ЗВО УНІВЕРСИТЕТ КОРОЛЯ
ДАНИЛА

Факультет суспільних та прикладних
наукКафедра інформаційних технологій

на правах рукопису

Адамович Андрій Романович

УДК 004.05

Застосування ентропії для ідентифікації символів за їх 2D зображеннями

Спеціальність 121 – «Інженерія програмного
забезпечення»Кваліфікаційна робота на здобуття
кваліфікації магістра

Нормоконтроль

_____к.т.н. Мануляк І. З.
(підпис, дата, розшифрування підпису)

Студент

_____ Адамович А.Р.
(підпис, дата, розшифрування підпису)

Допускається до захисту
Завідувач кафедри
_____к.т.н. Пашкевич О.П.
(підпис, дата, розшифрування підпису)

Керівник роботи :

_____д.т.н. Мельничук С.І
(підпис, дата, розшифрування підпису)

Івано-Франківськ – 2021

ЗВО УНІВЕРСИТЕТ КОРОЛЯ ДАНИЛА

Факультет суспільних та прикладних наук

Кафедра інформаційних технологій

Освітній ступінь: «магістр»

Спеціальність: 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____

« ___ » _____ 2021 року

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ (ПРОЕКТ) СТУДЕНТУ

Адамовичу Андрію Романовичу

(прізвище, ім'я, по батькові)

1. Тема магістерської роботи
Застосування ентропії для ідентифікації символів за їх 2D зображеннями
керівник роботи:
Мельничук Степан Іванович, доктор технічних наук
затверджена наказом вищого навчального закладу від « 30 » вересня 2021 року № 11/1 НВ
2. Термін подання студентом роботи 03.12.2021
3. Зміст магістерської роботи (перелік питань, які потрібно розробити)
1. Аналіз методів програмних засобів систем ідентифікації двомірних зображень.
2. Розробка структури та алгоритмів методу ідентифікації зображень символів цифр на основі ентропії.
3. Розробка дослідження ефективності програми для розпізнавання зображень цифр.

4. Дата видачі завдання 22.02.2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи (проекту)	Термін виконання етапів роботи	Примітка
1.	*Аналіз існуючих підходів та методів та методів розпізнавання символів цифр.*		*Виконано*
2.	*Аналіз переваг та недоліків існуючих методів ідентифікації символів.*		*Виконано*
3.	*Розробка методу на основі оцінки інформаційної ентропії.*		*Виконано*
4.	*Оформлення пояснювальної записки.*		*Виконано*
5.	*Оформлення графічного матеріалу.*		*Виконано*

Студент _____ *Адамович А.Р.*
(підпис) (прізвище та ініціали)

Керівник роботи _____ *Мельничук С.І.*
(підпис) (прізвище та ініціали)

Вихідні дані проекту: *Ruby on Rails, HTML5, CSS, MySQL*

Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

Сторінка	Опис графічного матеріалу	Сторінка	Опис графічного матеріалу
1	*Тема кваліфікаційної роботи*	*15*	*Висновок*
2	*Актуальність теми*		
5	*Алгоритмічне рішення розпізнавання зображень цифр*		
6	*Розпізнавання за інформаційною ентропією*		
7	*Розробка інтерфейсу та функцій порівняння зображень шаблонами*		
11	*Порівняльний аналіз*		

АНОТАЦІЯ

Завдання розпізнавання і перетворення текстової інформації при перекладі друкованого і рукописного тексту в машинні коди є одним з найважливіших складових проектів, що мають на меті автоматизацію документообігу.

Розроблено математичний метод на основі оцінки інформаційної ентропії, який покращує якість ідентифікації зображення при спотвореннях зображення зайвими пікселями та їх відсутністю.

Розроблена програма розпізнавання монохромних символів цифр використовує технологію OCR. Даний підхід вимагає створення шаблону для кожного шрифту. Програма розпізнає текст шрифту Calibri величиною шрифта 14. В процесі розпізнавання база еталоних зображень може поповнюватись новими видами шрифтів.

Ключові слова: OCR, Ruby on Rails, MySQL, ідентифікація зображення.

ANNOTATION

The task of recognizing and converting textual information when translating printed and handwritten text into machine code is one of the most important components of projects aimed at document automation.

A mathematical method based on information entropy estimation has been developed, which improves the quality of image identification in case of image distortion by extra pixels and their absence.

The developed program for recognizing monochrome character symbols uses OCR technology. This approach requires creating a template for each font. The program recognizes the text of the Calibri font with the size of font 14. In the process of recognition, the database of reference images can be replenished with new types of fonts.

Keywords: OCR, Ruby on Rails, MySQL, image identification.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

HTTP – Hyper Text Transfer Protocol;

HTML – HyperText Markup Language;

CSS – Cascading Style Sheets;

RoR – Ruby on Rails;

MVC – Model-View-Controller;

MySQL – Structured Query Language;

OCR – Optical Character Recognition.

ВСТУП

Актуальність теми. Завдання розпізнавання і перетворення текстової інформації при перекладірукописногоі друкованого тексту в машинні коди є одним з найважливіших складових проектів, які мають на меті автоматизувати документообіг. Разом з тим це завдання є одним з найбільш складних і наукоємних в області автоматичного аналізу зображень. Навіть людина, яка читає рукописні тексти, в відриві від контексту можезробитиблизько 4% помилок. Що стосується систем зчитування друкарських документів, то в таких відповідальних областях, як, для прикладу, автоматизація введення паспортно-візової інформації, потрібно забезпечити найвищу надійність розпізнавання (більше 98-99%) навіть приоцифровування початкового тексту і поганій якості друку.

У останні десятиліттябули розвинені нові методи для опрацювання зображень з використанням сучасних досягнень в області комп'ютерних технологій. Але створення кожного нового інструменту в даній області, все ще залишається творчим завданням та вимагає додаткових досліджень у зв'язку із специфічними вимогами щодо роздільної здатності,надійності розпізнавання і об'єму пам'яті, швидкодії, якими характеризується кожне конкретне вирішення інформаційної технології введення в комп'ютер паперової документації.

Мета дослідження. Підвищення ефективності обчислювальних методів ідентифікації об’єктів за їх монохромними зображеннями на основі розвитку ефективних технологій, методів та засобів опрацювання цифрових представлень з варіативною та змінною інформаційною ентропією. Розробка методу ідентифікації об’єктів на основі опрацювання імовірнісних представлень та проекцій, які формуються шляхом статистичного оцінювання значень інформаційної ентропії відповідних фрагментів їх характеристик.

Задача дослідження. Вдосконалення алгоритмічного забезпечення ідентифікації символівцифр за їх монохромними представленнями.

Для досягнення поставленої мети були вирішені наступні завдання:

1. Проведено аналіз існуючих підходів та методів розпізнавання

символів цифр з метою виявлення найбільш ефективних підходів до створення інформаційної системи їх опрацювання і розпізнавання.

2. Проаналізовано переваги та недоліки існуючих методів ідентифікації символів.

3. Розроблено математичний метод на основі оцінки інформаційної ентропії, який покращує якість ідентифікації зображення при спотвореннях зображення зайвими пікселями та їх відсутністю.

Об’єкт дослідження. Процеси ідентифікації символів тексту за їх 2D зображеннями з відсканованих документів.

Предмет дослідження. Засоби та методи ідентифікації об’єктів на основі монохромних двомірних зображень.

Методи дослідження. Основні результати і висновки зроблено на основі використання теорії інформації, теорії ймовірностей та математичної статистики, методів статистичного аналізу, цифрового опрацювання сигналів, а також за результатами моделювання в обчислювальному експерименті.

Наукова новизна. Отримали подальший розвиток, обчислювальні методи ідентифікації об’єктів на основі двовимірних масивів їх зображень.

Практичне значення одержаних результатів. Представлено метод, який покращує ідентифікацію символів при спотворенні зображення пікселями.

Апробація результатів дослідження. Матеріали кваліфікаційної роботи були представлені на конференції «Дослідження інновацій та перспективи розвитку науки і техніки у XXI столітті» яка була проведена 26 листопада 2021 року у м. Рівне.

1 АНАЛІЗ МЕТОДІВ ТА ПРОГРАМНИХ ЗАСОБІВ СИСТЕМ ІДЕНТИФІКАЦІЇ ДВОМІРНИХ ЗОБРАЖЕНЬ

1.1 Аналіз обчислювальних методів розпізнавання символів

Більшості програм оптичного розпізнавання символів (Optical Character Recognition - OCR) доводиться працювати із зображеннями документів, отриманих через сканер або факс-модем. Система OCR- повинна розбити сторінку на блоки, що являють собою графіку, текстовий зміст, таблиці та інші елементи з точки зору дій над зображеннями. Один із трудомістких і найскладніших процесів розпізнання пов’язаний з текстом, оскільки при його ідентифікації потрібно враховувати наявності декількох колонок, особливості вирівнювання тексту та інші елементи форматування.

Сам процес розпізнавання документа зображення починається з виявлення тексту, що являє собою набір елементарних зв’язаних областей (літер), які мають приблизно такий самий розмір та розташовані на площині вздовж паралельних прямих.

Блок розбивається на рядки після розпізнання текстового блока. Насправді це не є простою задачею, тому, що на практиці можна зустріти перекіс зображення сторінки, тобто маленький нахил навіть приводить до того, що лівий край біжучого рядка стає нижче правого краю іншого рядка, особливо при міжрядковому маленькому інтервалі. Виникає задача в цьому випадку, до якого саме, з двох рядків варто віднести даний елементарний фрагмент зображення - зв’язану область. Зв’язана область - не обов’язково літера. Наприклад, і та j складаються з двох простих фрагментів, окрім цього, кожна літера може опинитись розірваною на кілька простих областей в результаті дефектів сканування та друку. Це означає, що якщо десь між двома рядками опинилась окрема зв’язана область, це може бути крапка над і, відірвана частина літери нижнього рядка, чи частина букви верхнього рядка. Після того, як відбулося

розпізнавання рядків, наступним є розбиття їх на слова. На перший погляд досить проаналізувати відстань між зв'язаними областями (сусідніми символами) в рядку, обрати порогову величину, та якщо відстань між цими знаками перевищує це значення, потрібно рахувати, тут є пробіл. Але на практиці мають місце й такі помилки, як "злипання" і розрив різних слів. Саме тому досить часто відмовляються від одного порогового значення для всього рядка та проводиться обчислення локального порога по співвідношенню до декількох сусідніх символів.

Також необхідно вирішити завдання розпізнавання символів. Воно складне тим, що у зв'язку з непродрукуванням знаків та їх злипанням в слові неможливо знайти окремі знаки без застосування розпізнавання різних способів.

Методи розпізнавання символів. В загальному вигляді алгоритм розпізнавання - це є перевірка гіпотез та послідовне висунення, причому порядок висунення наступних гіпотез залежить від результатів перевірки минулих (рисунок 1.1). Гіпотеза майже кожна має оцінку числову або результат операції порівняння в алгоритмах розпізнавання систем OCR. Гіпотези послідовно звичайно висуваються, далі об'єднуються в список та сортуються на основі минулої оцінки. Кінцевий вибір гіпотези здійснюється в рамках контексту, із залученням, можливо навіть додаткових знань.

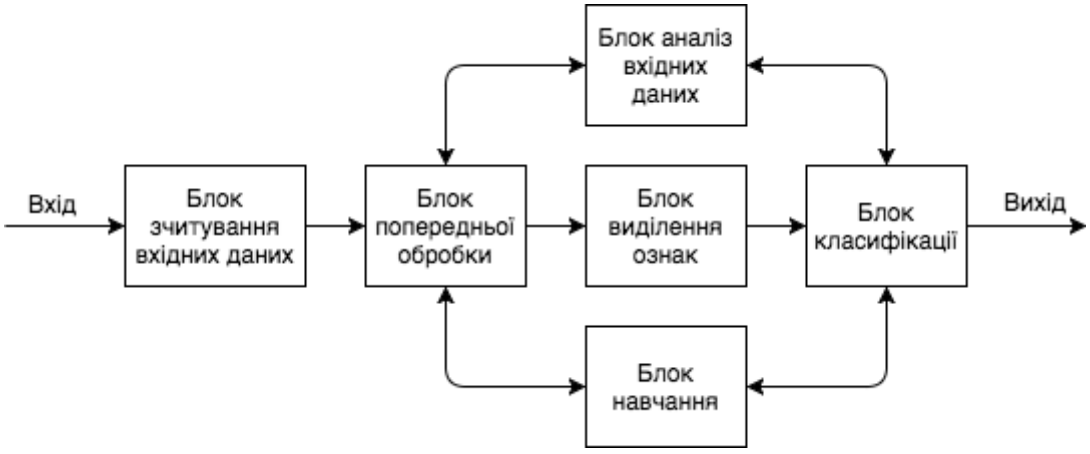


Рисунок 1.1 – Структурна схема розпізнаввня зображення

Розпізнавання символів різних зображень забезпечує рішення ряду

наукових і прикладних задач при ідентифікації різноманітних об'єктів. Теперишні способи розпізнавання символів використовуються для вирішення не тільки типових завдань, наприклад розпізнавання тексту, але й і спеціалізованих завдань, які орієнтовані на розпізнавання символьної інформації, нанесеної на поверхню різноманітних об'єктів. Зараз існує відносно велика кількість програм, призначених для розпізнавання тексту (наприклад, FineReader, Readiris, ScanSoft OmniPaeg, CuneiForm та ін.). Кожна з яких свою реалізацію пропонує рішення задачі обробки та розпізнавання зображень. В загальному ці програми є комерційними, що означає, що методи вирішення завдань, закладені в них, відомі тільки їх розробникам, тому неможливо визначити практично для яких завдань вони підходять та які не під силу їм завдання. Крім того, ці всі програми поставляються у вигляді модулів виконуваних, що робить неможливими аналіз працездатності програм, якості їх роботи і модифікацію використаних математичних моделей і алгоритмів. У даній роботі розглянемо найпопулярніші і поширені методи розв'язання задачі розпізнавання символів. На даний час виділяють три основні підходи для вирішення сформульованого завдання: структурний, ознаковий і шаблонний. У кожному методі присутні свої переваги та недоліки.

При розпізнаванні по шаблонах програма OCR працює зазвичай з великим растровим зображенням зі сканера сторінки. При цьому у більшості систем наявні шаблони, створені для різних накреслень. Після декількох розпізнаних слів, програма визначає використовуваний шрифт і шукає відповідні пари тільки для нього. Є випадки в яких, програмне забезпечення використовує чисельні значення частин символу (пропорцій), для того щоб визначити новий шрифт. Це також може поліпшити ефективність розпізнавання. Перероблення зображення сканованого в растрове є початковим етапом роботи шаблонного методу. Після нього виробляється його порівняння з всіма наявними шаблонами в базі системи. Найбільше підходящим шаблоном вважається той шаблон, у якого буде найменша кількість точок, відмінних від досліджуваного зображення. Шаблон зазвичай отримують для кожного класу,

усереднюючи зображення символів. У таких методах досить висока точність розпізнавання дефектних символів (склеєних або розірваних). Неможливість розпізнати шрифт, який відрізняється від закладеного в систему (розміром, нахилом або накресленням) є недоліком цього методу. Алгоритм, заснований на шаблонному методі, має заздалегідь знати шрифт, який наданий для розпізнавання. Оскільки друкованої продукції існує багато, в процесі навчання неможливо охопити всі шрифти та їх модифікації. Якщо пояснити іншими словами, цей фактор обмежує універсальність цих методів.

Розглянуто метод розпізнання на окремому прикладі шаблонного методу, який був розроблений ще в 60-ті роки минулого століття і застосовувався при створенні читаючого пристрою «РУТА 701», він також використовується і в даний час[1]. Мірою подібності в даному методі обраний коефіцієнт подібності зображення символу виражений наступною формулою:

$$R_s = \sum_{j+1}^{n} \left(ln \frac{P_{js}}{1-P_{js}}\right) x_j + \sum_{j=1}^{n} \ln\left(1 - P_{js}\right) \qquad (1.1)$$

де Rs - коефіцієнт подібності опізнаного символу до еталонного зображенню S-го класу символів, Pjs - імовірність появи чорного кольору в j -м елементі еталонного зображення S–го класу (виділяють три інтервали ймовірностей Pjs: 0,00 ÷ 0,25; 0,25 ÷ 0,75; 0,75 ÷ 1,00), Pxj - значення інтенсивності, відповідному j-му елементу розпізнаного символу. Зображення символу ототожнюється з еталонним класом, який дав максимальний коефіцієнт подібності R серед всіх Rs.

При структурному підході система розпізнавання OCR - використовує алгоритм, заснований на знаходженні загальних специфічних особливостей символів. Ця система містить близько 100 різних алгоритмів для ідентифікації 100 різних символів: нижнього та верхнього регістра від “A” до “Z”, записи чисел і символів пунктуації. Кожен з цих алгоритмів шукає “особливості” накреслень типу “півостровів”, “островів”, точок, прямих відбитків і дуг.

Експертні системи теж розглядають вертикальні та горизонтальні проекції відбитки літер та звертають увагу на основні особливості у створених кривих, підсумовуючи в них число темних пікселів [2]. Структурні методи розпізнавання зберігають інформацію про його топологію, а не про по-точкове написання символу. Тобто, еталон містить інформацію про взаємне розміщення окремих складових частин символу. Зрозуміло, що при цьому стає неважливим розмір розпізнаваної букви та навіть шрифт, яким вона є надрукована.

При цьому основною проблемою структурних методів розпізнавання залишається ідентифікація знаків, які мають дефекти (для прикладу, розрив лінії або злиття сусідніх ліній). Розглянуто методологію розпізнання на конкретному прикладі. Що розпізнається, знак піддається скелетизації процедурі (зменшення). Кожен отриманий контур скелетного уявлення у вигляді послідовного набору особливих точок описується і так званого ланцюгового коду, що складається з точки прив'язки, числа кодів і масиву напрямків з чергової точки на наступну точку. Особливі точки – це точки розгалуження (тріоди) та кінцеві точки, тобто точки, сусіди яких утворюють не менше трьох зв'язкових областей. У прикладі, представленому на рисунку 1.2, образ володіє двома внутрішніми контурами - однією кінцевою точкою і трьома тріодами.

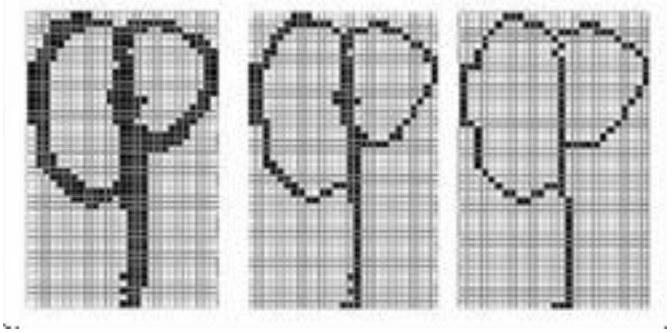


Рисунок 1.2 - Скелетизація образу, яка складається з 2-ох внутрішніх контурів та 1-го зовнішнього

В отриманому описі використовується округлення, що виконується видаленням коротких ліній, об'єднанні близьких тріодів, знищенні малих

внутрішніх контурів. Для зовнішнього контуру знаходиться його тип або топологічний код. Для цього він записується у вигляді послідовного набору номерів спеціальних точок, відповідних обходу за годинниковою стрілкою. Потім за допомогою перенумерування особливих точок і зміни початку контуру робиться спроба ототожнення контуру з одним з основних типів.

Ознакові методи базуються на тому, що зображенню ставиться у відповідність N–мірний вектор ознак. Розпізнавання полягає в порівнянні його з набором еталонних векторів тієї самої розмірності. Завдання розпізнавання, прийняття рішення про приналежність образу того або іншого класу, на підставі аналізу обчислених ознак, має велику кількість строгих математичних рішень в рамках імовірнісного і детерміністичного підходів. У системах розпізнавання символів досить часто використовується класифікація, заснована на підрахунку евклідової відстані між векторами ознак еталонного опису і вектором ознак розпізнаваного символу. Тип і кількість ознак у більшій мірі визначають якість розпізнавання. Формування вектора виробляється під час аналізу вже підготовленого зображення. Цей процес називають вийманням ознак. Еталон для всіх класів отримують в наслідок аналогічної обробки символів. Основні переваги ознакових методів - простота реалізації, хороша узагальнююча здатність, хороша стійкість до зміни форми символів, низьке число відмов від розпізнання, висока швидкодія.

Найбільш серйозний недолік цих методів - нестійкість до різних дефектів зображення. Крім того, ознакові методи мають інший серйозний недолік - на етапі вилучення ознак відбувається необоротна втрата частини інформації про символи. Вилучення ознак ведеться незалежно, тому інформація про взаємне розташування елементів символу втрачається.

Ідентифікація зображень на основі кореляційних методів обробки проходить наступним чином. Кореляційну обробку зображень представлених цифровими матрицями можна розділити на такі два типи: автокореляційна та взаємокореляційна.

Результати автокореляційної обробки відображають середньостатистичні

зв’язки між послідовностями відліків одного і того ж зображення. На практиці часто кореляційні оцінки використовуються, які простіші в обчисленні і відображають статистичні зв’язки з достатньою точністю. В основу обробки сигналів з різною базою покладено кореляційну функцію $R_{xx}(j)$:

Автокореляційна функція:

$$R_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n} \mathring{x}_i \cdot \mathring{x}_{i+j} \qquad (1.2)$$

Автоковаріаційна функція:

$$K_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n} x_i \cdot x_{i+j} \qquad (1.3)$$

Нормована автокореляційна функція:

$$\rho_{xx}(j) = \frac{R_{xx}(j)}{D_x} \qquad (1.4)$$

Знакова автокореляційна функція:

$$H_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n} \mathrm{sgn}[\mathring{x}_i] \cdot \mathrm{sgn}[\mathring{x}_{i+j}], \qquad (1.5)$$

де $\mathrm{sgn}[x_i] = \begin{cases} 1, x_i > 0; \\ 0, x_i = 0 \\ -1, x_i < 0 \end{cases}$

Полярна функція автокореляції:

$$P_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n} \mathring{x}_i \cdot \mathrm{sgn}[\mathring{x}_{i+j}] \qquad (1.6)$$

Структурна взаємокореляційна функція:

$$C_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n}(x_i - x_{i+j})^2 \qquad (1.7)$$

Модульна автокореляційна функція:

$$G_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n}\left| x_i - x_{i+j} \right| \qquad (1.8)$$

Автокореляційна функція еквівалентності:

$$\overset{\vee}{F}_{xx}(j) = \frac{1}{n}\sum_{i=1}^{n}\overset{\vee}{z}_i[x_i, x_{i+j}], \qquad (1.9)$$

$$\text{де } \overset{\vee}{z}_i = \begin{cases} x_i, \text{ при } x_i \le x_{i+j}, \\ x_{i+j}, \text{ при } x_i > x_{i+j}. \end{cases}$$

Обчислення кореляційної функції забезпечує високу точність визначення зв’язків між станами інформаційних сигналів.

