХ АЛГОРИТМІВ ПРИХОВУВАННЯ ІНФОРМАЦІЇ В ЗОБРАЖЕННЯХ
D. Reznik
COMPARATIVE ANALYSIS OF STEGANOGRAPHIC ALGORITHMS FOR HIDE INFORMATION IN IMAGES 89

А. Ринков, Л. Демків, Н. Кунанець
ІНТЕЛЕКТУАЛЬНА ІНФОРМАЦІЙНА СИСТЕМА ГЕНЕРАЦІЇ УМОВНИХ ЗНАКІВ У СТАНДАРТІ APP-6D
A. Rinkov, L. Demkiv, N. Kunanets
INTELLIGENT INFORMATION SYSTEM FOR GENERATION OF SYMBOLS IN THE APP-6D STANDARD 91

О. Головко, А. Станько
ТЕЛЕМЕДИЦИНА В ЕПОХУ COVID-19
O. Holovko, A. Stanko
TELEMEDICINE IN THE COVID-19 ERA 92

О. Яремчук
АЛГОРИТМ АСИНХРОНОГО АНАЛІЗУ ЦИКЛІЧНИХ КОЛИВАНЬ КОТИРУВАНЬ ЦІННИХ ПАПЕРІВ
O. YAREMCHUK
ALGORITHM OF ASYNCHRONOUS ANALYSIS OF CYCLIC OSCILLATIONS OF SECURITIES QUOTATIONS 94

О. Яремчук
МЕТОДИ ТА МОДЕЛІ ТОРГІВЕЛЬНИХ ІНДИКАТОРІВ ПОРТФЕЛІВ ЦІННИХ ПАПЕРІВ
O. YAREMCHUK
METHODS AND MODELS OF TRADING INDICATORS OF SECURITIES PORTFOLIO 95

СЕКЦІЯ 3. **КОМП'ЮТЕРНІ СИСТЕМИ ТА МЕРЕЖІ**

В Бурмістр, Г. Осухівська
ПОКРАЩЕННЯ ЯКОСТІ ЗОБРАЖЕННЯ РЕКВІЗИТІВ БАНКІВСЬКИХ КАРТ ДЛЯ ЇХ ОПТИЧНОГО РОЗПІЗНАВАННЯ
V. Burmistr, H. Osukhivska
IMPROVING THE IMAGE QUALITY OF BANK CARD DETAILS FOR THEIR OPTICAL RECOGNITION 96

П. Василишин, А. Редчук, А. Паламар
ІНФОРМАЦІЙНО-ВИМІРЮВАЛЬНА СИСТЕМА ДЛЯ КОНТРОЛЮ СТАНУ ТРАНСПОРТНИХ ЗАСОБІВ З ВИКОРИСТАННЯМ ТЕХНОЛОГІЇ INTERNET OF THINGS
P. Vasylyshyn, A. Redchuk, A. Palamar
INFORMATION AND MEASURING SYSTEM FOR MONITORING THE CONDITION OF VEHICLES USING INTERNET OF THINGS TECHNOLOGY 97

М. Луцків, М. Ващук
ГРАМАТИКА ПЕРЕТВОРЕННЯ ПАРАМЕТРІВ МОДЕЛІ МЕРЕЖІ ПЕТРІ У ПРОГРАМНИЙ КОД МОВИ С++
A. Lutskiv, M. Vashschuk
GRAMMAR OF PETRI NETWORK MODEL PARAMETERS TRANSFORMATION IN C ++ 98

