Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра програмного забезпечення

Пояснювальна записка

до магістерської кваліфікаційної роботи
магістр
(освітньо-кваліфікаційний рівень)

на тему: Методи та програмні засоби розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні

Виконав: студент II курсу
групи 2ПІ-18 м
спеціальності
121 – Інженерія програмного забезпечення
(шифр і назва напряму підготовки, спеціальності)
Воловик Б.П.
(прізвище та ініціали)

Керівник: к.т.н., доц. каф. ПЗ Рейда О.М.
(прізвище та ініціали)

Рецензент: к.т.н., доц. Колесницький О.К.
(прізвище та ініціали)

ВНТУ – 2019

Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення
Освітньо-кваліфікаційний рівень – магістр
Спеціальність 121 – Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О. Н.
"____"_____2019 року

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ

Воловику Богдану Петровичу

1. Тема роботи – методи та програмні засоби розпізнавання і аналізу інфомації про властивості матеріалів зондування земної поверхні.

Керівник роботи: Рейда Олександр Миколайович, к.т.н., доцент кафедри ПЗ, затверджені наказом вищого навчального закладу від "__"_____2019 року №___

2. Строк подання студентом роботи
_____

3. Вихідні дані до роботи: операційна система – Windows 8/10; середовище програмування Visual Studio 2015; мова програмування – C#; методи розпізнавання образів; методи розпізнавання тесту; методи попередньої обробки зображень; аналіз предметної області застосування; вихідні дані для розпізнавання; дані про кількість джерел цифрового шуму.

4. Зміст розрахунково-пояснювальної записки: аналіз методів розпізнавання об'єктів; аналіз методів розпізнавання тексту; аналіз методів попередньої обробки зображень; аналіз предметної області застосування; аналіз предметної області розробки; вибір засобів розпізнавання; формування задач до розробки; обґрунтування вибору засобів програмування; розробка архітектури; розробка модулів; розробка інтерфейсу програмного додатку; тестування програмного додатку; економічна частина.

5. Перелік графічного матеріалу: мета, об'єкт та предмет дослідження; задачі дослідження; методи дослідження; наукова новизна; практичне значення; обґрунтування доцільності розробки; метод ситуативного навчання; математична модель ситуативного навчання; засоби реалізації; розробка моделей системи; тестування; висновки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Рейда О.М., к.т.н., доцент кафедри ПЗ		
4	Бальзан М.В., к.е.н., доцент кафедри ЕПВМ		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської кваліфікаційної Роботи	Строк виконання етапів роботи	Примітка
1	Аналіз сучасних методів розпізнавання	07.10.2019 – 16.10.2019	Вик.
2	Розробка методу розпізнавання та аналізу інформації	17.10.2019 – 24.10.2019	Вик.
3	Розробка програмного забезпечення та його тестування	25.10.2019 – 14.11.2019	Вик.
4	Економічна частина	15.11.2019 – 27.11.2019	Вик.

Студент _____ Воловик Б.П.

(підпис) (прізвище та ініціали)

Керівник магістерської кваліфікаційної роботи _____ Рейда О.М.

(підпис) (прізвище та ініціали)

АНОТАЦІЯ

У магістерській кваліфікаційній роботі запропоновано новий метод модель розпізнавання на основі контурного аналізу та модель пошуку границь на зображенні.

Виконано аналіз сучасних методів та засобів розпізнавання та аналізу інформації.

Спроектовано архітектуру та інтерфейс програмного додатку. Створено програмний додаток відповідно до поставлених вимог, проведено його тестування.

Розроблені методи та моделі можуть бути використані при розробці сучасних систем розпізнавання інформації.

ANNOTATION

The master's qualification work proposes a new method of contour-based recognition model and boundary search model.

The analysis of modern methods and means of recognition and analysis of information is carried out.

The architecture and interface of the software application have been designed. A software application was created in accordance with the set requirements, its testing was carried out.

The developed methods and models can be used in the development of modern information recognition systems.

ЗМІСТ

ВСТУП

Обґрунтування вибору теми дослідження. На сьогодні існує зростаючий інтерес до систем розпізнавання інформації, це обумовлено необхідністю автоматизувати функції управління та контролю складними динамічними об'єктами в реальному часі.

Методи і алгоритми теорії розпізнавання широко використовуються в різних сферах життя людини. Однією з таких сфер є дистанційне зондування земної поверхні.

Дистанційне зондування Землі - незамінний інструмент для вивчення і постійного моніторингу нашої планети, який допомагає ефективно використовувати і керувати її ресурсами. Зображення, що передаються засобами дистанційного зондування земної поверхні, знаходять застосування в багатьох галузях: сільському господарстві, геологічних та гідрологічних дослідженнях, лісництві, охороні навколишнього середовища, плануванні територій, у військових та інших цілях. В експериментальних дослідженнях все зростаючу роль грає проблема автоматизації отриманої інформації з зображень, одержаних за допомогою різних фізичних приладів таких як: великі оптичні, радіо-, рентгенівські, гамма-телескопи та інші. Потенційні можливості цих приладів часто не реалізуються, якщо не забезпечити їх належними засобами обробки одержуваних зображень.