Результати взаємокореляційної обробки характеризують середньостатистичні зв’язки між двома числовими послідовностями. Досить часто виступає ідеальна послідовність в якості одного з процесів відліків, в якості іншого – отримані з каналу обміну даними реальні стани. Взаємокореляційні моделі реалізуються на основі взаємокореляційних функцій, які аналогічні до технологій побудови автокореляційних.

Взаємокореляційна функція:

$$R_{xy}(j) = \frac{1}{n}\sum_{i=1}^{n}\overset{\vee}{x}_i \cdot \overset{\vee}{y}_{i+j} \qquad (1.10)$$

Взаємоковаріаційна функція:

$$K_{xy}(j) = \frac{1}{n}\sum_{i=1}^{n} x_i \cdot y_{i+j} \qquad (1.11)$$

Нормована взаємокореляційна функція:

$$r_{xx}(j) = \frac{R_{xy}(j)}{\sqrt{D_x \cdot D_y}} \qquad (1.12)$$

Структурна взаємокореляційна функція:

$$C_{xy}(j) = \frac{1}{n}\sum_{i=1}^{n} (x_i - y_{i+j})^2 \qquad (1.13)$$

Модульна взаємокореляційна функція:

$$G_{xy}(j) = \frac{1}{n}\sum_{i=1}^{n} \left| x_i - y_{i+j} \right| \qquad (1.14)$$

Полярна функція взаємокореляції:

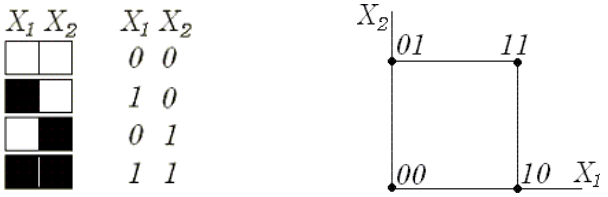
$$P_{xy}(j) = \frac{1}{n}\sum_{i=1}^{n} \dot{x}_i \, sign[\dot{y}_{i+j}] \qquad (1.15)$$

Взаємокореляційна функція еквівалентності:

$$F_{xy}(j) = \frac{1}{n}\sum_{i=1}^{n} z_j[x_i, y_{i+j}] \qquad (1.16)$$

Знакова, нормована та еквівалентні кореляційні функції є найменш чутливими до спотворення (шумів).

Розпізавання зображень об’єктів на основі методу потеціальних функцій має свою специфіку. Особливістю цього алгоритму є можливість його навчати. Для простоти доцільно розглядати тільки чорно-білі зображення. Нехай малюнок складається лише з двох пікселів. Практично відображення множини всіх об’єктів, фактично обмежується 4-ма об’єктами: (0,0), (0,1), (1,0), (1,1), де 1 — позначає чорний піксель, 0 — позначає білий, що видно з рис.1.3а.



а) б)

Рисунок 1.3 – Представлення двохпіксельного малюнка

Отже всі об’єкти універсальної множини можна розмістити у вершинах одиничного квадрату, рисунок 1.3б, таким чином множині фігур, що зображені на двохпіксельному полі, можна співставити множину точок двомірного простору. Ребру цього квадрату буде відповідати перехід від одного зображення до іншого. Отже для переходу з точки (1,1) до (0,0) треба обійти два окремі ребра, для переходу від (0,1) до (0,0) – тільки одне. Тобто кількість ребер в переході відображає кількість точок обох зображень, що відрізняються, отже відстань від одного зображення до іншого представляється кількістю неоднакових точок (пікселів) у них і це є так звана відстань Хемінга.

Якщо розглянути малюнок з трьох пікселів то коди будуть складатися із трьох значень а універсальна множина - з восьми елементів, які можна розташувати у вершинах одиничного куба. Аналогічно з попереднім випадком відстань за Хемінгом визначається так само. Якщо взяти малюнок 50х70=3500 пікселів то код будь-якого зображення складається з 3500 значень а

універсальна множина – із $2^{3500} = 4{,}027 * 10^{1053}$ елементів, які будуть розташовані у вершинах одиничної 3500-мірної фігури. Але суть обробки (розпізнавання) залишається незмінною. Основна ідея полягає в тому, що в цій багатомірній фігурі зображення, які відповідають конкретному образу, лежать недалеко один від одного – гіпотеза про компактність образів, рисунок 1.4.

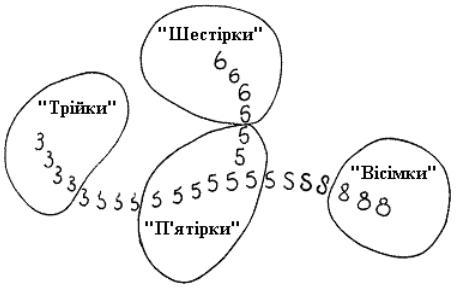


Рисунок 1.4 – Приклад гіпотези про компактність образів

Таким чином можна сформулювати кінцеву задачу: необхідно розділити універсальну множину на частини, деякі компактні множини, кожній з яких відповідає конкретний образ.

Реалізація програми відносно проста: в процесі навчання почергово задаються зображення (точки багатомірної фігури) і вказівки, до якого образу кожне зображення відноситься; в процесі розпізнавання програма шляхом обчислення відстані за Хемінгом встановлює, до якої з відомих компактних множин попало вхідне зображення[3].

Фактично розділяти універсальну множину не обов’язково, можна використати характеристику, яка показує віддаленість одного зображення до кожного з групи еталонних зображень. В якості міри віддаленості тут використовується потенціал, що визначається за формулою аналогічною до обчислення потенціалу точки простору в якому є електричний заряд:

$$P = a\frac{q}{R^2} \tag{1.17}$$

де a - деякий постійний коефіцієнт;

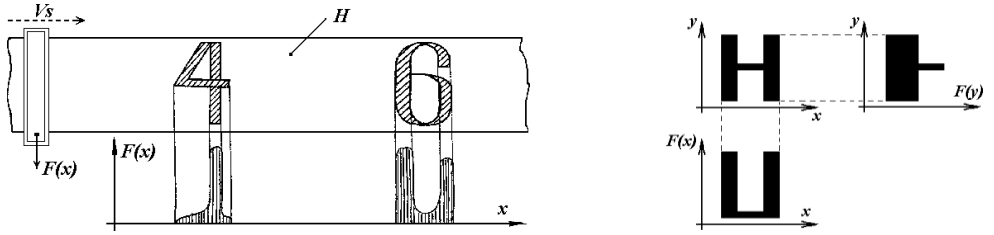
q- величина заряду;

R - відстань від даної точки до заряду.

Якщо електричне поле утворюється більшою кількістю зарядів, то потенціал в даній точці рівний сумі потенціалів кожного заряду. По аналогії – кожне зображення, на якому програма навчалась, утворює в просторі такої універсальної власний потенціал. Практично обчислюють такий потенціал, що формується в конкретній точці усіма об’єктами образу символу “а”, “б” і т.д. далі просто вибирають образ, що сформував максимальний потенціал.

Розпізавання зображень об’єктів на основі інтегральних методів. Інтегральні методи розпізнавання в якості суттєвих ознак використовують інтегральні характеристики геометричної форми знаків (форму хвилі струму чи напруги, аналогову функцію контуру, розподіл щільності зображення за досліджуваним полем тощо).

Наприклад, на рисунку 1.5а зображено знаки, що розпізнаються і відповідні закони інтегральної зміни проекції площі фігури при проектуванні її на горизонтальну вісь.



а) б)

Рисунок 1.5 – Приклад інтегральної зміни проекції площі фігури

В процесі обробки очікувана форма огинаючої кривої (так званої хвилі) кожного зчитаного знаку порівнюється з окремою еталонною хвилею кореляційним способом. Тобто при проходженні досліджуваного знака через зчитуючу матрицю отримується послідовність інтегральних значень елементів (0 – білий, 1 – чорний піксель) матриці, яка одночасно порівнюється з усіма наявними еталонами (інтегральними) і, як розпізнаний, обирається символ з максимальним значенням співпадінь, що вказує на високу степінь схожості досліджуваної фігури і еталону.

У випадку коли *δ(x,y)* є двійковою функцією, *F(x)* описує закон зміни площі букви вздовж осі *X*. Аналогічно *F(y)* є закон зміни площі вздовж осі *Y*. Тобто *F(x)*отримується в результаті сумування всіх значень *δ(x,y)* при фіксованому x_0, далі при деякому новому (наступному) x_1, що рівне x_0+dx і т.д. Функція, отримана таким чином, є однозначною.

Перегляд поля, на яке нанесено символ, здійснюється дискретно, тобто за рядками і стовпцями в результаті чого проекція задається сукупністю своїх стовпців (рядків):

$$F_i(x) = \left\{ f^i_1, f^i_2, ..., f^i_n \right\} \tag{1.18}$$

де *n* — число стовпців (рядків).

Далі здійснюється квантування за правилом:

$$f^i_j = \begin{cases} 1, f^i_j \geq \overline{f_j} \\ 0, f^i_j < \overline{f_j} \end{cases} \tag{1.19}$$

де $\overline{f_j}$ — порогове значення для даного стовпця проекції.

Еталонні знаки, за якими здійснюється порівняння, попередньо збережені в пам’яті. Порівняння кодів проекцій розпізнаваного знаку f^{oi}_j з еталонним $f^{эi}_j$ здійснюється послідовно, розряд за розрядом і при цьому число неспівпадань *a* з кожним із знаків.

Оскільки *δ* може приймати тільки значення 1 і 0, операція віднімання і підняття в квадрат двох *δ* здійснюється шляхом додавання за модулем 2 $(\sum \bmod 2)$Підрахунок числа неспівпадінь ведеться лічильником $C_ч a$. Далі відбувається порівняння за правилом:

$$Z = \begin{cases} 1, a_{\min} \leq q \cdot D_{ij\min} \\ 0, a_{\min} > q \cdot D_{ij\min} \end{cases} \tag{1.20}$$

де q — коефіцієнт пропорційності.

Якщо $Z = 1$, то рішення приймається на користь того знаку, з яким отримано a_{min} і цей знак видається як результат розпізнавання. Якщо ж $Z = 0$, то видається сигнал відмови.

Метод описаний вище також називають матричним методом розпізнавання оскільки тут знаки і символи представляються у вигляді матриць, що є сукупністю білих та чорних точок, і розпізнавання здійснюється шляхом безпосереднього співставлення матриці, отриманої при зчитуванні знаку, який розпізнається, з кожною з еталонних матриць наявної абетки.

Деяким способом подати наборами дискретних функцій в різних базисах: Хаара, Крейга, Уолша, Радемахера та інших.

Подальша обробка грунтується на реалізації розкладу таких об’єктів за власними функціями, методика розкладу наступна.

Формується набір власних функцій з довільними фазами (абетка), який утворює деяку множину (*V*), вданому випадку - функції (*V0... V6*), рисунок 1.6.

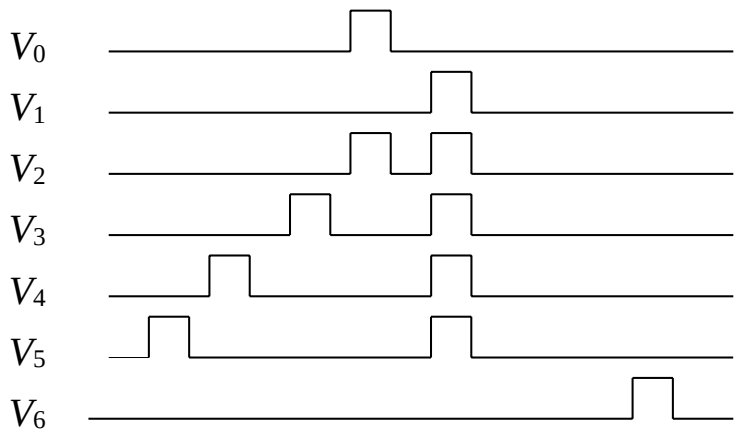


Рисунок 1.6 – Форми дискретних власних функцій з довільною фазою

Цей набір застосовується до відповідного зображення, внаслідок чого формується вектор розкладу. Приклад розкладу трикутного сигналу за наведеним вище набором власних функцій та подання його у вигляді відповідних векторів V_T подано на рисунок 1.7.

Звідки координати V_Tє наступні:V_T{*V5 V4 V3 V2V1*}.

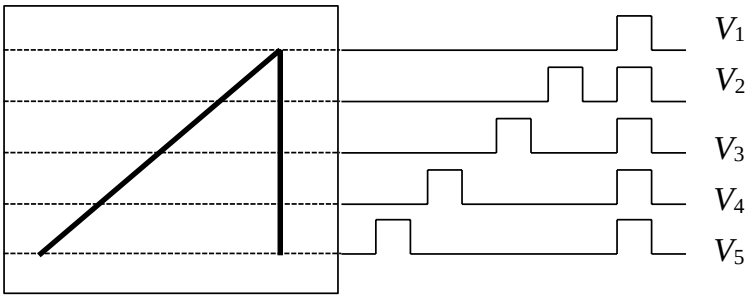


Рисунок 1.7 - Розклад трикутного зображення за власними функціями

Кожен еталонний образ (e_k) із множини еталонів (E) має свій визначаючий вектор (v_{ek}). В результаті сканування формується вектор розкладу цього об'єкту по власних функціях деякого графічного об'єкту (X). Відповідні координати цього вектора порівнюються з координатами векторів, що описують еталонні образи. При порівнянні значень кожної координати вектора графічного об'єкту з відповідними значеннями координат векторів еталонів, еталонам присвоюється ознаку співпадання (S_i^k):

$$S_i^k = \begin{cases} 1, якщо a_i^x = a_i^{e_к} \\ 0, якщо a_i^x \neq a_i^{e_к} \end{cases} \quad (1.21)$$

де k = 1,..., m - номер еталону;

m - загальна кількість еталонів в множині E.

Крім того, ознакам (S_i^k) можна надати нерівноцінні значення, які б враховували вагу кожної координати. В результаті отримується матриця ознак (S), розміром m х n.

$$S = \begin{vmatrix} s_1^1 & s_2^1 & \dots & s_n^1 \\ s_1^2 & s_2^2 & \dots & s_n^2 \\ \dots & \dots & \dots & \dots \\ s_1^m & s_2^m & \dots & s_n^m \end{vmatrix} \quad (1.22)$$

Сума елементів:

$$R_k = \sum_{i=1}^{n} s_i^k, \quad (1.23)$$

1.2 Аналіз методів розпізнавання реалізованих на основі нейромережевих структур та нечіткої логіки

Нейронні мережі - це є елементів структура пов'язаних, на яких задані функції перетворення сигналу, вони формують результуючі сигнали. Сигнал, який проходить через нейронну мережу, перетворюється також коефіцієнти, які можуть бути налаштовані на певний характер роботи. Деякі елементи структури виділені як вхідні: на них надходять сигнали ззовні, інші - вихідні: відповідно до формул на елементах мережі, і на виході формується відповідь [3]. Нейронна мережа класифікатор можна навчати, налаштовуючи коефіцієнти на елементах мережі, і саме цим чином, також може служити в системі розпізнавання тексту в якості класифікатора. Цей прагнути до ідеального результату розпізнавання. Нейронні мережі можуть застосовуватися в системах розпізнавання тексту, але є велика кількість недоліків, які перешкоджають їх частому застосуванню. Для того щоб побудувати елементів, що призводить до великих витрат пам'яті мережу, що забезпечує розпізнавання кожного символу тексту, необхідно побудувати досить велику мережу. В процесі розпізнавання сильніше витрачаються ресурси системи, всі випадки, але, це не може тому, що функції на елементах мережі працюють з числами з плаваючою крапкою. А ще, нейронні мережі необхідно навчати на гарантувати точного результату. І нарешті, робота нейронної мережі по розпізнаванню тексту багато в чому залежить від конфігурації функцій і мережі, заданих в елементах, що вимагає великих зусиль для побудови працюючої мережі.

Розвиток штучних нейронних мереж також надихається біологією, розглядаючи мережеві конфігурації і запозичені з принципів організації мозкової алгоритми, дослідники застосовують терміни, діяльності (біологічний прототип). На цьому розробникам мережі доводиться виходити аналогія закінчується. Знання про роботу мозку настільки обмежені, що мало б знайшлося точно доведених закономірностей для того, хто побажав би керуватися ними. Тому за межі сучасних біологічних знань для того, щоб

знайти структуру, здатну виконувати корисні функції. Інколи це призводить до необхідності відмови від біологічної правдоподібності, мозок стає просто метафорою, та створюються мережі, неможливі в живій матерії або вимагають неправдоподібно великих допущень про функціонування мозку та анатомію.

Не дивлячись на те, що зв'язок з біологією слабкий і часто несуттєвий, штучні нейронні мережі пізнанням, тому важко уникнути цієї аналогіїпродовжують порівнювати з мозком. Їх функціонування також має схожість з людським. Нажаль, ці порівняння неплодотворні та створюють очікування які є невиправданими, і неминуче ведуть до розчарування.

Нервова система людини, побудована з елементів, які називаються нейронами, і має приголомшливу складність. Приблизно8 – 9 нейронів беруть участь в зв'язках, які передаються, та унікальні здібності: приймати, обробляти мають довжину метр та більше. Кожний нейрон має багато властивостей, спільними з іншими органами тіла, але йому притаманні абсолютно і передавати електрохімічні сигнали по нервових шляхах, що утворюють комунікаційну систему мозку. На рисунку 1.7 подано структуру типових біонейронів.

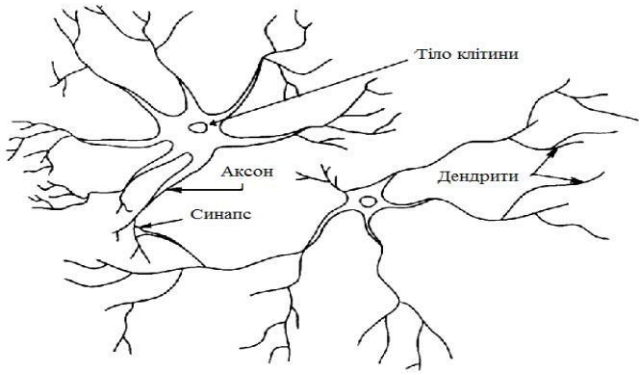


Рис. 1.7 – Структура пари біологічних нейронів

Дендрити йдуть від тіла нервової клітини до інших нейронів, де вони приймають сигнали в точках ронапередаються, та тут вони підсумовуються з'єднання, званих синапсами. Прийнятівхідні сигнали синапсом до тіла ней,

причому одні входи прагнуть порушити нейрон, інші - перешкодити його збудженню [4].

Після того як сумарне збудження в тілі нейрона перевищує деякий поріг, нейрон збуджується, посилаючи по штучних нейронних мереж моделюють лише ці прості властивості аксону сигнал іншим нейронам. У цієї основної функціональної схеми багато винятків і ускладнень, тим не менше, більшість.

Хоча нейрон і здатний виконувати прості процедури необхідно з'єднувати нейрони в мережі розпізнавання, але для серйозних обчислень нейронних (одношарові штучні нейронні мережі). Найпростіша мережа складається з декількох нейронів, якіяк показано утворюють шар в правій частині рисунку 1.8.

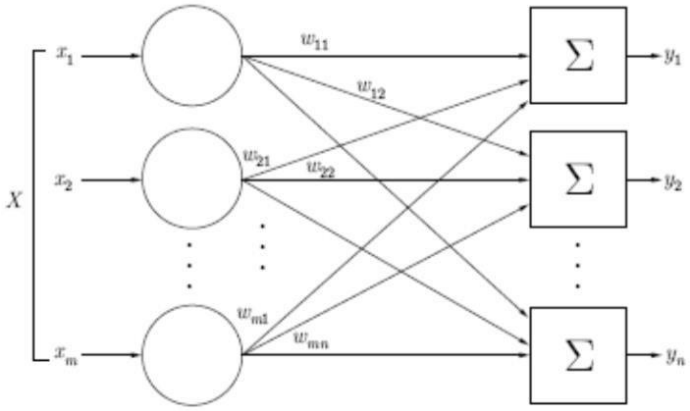


Рисунок 1.8 – Структурна схема одношарової нейронної мережі

Більші і складнімережі нейронні володіють, як правило, і великими обчислювальними можливостями (багатошарові тільки можна собі уявити, пошарова штучні нейронні мережі). Хоча створені мережі всіх конфігурацій, які організація нейронів копіюєструктури шаруваті певних були розроблені алгоритми відділів мозку. Виявляється, що такі багатошарові мережі володіють більшими можливостями, ніж одношарові, і в останні роки для їх навчання. Багатошарові мережі можуть не можуть багатошарові мережі привести до збільшення будуватися з каскадів шарів. Вихід одного шару є входом для наступного шару. Подібна мережа показана на рисунку 1.9. В порівнянні з

одношаровою мережею, потужності обчислювальної якщо активаціиньа функція між шарами лінійна. Обчислення виходу шару полягає в множенні вхідного вектора на першу (якщо відсутня нелінійна активаційна функція) результуючого вектора на вагову матрицю з подальшим множенням другу вагову матрицю [4].