Додаток Б. Лістинг коду

Вміст файлу ImageProcessing.cs:

```
using AForge.Imaging.Filters;
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Drawing.Drawing2D;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
namespace Tesseract_OCR
{
    class ImageProcessing
    {
        private Bitmap image;
        public struct RGB
        {
            public float R;
            public float G;
            public float B;
        }
        public Bitmap Image { get => image; set =>
image = value; }
        public void applyAllFilters(Bitmap image) {
            this.image = image;
            imageScaling(1.2);
            noiseCorrectionMedian();
            noiseCorrectionConvolution();
```

```
            greyScale();
            binarisationWithLocalMaxMin();
            noiseCorrectionMedian();
        }
        public void greyScale(){

            Bitmap bmap = (Bitmap)image.Clone();
            Color c;
            for (int i = 0; i < bmap.Width; i++)
            {
                for (int j = 0; j < bmap.Height; j++)
                {
                    c = bmap.GetPixel(i, j);
                    byte gray = (byte)(.299 * c.R +
.587 * c.G + .114 * c.B);

                    bmap.SetPixel(i,                j,
Color.FromArgb(gray, gray, gray));
                }
            }
            image = bmap;
        }
        public void noiseCorrectionMedian()
        {
                int Size = 5;
                System.Drawing.Bitmap   TempBitmap   =
Image;
                System.Drawing.Bitmap NewBitmap = new
System.Drawing.Bitmap(TempBitmap.Width,
TempBitmap.Height);
```

```
                    System.Drawing.Graphics NewGraphics =
System.Drawing.Graphics.FromImage(NewBitmap);
                    NewGraphics.DrawImage(TempBitmap, new
System.Drawing.Rectangle(0,     0,     TempBitmap.Width,
TempBitmap.Height), new System.Drawing.Rectangle(0, 0,
TempBitmap.Width,                    TempBitmap.Height),
System.Drawing.GraphicsUnit.Pixel);
                    NewGraphics.Dispose();
                    Random TempRandom = new Random();
                    int ApetureMin = -(Size / 2);
                    int ApetureMax = (Size / 2);
                    for (int x = 0; x < NewBitmap.Width;
++x)
                    {
                        for (int y = 0; y <
NewBitmap.Height; ++y)
                        {
                            List<int> RValues = new
List<int>();
                            List<int> GValues = new
List<int>();
                            List<int> BValues = new
List<int>();
                            for (int x2 = ApetureMin; x2 <
ApetureMax; ++x2)
                            {
                                int TempX = x + x2;
                                if (TempX >= 0 && TempX <
NewBitmap.Width)
                                {
```

```
                                    for    (int    y2    =
ApetureMin; y2 < ApetureMax; ++y2)
                                    {
                                        int TempY = y + y2;
                                        if  (TempY  >=  0  &&
TempY < NewBitmap.Height)
                                        {
                                            Color
TempColor = TempBitmap.GetPixel(TempX, TempY);

RValues.Add(TempColor.R);

GValues.Add(TempColor.G);

BValues.Add(TempColor.B);
                                        }
                                    }
                                }
                            }
                            RValues.Sort();
                            GValues.Sort();
                            BValues.Sort();
                            Color       MedianPixel       =
Color.FromArgb(RValues[RValues.Count / 2],
                                GValues[GValues.Count    /
2],
                                BValues[BValues.Count    /
2]);
                            NewBitmap.SetPixel(x,       y,
MedianPixel);
```

```
                    }
                }
                Image = NewBitmap;
            }

            public void binarisationWithLocalMaxMin()
            {
                BradleyLocalThresholding filter = new
BradleyLocalThresholding();

                Bitmap pic = new Bitmap(image);
                for (int y = 0; (y <= (pic.Height - 1));
y++)
                {
                    for (int x = 0; (x <= (pic.Width - 1));
x++)
                    {
                        Color inv = pic.GetPixel(x, y);
                        inv = Color.FromArgb(255, (255 -
inv.R), (255 - inv.G), (255 - inv.B));
                        pic.SetPixel(x, y, inv);
                    }
                }
                image                                    =
filter.Apply(Grayscale.CommonAlgorithms.BT709.Apply(new
Bitmap(pic)));
            }

            public void reverseImage()
            {
```

```
                Bitmap pic = new Bitmap(image);
                for (int y = 0; (y <= (pic.Height - 1));
y++)
                {
                    for (int x = 0; (x <= (pic.Width - 1));
x++)
                    {
                        Color inv = pic.GetPixel(x, y);
                        inv = Color.FromArgb(255, (255 -
inv.R), (255 - inv.G), (255 - inv.B));
                        pic.SetPixel(x, y, inv);
                    }
                }
                image = pic;
            }
            public void binarisationGlobal()
            {
                for (var x = 0; x < image.Width; x++)
                {
                    for (var y = 0; y < image.Height; y++)