Оскільки, на даний час, ідеальних технологій, як у швидкості, так і у якості розпізнавання інформації не існує, було вирішено покращити сучасні методи та програмні засоби розпізнавання інформації про властивості зондування земної поверхні, що в свою чергу покращить продуктивність процесу розпізнавання.

Зв’язок роботи з науковими програмами, планами, темами. Робота виконувалася згідно плану виконання наукових досліджень на кафедрі програмного забезпечення

Мета та завдання дослідження. Метою роботи є підвищення продуктивності розпізнавання та аналізу інформації про властивості матеріалів зондування земної поверхні.

Основними задачами дослідження є:

- провести аналіз існуючих методів і засобів розпізнавання для підвищення їх продуктивності;
- запропонувати покращені методи та засоби для підвищення швидкості та точності розпізнавання інформації;
- розробити відповідну систему розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні на основі запропонованих методів;
- провести експериментальні дослідження розроблених методів та засобів.

Об'єкт дослідження – процес розпізнавання інформації із зображення шляхом покращення існуючих аналогів.

Предмет дослідження – методи та засоби розпізнавання і аналізу інформації матеріалів зондування земної поверхні.

Методи дослідження. У процесі досліджень використовувались: теорія розпізнавання образів для розробки методів розпізнавання, методи розпізнавання тексту для покращення розпізнавання текстової інформації; методи попередньої обробки зображень для покращення алгоритмів обробки вхідної інформації, метод перетворень Хафа для збільшення продуктивності розпізнавання інформації, лінійна алгебра для рішення лінійних рівнянь, надання матриць, векторів.

Наукова новизна отриманих результатів.

1. Вперше запропоновано модель розпізнавання на основі контурного аналізу, яка при відповідному виборі лінійного простору для подання контуру, дозволяє зменшити трудомісткість операцій обробки та розпізнавання зображення, що в свою чергу, підвищує швидкість виконуваної операції(розпізнавання).
2. Вперше запропоновано модель попередньої обробки зображення, на основі оператора Собеля, який базується на більш м'якій апроксимації градієнта, що дозволяє збільшити точність розпізнавання інформації.

Практична цінність отриманих результатів. Практична цінність отриманих результатів полягає в тому, що на їх основі в магістерській кваліфікаційній роботі було розроблено моделі та програмний засіб для розпізнавання і аналізу інформації матеріалів зондування земної поверхні.

Апробація результатів роботи. Результати роботи були апробовані на науково-технічній конференції професорсько-викладацького складу, співробітників та студентів Вінницького національного технічного університету(м.Вінниця, Україна, 2019р.) та опубліковані у збірнику даної конференції.

Особистий внесок магістранта. Усі наукові результати отримано автором самостійно.

1 АНАЛІЗ СУЧАСНИХ МЕТОДІВ РОЗПІЗНАВАННЯ

1.1 Аналіз методів розпізнавання образів

Образ являє собою опис об'єкта. Кожної миті свого життя людина розпізнає все, що її оточує. Людська істота є дуже складною інформаційною системою - в певній мірі, це визначається розвиненою здатністю розпізнавати образи.

Розпізнавання можна розділити на два основних типи: розпізнавання конкретних і абстрактних об'єктів. Ми впізнаємо знаки, малюнки, музику та предмети, які нас оточують. Процес, що включає розпізнавання зорових і слухових образів, можна визначити як "сенсорне" розпізнавання. Методи такого типу дозволяють ідентифікувати та класифікувати просторові та часові образи. З іншого боку, із закритими вухами та очима ми можемо розпізнати старий аргумент або знайти рішення. Приклади просторових зображень включають символи, відбитки пальців, контурні карти, фізичні об'єкти та малюнки. Тимчасові зображення включають мову, змінні сигнали, електрокардіограми, властивості цілі та часовий ряд. Розпізнавання певних образів людиною може розглядатися як психофізіологічне завдання, пов'язане з процесом взаємодії індивіда з певним фізичним стимулом. Коли індивід сприймає образ, він реалізує процес індуктивного виводу та встановлює асоціативний зв'язок між своїм сприйняттям і певними узагальненими поняттями або «орієнтирами», наданими їм на підставі минулого досвіду. По суті розпізнавання людиною образів можна звести до питання оцінки відносних шансів на те, що вихідні дані відповідають одній або кільком із відомих множин статистичних сукупностей, що визначаються минулим досвідом людини і надають орієнтири і апріорну інформацію для розпізнавання. Таким чином, завдання розпізнавання образів можна розглядати як задачу встановлення відмінностей між вихідними даними, причому не за допомогою ототожнення з окремими образами, але з їх сукупностями.

Розпізнавання образів можна визначити як віднесення вихідних даних до певного класу за допомогою виділення істотних ознак або властивостей, які

характеризують ці дані, із загальної маси несуттєвих деталей. Прогноз погоди можна інтерпретувати як задачу розпізнавання образів. Вихідні дані в цьому випадку приймають вид синоптичних карт. Система інтерпретує їх, виділяючи істотні ознаки і формуючи на їх основі прогноз. Постановку медичного діагнозу також можна розглядати як задачу розпізнавання образів. Симптоми служать вихідними даними для системи, що розпізнає, яка на основі їх аналізу ідентифікує захворювання. Система розпізнавання символів являє собою систему розпізнавання образів, в яку в якості вихідних даних вводяться оптичні сигнали і яка ідентифікує назви символів. В системі розпізнавання мови вимовлене слово ідентифікується за допомогою аналізу сприйнятого системою звукового сигналу.