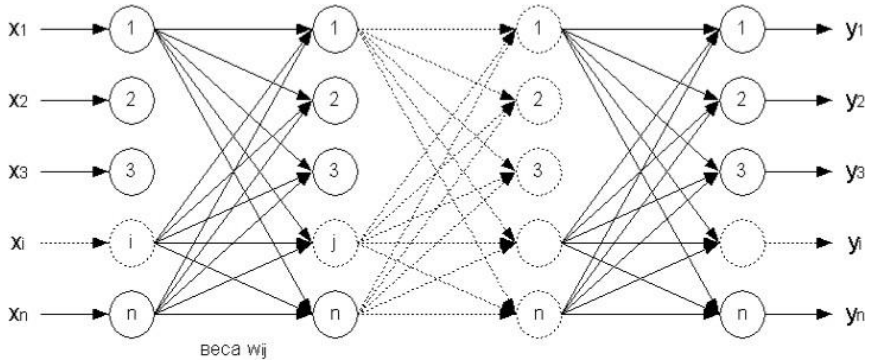


Рисунок 1.9 – Структура багатошарової штучної нейронної мережі

Розпізнавання тексту це комплекс які необхідно виконати завдань, для отримання кінцевого результату (тексту). Деякі комерційні системи розпізнавання тексту використовують набір алгоритмів, а інша частина може використовувати переваги клітинних автоматів, і нарешті які дають точний результат в сукупності. Одна частина функціонує на основі нейронної мережі системи розпізнавання, , третя частина системи накопичувати статистику та на її основі видавати результат. Комплекс заходів і алгоритмів, безперечно, дозволить кращого результату добитися, ніж окремо взятий принцип чи алгоритм. Клітинні автомати можуть бути частиною подібного комплексу, обчислення, легкість і простота правил, на основі яких вони побудовані, можливість які мають безперечні переваги, такі як можливість паралельного реалізації багатьох складних алгоритмів обробки зображень. Основна ідея запропонованого алгоритму розпізнавання символів за допомогою клітинних автоматів полягає в тому, що символи будь-якої мови , якщо деформація не змінює взаєморозташування ліній, ознака відрізняються один від одного характерними положеннями ліній одна відносно одної [5]. Даний підхід дає цілу

низку переваг при деформуванні або накладанні символів. У випадку символу не змінюється. Наприклад на рисунку 1.10 зображено спотворені символи EP, вкладені символи N та O, накладені символи VZ. Людина, легко ідентифікує ці символи на відміну від існуючих систем розпізнавання.

Рисунок1.10 - Спотворені символи англійського алфавіту

В даному алгоритмі використовується система з таких типів клітинних автоматів, що описують відповідні символи, тобто траєкторія руху такого автомата співпадає з певним символом. Окрім того, задаються такі правила взаємодії клітинних автоматів та розпізнавання зводиться до аналізу типів множин автоматів в тій чи іншій області функціонування, які переводять до стаціонарного стану систему, коли на кожному символі накопичуються автомати певного типу. Тому, задача клітинноавтоматного поля. Це зручно виконувати співставленням певного кольору тому чи іншому типу клітинного автомату. Тоді окремі символи в процесі алгоритми для функціонування клітинних автоматів розпізнавання будуть набувати характерний їм колір. Розглянуто у вигляді графів переходів імовірнісного автомата Мура (рисунок 1.11).

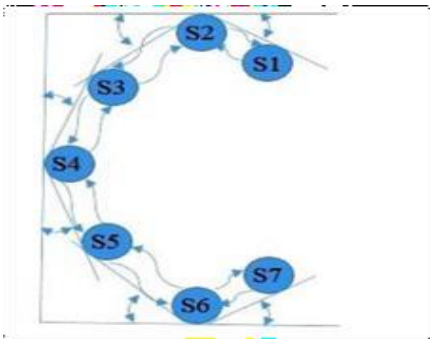


Рисунок 1.11 - Приклади графів переходів імовірнісних автоматів Мура для символа “С”

Вхідним сигналом тут (до іншого стану автомата умовою переходу) є або перебування клітинного автомату , яке має місце в символах в точці розгалуження: A, B, E, F та ін., або досягнення клітиною кінця лінії символу. Автомат переходить в один з набору рівно-ймовірних станів при цьому (згідно переходів графу). Сигнал руху клітини автомати можуть лише в межах клітин при цьому, в даний момент часу про напрямок (зображено стрілками на графі переходів біля станів:↓напрямок вниз, →напрямок вправо, ↑напрямок вгору, ←напрямок вліво), та перевірки кута між станами переходу клітинного автомата є вихідною реакцією клітинного автомата. Пересуватися клітинні а які відповідають їх символів. Цілком очевидно, що клітинний автомат, заданий графом, зображеним на рисунок 1.12, буде описувати символ „C”. Подібним чином можна побудувати графи переходів клітинних автоматів, які будуть описувати решту символів. З іншої сторони, задача зазначено, забезпечити слід такий розпізнавання апріорних відомостей щодо відношення тих чи інших символів до відповідного класу не передбачає. Тому, як вже було алгоритм функціонування та взаємодії клітинних автоматів, щоб в процесі роботи автомати конкретного типу алгоритму накопичувалися тих символів у клітинах, яким ці типи відповідають найбільше.

Приклади програмної реалізації. Фільтрація та обробка.

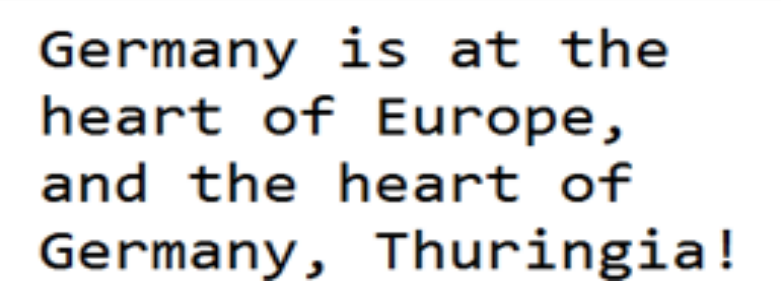


Рисунок 1.12 – Приклад зображення для фільтрації та обробки

Двома фільтрами дане зображення обробляється. Медіа та змінена версія медіанного фільтра з збільшенням монохромом, у додатку використовувалася компоненту значення червоного кольору.

```
public static void Median(ref Bitmap image{
var arrQ = new int[8];
 var arrW = new int[8];
 var arrE = new int[8];
 var outImage = new Bitmap(image);
 for (int x = 1; x< image.Width - 1; x++)
for (int y = 1; y< image.Height - 1; y++){
for (int x1 = 0; x1 < 2; x1++){
for (int y1 = 0; y1 < 2; y1++){
var p = image.GetPixel(x + x1 - 1, y + y1 - 1);
arrQ[3* x1 + y1] = ((p.Q + p.W + p.E) / 3) & 0xff;
arrW[3* i1 + j1] = ((p.Q + p.W + p.E) / 3) >> 8 & 0xff;
arrE[3* i1 + j1] = ((p.Q + p.W + p.E) / 3) >> 16 & 0xff;}
Array.Sorting(arrQ);
Array.Sorting(arrW);
Array.Sorting(arrE);
outImage.SetPixel(x, y, Color.FromArgb(arrQ[3], arrW[4], arrE[5]));}
   image = outImage}
```

Даний фільтр застосуються для змазування гострих кутівлітер (зарубок і т.п.) і зменшення шуму. Іншими словами відбувається чітка бінаризація, при чому межі літерфіксуються чітко. Потімкартинка обробляє монохром.

```
public static void Monochrome(ref Bitmap image, int level) {
for (int y = 0; y< image.Height; y++){
for (int x = 0; x< image.Width; x++){
var color = image.GetPixel(x, y);
 int sr = (color.Q + color.W + color.E) / 3;
 image.SetPixel(x, y, (sr < level ? Color.Black : Color.White));}
}
}
```

Сегментація зображення після обробки відбувається в процесі

розпізнавання. Знову ж таки, через те, що опущений етап виявлення - для процесу сегментації прийнята наступна евристика. Передбачається, що пропозиції тексту горизонтально розташовані тому вони не створюють пересічень один з одним. Завдання сегментації тоді труднощів не складає.

Задається середнє значення відстані між двома літерами в слові. Потім зображення ділиться на рядки шляхом пошуку слова передаються на заключний етап і вони діляться повних білих смуг. Далі ці смуги діляться на слова шляхом пошуку білих смуг певної ширини. Після всього цього виділені на літери. Таким чином на виході модуля сегментації отримується весь текст представлений зображеннями букв цього тексту (рисунок 1.13).

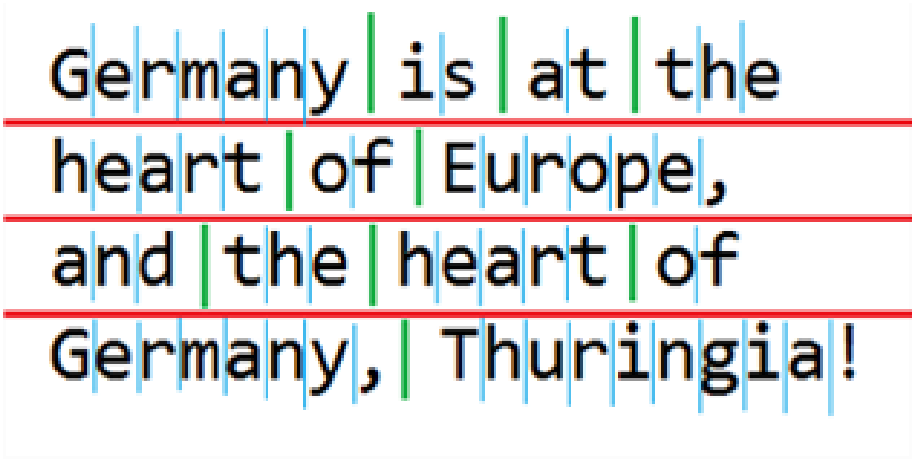


Рисунок 1.13 – Приклад зображення для сегментації

До розмірів шаблонів підготовлених зображення приводиться заздалегідь та нормалізується перед розпізнаванням. Наступним є сам процес розпізнавання. Два вибори доступні для користувача, за допомогою нейронної мережі та за допомогою метрик.

Метрика — визначає в просторі положення об'єкта деяке умовне значення функції, яке. Отже, якщо два об'єкти один від одного розташовані близько, тобто схожі (наприклад, дві букви А написані різним шрифтом), то метрики для таких об'єктів збігатися будуть або бути гранично схожими. Для розпізнавання в цьому режимі була обрана метрика Хеммінга[8].

Метрика Хеммінга — це метрика яка показує, не схожі об'єкти на скільки

сильно між собою. Таку метрику можна побачити при інформації кодуванні та даних передачі. Для прикладу, після сеансу передачі на виході є наступна послідовність біт (1001001), також нам відомо, послідовність біт що повинна прийти інша (1000101). Обчислюється метрика відповідними місцями з іншої послідовності шляхом порівняння частин послідовності з. Таким чином метрика Хеммінга в даному випадку дорівнює два. Так як об'єкти відрізняються у двох позиціях. 2 — це ступінь несхожості, чим більше, тим гірше в даному випадку.

Отже потрібно її метрику знайти з усіма шаблонами готовими, щоб визначити, зображена яка букви. І той шаблон, чия до 0 буде відповіддю метрика виявиться найбільш близькою.

Але як показала практика підрахунок однієї лише метрики не дає позитивного результату, так багато літери схожі між собою, наприклад “j” “i”, що призводить до помилкового розпізнавання.

Тоді було рішення прийнято що дозволяють розмежувати в окремий клас деякий безліч букв придумати нові метрики,. Зокрема, були реалізовані метрики , горизонтального переважання ваги і вертикального, відбитки горизонтального і вертикального).

Після проведення експерименту було з'ясовано, що такі літери як “H”, “I”, “i”, “O”, “o”,“X”,“x”,“1” мають суперсиметрією (повністю збігаються зі своїми відображеннями і значущі пікселі розподілені рівномірно по всьому зображенню), тому, що скорочує перебір всіх вони були винесені в окремий клас метрик приблизно в 6 разів. Такі ж дії були проведені відносно інших букв. У середньому зменшення перебору досягає приблизно 3 рази[6].

Також є така як «J» унікальна буква, яка знаходиться в класі своєму одна, і ідентифікуються значить однозначно. Далі, для кожного класу вираховується метрика Хеммінга, яка на даному етапі дає кращі показники ніж при прямому застосуванні.

При створенні шаблонів шрифт “consolas” використовувався, тому точність розпізнавання має порядку 99 відсотків, якщо розпізнаваний текст написаний цим шрифтом,. При зміні шрифту, падає до 70 відсотків точність (рисунок 1.14).

Рисунок 1.14 – Приклад зниження точності зображення

Наступний спосіб розпізнавання — за допомогою нейронної мережі. Сказати в математичному сенсі нейронна можна те, що мережа — це лише модель біологічного визначення.

Принцип роботи мережі нейронної такий, що повчивши нейронів на вхідний шар нове зображення мережа реагує імпульсом того чи іншого нейрона. Так термінологію мереж можна сказати як всі нейрони пойменовані значеннями букв, отже, зреагувавши, нейрон несе відповідь розпізнавання. Заглиблюючись в, що нейрон крім виходу має також безліч входів. Дані входи описують значення пікселя зображення. Тобто, якщо є16х16 зображення, входів має бути 256 у мережі.

Кожен вхід сприймається з певним коефіцієнтом і в результаті, на кожному нейроні скупчується певний заряд по закінченню розпізнавання, ніж заряд буде більше той нейрон і випустить імпульс[7].

Але щоб були налаштовані правильно необхідно коефіцієнти входів навчити мережу спочатку. Цим окремий модуль навчання займається. Даний модуль бере чергове зображення з навчальної вибірки і згодовує мережі. Мережа вирівнює коефіцієнти мінімізуючи помилку збіги аналізує всі позиції чорних пікселів і методом градієнта, після чого певного нейрона зіставляється дане зображення. Процес навчання проходить наступним чином:

```
public void Teach(Bitmap img, Neuron correctNeuron){
var vector = GetVector(img);
for (int i = 0; i < vector.Length; i++)
{
vector[i] *= 10;
correctNeuron.Weigths[i] = correctNeuron.Weigths[i] + 0.5 * (vector[i] -
```

```
correctNeuron.Weigths[i]);
        }
        }
```

По закінченню кожен нейрон схожий на полотно навчання, де на місцях, в яких найчастіше зустрічалися чорні пікселі найбільш темна фарба (значення заряду більше), а там де рідше - зовсім світлий тон.

Всі вирівняні і готові сприймати зображення коефіцієнти. Точність розпізнавання при цьому методі досягає 80 відсотків. Слід зауважити залежить від навчальної вибірки, що точність розпізнавання, як від кількості, так і від якості.

Алгоритми комп’ютерної побудови системи розпізнавання образів з використанням нейромережних паралельних обчислень. На сьогоднішній день досить багато потужних програм існує по розпізнаванню символів, але слід зазначити, що текст низької якості дотепер перевершує здатність людини читати друкований здатності комп'ютера. Кожен текст друкований має первинну властивість — шрифти, якими він надрукований. Виходячи з цього, існують два класи алгоритмів друкованих символів розпізнавання: шрифтовий та безшрифтовий. Шрифтові або шрифтозалежні що повинна бути надана повноцінна вибірка тексту алгоритми використовують апріорну інформацію про шрифт, яким надруковано букви. Це означає, програмі, надрукованого даним шрифтом. Програма вимірює й аналізує характеристики різного шрифту й заносить процесу шрифтова програма оптичного їх у свою базу еталонних характеристик. По закінченні цього розпізнавання символів готова до розпізнавання конкретного даного шрифту[8].

Недоліки цього підходу наступні:

— алгоритм повинен заздалегідь шрифт знати, що повинен зберігати в базі різні характеристики йому представляють для розпізнавання, тобто він цього шрифту;

— якість розпізнавання тексту, характеристик цього шрифту зі шрифтами надрукованого довільним шрифтом, буде прямо пропорційна кореляції, наявними в базі програми.

Дані фактори обмежують універсальність таких алгоритмів.

Необхідний блок настроювання на конкретний шрифт для роботи програми розпізнавання. Зрозуміло, що вносити свою частку помилок цей блок буде в інтегральну оцінку якості розпізнавання, або функцію встановлення шрифту доведеться покласти на користувача.

Програма, заснована на алгоритмі шрифтовому розпізнавання символів, вимагає від користувача спеціальних знань про шрифти паперовий документ самим користувачем взагалі, про їхні групи й відмінності один від одного, про шрифти, якими надруковано документ користувача. У випадку якщо не створений, а прийшов до нього ззовні, не існує загального способу довідатися, з використанням шрифтів яких цей документ був надрукований[9].

З іншого боку, у шрифтового підходу є перевага, завдяки якій його активно використовують і, очевидно, будуть використовувати в майбутньому. А саме, на відміну від безшрифтового, надійність розпізнавання символу є інтуїтивно ясною й математично вираженою величиною маючи детальну апріорну інформацію про символи, можна побудувати досить точні й надійні алгоритми розпізнавання. Взагалі, при побудові шрифтового алгоритму розпізнавання,. Ця величина визначається у якому-небудь метричному просторі від еталонного символу як відстань, пред'явленого програмі в процесі навчання, до символу, що програма намагається розпізнати.

Другий клас алгоритмів — безшрифтові або шрифтонезалежні, тобто алгоритми, вимірюють й аналізують різні характеристики (ознаки), що властиві буквам безвідносного шрифту що не мають апріорних знань про символи, що надходять до них на вхід. Ці алгоритми й абсолютного розміру, яким вони надруковані. У граничному випадку для шрифтонезалежного алгоритму процес навчання може бути відсутнім. У цьому випадку характеристики символів вимірює, кодує й поміщає в базу програми сама людина. Однак, коли такий шлях вичерпно вирішує поставлене завдання. Більш загальний шлях створення характеристик рідко на практиці зустрічаються випадки, бази полягає в навчанні програми реальних символів на вибірці [10].

Недоліком такого підходу є нижча якість розпізнавання узагальнення характеристик при вимірах символів більший набагато, ніж у шрифтових алгоритмів. Це пов'язане з тим, що рівень, ніж у випадку шрифтозалежних алгоритмів. Фактично це означає, що різні допуски й спрощення при вимірах характеристик символів для роботи безшрифтових алгоритмів можуть бути в 2-20 разів більші в порівнянні зі шрифтовими.

Переваги цього підходу тісно пов'язані з його недоліками. Основними перевагами можна відзначити: універсальність та практичність.

Це означає, з одного боку, можливість застосування цього підходу у випадках великої різноманітності символів, які можуть екстраполювати знання накопичені за межі навчальної вибірки надійти на вхід системи; з іншого боку, за рахунок закладеної в них здатності узагальнювати, такі алгоритми можуть, тобто стійко розпізнавати символи, на вигляд далекі від тих, які були в навчальній вибірці присутні. Процес навчання шрифтонезалежних алгоритмів звичай но є більше простим й інтегрованим у тому розумінні, що навчальна вибірка не фрагментована на різні класи. При цьому відсутня необхідність умови спільного існування цих класів підтримувати в базі характеристик різні (некорельованість, незмішуваність, систему унікального іменування й т.п.). Проявом технологічності також є той факт, що часто вдається майже повністю створити автоматизовані процедури навчання.

Зручність у процесі використання програми. У випадку, якщо програма побудована на шрифтонезалежних алгоритмах, інтерфейс користувача програми за рахунок відсутності набору опцій і діалогів зобов'язаний знати що-небудь про сторінку, яку він хоче ввести в комп'ютерну пам'ять і повідомляти програму про ці знання. Також спрощується, що обслуговують навчання й керування базою характеристик. У цьому випадку можна представляти користувачеві процес розпізнавання як “чорний ящик” (при цьому користувач повністю не має змоги керувати, або якимось чином модифікувати хід процесу розпізнавання). У підсумку це приводить до розширення кола потенційних користувачів за рахунок, що наділенікомп'ютерною мінімальною включення людей в нього, грамотністю.

При розпізнаванні символів використовуються досить часто штучні нейронні

мережі. Алгоритми, що використовують нейронні мережі для розпізнавання символів, часто будуються в такий спосіб. Зображення приводиться до деякого стандартного розміру символу (растр), що є вхідним для розпізнавання,. Як правило, використається растр розміром 16х16 пікселів.

Значення яскравості у вузлах нормалізованого растра використовуються як вхідні параметри мережі нейронної. Число вихідних параметрів нейронної мережі дорівнює числу розпізнаваних символів. Результатом розпізнавання є символ, або з пошуком більш інформативних вхідних ознак, якому відповідає найбільше зі значень вихідного вектора нейронної мережі. Підвищення надійності таких алгоритмів пов'язано, як правило, або з ускладненням структури нейронної мережі.

Надійність розпізнавання й потреба програми в обчислювальних ресурсах залежать багато в чому від вибору структури й параметрів нейронної мережі. Нижні шари мережі не є пов‘язаними. Зображення цифр приводяться до єдиного розміру (16х16 пікселів). Отримане зображення нейронної мережі подається на вхід, що має три внутрішніх рівні й 10 вузлів у верхньому рівні. Вузли нижчого рівня спільно використовують загальний набір ваг. Все це, за задумом розробників, повинне підвищити здатність нижчих рівнів мережі до виділення первинних ознак у зображеннях. Отримана нейронна мережа в такий спосіб має 1256 вузлів й 9760 незалежних параметрів. Для збільшення і пам'яті проводиться видалення мало використовуваних ваг здатності мережі до узагальнення й зменшення обсягу необхідних обчислень. У результаті число незалежних параметрів зменшується в чотири рази. Навчання нейронної мережі проведено на наборі з 7300 символів,

тестування на наборі з 2000 символів. Помилки розпізнавання становлять приблизно 1% на навчальному наборі й 5% на перевірочному [11].

Як вхідні параметри нейронної мережі, замість значень яскравості у вузлах нормалізованого растра можуть використовуватися значення, що характеризують перепад яскравості. Такі вхідні краще виділяти межі букви параметри дозволяють. Об‘єкти розпізнавання приводяться до розміру 16х16 пікселів. Після цього метою виділення ділянок з найбільшими вони піддаються додатковій обробці з перепадами в яскравості.

Одним із широко використовуваних підвищення декількох різних розпізнавальних модулів точності розпізнавання методів є одночасне використання і наступне об'єднання отриманих результатів (наприклад, шляхом голосування). При цьому дуже важливо, щоб алгоритми, використовувані цими модулями, були як можна більше незалежні. Це може досягатися, що використовують принципово різні алгоритми розпізнавання, як за рахунок використання розпізнавальних модулів, так і спеціальним підбором навчальних даних.

Один з таких методів був запропонований кілька років тому і заснований на використанні трьох розпізнавальних модулів (машин) [12]. Перша машина навчається звичайним чином. Друга машина чином, що друга машина бачить суміш символів навчається на символах, які були відфільтровані першою машиною таким, 50% з яких були розпізнані першою машиною вірно й 50% невірно. Нарешті, третя машина навчається на символах, на яких результати розпізнавання 1-ої й 2-ий машин різні. При тестуванні на вхід всім трьом машинам. Оцінки, одержувані розпізнавані символи подаються на виході всіх трьох машин складаються. Символ, що одержав найбільшу сумарну оцінку видається як результат розпізнавання[13].