                    {
                        var pixel = image.GetPixel(x, y);
                        if (pixel.R < 162 && pixel.G < 162
&& pixel.B < 162)
                            image.SetPixel(x,           y,
Color.Black);
                        else if (pixel.R > 162 && pixel.G
> 162 && pixel.B > 162)
                            image.SetPixel(x,           y,
Color.White);
```

```
                }
            }
        }

        public void imageScaling(double
scaleCoefficient)
        {
            int newWidth = Convert.ToInt32(image.Width
* scaleCoefficient);
            int newHeight =
Convert.ToInt32(image.Height * scaleCoefficient);
            var newbmp = new Bitmap(newWidth,
newHeight);
            var g = Graphics.FromImage(newbmp);
            g.SmoothingMode = SmoothingMode.None;
            g.InterpolationMode =
InterpolationMode.NearestNeighbor;
            g.PixelOffsetMode =
PixelOffsetMode.HighQuality;
            g.CompositingQuality =
CompositingQuality.AssumeLinear;
            g.DrawImage(image, new Rectangle(0, 0,
newWidth, newHeight));
            image = newbmp;
        }
        public void noiseCorrectionConvolution()
        {
            UInt32[,] pixel;
            pixel = new UInt32[image.Height,
image.Width];
```

```
                for (int y = 0; y < image.Height; y++)
                    for (int x = 0; x < image.Width; x++)
                        pixel[y,           x]           =
(UInt32)(image.GetPixel(x, y).ToArgb());
                int W = image.Width;
                int H = image.Height;
                const int N = 3;
                double[,] matryx = new double[N, N] {{-1,
-1, -1},

{-1,  9, -1},

{-1, -1, -1}};
                int i, j, k, m, gap = (int)(N / 2);
                int tmpH = H + 2 * gap, tmpW = W + 2 * gap;
                UInt32[,]  tmppixel  =  new  UInt32[tmpH,
tmpW];
                UInt32[,] newpixel = new UInt32[H, W];
                for (i = 0; i < gap; i++)
                    for (j = 0; j < gap; j++)
                    {
                        tmppixel[i, j] = pixel[0, 0];
                        tmppixel[i,  tmpW  -  1  -  j]  =
pixel[0, W - 1];
                        tmppixel[tmpH - 1 - i, j] = pixel[H
- 1, 0];
                        tmppixel[tmpH - 1 - i, tmpW - 1 -
j] = pixel[H - 1, W - 1];
                    }
                for (i = gap; i < tmpH - gap; i++)
```

```
                    for (j = 0; j < gap; j++)
                    {
                        tmppixel[i, j] = pixel[i - gap, j];
                        tmppixel[i, tmpW - 1 - j] = pixel[i
- gap, W - 1 - j];
                    }
                for (i = 0; i < gap; i++)
                    for (j = gap; j < tmpW - gap; j++)
                    {
                        tmppixel[i, j] = pixel[i, j - gap];
                        tmppixel[tmpH - 1 - i, j] = pixel[H
- 1 - i, j - gap];
                    }
                for (i = 0; i < H; i++)
                    for (j = 0; j < W; j++)
                        tmppixel[i + gap, j + gap] =
pixel[i, j];
                RGB ColorOfPixel = new RGB();
                RGB ColorOfCell = new RGB();
                for (i = gap; i < tmpH - gap; i++)
                    for (j = gap; j < tmpW - gap; j++)
                    {
                        ColorOfPixel.R = 0;
                        ColorOfPixel.G = 0;
                        ColorOfPixel.B = 0;
                        for (k = 0; k < N; k++)
                            for (m = 0; m < N; m++)
                            {
```

```
                                ColorOfCell             =
calculationOfColor(tmppixel[i - gap + k, j - gap + m],
matryx[k, m]);
                                ColorOfPixel.R          +=
ColorOfCell.R;
                                ColorOfPixel.G          +=
ColorOfCell.G;
                                ColorOfPixel.B          +=
ColorOfCell.B;
                            }
                        if     (ColorOfPixel.R     <     0)
ColorOfPixel.R = 0;
                        if    (ColorOfPixel.R    >    255)
ColorOfPixel.R = 255;
                        if     (ColorOfPixel.G     <     0)
ColorOfPixel.G = 0;
                        if    (ColorOfPixel.G    >    255)
ColorOfPixel.G = 255;
                        if     (ColorOfPixel.B     <     0)
ColorOfPixel.B = 0;
                        if    (ColorOfPixel.B    >    255)
ColorOfPixel.B = 255;
                        newpixel[i  -  gap,  j  -  gap]  =
build(ColorOfPixel);
                    }
                for (int y = 0; y < image.Height; y++)
                    for (int x = 0; x < image.Width; x++)
                        image.SetPixel(x,               y,
Color.FromArgb((int)newpixel[y, x]));
            }
```

```
            public static RGB calculationOfColor(UInt32
pixel, double coefficient)
            {
                RGB Color = new RGB();
                Color.R = (float)(coefficient * ((pixel &
0x00FF0000) >> 16));
                Color.G = (float)(coefficient * ((pixel &
0x0000FF00) >> 8));
                Color.B = (float)(coefficient * (pixel &
0x000000FF));
                return Color;
            }

            public static UInt32 build(RGB ColorOfPixel)
            {
                UInt32 Color;
                Color       =       0xFF000000       |
((UInt32)ColorOfPixel.R << 16) | ((UInt32)ColorOfPixel.G
<< 8) | ((UInt32)ColorOfPixel.B);
                return Color;
            }
        }
    }
```