Розділяють три основних групи методів розпізнавання образів:

1. Порівняння з зразком. До цієї групи належать структурні методи та методи, які використовують наближення і відстань .
2. Статистичні методи. Прикладом цієї групи служить метод Баєса для прийняття рішення. Статистичні методи засновані на обчисленні ймовірності.
3. Нейронні мережі. Окремий клас методів розпізнавання. На відміну від інших методів, нейронні мережі здатні навчатися вже в процесі розпізнавання і володіють хорошим потенціалом розвитку.

Нижче розглянемо класифікацію цих методів.

- **Контурний аналіз.**

1. Метод активних контурів: активні контури широко застосовуються в задачах виділення контурів, меж і сегментації зображень. Для виявлення контурів на зображенні тут використовуються криві мінімальної енергії, або змійки. Алгоритм наступний: спочатку контур ініціалізується як проста лінія, а потім він деформується для створення області об'єкта. Точки в контурі прагнуть до кордону об'єкта при мінімізації енергії контуру. Для кожної точки v_i енергія

$$E_i = \alpha E_{int}(v_i) + \beta E_{ext}(v_i); \qquad (1.1)$$

де α, β - константи, що забезпечують відносну корекцію енергії; $E_{int}(v_i)$ - функція енергії, що залежить від форми контуру; $E_{ext}(v_i)$ - функція енергії, що залежить від властивостей зображення і типу градієнта в околиці точки.

Величини E_i, $E_{int}(v_i)$, $E_{ext}(v_i)$ є квадратними матрицями. Значення в центрі кожної матриці відповідає енергії контуру в позначці v_i.

Кожна вершина v_i потенційно може перейти в будь-яку точку v'_i, відповідну мінімальної енергії E_i.

2. Метод активних контурів без попереднього виділення кордонів: на відміну від звичайного методу активних контурів цей метод не вимагає попереднього виділення кордонів об'єкта зображення, а вихідне зображення не обов'язково згладжувати. Крива, або змійка (замкнутої округлої форми), рухається з довільної точки зображення. При перетині кордону вона починає деформуватися і приймати форму об'єкта на зображенні, як би заповнюючи внутрішню його частину [5].

3. Відстеження контурів: метод полягає в послідовному кресленні кордону між об'єктом і фоном. Точка, яка відстежує контур повзає по зображенню до тих пір, поки не доходить до темної області (об'єкту). Тоді вона повертається ліворуч і рухається по кривій, поки не досягне меж об'єкту, після цього повертається направо і повторює процес, поки не досягне початкової точки

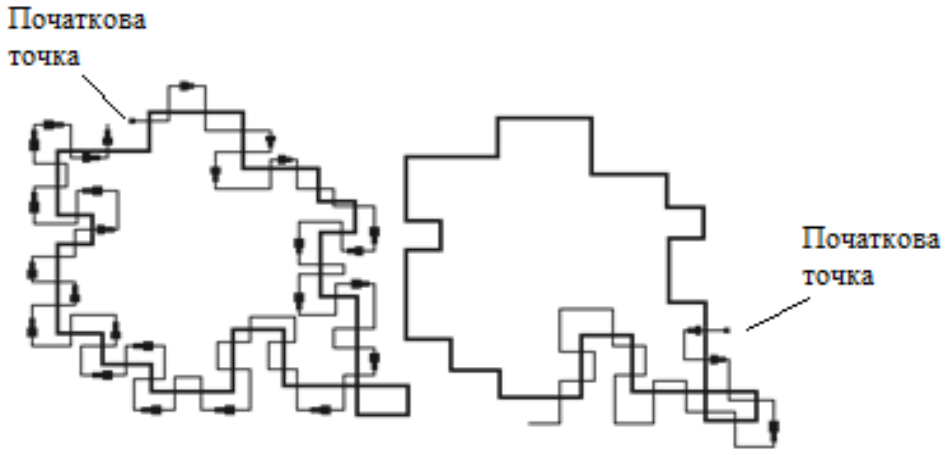


Рисунок 1.1 – Метод відстеження контурів

4. Локальна обробка. Методи виявлення кордонів виділяють в зображенні тільки пікселі, що лежать на контурі. На практиці це безліч пікселів рідко відображає контур досить точно через шумів, розривів контурів через неоднорідність освітлення. Тому алгоритми виявлення контурів зазвичай доповнюються процедурами зв'язування, щоб сформувати безлічі контурних точок.

Один із способів зв'язування точок контуру полягає в аналізі характеристик пікселів в невеликій околиці кожної точки зображення, яка була відзначена як контурна. Всі точки, які є подібними відповідно до деякими критеріями, зв'язуються і утворюють контур, що складається з відповідають цим критеріям пікселів. При цьому використовуються два основних параметри для встановлення подібності пікселів контуру: відгук оператора градієнта, що визначає значення пікселя контуру, і напрямок вектора градієнта.