Як правило, алгоритм розпізнавання заснований на виділенні з растра із зображенням букви первинних ознак і наступному використанні штучної нейронної мережі для оцінки близькості вхідного зображення із символами із символами із заданого набору символів заданого набору букв. Результатом роботи є набір оцінок, що відбивають ступінь близькості розпізнаваного символу із. Набір розпізнаваних символів може включати букви й цифри. Вхідні матеріали для розпізнавання зображення символів перетворюються до єдиного розміру.

Відмінною рисою реалізованого алгоритму є використання нейронної мережі з досить великою кількістю вхідних ознак. На вихідному зображенні виділяються первинні ознаки, що характеризують перепади яскравості у вузлах растра. Нейронна мережа має один внутрішній внутрішнього рівня з'єднаний з усіма вхідними рівень, що містить 100 вузлів і є загальнопов‘язаною, тобто кожен вузол вузлами, а кожен вузол верхнього рівня з'єднаний з усіма вузлами внутрішнього рівня. Для зменшення обсягу обчислень при розпізнаванні для

кожного розпізнаваного зображення символу використовуються не всі вхідні ознаки, а тільки частина, іншими словами вектор вхідних параметрів нейронної мережі є сильно розрідженим [14].

Навчання нейронної мережі відбувається звичайним чином, тобто використовується алгоритм зворотнього поширення помилки. Програма навчання одержує на вхід файл із зображеннями символів. При навчанні символи із мережі при навчанні виконується після кожного символу цієї бази перебираються циклічно. Для кожного зображення з бази виділяються первинні ознаки, після чого виконуються прямий і зворотний проходи по мережі. Модифікація ваг. Крок зміни ваг мережі постійний. Для прискорення й покращення навчання погано розпізнавані символи проглядаються частіше за інші. У цьому випадку використовується кеш, у якому зберігаються важко розпізнавані зображення. Растри для розпізнавання, тобто погано розпізнавані символи навчання вибираються як із вхідного файлу, так і з кешу. Вибір символу з кешу відбувається з урахуванням якості його вибираються частіше. Крім того, при навчанні мережі використовується регулярізація ваг мережі, тобто вводиться їхнє експонентне згасання[15].

Якість розпізнавання залежить не тільки від алгоритмів, що використовуються програмами розпізнавання й навчання нейронної мережі, але й від того, яким чином навчалася нейронна мережа. На якість навчання нейронної мережі бази з навчальними растрами, розмір, спосіб відбору впливають наступні фактори: параметри растрів, порядок растрів у базі, наявність брудних символів і помилок у розмітці [16].

Використання на різних етапах навчання різних оптимізуючих факторів можливе:

— відносна частота вибору растрів з навчальної бази даних і з кешу поганих символів;

— історія навчання мережі, розмір кешу поганих растрів;

— використання мережі регуляризації;

— зміна мережі коефіцієнтів;

— використання перекручувань символів і додаткового шуму;

— момент зупинки навчання. Бажано уникати як недостатнього навчання мережі, так і перенавчання.

Параметри навчання взаємозалежні й повинні вибиратися узгоджено. Так, наприклад, при невеликому розмірі навчальної бази використання пере кручувань символів може приводити до поліпшення якості навчання, з навчальними символами більша частина символів з бази а при збільшенні розміру бази приводить до його погіршення. Використання кешу поганих символів на самому початку навчання не має особливого сенсу. Навпаки, після декількох проходів по базі розпізнається з дуже великою надійністю. Зміна ваг мережі відбувається головним чином за рахунок растрів, що втримуються в кеші поганих символів [17].

Для визначення найкращого моменту зупинки мережі можна періодично тестувати якість розпізнавання на невеликій незалежній базі даних. Регуляризація дуже невеликого коефіцієнта згасання дозволяє підвищити стійкість (тобто введення експонентного згасання ваг при навчанні) приводить до деякого погіршення якості розпізнавання. Однак використання мережі без помітних втрат для розпізнавання. Порівняння якості різних алгоритмів розпізнавання символів ускладнене тим, що відносне значення числа правильно розпізнаних символів технологія навчання нейронної мережі, методика й алгоритми виділення первинних ознак, істотно залежить від конкретної бази даних, на якій проводиться тестування. На якість розпізнавання впливають також такі фактори: обсяг набору розпізнаваних символів, технологія підготовки навчальної бази даних й інші фактори.

Алгоритм шляхом пошуку більш адекватного подання структурних ознак розпізнаваних символів може бути вдосконалений. Збільшення пам'яті нейронної мережі і використання більшої навчальної бази даних також може дати поліпшення якості розпізнавання [18] .

Проектована система, близькому до реального часу, має працювати у режимі, а отже розроблюваний алгоритм має бути досить швидким і, в той же час, мати достатню точність розпізнання.

1.3 Огляд програми автоматичних систем розпізнавання символів

OCR CUNEIFORM — це безкоштовна програма сканування і розпізнавання тексту російського розробника Cognitive Technologies.

Спочатку OCR CuneiForm розроблявся як комерційний продукт, але, потім стала поширювати програму безкоштовно, а незабаром відкрила вихідний код програми. Ця OCR програма додається в комплекті з деякими моделями сканерів фірм Canon, Hewlett Packard, Oki, Olivetti. Технології розпізнавання компанії Cognitive використовуються в популярному видавничому пакеті Corel Draw.

OCR CuneiForm забезпечує швидке, зручне і якісне розпізнавання тексту із збереженням вихідного вигляду документа. Підтримується розпізнавання з більше 20 мов, серед яких російська, українська, англійська і т.д. На рисунку 1.15 представлені скріншоти даної програми.

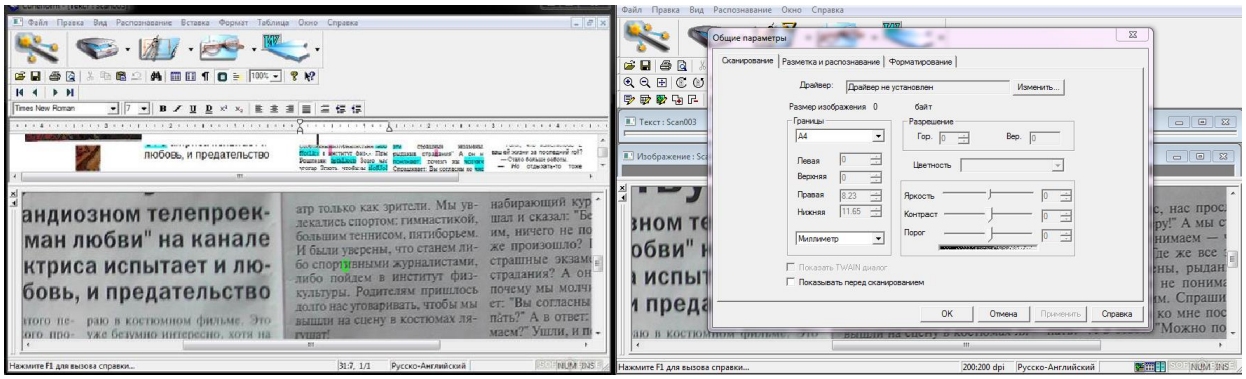


Рисунок 1.15 – Скріншоти програми OCR CuneiForm

Finereader — популярна програма розпізнавання тексту російської компанії ABBYY.

Finereader забезпечує якісне розпізнавання і збереження оформлення документів. Існують три версії цього пакету розпізнавання: Home Edition, Professional Edition і Corporate Edition, які відрізняються своїми можливостями, призначеним для користувача інтерфейсом, ціною і типом ліцензії.

Версія Home Edition призначена лише для домашнього використання і згодиться тим, кому час від часу потрібно отримати розпізнану копію сторінок

книги, підручника, статті з журналу для подальшого редагування в поширених офісних програмах. Інтерфейс програми спрощений, для роботи можна вибрати один з типових способів обробки зображення і натисненням однієї кнопки швидко отримати результат.

Professional і Corporate Edition мають професійний інтерфейс, додатково містять підтримку розпізнавання PDF файлів, вбудований редактор тексту, перевірку орфографії. Corporate версія орієнтована на використання в організаціях, підтримуються мережеві сканери і багатофункціональні пристрої, додані можливості для спільної роботи користувачів.

Програма робить розпізнавання тексту з більше 180 мов, для 38 з них передбачена вбудована перевірка орфографії. Починаючи з версії Professional, розпізнаються іврит, японська, тайська, китайська мови. Finereader відкриває файли графічних форматів (TIFF, JPEG, PFD, PNG і ін.) у тому числі DjVu — компактний формат для зберігання відсканованих документів, книг. На рисунку 1.16 наведено приклад використання даної програми.

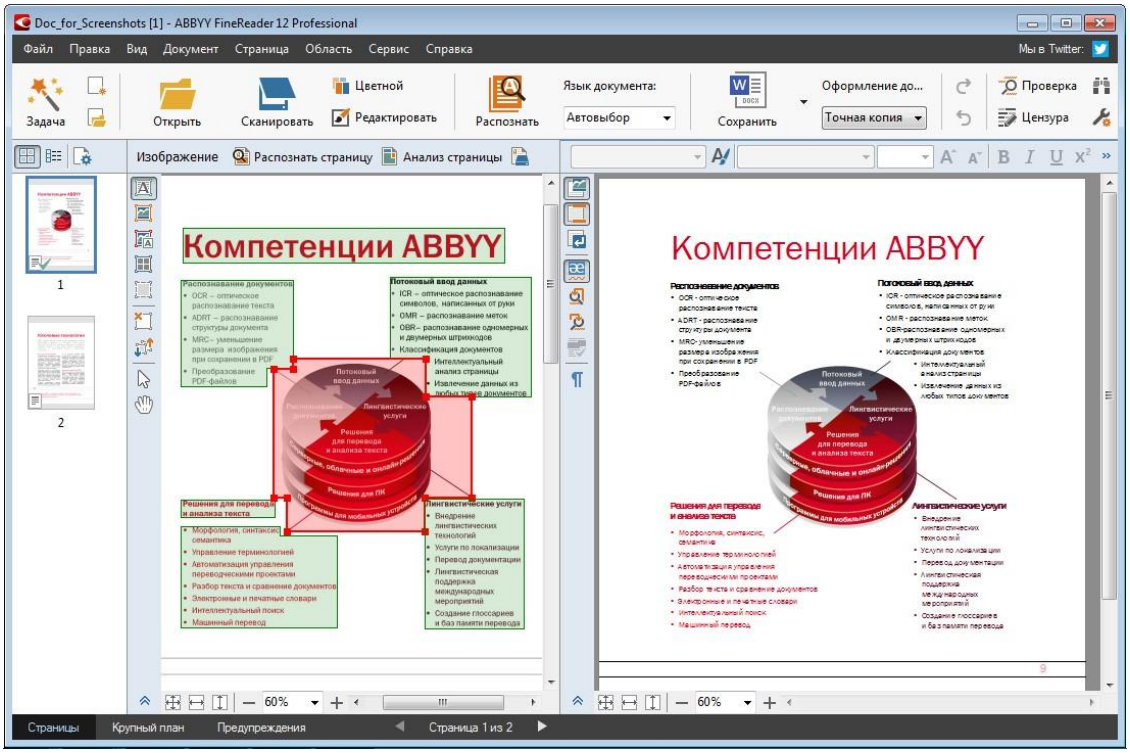


Рисунок 1.16 – Скріншот програми Finereader

Omnipage — програма сканування і розпізнавання тексту компанії Nuance

Communications.

Програма відрізняється високою точністю розпізнавання і швидкістю. Розпізнаються більше 110 мов з різними алфавітами: китайська, кирилиця, японська і корейська мови, та латинський, грецький алфавіти. Як і FineReader, OmniPage упевнено розпізнає документи, отримані за допомогою цифрових камер за допомогою технології корекції зображення «3D Correction».

У даній програмі є підтримка паралельної роботи з декількома документами: можна відкривати, розпізнавати, коректувати і зберігати декілька документів одночасно.

OmniPage випускається в трьох різних версіях: Standard, Professional, Enterprise. Версія Professional, на відміну від Standard, включає засіб для управління документами ParerPort, і програму PDF Create, що дозволяє створювати PDF документи. У версії Enterprise додані інтеграція з Microsoft SharePoint Server, додаткові мережеві функції.

Readiris — програма компанії I.R.I.S сканування і розпізнавання тексту.

Так само, як і інші програми розпізнавання тексту, Readiris перетворить відскановані зображення документів в редагований формат. Readiris упевнено розпізнає документи, що містять складну верстку, таблиці, ілюстрації.

Існують Pro і Corporate версії цього продукту, а також додаткові модулі розпізнання близькосхідних і східних мов. Версія Corporate відрізняється від Pro покращуваною роботою з PDF, підтримкою стискування вихідних файлів, розпізнаванням одного пакету в декілька файлів, індексуванням розпізнаних документів і іншими можливостями. У Corporate версії є зручний засіб для автоматичного розпізнавання файлів, що потрапляють в певну теку (Watched folder).

Підтримується розпізнавання тексту з більше 120 мов розпізнавання, включаючи російську, а також близькосхідні мови — арабську, іврит, фарсі (у версії Middle-East) і японську, китайську, корейську (у версії Asian). Є версія Readiris для Macintosh. Разом з підтримкою розпізнавання популярних форматів картинок, розпізнаються файли PDF і DjVu.

Microsoft Office Document Imaging — програма розпізнавання тексту компанії Microsoft.

Microsoft Office у складі має інструменти для сканування і розпізнавання тексту — MS Office Document Scanning та Document Imaging. Ця програма має високу міру інтеграції з іншими додатками MS Office, підтримуються засоби індексування документів. Document Imaging може відкривати файли MDI (власний формат Microsoft для стискування зображень) і TIFF, можлива передача результатів розпізнавання в Microsoft Word. На рисунку 1.17 наведено скріншот даної програми.

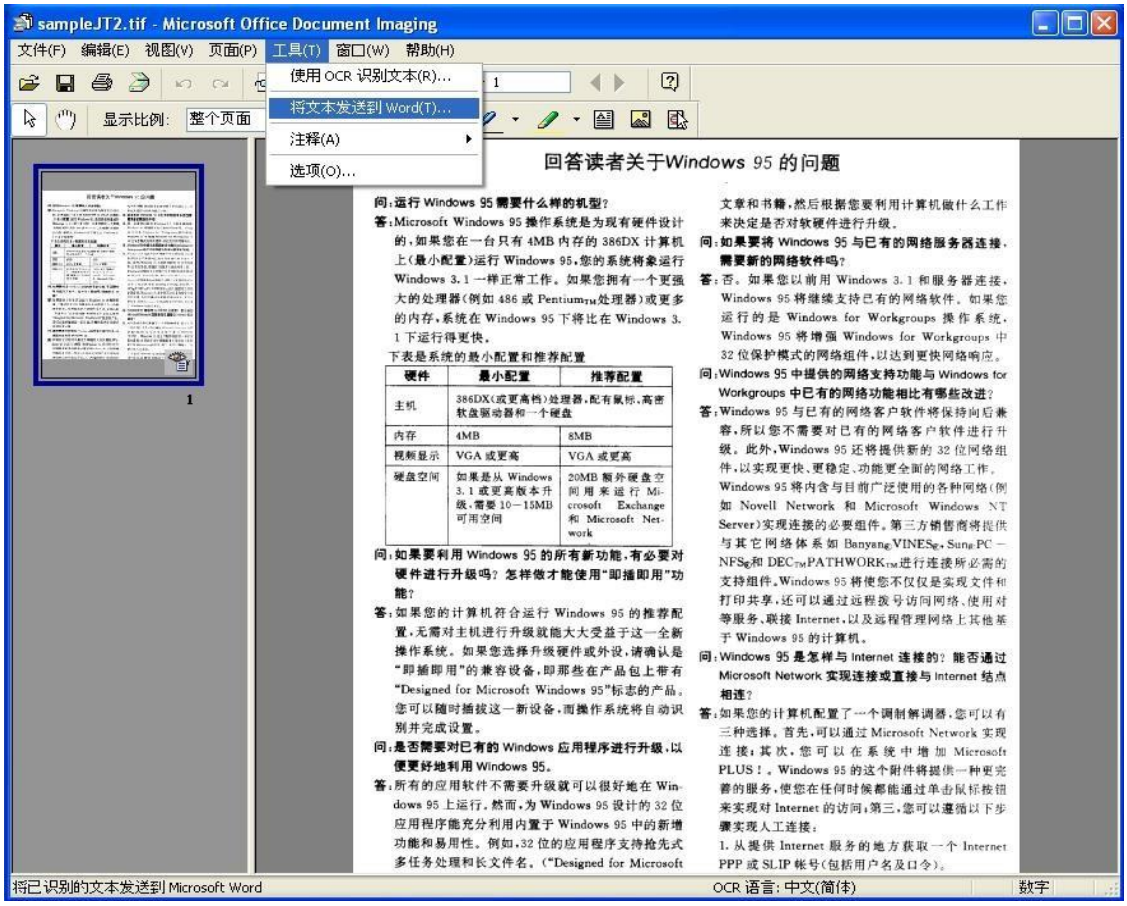


Рисунок 1.17 – Скріншот програми Microsoft Office Document Imaging

Document Imaging мрограма може працювати тільки з двома мовами: з мовою локалізації MS Office та англійською мовою. Потрібно додатково завантажити пакет Multilingual User Interface (MUI) для підтримки різних мов. В програмі OCR налаштувань майже немає, вона автоматично підтримує розпізнавання картинок,

шрифтів і простих таблиць. Суттєві проблеми, пов'язані з розпізнаванням символів. Існує декілька проблем пов’язаних з розпізнаванням рукописних і друкованих символів. Найбільш важливі з них наступні:

— різноманітність форм накреслення символів;

— спотворення зображень;

— варіації розмірів;

— варіації масштабу символів.

Кожен окремий знак може бути написаний різними стандартними шрифтами, наприклад (Orator, Gothic, Courier,Elite), спеціальними шрифтами, які використовуються в системах OCR, а також багатоінших шрифтів. Крім того, різні символи можуть володіти подібними обрисами. Наприклад, U і V, S і 5 raquo, G і 6, iZ 2 raquo. Спотворення форми: непропечатаність символів, розірваність рядків, ізольованість окремих точок, зміщення символів чи їх частин щодо місця розташування в рядку, неплоский характер інформаційного носія (для прикладу, ефект жолоблення);грубим дискретом оцифрування зображень; обертання з зміною нахилу символів. Радіометричні спотворення: тіні, дефекти освітлення, нерівномірний фон, відблиски, помилки при зйомці відеокамерою чи скануванні.

Також значним є й вплив вихідного масштабу друку. У прийнятій термінології масштаб 17, 12 або 10 означає, що в дюймі рядка поміщаються 17, 12 або 10 символів. При цьому, наприклад, символи масштабу 10 звичайно ширше та крупніше символу масштабу 12.

OCR система, окрім зазначених проблем, повинна виділяти на зображенні текстові області, розпізнавати їх та бути нечутливою до способу верстки і відстані між рядками, в них виділяти окремі символи.

Висновки до розділу 1

В даному розділі було розглянуто методи програмних засобів систем ідентифікації двомірних зображень. А саме:

— з'ясовано та порівняно основні недоліки та переваги OCR систем;

— проаналізовано методи розпізнавання реалізованих на основі нейромережевих структур та нечіткої логіки;

— розглянуто програми автоматичних систем розпізнавання символів(OCR CUNEIFORM, Finereader, Omnipage, Readiris).

Також було з'ясовано, що основною проблемою розглянутих OCR програм є велика чутливість до спотворення зображення(при втраті пікслів, чи надлишкових пікселів.)

2 РОЗРОБКА СТРУКТУРИТА АЛГОРИТМІВ МЕТОДУ ІДЕНТИФІКАЦІЇ ЗОБРАЖЕНЬ СИМВОЛІВ ЦИФР НА ОСНОВІ ОЦІНОК ЕНТРОПІЇ

20.1 Представлення зображень символів двомірними бінарними матрицями

В модернізованій машинній графіці використовуються десятки спеціалізованих форматів даних. Деякі з яких створені окремими фірмами під спеціальні програми, інші створені науково-дослідними установами, у більшій чи меншій мірі пов'язаними співпрацею з Міжнародною організацією стандартів. Проте у повсякденній практиці зустрічається всього лише декілька.

Один з простих формат — BMP (BitMaP, тобто бітова карта), що з'явився з першими версіями ОС Microsoft Windows. Аналогічним є формат ico для зображення у ОС Windows іконок — маленьких значків-логотипів програм. Так як зіскановане зображення може містити різні спотворення (рисунки 2.1 та 2.2) в даній роботі було обрано формат bmp, так як він є найпростіший для демонстраційних потреб.

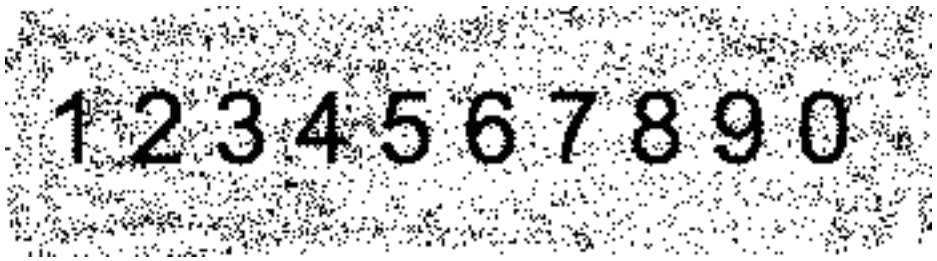


Рисунок 2.1 – Зображення із надлишковими спотвореннями

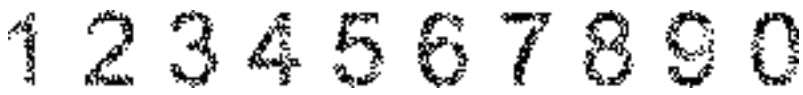


Рисунок 2.2 – Зображення із втратами частин символів

Формат файлу BMP може зберігати 2D цифрові зображення довільної ширини, висоти та роздільної здатності, як монохромні так і кольорові, різної глибини кольору, і, необов’язково, зі стисненням даних, керуванням кольору і альфа-каналом. Специфікація (WMF) Windows Metafile захоплює файл форматуBMP, wingdi.h визначає структури і константи-BMP.

BMP (Bitmap) — DIB (англ. Device independent bitmap)абобitmap-формат— формат файлу зображень растрової графіки, в якому зображення зберігається у вигляді двовимірного масиву пікселів. Запам’ятовує одно і багатокольорові (RGB) ілюстрації у формі Pixel. BMP-формат використовується в операційній системи Windows та OS/2. Дані цього формату включаються в двійкові файли ресурсів RES і в PE-файли.

На кожний піксель в різних файлах може приходити різна кількість біт (глибина кольору). Microsoft пропонує бітності 64, 48, 32, 24, 16, 8, 4, 2, 1. У такому форматі зберігати можна тільки одношарові растри. В бітності нижче 8 він вказується індексом з таблиці кольорів (палітри), а при великих: безпосереднім значенням. Колір в будь-якому випадку можна встановити тільки в колірній моделі RGB, але в бітності 32 та 16 можна отримати Grayscale з глибиною до 16 і 32-ох біт відповідно. Мізерна прозорість реалізована альфа-каналом різних бітностей, та при цьому прозорість без градацій можна отримати кодуванням-RLE.

Для бітності 4 і 8 доступно RLE-кодування, яке може зменшити їх розмір. Формат BMP так само підтримує вбудовування даних у форматах PNG і JPEG. Зазвичай пікселі зберігаються у вигляді відносно простого двовимірного масиву. Але останнє скоріше більше призначене для обходу обмежень архітектури GDI, а не для компактного зберігання, яка не передбачає пряму роботу із зображеннями форматів відмінних від BMP.

В нових версіях BMP формату теж з'явилися можливості за допомогою яких можна керувати кольором. Також, можна виробляти гамма-корекцію, вбудовувати колірні профілі ICC та вказувати кінцеві точки.

Офіційну інформацію BMP формату можна відшукати в довідці Microsoft Windows SDK (може йти в комплекті з IDE) чи в MSDN. У файлі під назвою

wingdi.h, що від компанії Microsoft можна побачити всі оголошення мовою C++, які відносяться до даного формату. Сюди не були включені оголошення типів, так як від цього він може бути досить великим. При чому деякі розробники можуть вважати незручними офіційні оголошення і тому їх необхідність сумнівна.

Найбільший розмір неподільних комірок (виключаючи поля бітових структур): 32 біта і тому формат можна вважати як 32-бітний.Цілі числа записуються в прямому коді, зі знаком — в доповняльному. Якщо порівнювати з апаратними архітектурами, то порядок байт і формат чисел відповідає архітектурі x86. Винятком можуть бути 64-бітові пікселі, але значення їх каналів можна обробляти і 16-бітними словами. Порядок байтів в 16-бітних і 32-бітових комірках усюди від меншого до більшого.

Можна зустріти такі числові типи:

— 32-бітове ціле зі знаком — LONG;

— 8-бітове ціле беззнаковое — BYTE;

— 32-бітове ціле беззнаковое — DWORD;

— 16-бітове ціле беззнаковое — WORD.

У деяких елементів формату вказана версія Windows, починаючи з якої він підтримується. Мова йде в першу чергу про основні бібліотеки WinAPI такі як gdi32.dll, shell32.dll, user32.dll і kernel32.dll. Інші компоненти операційної системи (наприклад, GDI+, .NET, DirectX) можуть мати інші більш широкі можливості.

У форматі Windows Bitmap під структурами розуміється блок з послідовними комірками різного фіксованого розміру, у яких є умовні імена (є в багатьох мовах програмування), а не щось складніше (наприклад, потік команд довільного розміру).

На рисунку 2.3 зображено структуру файлу-BMP, він включає в себе чотири частини:

— заголовок зображення (BITMAP_INFO_HEADER, може бути відсутнім). BITMAP_V4_HEADER (Win_95, NT4.0) BITMAP_V5_HEADER (Win98/Me, 2000/XP);

— саме зображення;

— заголовок файлу (BITMAP_FILE_HEADER);

— палітра.

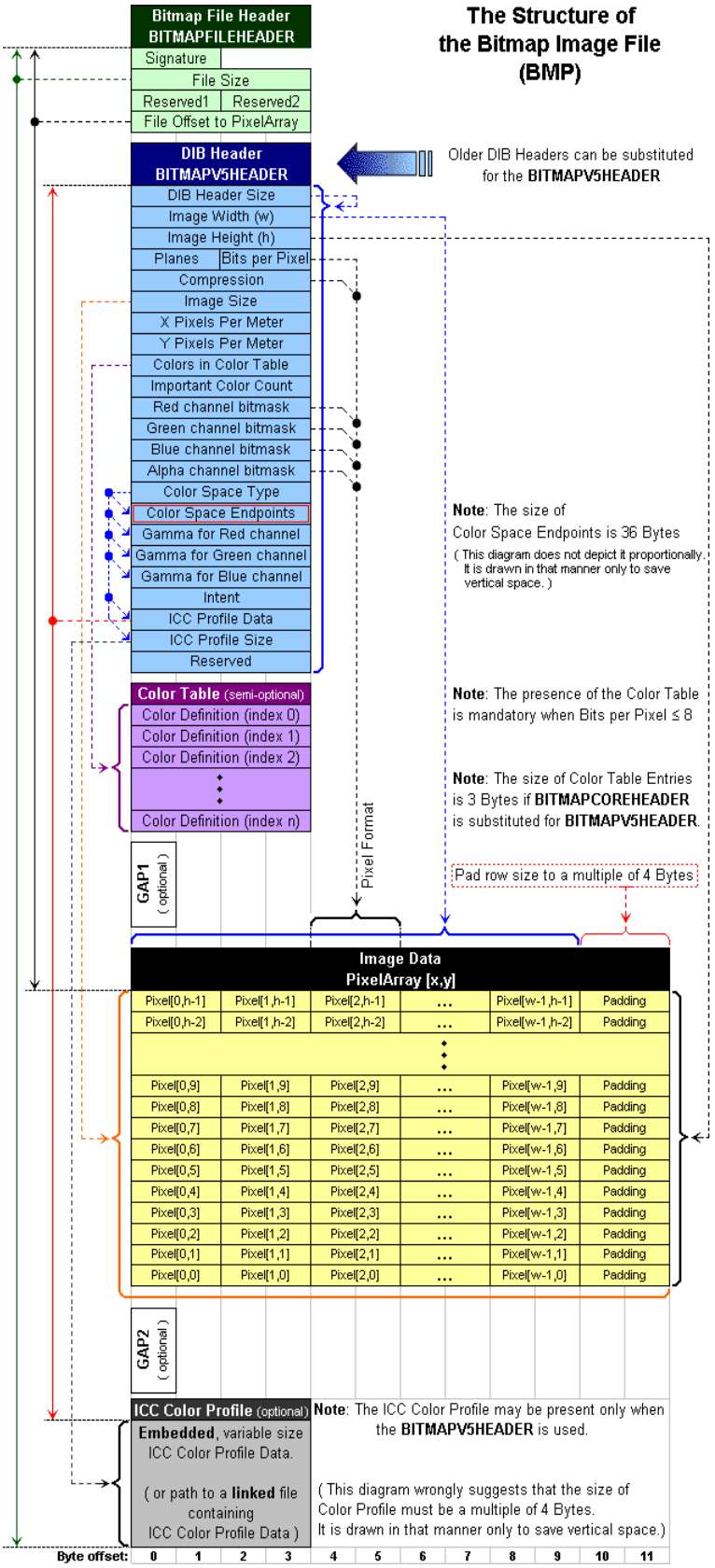


Рисунок 2.3 – Структура BMP файлу

BITMAP_FILE_HEADER — це 14-байтна структура (таблиця 2.1), що розташована на початку файлу. Можна побачити, що на початку структури збивається вирівнювання комірок. Якщо це важливе, то в ОЗУ даний заголовок розташовуйте по парними адресами, які не кратні 4 (тоді 32-бітові комірки потраплять на вирівняні позиції).

Таблиця 2.1 – 14-ти байтна структура BITMAPFILEHEADER

Поз. (hex)	Розмір (байт)	Ім'я	Тип WinAPI	Опис
0	2	bfType	WORD	Позначення формату для відмінності від інших (сигнатура формату). Може містити єдине значення $4D42_{16}$/$424D_{16}$ (little-endian/big-endian), що відповідає значенню BM в стандарті ASCII.
2	4	bfSize	DWORD	Розмір файлу в байтах.
6	2	bfReserved	WORD	Зарезервовані і повинні містити нуль.
6	2	bfReserved	WORD	
A	4	bfOffBits	DWORD	Положення піксельних даних відносно початку даної структури (в байтах)).

При перегляд і сигнатура формату вмісту файлу текстом в двійковому режимі виглядає як пара ASCII-символів “BM”.

BITMAP_INFO у файлі йде відразу за BITMAP_FILE_HEADER. Дана структура є основною і описовою у форматі BMP і тому коли просто згадано ім'я поля, то мова йде про поле в даній структурі. Адреса даного блоку безпосередньо в пам'яті так само передається деяким функціям WinAPI (наприклад, SetDIBitsToDevice або CreateDIBitmap). Крім того, цей же блок використовується у форматах курсорів та значків Windows, але в даній статті цей момент не розглядається (див. окремі описи цих форматів).

Блок BITMAP_INFO складається з таких частин:

— структура з інформативними полями;

— сіткові маски для вилучення значень колірних каналів;

— таблиця кольорів (присутня не завжди).

Через ідентичності полів у версіях 3, 4 і 5 здається, що можна регулювати кількість полів за допомогою поля Size, видаляючи невикористовувані. Насправді це не так, тому, що тут розмір грає роль версії (у версії CORE хоч і теж ідентичні поля, але іншого розміру і типу). Ніхто не може гарантувати, що не можуть попастися заголовки менших або більших розмірів з іншим набором полів. Проте, Adobe Photoshop може при збереженні файлів BMP записувати структури інформаційних полів з розмірами 52 і 56 байт. По суті це урізана 4-та версія, яка містить тільки бітові маски каналів (56 байт — версія з альфа каналом).

16-бітові інформаційні поля (версія CORE). Варто звернути увагу на те, що тут поля ширини і висоти містять беззнакові цілі, у той час як 32-бітові структури зберігають значення зі знаком.

Для обробки зображення необхідно представити його у вигляді двовимірної бінарної матриці, це надасть можливість легше обробляти його і опрацьовувати різними методами в подальшому. Так як зображення є монохромними, матриця буде представлена у вигляді набору одиниць і нулів.

Насамперед необхідно розбити зображення на пікселі, для цього використаються класи мови програмування Ruby ImageList та Image. В даній роботі кожний колір представлений у форматі RGB (RED, GREEN, BLUE відповідно червоний, зелений, блакитний). Так як зображення є монохромним отримується тільки два кольори – чорний і білий. Враховуючи той момент, коли зображення не є монохромним, а містить і інші відтінки, відмінні від білого і чорного кольорів, ставиться умова для формування матриці. Темніші кольора вважатимуться одиницею, світліші – нулем.

Програма, написана на фреймворку Ruby On Rails, надає повний пакет інструментів для роботи із зображеннями. Серед найвідоміших пакетів (gem) для роботи із зображеннями на даний момент є Paperclip та Carrierwave. В даній роботі використовується paperclip, так як він є більш зручнішим та гнучкішим для даного завдання. Насамперед необхідно сформувати шаблони матриць для симовлів (цифр). Так як буде виключно розпізнавання цифр, буде

підготовленно шаблони для 0 – 9. Для цього буде підготовлено зображення кожного символу без жодного спотворення та ідеальних параметрів та розмірів. В даній роботі використається шрифт 14-го розміру та текст формату Calibri (Body).

Для створення самих шаблонів буде відскановано кожен символ окремо та переведено його в двовимірну матрицю нулів та одиниць. При виявленні будь яких спотворень або відхилень, всі поправки вводяться вручну (заміна 0 на 1 або ж навпаки). В даній програмі всі шаблони символів зберігаються в базу даних з розширенням типу text. При кожному виклику значення буде переведено в матрицю. Це спростить роботу, так як зберігати дані типу Array можливості немає. В даній роботі використовується база даних MySQL. Для її швидкого підключення і налаштування використається gem mysql2. Значну частину роботи виконає даний пакет. Залишається лиш налаштувати файл database.yml, який міститиме параметри доступу до бази даних для подальшої роботи.

Сам процес сканування відбувається наступним чином. Підготувавши тестову форму для завантаження зображення (відсканованого тексту), виводиться поле типу file для завантаження зображення. Добавляється кнопка submit для відправки самої форми з даними. В даному випакду використовується запит типу POST, так як створюється новий запис в базі даних. Після відправлення, форма потрапляє на вказану url адресу. В даному випадку за допомогою RoR, форма переходить до екшену create контролера через який зберігаються дані. Значну частину роботи виконає сам фреймворк. Зі сторони розробника доведеться провести лише конфігурацію. Пройшовши по горизноталі формується почергово кожен новий рядок (масив) з одиниць і нулів. Таких рядків на одне зображення в даному випадку (підібрано шрифт і розмір тексту) припадає порядку 100 символів (висота зображення). Після того як матриця є сформованою, необхідно зберегти її до бази даних. В цьому також допоможе фреймворк. Він надає повний пакет інструментів для створення та редагування баз даних та наповнення їх інформацією. Процес відбувається

наступним чином. Насамперед створюється база даних. Після цього створюються необхідні таблиці з полями. Для роботи з даними RoR надає свої інструменти.

Важливими компонентами застосунків Ruby on Rails є контролер (controller), вид (view) і модель (model). Модель надає решті компонентів програми об'єктно-орієнтоване представлення даних (таких як каталог продуктів або список замовлень). Об'єкти моделі здійснюють збереження даних та завантаження в реляційній базі даних. Завдяки динамічної типізації можливостям в мові Ruby розробникові досить успадкувати свій клас моделі від базового класу ActiveRecord::Base. RoRстворює атрибути об'єктів для відповідних полів таблиці і пов'язує класи моделі автоматично з таблицями в базі даних. Вся робота проходитиме виключно через модель, для цього створюється модель з певною назвою, яка буде надалі співпрацювати з таблицею яка підходить по назві. Для прикладу модель з назвою Blog буде звертатися до таблиці з назвою blogs. Для того щоб підключити для моделі іншу таблицю необхідно виконати деякі мінімальні зміни, але в даному випадку буде використовуватися стандартний підхід. В даній роботі створюється модель з назвою number куди і зберігатимуться усі матриці для шаблонних методів. Для зберігання матриць необхідний певний тип поля. Для цього використовується тип поля text, дані з якого в подальшому будуть переводитись в матриці за допомогою певних методів. Тип поля text дозволяє зберігати досить великий об'єм даних, який цілком вистачить для зберігання даних матриці. У фреймворку RoR доступні методи для переведення тексту у масив і його обробки.

Наприклад:

```
class Number < ActiveRecord::Base
  def text_to_array
    self.split(',')
  end
end
```

В даному випадку є клас з назвою Number який наслідується від класу ActiveRecord::Base. Дане наслідування дозволятиме використовувати різні методи які включені у даний клас. В середині класу знаходиться метод з назвою text_to_array, який виконує функцію розбивання тексту на елементи масиву за допомогою метода split(). Для прикладу якщо буде звернення об'єкта до даного методу з текстом, відбудеться його розбивка на елементи масиву.

Ruby on Rails надає дуже корисний інструмент для роботи з базами даних. CRUD — (англ. createreadupdatedelete) 4 базові функції управління даними “створення, зчитування, зміна і видалення”.

Термін “CRUD” також вживають стосовно інтерфейсу користувача. Для прикладу, в програмі адресної книги, базовий об'єкт — це запис з контактними даними.

Як мінімум, програма повинна надавати користувачу функції для:

— додання записів;

— пошук і зчитування записів;

— редагування записів;

— видалення існуючих записів.

Без цих базових операцій програма не може вважатися придатною до користування. Так як деякі спотворення зображень уникнути повністю не є можливим, в подальшому можна буде вручну виправити згенеровані матриці, відредагувавши їх через командну консоль або ж іншими методами.

Так як це не є основною задачею, даний пункт можна буде пропустити не писавши програмного коду для коректування матриці, так як в даному проекті розглядаються тільки символи цифр. На рисунку 2.4 приведена блок схема процесу отримання еталонних матриць символів.

Для того щоб перевірити генерування матриці необхідно використати інтерфейс користувача — засіб зручної взаємодії користувача з інформаційною системою. У графічних системах інтерфейс користувача реалізовується багатовіконним режимом, змінами кольору, розміру, видимості (прозорість,

напівпрозорість, невидимість) вікон, їхнім розташуванням, сортуванням елементів вікон тощо [12].

Рисунок 2.4 – Блок-схема створення еталонних матриць

Для цього буде використано інструмент фреймворка. Вид (view) створює інтерфейс користувача для відображення отриманих від контролера даних. У RoR вид описується за допомогою шаблонів RHTML. Вони є файлами HTML з додатковими включеннями фрагментів коду Ruby (EmbeddedRuby або ERb). Вивід, згенерований вбудованим кодом Ruby, включається в текст шаблону сторінки HTML, яка після цього повертається користувачеві. Види можуть використовувати фрагменти інших видів і, у свою чергу, бути включеними в шаблон (layout) вищого рівня.

Але самого виду і моделі буде недостатньо для того щоб відобразити сформовані матриці. Для цього необхідно використати інструмент фреймворка контроллер.Контролером в RoRє клас, успадкований від ActionController::Base.

Відкриті методи контролера є так званими діями (actions). Action часто відповідає окремому видові. Наприклад, по запиту користувача admin/list буде викликаний метод list класу AdminController і потім використаний вид list.rhtml.

Нижче наведено приклад контроллера, який використовується в проекті:

```
class StandartController < ApplicationController
before_action :get_standarts, only: :index
before_action :set_standart, only: :show
def index
end
def show
end
def get_standarts
@standarts = Number.all
end
def set_standart
@standart = Number.find(params[:id])
end
end
def get_standarts
def index
def show
end
end
```

Даний контролер містить два екшена index та show. Кожен з них відповідає за відображення певних даних. Зверху після назви контролера описується колбеки, які будуть викликатись перед екшенами. Перший з них надає список усіх шаблонів, другий – тільки той, який необхідний. Він буде ідентифікуватись по :id, тобто по записі в базі даних в таблиці numbers.

Для подальшої роботи відображення еталонів не є обхідним. Вони зберігатимуться в базі даних і будуть зчитуватись тільки тоді, коли буде відбуватись процес порівняння або інші методи розпізнавання, які потребують застосування даних шаблонів. Дані матриці необхідні для розпізнавання за еталонним методом [13].

20.2 Розробка методу опрацювання матриць символів за оцінками ентропії

Спосіб за імовірнісними характеристиками ідентифікації об’єктів їх 2Dшаблонів – поставлена задача розробки нового способу розпізнавання об’єктів шляхом використання значення однієї або декількох сумісно імовірнісних характеристик фрагментів зображення обчислених за та забезпечити покращення достовірності розпізнавання у випадках суттєвої зашумленості, еталонними шаблонами образів, саме це дозволяє полегшити програмну і алгоритмічну реалізацію. В якості таких характеристик використовують ентропію, розподіл ймовірностей станів, дисперсію, середнє квадратичне відхилення[14].

Вирішення поставленої задачі стає можливим завдяки тому, що при опрацюванні послідовності непохідних елементів, яка використовують значення однієї або декількох представляє розпізнаваний об’єкт, для порівняння сумісно імовірнісних характеристик фрагментів зображення обчислених за еталонними шаблонами образів.

Значення імовірнісної характеристики стаціонарних процесів фрагментів зображення прямує до постійної величини, що обчислюються за елементами шаблонних, яка залежить від характеристик зображення та таких фрагментів відбувається зміна результуючого значення наявних спотворень. В ході опрацювання відповідної імовірнісної характеристики, що використовується як ознака форми об’єкту.

В результаті запропоновано спосіб, у якому зображення обчислених за еталонними шаблонами образів, що забезпечує покращення достовірності ідентифікацію об’єктів здійснюють шляхом оцінювання значення імовірнісних характеристик фрагментів а також кількісних та якісних характеристик цифрових систем розпізнавання [15].

Суть винаходу пояснюється тим, що при опрацюванні послідовності непохідних елементів, яка представляє розпізнаваний об’єкт, використовують

значення однієї або декількох сумісно імовірнісних характеристик фрагментів зображення обчислених за еталонними шаблонами образів.

Винахід ілюструється зображено схему обчислення оцінки ентропії за еталонним шаблоном кресленнями, де на рисунку 2.5.

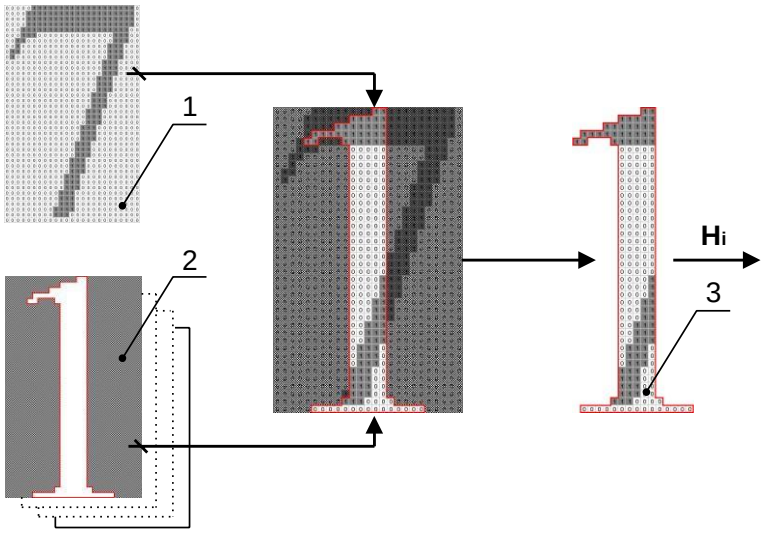


Рисунок 2.5 - Схема обчислення оцінки ентропії за еталонним шаблоном

Для кожного образу створюється окремий еталонний шаблон. Для ідентифікації (розпізнавання) об’єкту, за його зображенням 1 (зображення об’єкту, який необхідно ідентифікувати), (код) кольору використовують цифрові значення відповідних точок (пікселів), отриманих при оцифровуванні. При опрацюванні шаблон 2 (еталонні шаблони образів) визначає фрагмент зображення 3 (фрагмент зображення за елементами якого розраховується оцінка ентропії H_i), значення якої в подальшому використовується в якості за яким обчислюється оцінка ентропії, ознаки форми [16, 17].

У випадку ідентифікації одного об’єкту, наближення оцінки ентропії до нульового значення визначає ступінь його відповідності до задіяного еталонного шаблону образу об’єкту.

У випадку множинної ідентифікації об’єктів використовують набір еталонних шаблонів, за яким розраховується набір оцінок визначає приналежність отриманого зображення ентропії мінімальне значення якого, до відповідного образу об’єкту.