Піксель контуру (x_0, y_0), розташований всередині заданої околиці точки (x, y), вважається схожим з пікселем (x, y) по модулю градієнта, якщо:

$$\nabla f\ (x\ y,) -\ \nabla f(\ x_0, y_0) \leq\ E; \tag{1.2}$$

де E - заданий невід'ємних поріг, і по напрямку градієнта, якщо

$$\alpha(x\ y,) -\ \alpha(\ x_0, y_0) \leq\ \text{A}; \tag{1.3}$$

де $\alpha\ (x, y) =\ \ arctg\ \frac{\partial x}{\partial y}$; A - заданий невід'ємний кутовий поріг.

Піксель в заданій околиці об'єднується з центральним пікселем (x, y), якщо виконані критерії подібності і за значенням, і за напрямком. Цей процес повторюється в кожній точці зображення з одночасним запам'ятовуванням знайдених пов'язаних пікселів при русі центру міста.

Простий спосіб обліку даних полягає в тому, що кожному безлічі пов'язують пікселів контуру присвоюється своє значення яскравості [5].

5. Аналіз за допомогою графів: підхід до виявлення і зв'язування контурів на основі подання у вигляді графа і пошуку на цьому графі шляхів з найменшою вартістю, які відповідають значущим контурам, дозволяє побудувати метод, добре працює в присутності шуму. Така процедура виявляється досить складною і вимагає великого часу обробки [4].

Елемент контуру - межа між двома пікселями p і q, які є сусідами.

Елементи контуру ідентифікуються координатами точок p та q. На рис. 1.2 елемент контуру визначається парами (xp, yp)(xq, yq).

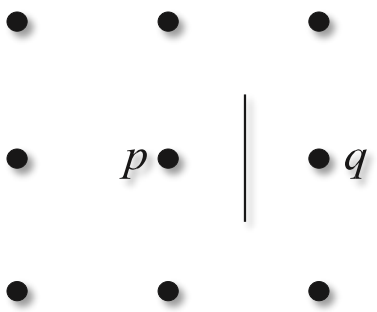


Рисунок 1.2 – Елемент контуру, що знаходиться між пікселями p і q

Контур - послідовність з'єднаних між собою елементів. Кожному елементу контуру, заданому пікселями p і q, відповідає деяка вартість

$$c(p, q) = H - [f(p) - f(q)], \quad (1.4)$$

де H - максимальний рівень яскравості в зображенні; f (p), f (q) - яскравості пікселів p і q відповідно.

Завдання відшукання на графі шляху мінімальної вартості є нетривіальною по обчислювальної складності, тому доводиться жертвувати оптимальністю на користь швидкості обчислень.

Складність реалізації і велика ресурсомісткість - основні недоліки такого аналізу, перевагою якого є слабка чутливість до шумів [3].

- **Метод відстані до найближчого сусіда**. Цей метод відносить невідомий вектор ознак до класу, окремі зразки якого знаходяться ближче всіх. Такі зразки називаються найближчими сусідами. При класифікації за найближчому сусіду не потрібно знати моделей розподілу класів в просторі, необхідна тільки інформація про еталонні зразки.

Принцип роботи алгоритму побудований на визначенні мінімальної відстані до зразка ознаки з бази даних. Так само рішення можна поліпшити, якщо шукати серед сусідів. Для класифікації кожного з об'єктів вибірки необхідно послідовно виконати наступні операції:

- Обчислити відстань до кожного з об'єктів навчальної вибірки.

- Відібрати декілька об'єктів навчальної вибірки, відстань до яких мінімальна.

- Клас об'єкту, що класифікується - це клас, який найчастіше трапляється серед декількох найближчих сусідів.

Переваги даного підходу:

- коли завгодно можна додати нові зразки;

- деревовидні та растрові структури даних зменшують кількість обчислених відстаней.

Структурний. Структурний (синтаксичний) підхід застосовується до завдань розпізнавання образів, в яких важлива інформація про структуру конкретного об'єкта. Від процедури розпізнавання потрібно не тільки, щоб вона могла визначити об'єкту його клас, а й виявити таку інформацію про об'єкт, яка не дозволяє віднести його до іншого класу[1].

В рамках цього підходу вважається, що образи складаються з різних підобразів, також як фрази і пропозиції будуються шляхом з'єднання слів.

Мова опису образів - мова, яка забезпечує структурний опис образів в термінах безлічі непохідних елементів і операцій композиції цих елементів.

Структурний розпізнавання образів часто реалізується за допомогою алгоритмів зіставлення графів. Саме відношення між двома примітивами може використовуватися як елементарний ознака і входити в вектор ознак.

Найпростіший спосіб - порахувати кількість заданих відносин між двома ознаками різних типів. Метод буде корисний для розпізнавання складних образів, що складаються з багатьох простих.

- **Баєсівський підхід.** Якщо розглядати векторне подання образів, то в загальному випадку, вектор ознак складається з компонент ознак, кожен з яких і їх сукупність в цілому характеризують образ з тим або іншим ступенем визначеності. Ця невизначеність може, зокрема, носити ймовірнісний характер.