Описаний а також зменшення чутливості до рівномірного спотворення зображень, спосіб забезпечує спрощення алгоритмічної та програмної реалізації, можливості використання в малопродуктивних обчислювальних системах що зумовлена впливом навколишнього середовища при оцифровуванні.

За результатами проведених досліджень здійснено розроблення нового методу ідентифікації об’єктів (сигналів) шляхом використання характеристик при опрацюванні їх двомірних однієї або декількох сумісно імовірнісних цифрових представлень [18, 19]. У якості таких характеристик використано ентропію, розподіл ймовірностей станів, дисперсію, середнє квадратичне відхилення.

Тобто при побудові проекції формують відображення характеристик фрагментів цифрового представлення розташованих зображення у вектор, значення якого представляють як результат обчислення однієї або декількох сумісно імовірнісних вздовж визначених напрямків.

Множина усіх монохромних зображень розміру $n \times m$ пікселів знаходиться у бієктивній відповідності з множиною усіх бінарних матриць порядку $n \times m$, тобто є векторним простором розмірності $n \times m$ над скінченним полем. Означено відображення простору $\{0,1\}^n$ (простір впорядкованих наборів з “0” та “1” довжиною n) у відрізок $[0,1]$ (ентропії оцінок інформаційної таких наборів), тобто відображення

$$\hat{I}_n : \{0,1\}^n \to [0,1] \tag{2.1}$$

$$\hat{h}_n(i_1, i_2, ..., i_n) = P^{"0"} \cdot \log_2 P^{"0"} + P^{"1"} \cdot \log_2 P^{"1"}, \tag{2.2}$$

де $i_j \in \{0,1\}$,

$$P^{"1"} = \frac{1}{n} \sum_{j=1}^{n} i_j, \tag{2.3}$$

$$P^{"0"} = 1 - P^{"1"}. \quad (2.4)$$

Нехай бінарна матриця $X \in \{0,1\}^{n \times m}$ є представленням якогось об'єкту. Позначимо тепер таким чином відображення оцінок ентропії:

$$\hat{h}_h(X) = (\hat{h}_m(x_{11}, x_{12}, ..., x_{1m}), \hat{h}_m(x_{21}, x_{22}, ..., x_{2m}), ..., \hat{h}_m(x_{n1}, x_{n2}, ..., x_{nm})) \quad (2.5)$$

де x_{ij} – елемент матриці X або отриманої матриці з X незалежними перестановками елементів у її стовпцях.

$$\hat{h}_v(X) = (\hat{h}_n(x_{11}, x_{21}, ..., x_{n1}), \hat{h}_n(x_{12}, x_{22}, ..., x_{n2}), ..., \hat{h}_n(x_{1m}, x_{2m}, ..., x_{nm})) \quad (2.6)$$

де x_{ij} – елемент матриці X або отриманої матриці з X незалежними перестановками елементів у її рядках.

Індекси h та v у позначенні $\hat{h}$ відображення фактично вказують на те, що $\hat{h}_h$ – діє на рядки а $\hat{h}_v$ – діє на стовпці матиці X.

Таким чином кожному бінарному зображенню об'єкта (рисунок 2.6) в відповідність ставиться набір проекцій – оцінок векторів ентропії, що отримуються через застосування відображень $\hat{h}_h$ і $\hat{h}_v$ до матриці X та її заданих перестановок [20, 21].

На основі цих проекцій формують еталонні представлення для певної кількості k об'єктів $H^e : \{H^{e1}, H^{e2}, ..., H^{ek}\}$ з використанням яких, далі, розраховуються коефіцієнти парної кореляції з об'єктом, який ідентифікується.

Доцільно зазначити, що обов'язковим є виконання умови $|H^{ob}| = |H^{ei}|$ для $i = \overline{1,k}$.

Нехай $\hat{h}^{ob} \in H^{ob}$ і $\hat{h}^{e} \in H^{ei}$ відповідні вектори оцінок інформаційної ентропії об'єкта та відповідно одного з еталонів (рисунок 2.6).

В цьому випадку:

$$R_{xy}(\hat{h}^e, \hat{h}^{ob}) = \frac{1}{q}\sum_{j=1}^{q} \overset{o}{\hat{h}}^e[j] \cdot \overset{o}{\hat{h}}^{ob}[j], \qquad (2.7)$$

де q – розмірність векторів

$\hat{h}^e$ та $\hat{h}^{ob}$, $\overset{o}{\hat{h}}^e$ та $\overset{o}{\hat{h}}^{ob}$ – центровані вектори векторів $\hat{h}^e$ та $\hat{h}^{ob}$ відповідно.

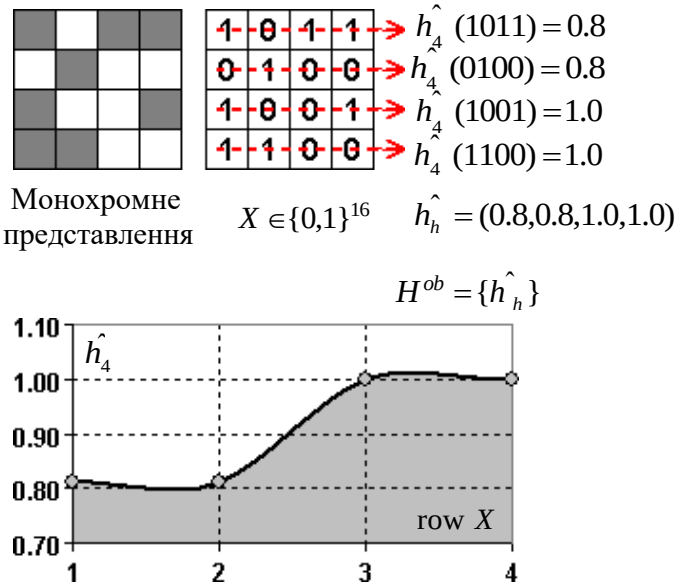


Рисунок 2.6 – Схема формування та графічного представлення вертикальної проекції $\hat{h}_h$ бінарної матриці X монохромного об'єкта зображення

В результаті формується коефіцієнтів кореляції задіяних матриця проекцій об'єкту H^{ob} та наявних еталонів H^e:

$$\begin{vmatrix} R_{xy}(\hat{h}^{e1}, \hat{h}^{ob}) & R_{xy}(\hat{h}^{e1}, \hat{h}^{ob}) & \cdots & R_{xy}(\hat{h}^{e1}, \hat{h}^{ob}) \\ R_{xy}(\hat{h}^{e2}_1, \hat{h}^{ob}_1) & R_{xy}(\hat{h}^{e2}_2, \hat{h}^{ob}_2) & \cdots & R_{xy}(\hat{h}^{eg}_g, \hat{h}^{ob}_g) \\ \vdots & \vdots & \ddots & \vdots \\ R_{xy}(\hat{h}^{ek}_1, \hat{h}^{ob}_1) & R_{xy}(\hat{h}^{ek}_2, \hat{h}^{ob}_2) & \cdots & R_{xy}(\hat{h}^{ek}_g, \hat{h}^{ob}_g) \end{vmatrix} \qquad (2.8)$$

Відповідність об'єкту до одного з уже наявних еталонів максимальних значень коефіцієнтів парної визначається за кількістю кореляції $R_{xy}(\hat{h}^e, \hat{h}^{ob})$ в

одному рядку. Якщо виявлено в матриці рядки з одинаковою кількістю максимальних $R_{xy}(\hat{h}^e, \hat{h}^{ob})$ то відповідність між ними не встановлено, що означає, що інформації отриманої за проекціями об'єкту H^{ob}замало для його ідентифікації [22,23]. Схему реалізації методу вище згаданого відображено на рисунку 2.7.

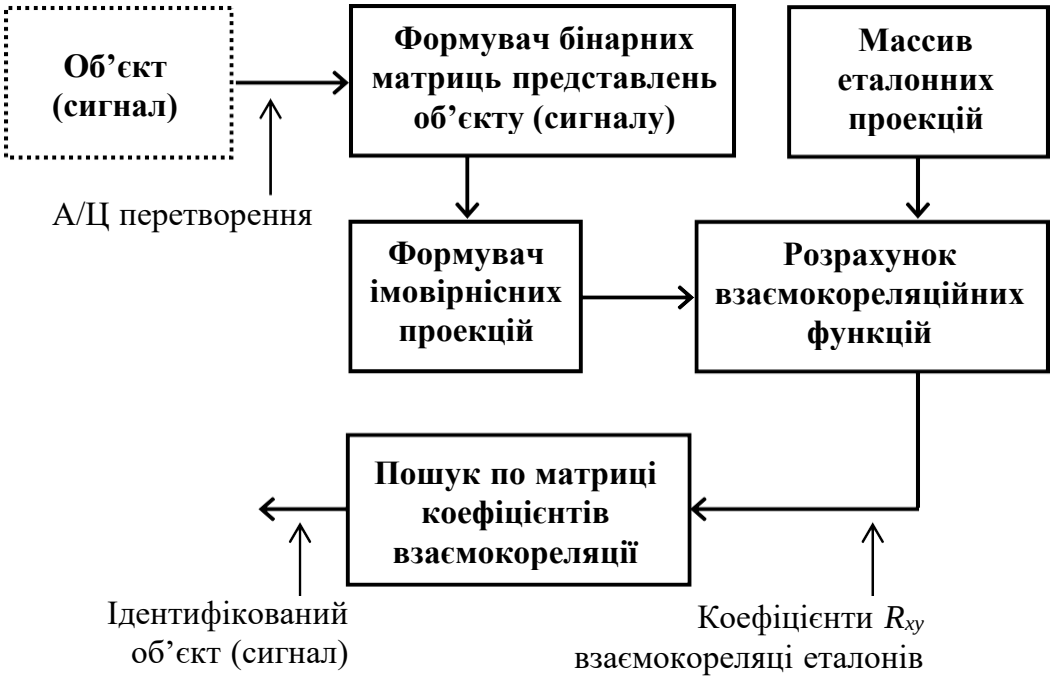


Рисунок 2.7 – Схема ідентифікації об'єкту (сигналу) за його імовірнісними проекціями

Запропонований підхід побудови проекцій, що розраховується за Шеннона формулою за оцінками інформаційної ентропії, до інверсного представлення об'єкту є інваріантним [24].

20.3 Розробка алгоритмічних рішень розпізнавання зображень цифр

Процес розпізнавання розбивається на декілька етапів (рисунок 2.8):

— попереднє створення еталонів для порівняння зображень;

— корекція та зберігання еталонів в базу даних;

— завантаження забраження для розпізнавання;

— зберігання зображення в базу даних;

— розпізнавання зображення за певним методом (в даному випадку за інформаційною ентропією).

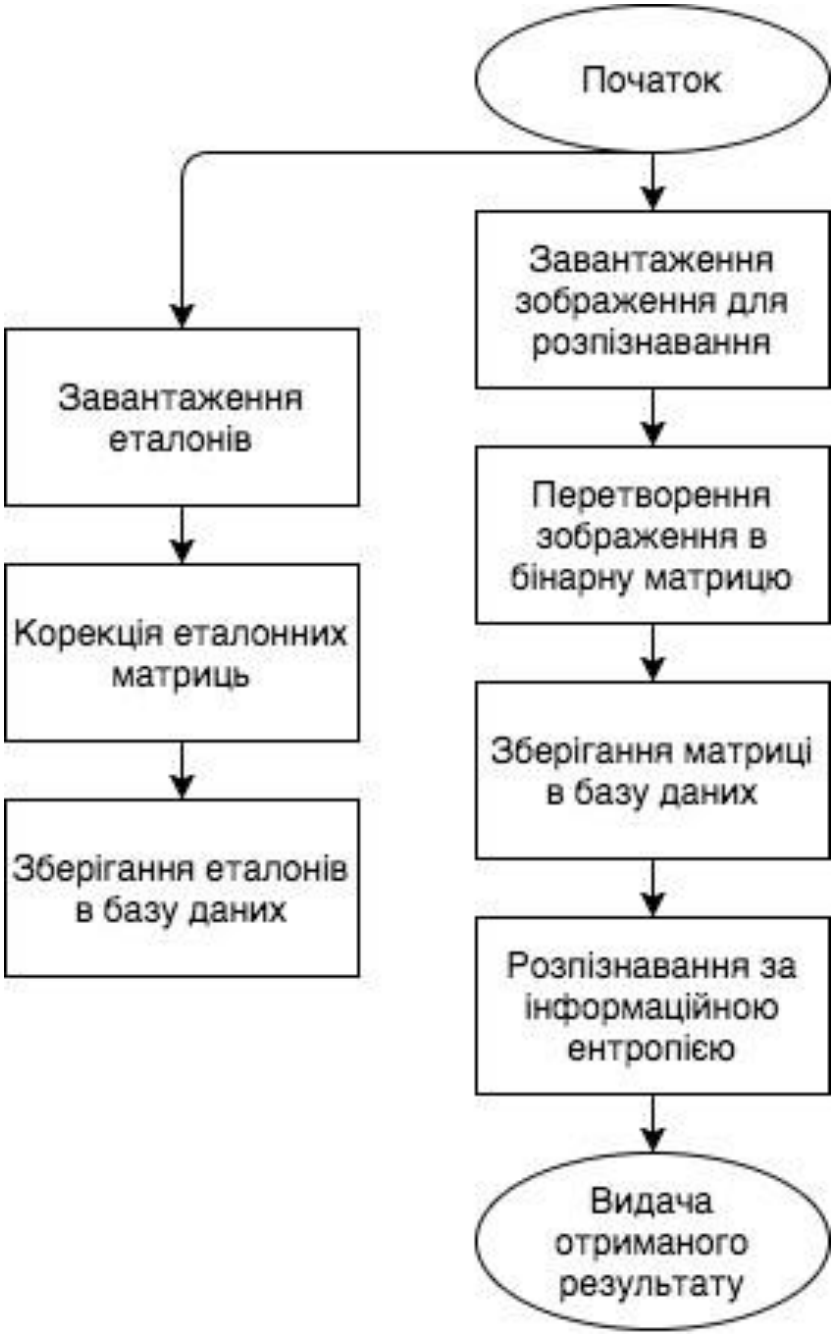


Рисунок 2.8 – Структура системи розпізнавання зображення

Процес завантаження еталонів відбувається аналогічним чином до завантаження зображень для розпізнавання.

Процес буде починатись із завантаження зображення через сторінку користувача. Після цього зображення перетворюватиметься в бінарну матрицю. Процес обробки зображення для розпізнавання зображено на рисунку 2.9.

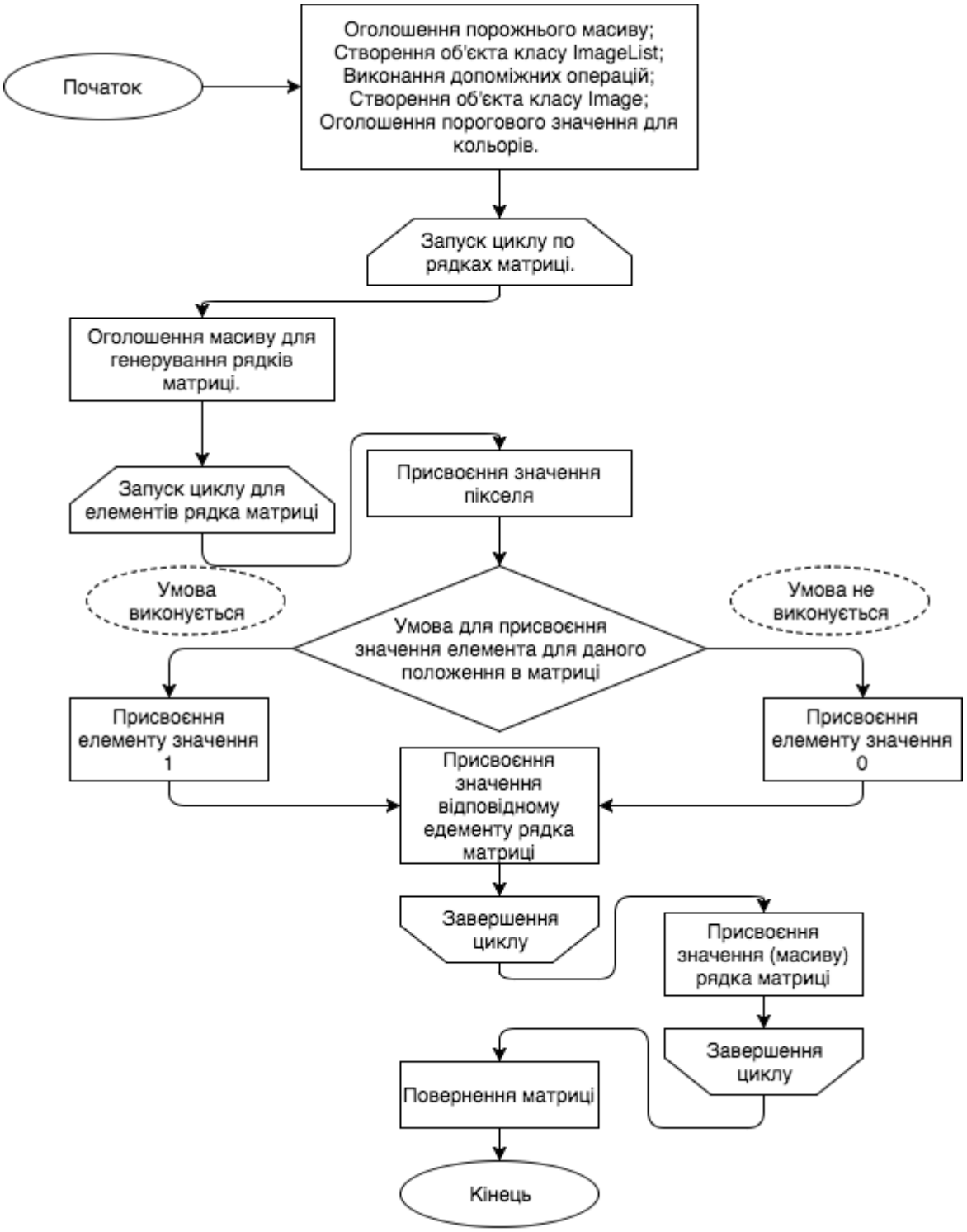


Рисунок 2.9 – Перетворення зображення в бінарну матрицю

Після цього дані надсилатимуться на сервер де зберігатимуться в базу даних. Перед зберіганням зображення, викликатимуться методи для

розпізнавання. В даному випадку основним є метод розпізнавання за інформаційною ентропією.

Сам процес показаний на рисунку 2.10.

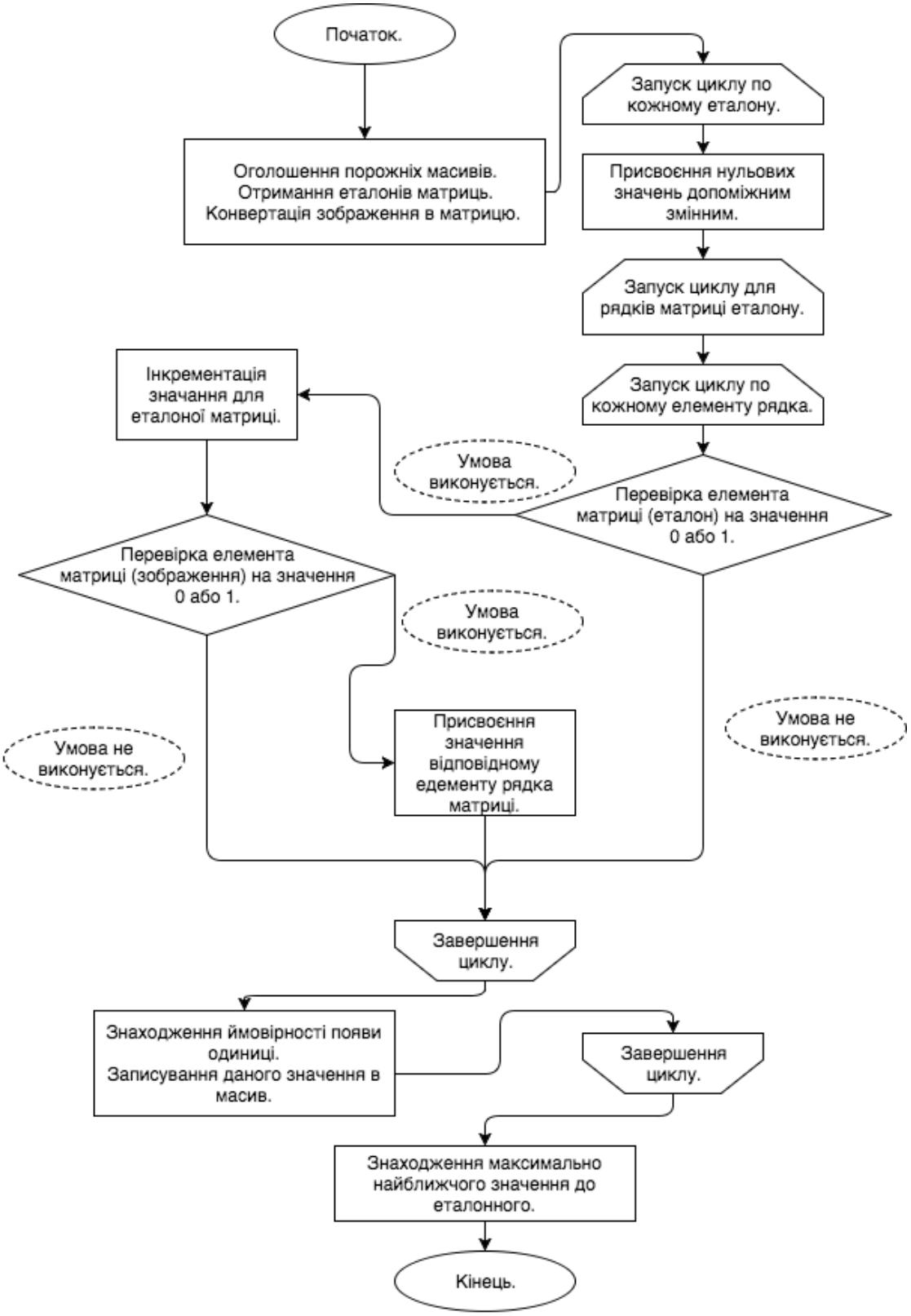


Рисунок 2.10 – Блок схема розпізнавання за оцінкою інформаційної ентропією.