Поява того чи іншого способу є випадковою подією і ймовірність цієї події можна описати за допомогою закону розподілу ймовірностей багатовимірної випадкової величини ξ в тій чи іншій формі. Вид і параметри функції щільності визначаються конкретної середовищем, в якій працює система розпізнавання. Знаючи елементи навчальної вибірки можна відновити імовірнісні характеристики цього середовища. Баєсівський класифікатор на основі спостережуваних ознак відносить об'єкт до класу, до якого цей об'єкт належить з найбільшою ймовірністю[2].

- **Нейронні мережі.** Розглянуті до поточного моменту методи розпізнавання були засновані на навчанні з людиною. Однак навчання без втручання людини також можливо. При такому підході машина сама повинна визначити структуру класів і приналежність кожного образу певному класу. Нейронна мережа - це метод параметричної апроксимації, що довів свою корисність в побудові моделей щільності.

Нейронні мережі зазвичай апроксимують деяку векторну функцію f деякого вхідного вектора x поруч рівнів. Кожен рівень формує вектор виходів, кожен з яких представляє собою результат дії певної нелінійної функції.

При розпізнаванні методом найближчого сусіда результат методу був повністю визначений еталонними зразками, завантаженими в пам'ять перед обробкою. Тоді як при застосуванні нейронного методу розпізнавання кінцевий результат заздалегідь не відомий. Алгоритми нейронний мереж самі вирішують задачу вибору ознак. З цим пов'язаний один з недоліків - перенавчання.

Нейронні мережі добре підходять для розпізнавання осіб. Якщо розглянути безліч осіб у фронтальній проекції видно, що практично всі особи виглядають самотнього (при досить грубій шкалою). Є світлі і темні області. Так наприклад очі темніше чола, щоки світліше носа. Безліч таких ознак дозволяють використовувати нейронні мережі в задачі розпізнавання осіб. Стандартні нейронні мережі показують результат гірше ніж у алгоритму Віоли-Джонса[3].

1.2 Аналіз методів розпізнавання тексту

Класифікація методів розпізнавання тексту:

1. **Метод аналізу проекційних профілів.** Цей підхід ґрунтується на припущенні, що текст побудований уздовж паралельних прямих. Принципова схема алгоритму цього методу передбачає розрахунок профілю проекції для кожного кута нахилу, визначення функції та вибір кута, який приводить його в оптимальний стан. Цей підхід вимагає відносно великої кількості обчислювальної потужності.
2. **Метод перетворення Хафа.** Перетворення Хафа – чисельний метод, який застосовується для вилучення елементів з зображення[4]. Використовується в аналізі зображень, цифровій обробці зображень і комп'ютерному зорі. Призначений для пошуку об'єктів, що належать певному класу з використанням процедури голосування.

Процедура голосування застосовується до простору параметрів, з якого і виходять об'єкти певного класу по локальному максимуму в, так званому, накопичувальному просторі, яке будується при обчисленні трансформації Хафа.

Класичний алгоритм перетворення Хафа пов'язаний з ідентифікацією прямих в зображенні, але пізніше алгоритм був розширений можливістю ідентифікації позиції довільної фігури, найчастіше, еліпсів і кіл. Перетворення Хафа, в тому вигляді, якому воно використовується тепер, було винайдено в 1962 році[5].

Мінуси алгоритму:

1) Перетворення Хаффа впливає лише на велику кількість «ударів» у відповідний елемент простору, і тоді ви можете сміливо ідентифікувати елемент і ігнорувати фоновий шум. Це означає, що розмір елемента не повинен бути надто малим, оскільки в іншому випадку деякі значення потрапляють до сусідніх елементів і видимість потрібного елемента знижується.

2) Якщо кількість параметрів становить три і більше, середня кількість звернень у елементі невелика, і тому потрібний нам елемент суттєво не відрізняється від сусідніх елементів. Тому алгоритм слід використовувати з максимальною обережністю, щоб уникнути ідентифікації абсолютно непотрібного елемента.

3) У більшості випадків продуктивність алгоритму залежить від якості вхідних даних: межі елементів на фазі попередньої обробки повинні бути чітко визначені. Використання перетворення Хаффа для зображень шуму є складним. Для зображень з високим рівнем шуму необхідний крок попередньої обробки для придушення шуму.

3. **Методи кластеризації найближчих сусідів.** Методи цього класу ґрунтуються на загальному припущенні, що символи вирівнюються в ряд і розташовуються близько один до одного. Вони характеризуються обробкою знизу вгору, яка починається з багатьох об'єктів, з'єднаних компонентів або їх точок, і використовує їх взаємні відстані та просторові відносини для оцінки кута нахилу.

Також текст розпізнають за допомогою нейронних мереж.

Використання штучних нейронних мереж пропонує високу якість розпізнавання. До переваг методу можна віднести те, що він фіксує приховані зв’язки між ознаками переданого йому зображення. У процесі навчання нейронної мережі немає необхідності ретельно вивчати ці сполуки. Досить стежити за результатом нейронної мережі. Метод також дозволяє підвищити якість розпізнавання за рахунок збільшення кількості переданих даних - особливостей зображення.

Основна проблема методу - ресурсомісткість. Щоб обробити один символ, ви повинні обробити всі нейрони та зв’язки між ними. Крім того, виявлення кешем забезпечується шляхом визначення зв’язків між ознаками, на які безпосередньо впливає мережева структура, та кількістю нейронів, тому зменшення кількості нейронів, як правило, неприпустимо.

1.3 Аналіз методів попередньої обробки зображень

Обробка зображень, зазвичай, передує операції розпізнавання певних об'єктів на зображеннях для створення умов, які підвищують ефективність і якість цього процесу.

Задача розпізнавання ділиться на кілька великих підзадач:

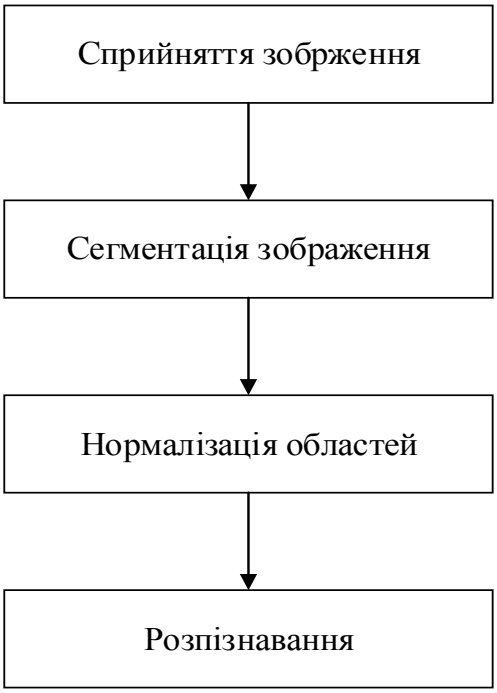


Рисунок 1.3 – Етапи роботи різного роду алгоритмів розпізнавання

Основним завданням розпізнавання є – зрозуміти, чи належать дані на зображенні до шуканих об'єктів. Основними технічними складнощами, що можуть виникнути в даному випадку є:

1. Вхідне зображення має складне тло;
2. Вхідне зображення має цифровий шум;

3. Різні області зображення мають різні характеристики (засвічення, різну освітленість).

Для вирішення завдання на різних етапах застосовують різні підходи і методи сегментації, нормалізації і розпізнавання. Розглянемо їх більш детально:

1. Попередня обробка - майже завжди використовується після видалення інформації із зображення, щоб зменшити шум зображення шляхом вибірки, квантування та придушення зовнішнього шуму. Зазвичай ці операції використовуються для усереднення та вирівнювання гістограм.

2. Сегментація відноситься до знаходження однорідних областей на зображенні. Ця фаза є дуже складною та не алгоритмізованою для зображення. Більшість методів сегментації були створені для визначення однорідних кольорів, текстур.

3. Покращення / фільтрація – може застосовуватися після процесу сегментації. Знижує перешкоди на зображенні, які були результатом вибірки та квантування, чи усунення зовнішнього шуму.

4. Розпізнавання – у більшості випадків, останній етап обробки, що лежить в основі процесів інтерпретації і розуміння. Вхідними для розпізнавання є зображення, виділені в результаті сегментації і, частково, відреставровані, вони відрізняються від еталонних зображень геометричними спотвореннями та спотвореннями яскравості, також на них можуть залишитися шуми.

Методи попередньої обробки досить різноманітні та включають в себе:

- виділення найбільш інформативних фрагментів,
- збільшення інформативних фрагментів,
- отримання 3-мірних зображень,
- картування кольору,
- підвищення розподільної здатності,
- підвищення контрастової здатності,
- покращення якості зображень.

Корекція яскравості і контрастності зображень. Зображення, введені у ваш комп'ютер, часто мають низьку контрастність. Низький контраст, зазвичай

викликаний широким діапазоном відтворюваної яскравості, часто поєднується з нелінійністю характеристик передачі рівнів. Характер залежності зміни яскравості палітри пікселів від мінімального значення до максимального також впливає на якість зображення. Функція зміни інтенсивності лінійного пікселя є оптимальною. З увігнутою характеристикою зображення темніше, з опуклою – світліше. В обох випадках особливості об’єктів можуть бути спотворені та не чітко визначені. Корекція яскравості палітри значною мірою покращує якість зображення[6].

Низький контраст може бути пов’язаний з тим, що варіації функції яскравості пікселів на зображенні значно менше, ніж прийнятний діапазон шкали яскравості. У цьому випадку контрастність зображення збільшується за рахунок «розтягування» реального динамічного діапазону яскравості на всю шкалу через лінійне до елементарне перетворення.

Ще одним способом регулювання яскравості палітри є інвертування вхідного зображення. Оскільки важко відрізнити слабкі сигнали на темному тлі, обернене зображення таких зображень має іншу гістограму яскравості, більш прийнятну для спостереження та візуальної ідентифікації.

У деяких завданнях з обробки зображень напівтонове зображення (багато градацій яскравості) перетворюється на бінарне зображення (дві градації). Перетворення виконується для зменшення надмірності інформації зображення, залишаючи лише інформацію, необхідну для вирішення конкретної задачі. Двійкове зображення повинно зберігати певні деталі (наприклад, контур зображених об'єктів) та виключати підпорядковані функції (фон).