Зі сторони програмного коду все виглядає наступним чином. Процес переведення зображення в матрицю описується наступним кодом:

```
def self.convert_image_to_pixel image_path
  pixels = []
# створення нового об'єкту класу ImageList
  image  = ImageList.new(image_path)
# розпізнівання кольору попіксульно
  image3 = image.quantize(number_colors = 256)
  image2 = Image.new(image.columns, image.rows)
  limit = 1
# запуск циклу по кожному рядку бінарної матриці зображення
  (0..image.rows).each do |x|
    array_row = []
# запуск циклу по кожному елементу рядка бінарної матриці зображення
    (0..image.columns).each do |y|
      pixel = image3.pixel_color(y, x)
      if (limit > pixel.red) && (limit > pixel.blue) && (limit > pixel.green)
# присвоєння 1 для матриці в разі чорного кольору
        el = 1
      else
# присвоєння 0 для матриці в разі білого кольору
        el = 0
      end
# додавання значення в масив
      array_row.push(el)
    image2.pixel_color(x, y, pixel)
    end
# додавання масиву (рядка) в матрицю
    pixels.push(array_row)
  end
```

```
# повернення сформованої матриці
        return pixels
      end
```

Процес порівняння матриці зісканованого зображення з еталонним за інформаційною ентропіює:

```
      def self.entropy_method image_path
# набір усіх еталонів
        standart_arrays = Number::STANDART_ARRAY
# зіскановане зображення представлене у вигляді бінарної матриці матриці
        image_pixels = ImagePixel.convert_image_to_pixel(image_path)
# оголошення допоміжних масивів
        res_arr = []
        array = []
# запуск циклу по всі еталонах
        standart_arrays.each do |obj|
# початкове присвоєння нульових значень ймовірності появи одиниці
          all_one = 0
          img_one = 0
# запуск циклу по кожному рядку еталонної матриці
          obj[1].each_with_index do |arr, index_arr|
# запуск циклу по кожному елементу рядка еталонної матриці
            arr.each_with_index do |el, index_el|
# перевірка появи одиниці по кожному елементу еталонної матриці
              if el == 1
                all_one += 1
# перевірка появи "1" по кожному елементу матриці завантаженого зображення
                if image_pixels[index_arr][index_el] == 1
                  img_one += 1
                end
              end
```

```
            end
          end
# ймовірність появи одиниці
          percent_one = img_one.to_f/all_one.to_f
          percent_zero = 1.0 - percent_one.to_f
# записування в масив ймовірностей появи одиниці
          array.push([percent_one, percent_zero])
          h = (percent_zero * Math.log(percent_zero, 2) +
              + percent_one * Math.log(percent_one, 2)) * (-1.0)
          res_arr.push(h)
        end
# знаходження максимально схожого значення
        el = res_arr.max
        el_index = res_arr.index(el)
        result_number = standart_arrays[el_index][0]
# повернення знайденого значення
        return result_number
      end
```

Таким чином, процес переведення зображення в матрицю реалізовано згідно блок-схеми поданої на рис.2.10.

Висновки до розділу 2

В даному розділі було розроблено структуру та алгоритм методу ідентифікації зображень символів цифр на основі оцінок ентропії.

А саме нового способу розпізнавання об’єктів шляхом використання значення однієї або декількох сумісно імовірнісних характеристик фрагментів зображення обчислених за еталонними шаблонами образів, що дозволяє спростити алгоритмічну і програмну реалізацію, а також забезпечити покращення достовірності розпізнавання у випадках суттєвої зашумленості.

3 РОЗРОБКАДОСЛІДЖЕННЯ ЕФЕКТИВНОСТІ ПРОГРАМИ ДЛЯ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ ЦИФР

3.1 Розробка інтерфейсу та функцій порівняння зображень із шаблонами

Для даного проекту було вибрано мову програмування Rudy, а точніше фреймворк написаний на ній Ruby on Rails. Ruby (в перекладі з анг. – Рубін) – це динамічна, рефлективна, об‘єктно-орієнтована, високорівнева мова програмування. Дана мова має реалізацію багато потоковості незалежну від операційної системи, сувору динамічну типізацію та багато інших можливостей. За особливостями синтаксису близька до Dylan, Python, Lisp, Perl.[25].

RoR – фреймворк, написаний мовою програмування Ruby. Реалізовано представлення за архітектурою модель – представлення – контролер, для веб-аплікацій, і забезпечує інтеграцію із базою даних та веб-сервером.

Ruby – це добрий вибір при створені Інтернет-аплікації саме через ряд своїх переваг, які є немало-важливими для розробки навчальних ресурсів. До основних переваг можна віднести:

— Ruby здатен вливатися в HTML-розмітку, саме це надає розробнику ширші можливості маніпуляції над розроблюваною аплікацією;

— відкритість значить, що є можливість використовувати без отримання ліцензії на використання розробки, та дослідити весь код мови;

— відноситься до мов програмування надвисокого рівнем абстракції і предметним підходом рівня, тобто супроводжується високим в реалізації алгоритмів, чим забезпечує якість програмного продукту;

— містить інтегровані методи полегшують роботу програміста для роботи із масивами. Що значно над проектом;

— має вбудований відлагоджувач програмного коду;

— реалізує об’єктно-орієнтовану парадигму;

— надає просунуті засоби роботи із текстом і строками, прискорюючи правельність реалізації програми;

— можливості мови можна розширити за допомогою бібліотек написаних на C чи Ruby;

— з його допомогою легко інтегрувати в проект бази даних (DB2, MySQL, Oracle I Sybase) [26];

— за допомогою VHLL аплікації розроблені на Ruby легко супроводжуються та масштабуються;

— надає додаткові можливості для забезпечення безпеки роботи аплікації;

— завдяки чистому та простому синтаксису програмістам полегшується перші кроки в освоєнні даної мови;

— містить зручний та простий інтерфейс для створення багато потокових аплікацій;

— також може працювати на різних платформах, що робить його більш зручним і збільшує його поширюваність серед розробників. Саме це усуває ряд проблем пов'язаних з інтеграцією прогами з іншими програмами, сильно не турбуючись через його орієнтованість на платформу.

Якщо говорити про популярність мови, згідно з офіційним сайтом Ruby, дана мова входить в десятку найбільш популярних мов програмування. Приріст популярності самої мови залежить від популярності програмних продуктів написаних на Ruby. Як висновок, використання мови програмування Ruby та її додатків Ruby on Rails для розробки навчальних ресурсів є доцільним. Оскільки дана мова є актуальною, підтримуваною, та повністю підходить для прихильників об’єктно-орієнтованого підходу в програмуванні.

Варто згадати кілька слів про MVC (Model-View-Controller), тому що використання саме цього підходу може відштовхувати людей від використання фреймворків типу Rails. MVC відносно серйозна тема в програмуванні [27].

Рівень моделі – це рівень, де ви визначаєте класи для даних, що буде буде використовувати додаток. Наприклад, якщо необхідно зберегти повідомлення на блозі, буде використовуватись модель "Post". Модель має можливість взаємодіяти з базою даних, для отримування і зберігання даних. Ця функціональність досягається шляхом успадкування його від суперкласу ActiveRecord. Будь-які методи, які діють на цих даних, повинні також бути розміщені в моделі.Рівень відображення має одну головну мету - повернути відповідний HTML, який віддаватиметься користувачеві у браузер. У Rails View зберігаються в ERB файлах, що містять як HTML, так і вкладені Ruby конструкції.Але нічого не станеться без контролера. Контролер взаємодіє з моделлю, щоб вибирати та зберігати дані. Він буде передавати будь-які дані, отримані від моделі, у відображення. View повертає отриманий HTML до контролера і контролер посилає це назад користувачам браузера.

Так як основною задачею даної роботи є не написання нової системи розпізнавання зображень, а покращення існуючих, до уваги візьметься сама функціональність, а не представлення цілого вихідного програмного забезпечення. Перш ніж зробити що-небудь, потрібно створити новий проект. У Rails проект являє собою структуру папок, яка використовується для зберігання всіх файлів, які мають відношення до веб-додатку. Щоб створити проект, потрібно ввести наступну команду в командному рядку:

rails new myapp –d mysql

В цьому випадку використовується MySQL як СКБД. Ця команда створить папку з ім'ям myapp, який містить всі відповідні папки проекту.

Скафолдинг – швидка генерація, що може бути використаний для швидкого створення/прототипування для клієнту. Також скафолдинг може бути корисним для вивчення Rails. Однією простою командою можна створити модель, контроллер і кілька відображень. Для створення відповідних скафолд файлів для записів, можна використати команду:

rails generate scaffold blog title:string

Ця команда створює багато файлів. Вона створює контролер для записів під назвою blogs_controller.rb і модель під назвою blog.rb. Він також створює під папку в папці відоражень під назвою blogs. Вона містить чотири файли, index.html.erb, edit.html.erb, show.html.erb, new.html.erb. Також вказано необхідне поле title з типом string.

В процесі розробки Rails додатку необхідно думати про те, як складові структури даних будуть взаємодіяти одна з одною. Є різні рівні проектування, які можна виконати.

Даний додаток повинен завантажувати працювати із зображеннями, це буде першим модулем. Він розіб'ється на дві частини:

— завантаження зображень для отримання шаблонів;

— завантаження зображень для розпізнавання.

В проекті міститимуться контроллери для зв'язку моделей з відображеннями:

— BlogsController – на ньому будуть відображатись усі проведені процедури розпізнавання та відображення результатів;

— ImageFilesController – створений для роботи із зображеннями (завантаження, обробка);

— StandartController – для відображення існуючих еталонів;

В проекті міститимуться також моделі для роботи з базою даних.

— Blog – міститиме деякі методи для обрізання та коректного відображення зображень та даних;

— ImageFile – служить для зберігання зображень в базу даних;

— ImagePixel – містиме методи для порівняння зображень з шаблонами, а також для обробки зображень;

— Number – існує для роботи із шаблонами бінарних матриць.

Крім того в проекті створено папку із відповідними темплейтами. В папочці app/views/ доступні всі файли для відображення даних:

— layouts/ - міститиме файл application.html.haml – це загальний лейаут, на який підвантажимуться усі інші види;

— blogs/ - містить усі види котроллера blogs_controller, а це show.html.haml, index.html.haml, new.html.haml, cropper.html.haml (міститиме сторінку для обрізання зображення) та інші;

— image_files/ - деякі допоміжні види для контроллера image_controller;

— standarts/ - містить сторінки для відображення еталонів.

Варто зауважити, що всі файли для відображення даних користувачу в проекті RoR за замовуванням мають розширення .html.erb – html-сторінка, яка може містити код ruby. В даному проекті використано gem haml для спрощення роботи із видами. Тому тут всі файли в папочці views мають розширення .html.haml.Допоміжним елементом в проекті є хелпери. В папці helpers міститься файл blog_helper.rb для правильного відображення деяких даних.

Однією з найважливіших конфігурацій є налаштування Gemfile, він міститиме найважливіші пакети (gems), без яких робота проекту неможлива. Одними з базових є:

— gem 'rails', '4.2.6' – версія rails в проекті;

— gem 'mysql2' – база даних для проекту;

— gem 'sass-rails', '~> 5.0' – препроцесор для css;

— gem 'jquery-rails' – для підключення бібліотеки jQuery;

— gem 'haml', '4.1.0.beta.1' – підключення haml;

— gem 'html2haml', '1.0.1';

— gem 'haml-rails', '0.5.3';

— gem 'bootstrap-generators', '~> 3.3.4' – підключення бібліотеки bootstrap та деяких допоміжних компонентів;

— gem 'bootstrap-slider-rails';

— gem 'jquery-ui-rails', '5.0.3';

— gem 'rmagick' – важливий для роботи із зображеннями;

— gem 'paperclip', '~> 3.0' – інструмент для завантаження та обробки зображень.

Так як в проекті буде проводитись робота із завантаженням та обрізанням зображень, буде підключено модуль cropper. Міститиме він наступний код:

```
module PapOne
  class Cropper < Thumbnail
   def transformation_command
     if crop_command
       crop_command + super.join(' ').sub(/ -crop \S+/, '').split(' ')
     else
      super
     end
   end
   def crop_command
    target = @attachment.instance
     if target.cropping?
       ["-crop",
"#{target.crop_w}x#{target.crop_h}+#{target.crop_x}+#{target.crop_y}"]
     end
   end
  end
end
```

Це допоможе в обрізанні зображень, так як порівнюватиметься не цілий текст, а лише окремі символи (цифри), які необхідно буде вирізати із зображення.

Для кращого представлення даних будуть використовувати стилі (css). В проекті міститься папка assets/, яка містить усі файли карнтинок, javascripts та stylesheets. Для обробки зображень створено різні методи, ось деякі найважливіші з них:

— для отримання зображення у вигляді пікселів:

```
def self.get_pixels image_path
  pixels = []
```

```
    image  = ImageList.new(image_path)
    image3 = image.quantize(number_colors = 256)
    image2 = Image.new(image.columns, image.rows)
    limit  = 40000
    (0..image.rows).each
do |x|
        (0..image.columns).each
do |y|
          pixel = image3.pixel_color(y, x)
          pixels.push([pixel.red, pixel.blue, pixel.green])
          image2.pixel_color(x, y, pixel)
        end
    end
    return pixels
end
—    сумування елементів масиву:
def self.get_sum_array arr
    arr.inject(0)
{
|sum,x| sum + x
}
end
—    сума елементів масиву по вертикалі та горизонталі:
def self.get_array_sum_by_verticale image_array
    sum_arr = []
    for i in 0..image_array[0].length
      sum = 0
      image_array.each
do |arr|
          sum = sum.to_i + arr[i].to_i
```

```
      end
    sum_arr.push(sum)
    end
    return sum_arr
end
def self.get_array_sum_by_horizon image_array
    sum_arr = []
    image_array.each
do |arr|
      sum_arr.push(arr.inject(0){
|sum,x| sum + x
})
    end
    return sum_arr
end
```

— вирівнювання масиву:

```
def self.smoothing_array array
    smoothing_array = []
    sum = array.inject(0)
{|sum,x| sum + x
}
    middle_value = sum/array.length
      array.each do |el|
      smoothing_array.push(el-middle_value)
      end
    return smoothing_array
end
```

Інтерфейс користувача буде поділено на два модуля: завантаження еталонних зображень для методів та завантаження зображень для розпізнавання (рисунок 3.1) [28].

Рисунок 3.1 – Структурна схема організації інтерфейсу

Зі сторони користувача все буде виглядати наступним чином. На початку користувач заходить на сторінку завантаження зображення в браузері де бачить форму для завантаження (рисунок 3.2).

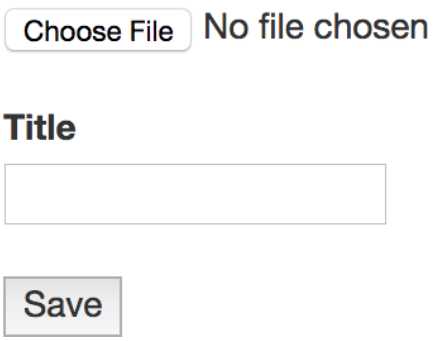


Рисунок 3.2 – Форма для завантаження зображення

Форма містить поле типу file та кнопку типу submit. Також додано поле Title для вказування назви зображення (для подальшої ідентифікації в майбутньому). Після вибору зображення з текстом, форма відправляється на

сервер для подальшої обробки. Але перед цим рендериться сторінка з можливістю обрізання певної частини зображення (в даному випадку для цифри, рисунок 3.3).

Рисунок 3.3 – Обрізання зображення

Після того як зображення пройшло процес перевірки, отримується результат на окремій сторінці браузера (рисунок 3.4).

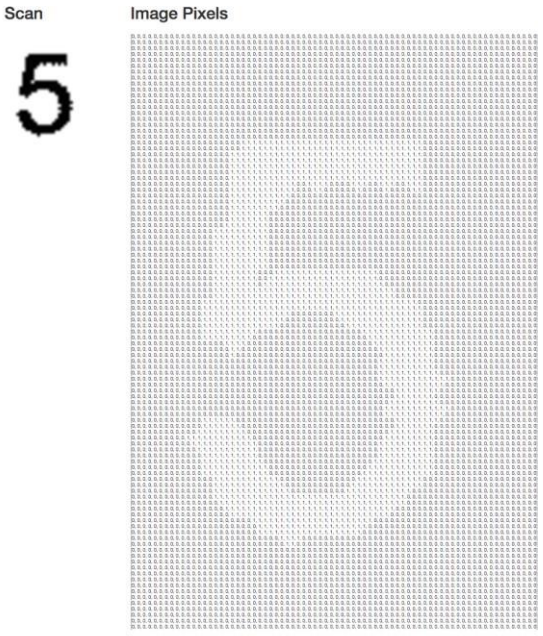


Рисунок 3.4 – Зіскановане зображення та його бінарна матриця

Крім того на сторінці буде показано результат розпізнавання з даним методом, а також за іншими методами для порівняння їх ефективності (рисунок 3.5).

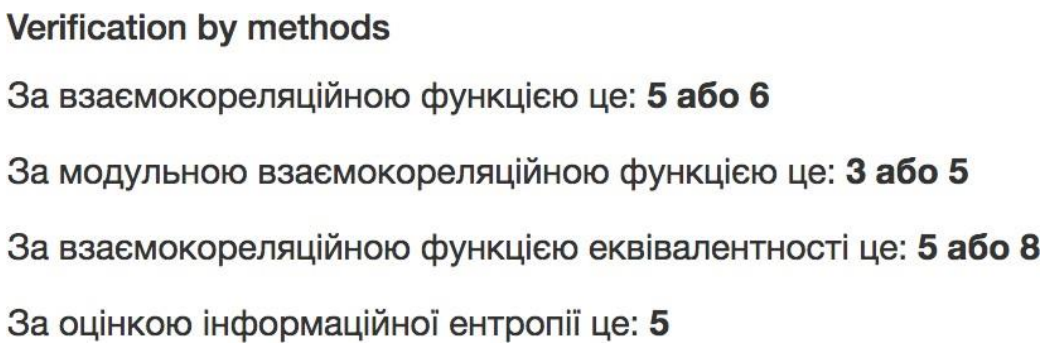


Рисунок 3.5 – Результат розпізнавання за декількома методами

Після даного процесу всі випадки розпізнавання, які проводились раніше, доступні на окремій сторінці (рисунок 3.6).

На ній зображено назва для ідентифікації, саме зображення та посилання для перегляду та видалення.

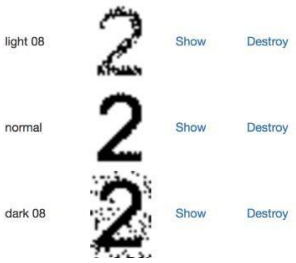


Рисунок 3.6 – Список зісканованих зображень

Надалі користувач має можливість переглядати попередні результати досліджень, на яких буде відображено зісканований малюнок (область обрізання), згенеровану бінарну матрицю, а також результати по кожному з наявних методів.

3.2 Порівняльний аналіз впливу втрат пікселів зображення на адекватність ідентифікації

Доцільно зазначити, що при формуванні графічних представлень об’єкту різного роду виникають спотворення зумовлені впливом навколишнього

апаратних та конструктивних обмежень сенсорів, похибок перетворювачів, середовища тощо.

Для того, щоб перевірити ефективність даного підходу проведено імітаційне моделювання в обчислювальному експерименті, під час якого здійснювалось спотворення графічних представлень (рисунок 3.7) з подальшим оцінюванням ймовірності вірної ідентифікації P_{is} символу.

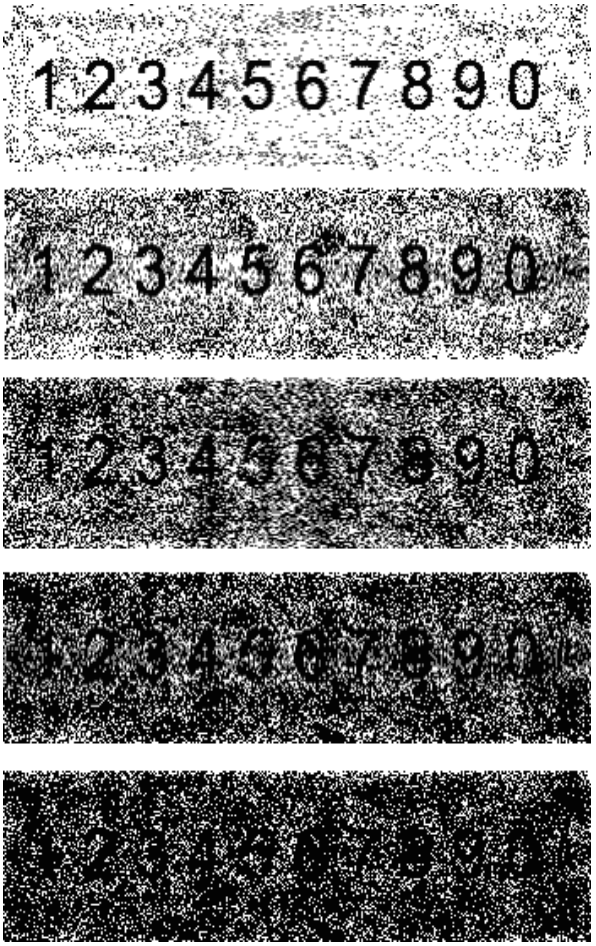


Рисунок 3.7 – Спотворення зображення чорними пікселями.

Для того щоб довести ефективність використання методу порівняння за ентропією, проводиться порівняння ефективності використання кожного з методів при спотворенні зображення.

При побудові проекцій адекватність ідентифікації спостерігається за інтегральними характеристиками прийнятна при інформаційних втратах

графічних представлень, що не становлять не більше 40% при спотворенні чорними пікселями.

Під час використання інших статистичних характеристик, зокрема середнього значення, серед вище згаданих найкращий середнього квадратичного відхилу, дисперсії адекватність розпізнавання значно погіршується, результат, 20% при спотворенні чорними пікселями, побудованих за дисперсією спостерігається для проекцій.

Для наглядної характеристики ефективності методів було проведено ряд дослідів з певним рівнем спотворення зображення.

Нижче наведено результати (таблиця 3.1).

Таблиця 3.1 – Ймовірність достовірності методів розпізнавання при спотворенні зображення надлишковими пікселями

Величина спотворення (%)	За взаємокореляційною функцією (%)	За модульною взаємокореляційною функцією (%)	За взаємокореляційною функцією еквівалентності (%)	За оцінкою інформаційної ентропії (%)
0	85.10	95.07	85.10	95.04
20	85.02	95.02	85.02	95.04
40	75.10	75.02	60.03	95.08
60	75.05	75.04	55.02	95.01
80	60.04	60.05	55.02	80.01
90	40.02	30.03	30.04	80.01

Як можна побачити, використання спрощених кореляційних функцій суттєво погіршує імовірність коректної ідентифікації символів.