Обробка порогу напівтонового зображення складається з підрозділу всіх елементів зображення на основі яскравості з кордоном А-межі на два класи А1 та А2 та виконання відповідної порогової фільтрації із заміною пікселів зображення на встановлену яскравість класів. Вибір кордону визначається типом яскравості рядка вихідного зображення. Для найпростіших зображень, таких як креслення, машинописні тексти з бімодальним розподілом, межа між режимами розподілу зводиться до мінімуму. У загальному випадку зображення може бути

багатомодальним, і якщо встановлюється досить надійне відповідність між об'єктами і відповідними модами їх яскравості, то порогова фільтрація також може передбачати кілька класів яскравості пікселів[6].

Діапазон яскравості зображення на комп'ютері може відрізнятися від початкового діапазону яскравості, наприклад через недостатню експозицію. Є два способи регулювання яскравості. Після першого методу зображення відображається лінійно в діапазоні яскравості виводу. Другий метод передбачає обмеження яскравості пікселів в оброблюваному зображенні до максимального та мінімального порогів, і використовується ширше. Наявність зображення в найяскравіших і темних тонах створює враження хорошого контрасту, але надмірний контраст спричиняє максимальні градації на півтони, і більшість деталей зображення намальовані в півтонах, а надмірний контраст може призвести до втрати цих деталей вести або ускладнювати їх вибір.

Гістограми яскравості. Інструментом для оцінки рівнів інтенсивності пікселів є гістограма - графік кількісної характеристики розподілу ймовірності інтенсивності (яскравості) пікселів у вибраній області зображення. Максимальному значенню інтенсивності пікселів присвоюється рівень градації інтенсивності 255 (білий), а найбільш темне значення - 0 (чорне). Інтенсивності в діапазоні від 0 до 255 мають лінійну шкалу змін або встановлюються відповідно до прийнятої функції зміни, z діапазони інтересів.

Відомий метод покращення зображень, заснований на обчисленні логарифма спектральних коефіцієнтів перетворення Фур'є вихідного зображення (обчислення кепстра). Кепстр – це спектр логарифма спектра початкового сигналу[7]. При зворотному перетворенні кепстра в зображення відбувається вирівнювання гістограми зображення за рахунок логарифмічного перетворення спектру зображення.

Для багатьох зображень характерні гістограми з високою лінійною концентрацією в певних зонах розподілу інтенсивності. Часто гістограма розподілу яскравості зображення зміщується в бік менших значень (більшість елементів нижче середнього). Одним із способів поліпшити якість таких

зображень є зміна їх гістограми. Вирівнювання гістограми може здійснюватися на основі суми спектральних коефіцієнтів модуля перетворення Фур'є зображення при збереженні знакових та фазових коефіцієнтів. Якщо позначити показник ступеня α, то при α <1 операція витягання кореня ступеня α зменшує великі спектральні коефіцієнти і збільшує малі. Такий перерозподіл енергії в частотній площині зображення призводить до більш ефективного використання динамічного діапазону інтенсивностей пікселів зображення в просторовій області[6].

Вибір хорошої маски для регулювання гістограми по інтенсивності пікселів покращує контрастність, що покращує роздільну здатність контрасту частин. У програмах обробки є команди, які дозволяють регулювати кольори, коли ви співставляєте кольори зображень з плавними або зворотно різкими переходами відображених деталей у потрібній області. У поєднанні з контрастним контрастом, який перетворює негативне зображення в позитивне зображення, цей спосіб також покращує контраст дрібних та середніх деталей зображення.

Існує значний арсенал математичних моделей та алгоритмів, реалізація програмного забезпечення яких може значно покращити роздільну здатність контрасту зображень. Ці алгоритми засновані на лінійних та нелінійних процесах фільтрації зображень, які перетворюють гістограму інтенсивності.

Вирівнювання освітленості зображень. Часто деякі ділянки на малюнку занадто темні, щоб їх не було видно. Якщо ви додасте яскравість усьому зображенню, то спочатку можуть бути освітлені яскраві області. Для поращення зовнішнього вигляду зображення в таких випадках використовується метод вирівнювання світла.

Освітлення повільно змінюється в приміщенні і може розглядатися як сигнал низької частоти. Ця ж картина може розглядатися як середній сигнал високої частоти. Якщо ці сигнали були додані під час фотографування, їх можна було б розділити звичайними фільтрами. Однак, власне фото - це твір зображення, яке ми хочемо побачити, і світлих карт. А оскільки ці сигнали не

складаються та не примножуються, усунути нерівномірність освітлення неможливо простою фільтрацією.

Для вирішення таких завдань застосовується гомоморфна фільтрація. Гомоморфна фільтрація – це узагальнена техніка для цифрової обробки сигналів і зображень, за участю нелінійного відображення в інших просторах, в яких теорія лінійних фільтрів може бути застосована і відображена назад у первинний простір[9]. Ідея обробки полягає в зведенні нелінійної задачі до лінійної. Наприклад, можна звести задачу поділу перемноження сигналів до задачі розділення складених сигналів. Для цього потрібно взяти логарифм від твору зображень, який буде дорівнює сумі логарифмів співмножників. При цьому завдання поділу твори сигналів зводиться до задачі розділення суми НЧ-і ВЧ-сигналів і вирішується за допомогою ВЧ-фільтра, який видалить з суми сигналів низькі частоти. Залишиться взяти від отриманого сигналу експоненту, щоб повернутися до вихідного масштабу амплітуд[6].