Результати представлені графіком на рисунку 3.8.

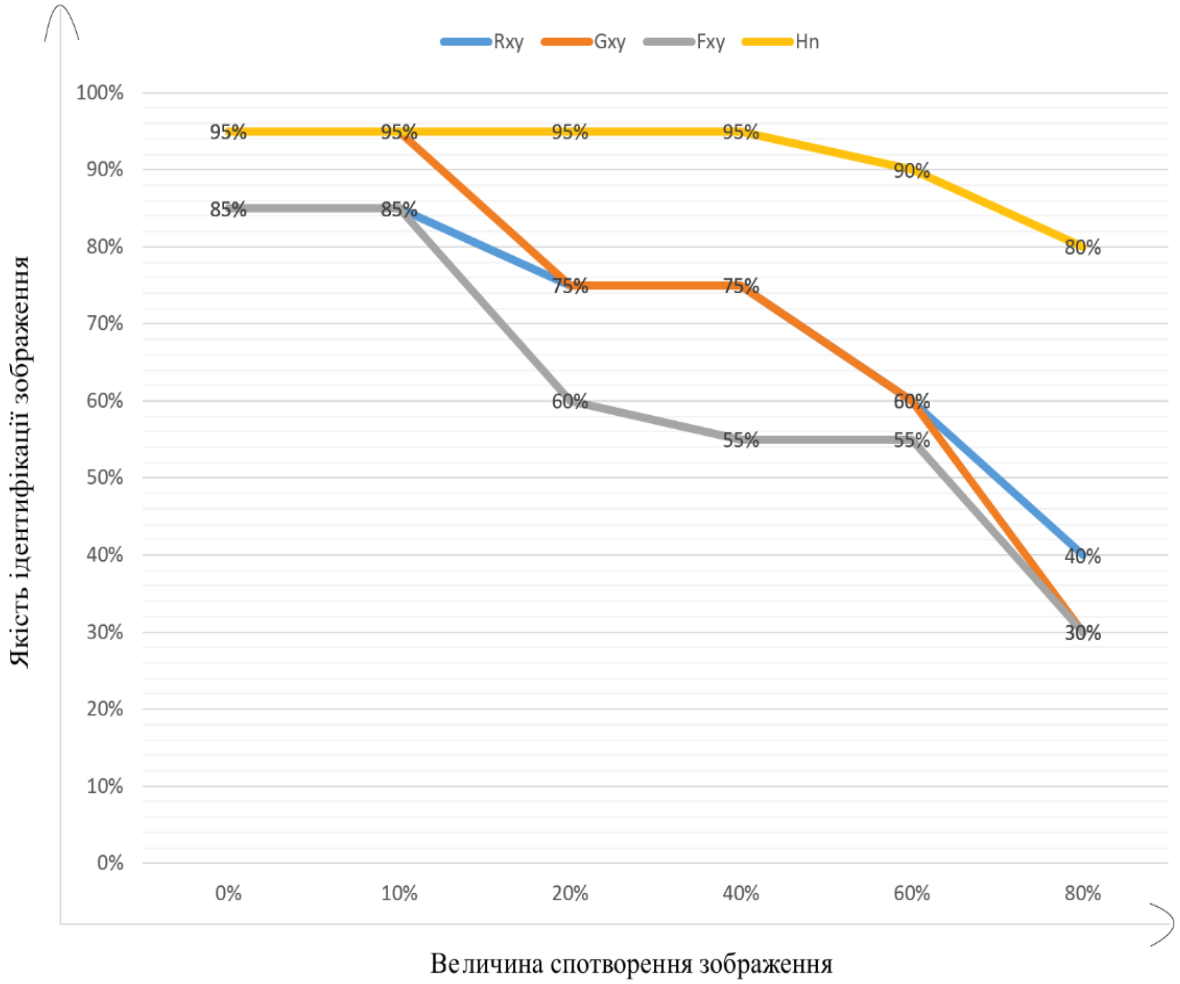


Рисунок 3.8 – Залежність достовірності методів розпізнавання при спотворенні зображення надлишковими пікселями

Як видно з рисунка 3.8, результати розпізнавання за кожним з наведених методів не є ідентичними, а змінюються по різному в залежності від величини спотворення.

Найдостовірнішим є результат методу за оцінкою інформаційної ентропії, так як за величини спотворення 80% якість ідентифікації знизилась лише на 20%, що є набагато вищим результатом по відношенню до решти методів.

3.3 Порівняльний аналіз впливу наявності спотворень зображень на адекватність ідентифікації

Досить часто на якість розпізнавання зображень впливають не тільки надлишкові спотворення, але й втрата якості самого зображення через втрату певних частин символів.

Для прикладу доцільно провести експериментальні дослідження, додавши до зображення надлишкові білі пікселі. Спотворення проводитиметься також в 5 рівнів (рисунок 3.9)

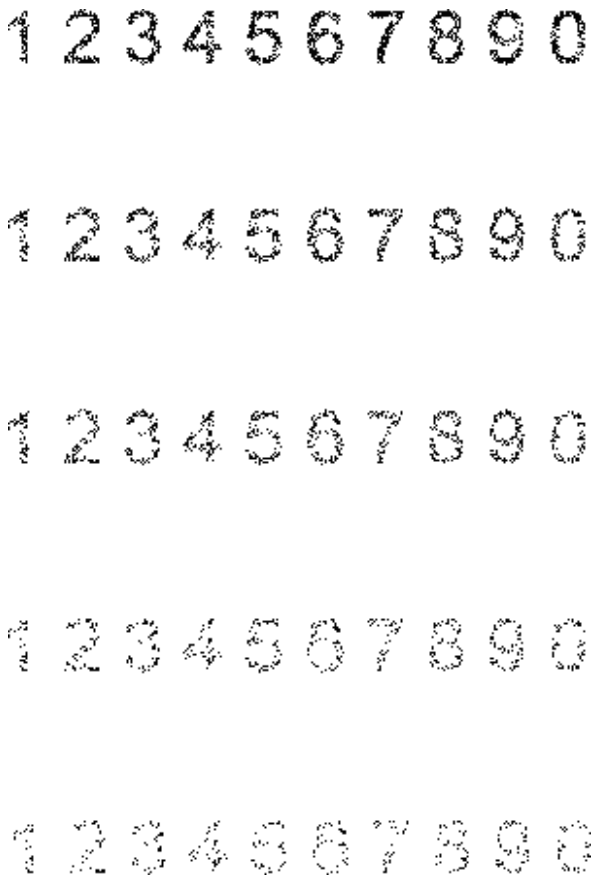


Рисунок 3.9 – Спотворення зображення білими пікселями.

З’ясовано, що під час побудови за інтегральними характеристиками проекцій прийнятна адекватність ідентифікації спостерігається при графічних представлень інформаційних втратах, які не перевищують 30% при спотворенні білими пікселями.

При використанні зокрема середнього значення, середнього квадратичного відхилу, дисперсії, центральних моментів вищих порядків - інших статистичних характеристик, адекватність розпізнавання погіршується, серед згаданих найкращий результат, 12% при спотворенні білими пікселями, спостерігається для проекцій побудованих за дисперсією.

Для наглядної характеристики ефективності методів було проведено ряд дослідів з певним рівнем спотворення зображення.

Нижче наведено результати (таблиця 3.2).

Таблиця 3.2 – Ймовірність достовірності методів розпізнавання при спотворенні зображення втратою пікселів

Величина спотворення (%)	За взаємокореляційною функцією (%)	За модульною взаємокореляційною функцією (%)	За взаємокореляційною функцією еквівалентності (%)	За оцінкою інформаційної ентропії (%)
0	85.04	95.02	85.05	95.01
20	85.02	95.01	85.05	95.01
40	50.02	70.01	50.04	95.02
60	50.01	70.02	50.03	95.01
80	50.05	50.02	50.05	85.21
90	40.04	30.05	30.05	85.01

Як можна побачити, використання спрощених кореляційних функцій суттєво погіршує імовірність коректної ідентифікації символів. Результати представлені графіком на рисунку 3.10.

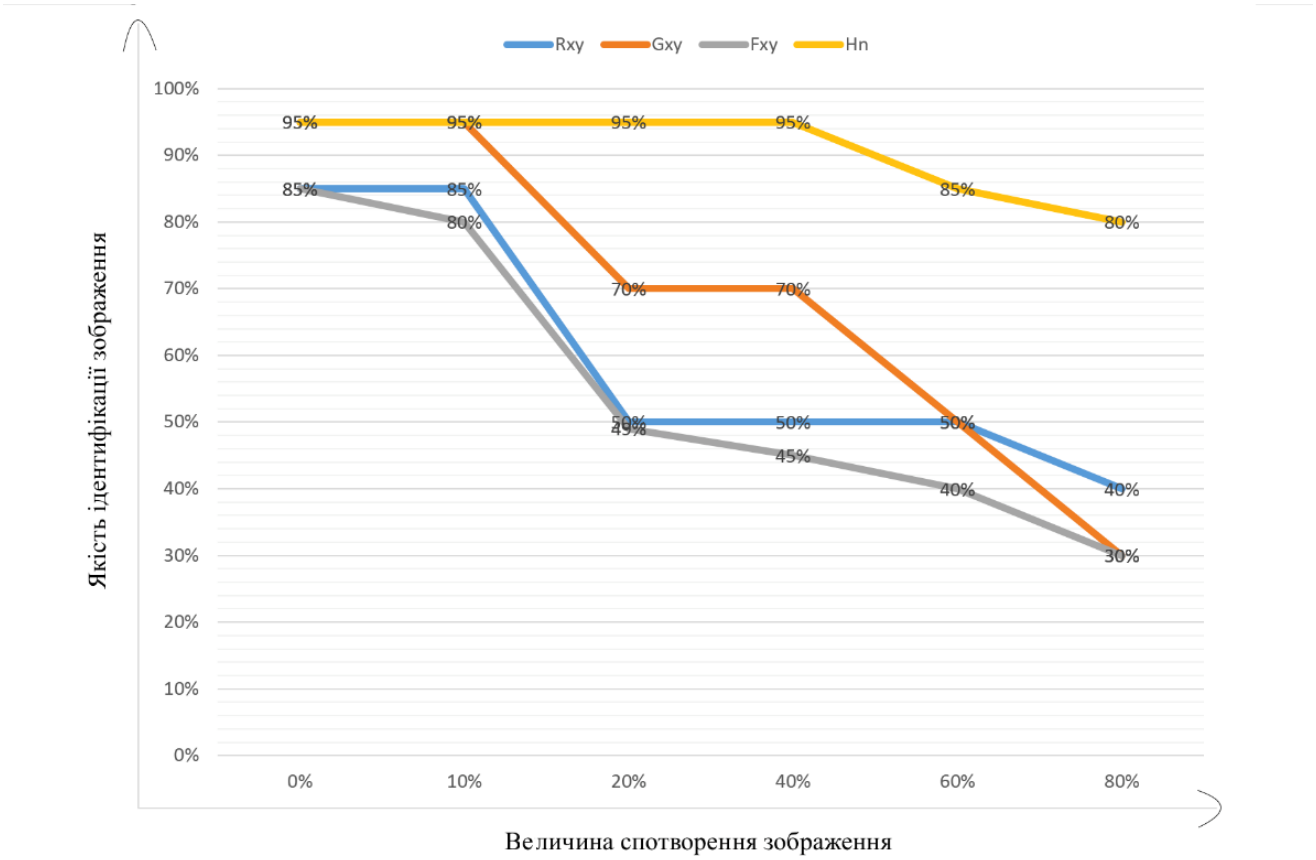


Рисунок 3.10 – Залежність достовірності методів розпізнавання при спотворенні зображення втратою пікселів

Як і в результатах при спотворенні надлишковими пікселями, на даному графіку видно значно вищу якість ідентифікації метода за оцінкою інформаційної ентропії.

Крім того видно як падає якість ідентифікації при спотворенні лише на 20 % в інших методах, при тому що в запропонованому методі вона залишається незмінною аж до величини спотворення 40%.

Висновки до розділу 3

В даному розділі було проведено дослідження ефективності програми для розпізнавання зображень цифр.

А саме:

— розроблено інтерфейс програмного забезпечення та функції порівняння

зображень із шаблонами;

— проведено порівняльний аналіз впливу втрат пікселів зображення на адекватність ідентифікації;

— проведено порівняльний аналіз впливу наявності спотворень зображень на адекватність ідентифікації.

Програма розпізнає шрифт Calibri величиною 14, також база еталоних зображень може поповнюватись новими видами шрифтів при розпізнаванні.

ВИСНОВКИ

В ході виконання магістерської випускної роботи було виявлено, що деякі методи розпізнавання тексту не є досить ефективними і якість ідентифікації залежить значною мірою від багатьох чинників, в тому числі спотворень.

В процесі розробки модуля було запропоновано новий метод розпізнавання, а саме за оцінкою інформаційної ентропії. Запропонований метод дозволяє збільшити адекватність ідентифікації порівняно з відомими проекційними методами, а також забезпечує сумарну ефективність порівняно з обчислювальними кореляційними методами за рахунок використання імовірнісного, в розглянутому випадку ентропійного, представлення при побудові проекцій.

Крім того використання імовірнісних характеристик дозволяє суттєво зменшити, а в окремих випадках усунути, вплив одиничних спотворень.

Програма розпізнавання розроблена символів цифр монохромних використовує OCR технологію. Даний підхід вимагає для кожного шрифту створення шаблону.

Розроблена програма розпізнає шрифту Calibri величиною шрифта 14. База еталоних зображень може поповнюватись новими видами шрифтів при розпізнаванні.

Дослідження результатів описаної проблеми було використано базу даних з еталонних образів, зокреме що складається з десяти арабських цифр. Хоча для запропонованого методу можна наповнювати базу і новими символами.

ПЕРЕЛІК ПОСИЛАНЬ НА ДЖЕРЕЛА

1. Фу К. Структурные методы в распознавании образов : пер. с англ. / Фу К. М. : Мир, 1977. 320 с.

2. Rudnytska T.H. The Benefits of Voice Recognition by Spiking Neural Network for Students with disabilities / T. H. Rudnytska, T. S. Lishchenko // Іноземні мови у вищому навчальному закладі: теоретичні засади та прикладні аспекти: Матеріали Всеукраїнської науково-теоретичної конференції / Гол. ред. Ямчинська Т. І. – Вінниця, 2013. 283с.

3. Kozemiako V. P. Optoelectronic spiking neural network / V. P. Kozemiako, O. K. Kolesnytskyj, T. S. Lischenko, W. Wojcik, A. Sulemenov / Optical Fibers and Their Applications. Poland.– 2012. С. 40 – 43.

4. Бардаченко В.Ф. Таймерні нейронні елементи та структури /В.Ф. Бардаченко, О.К.Колесницький, С.А. Василецький. –Вінниця, 2005. 126 с. – (Монографія).

5. Мельничук С.І. Ідентифікація об’єктів за імовірнісними характеристиками фрагментів їх монохромних зображень. / С.І Мельничук. // XI міжнародна науково-практична конференція: Математичне та програмне забезпечення інтелектуальних систем, м. Дніпропетровськ, 20-22 листопада 2013 р. / Дніпропетровськ: ТОВ «Роял Принт», 2013. С. 159 – 160.

6. O. Alsharif and J. Pineau. End-to-end text recognition with hybrid HMM maxout models. – ICLR, 2014. 205 с.

7. Применение нейросетей в распознавании изображений [Електронний ресурс] – URL: http://habrahabr.ru/post/74326/ (дата звернення: 10.11.2021)

10. Kolesnytskyj O. K. Optoelectronic Implementation of Pulsed Neurons and Neural Networks Using Bispin-Devices / O. K. Kolesnytskyj, I. V. Bokotsey, S. S. Yaremchuk // Optical Memory & Neural Networks (Information Optics), 2010. — Vol.19. — No 2. 165 с.

11. Гороховатский А.В. Распознавание изображений символов на

основе линейного описания структурных характеристик / А.В. Гороховатский, Е.О. Передрий // Системы обработки информации. – Х.: ХУ ПС, 2012. – Вип. 7(105). 203-206 с.

12. Бутаков Е. А. Обработка изображений на ЭМВ / Е. А. Бутаков // Радио и связь – Россия, 1987. 45 с.

13. Левенштейн В.И. Двоичные коды с исправлением выпадений, вставок и замещений символов / В.И. Левенштейн // Доклады Академии Наук СССР. – 1965. – No 163.4. 845-848 с.

14. A. Rodriguez-Serrano. Label embedding: A frugal baseline for text recognition. – IJCV, 2015. 193-207 с.

15. Горелик А. Л. Методы распознавания / Л. А. Горелик – М.: Высшая школа, 2004. 262 с.

16. Бонгард М. М. Проблема узнавання / М. М. Бонгард – М.: Наука, 1967. 320 с.

17. Бардаченко В. Ф. Перспективи застосування імпульсних нейронних мереж з таймерним представленням інформації для розпізнавання динамічних образів / В.Ф. Бардаченко, О.К. Колесницький, С.А. Василецький // УСіМ. – 2003, 82 с.

18. Імад І.А. Сбіех. Реконструкція тривимірних об’єктів на растрових зображеннях / Ємець В.Ф., Мороз І.В., Імад І.А. Сбіех // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова. Вип. 29. – К.: 2005. 132 с.

19. Імад І. А. Сбіех. Підходи до побудови сучасних систем оптичного розпізнавання символів та покращення їх характеристик з використанням некоректних задач /Імад І. А. Сбіех// Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова. Вип. 40. К.: 2007. 171 с.

20. Імад І.А. Сбіех. Побудова систем розпізнавання символів з використанням методів розв’язку некоректних задач / Імад І.А. Сбіех // Матеріали конференції: XXVI науково-технічна конференція “Моделювання“, ІПМЕ НАН України, Київ, 2007. 81 с.

21. Имад И.А. Сбиех. Улучшение характеристик современных систем

распознавания символов с использованием методов решения некорректных задач / Имад И.А. Сбиех// Материалы семинара: Научно-технический семинар «Системы синхронизации, формирования и обработки сигналов для связи и вещания». Одесса: 2007. 192 с.

22. Пат. 103281 Україна, МПК(2006) G06K 9/64. Спосіб ідентифікації об’єктів за імовірнісними характеристиками фрагментів їх зображень / Мельничук С.І. (Україна). – заявка № а 2012 12161; заявл. 23.10.2012, опубл. 25.09.2013, Бюл. № 18.

23. Імад І.А.Система реконструкції тривимірних об'єктів за невпорядкованими даними аерозображень / І. А. Імад, В. Ємець, О. Карпін// Матеріали І Міжнародної конференції: Modern problems of radioelectronics, telecommunications and instrument making (MPRTI-2005). Vinnitsa 2-5 June 2005. 105 с.

24. Журавлев Ю.И. Распознавание. Математические методы. Программная система. Практические применения. / В.В. Рязанов, О.В. Сенько – М.: Фазис, 2005. – 159 с. 2. Рябенький В.М. Комбіновані системи розпізнавання образів / В.М. Рябенький, О.І. Захожай // Журнал «Проблеми інформаційних технологій» № 01 (009). – Херсон: ХНТУ – 2011. 160 с.

25. J. Gllavata, R. Ewerth, B. Freisleben. Text Detection in Images Based on Unsupervised Classification of High-Frequency Wavelet Coefficients – IEEE Conf. Pattern Recognition, – 2004, 424 с.

26. Захожай О.І. Основні аспекти структурної організації комбінованих систем розпізнавання образів / О.І. Захожай, Ю.Е. Паеранд // Вестник ХНТУ №1 (44). – Херсон: «Олди-Плюс». – 2012. 225 с.

27. Руби С. Гибкая разроботкавеб-приложений в среде Rails, 4-е издание / С. Руби, Д. Томас, Д. Хенссон–Питер– 2012. 40 с.

28. Дюбуа П. MySQL сборник рецептов / П. Дюбуа –Санкт-Петербург– 2010. 128 с.

29. Model View Controller. [Електронний ресурс]. – 2017. – URL: https://uk.wikipedia.org/wiki/model-view-controller (дата звернення 14.11.2021).

30. Інтерфейс користувача. [Електронний ресурс]. – 2017. – URL: https://uk.wikipedia.org/wiki/інтерфейс-користувача (дата звернення 14.11.2021).

31. Стандарт підприємства. Курсовий і дипломний проект. Вимоги до змісту та оформлення. СТП 02070855-03-99 – Івано-Франківськ, ІФДТУНГ, 1999.- 72 с.

32. A. Mishra, K. Alahari. Scene text recognition using higher order language priors. – BMVC, 2012. 111 с.

33. Фу К. Структурні методи в розпізнаванні образів. Пер. с англ. / К. Фу – М.: Мир, 1977. 77 с.

34. Мінський М. Фрейми для представлення знань. Пер. с англ. / М. Мінський – М.: Мир, 1979. 190 с.

35. Імад І.А. Сбіех. Реконструкція тривимірних об’єктів на растрових зображеннях / Ємець В.Ф., Мороз І.В., Імад І.А. Сбіех // Зб. наук. пр. ІПМЕ ім. Г.Є. Пухова. Вип. 29. – К.: 2005. 127 с.

36. Козак Ю., Мацкул В. Математичні методи та моделі для магістрів// Центр навчальної літератури, 2019. 223 с.

37. X. Chen, J. Yang, J. Zhang. Automatic Detection and Recognition of Signs from Natural Scenes – IEEE Image Processing, vol. 13, no. 1, 2004. 98 с.

38. C. Garcia, X. Apostolidis. Text Detection and Segmentation in Complex Color Images – in Proc. IEEE Conf. Acoustics, Speech and Signal Processing, 2000. 221 с.

39. Гонсалес Р. Цифрова обробка зображень. Пер. с англ. / Р. Гонсалес, Р. Вудс – М.: Техносфера, 2005. 95 с.

40. Hunspell spell chacker dictionary. // [Електронний ресурс]. – URL: http://hunspell.sourceforge.net/ (дата звернення 15.11.2021).

41. Захожай О.І. Основні аспекти структурної організації комбінованих систем розпізнавання образів / О.І. Захожай, Ю.Е. Паеранд // Вестник ХНТУ №1 (44). – Херсон: «Олди-Плюс». – 2012. 135 с.

42. Васильєв, В. Н. Математичні методи і алгоритмічне забезпечення аналізу та розпізнавання зображень в інформаційно-телекомунікаційних

системах / В. М. Васильєв, І. П. Гуров, А. С. Потапов, 2008. 25 с.

43. Хант Е. Штучний інтелект. Пер. с англ. / Е. Хант – М.: Мир, 1978. 111 с.