ВЧ-фільтр можна реалізувати наступним чином. Спочатку до зображення застосовується операція розмиття (НЧ-фільтр), а потім з вихідного зображення віднімається розмите. Найкращий радіус розмиття залежить від конкретного зображення. Можна почати експерименти з радіусу близько десяти пікселів.

Зазвичай для розмиття зображення застосовується двовимірний гауссівский фільтр, що має вигляд :

$$y(m,n) = \frac{1}{2\pi r^2}\sum_{u,v} e^{\frac{-(u^2+v^2)}{2\gamma^2}}\, x(m+u, n+v); \qquad (1.5)$$

Прямий розрахунок двовимірної згортки з таким ядром вимагає великих обчислень навіть при порівняно невеликому розмірі ядра. Однак еквівалентний ефект можна отримати, спочатку фільтруючи лінії зображення, а потім стовпці отриманого зображення з одновимірним гауссовим значенням. Ефект корекції освітлення може бути занадто сильним (темні ділянки мають таку ж яскравість,

як і світлі). Для зменшення ефекту можна просто змішати відредаговане зображення у певному співвідношенні з оригіналом.

Збільшення розподільної здатності. Інтерпретація зображень тісно пов'язана з якістю представлення дрібних невикривлених частин. У той же час, коли фрагменти збільшуються, потрібно, щоб роздільна здатність зображення не погіршувалася при виконанні математичних операцій 2D інтерполяційної функції просторового розподілу інтенсивності пікселів у рядках і стовпцях матриці зображення. Важливим фактором ідентифікації об'єктів є положення та відображення областей однакової яскравості чи кольору, навіть якщо ці області мають розмір декількох пікселів[6].

Чіткість зображень у професійних програмах зазвичай регулюється шляхом пошуку оптимальних значень яскравості та контрасту, шляхом вибору відповідних параметрів:

а) «величини» - ступеня впливу на різкість зображення;

б) «радіусу» величина різкості контуру;

в) «порогу дискримінації» - визначення країв об’єктів, визначивши значення інтенсивності сусідніх пікселів, що буде достатньо для збільшення контрасту між ними.

Деякі програми автоматично встановлюють оптимальне співвідношення контрастності та яскравості для забезпечення бажаної чіткості зображення.

При обробці зображень алгоритми фільтрування рангів відіграють важливу роль у наданні можливості усунути нечіткі деталі (покращити їх фокус) шляхом вибору двовимірної маски n×n пікселів, виконання операцій зі значеннями інтенсивності пікселів у межах заданої маски та призначення центрального значення пікселя. Середня техніка фільтрації, яка виключає некорельовані випадкові сигнали та імпульсний шум без "розмивання" різких відмінностей у краях об'єкта, також класифікується.

Інформаційне застосування лінійної фільтрації пояснюється її здатністю виправляти різного роду спотворення, що виникають внаслідок недосконалості пристроїв візуалізації. Лінійна фільтрація зменшує вплив флуктуаційного шуму

та інших дефектів на відтворювані зображення з низькоконтрастними частинами, що підсилюють контраст при збільшенні масштабу інтерфейсу[6].

Корекція апертурних спотворень зображень в разі відсутності шумової складової сигналу здійснюється шляхом інверсної фільтрації (зворотньої згортки). Зворотня згортка, розгортка — математична операція, зворотна згортці сигналів, зворотня згортка широко використовується в обробці сигналів, і зображень, а також для інших інженерних і наукових застосувань[8]. Однак слід враховувати, фільтри зворотньої згортки мають коефіцієнт посилення дисперсії шумів, більший 1, і замість покращення зображення може збільшитися зашумленість.

Перспективними є нелінійні методи фільтрації, які засновані на частотних масках, які дають можливість зменшити вплив низькочастотних компонентів сигналу зображення на позитивні ефекти високочастотних компонентів та покращити однакове просторове розділення частин. Також досить перспективним є використання детектора Кенні.

1.4 Висновки

В даному розділі було детально проаналізовано методи розпізнавання образів, тексту та інформації вцілому. Також проаналізовано основні методи та алгоритми попередньої обробки зображень.

2 АНАЛІЗ ЗАСОБІВ РОЗРОБКИ

2.1 Аналіз предметної області застосування та розробки

Контур має у собі всю потрібну інформацію про форму об’єкта. Це обмежує область алгоритмів контурного аналізу, однак контурність дозволяє перейти з двовимірного простору зображення в контурний простір, що зменшує складність обчислень та алгоритмів. Контурний аналіз дозволяє ефективно вирішувати основні проблеми розпізнавання. Системи розпізнавання використовують різні методи контурного кодування. Найпопулярнішим є код Фрімена, 2Д та полігональне кодування. Однак контурний аналіз використовує не всі ці методи кодування. Натомість у ньому контур кодується послідовністю складних чисел. По контуру фіксується точка, яка називається початковою точкою. Потім контур обходить по колу (передбачається за годинниковою стрілкою) і кожен вектор переміщення записується комплексним числом a + ib. Де *a* - зсув точки по осі X, а *b* - зсув по осі Y.

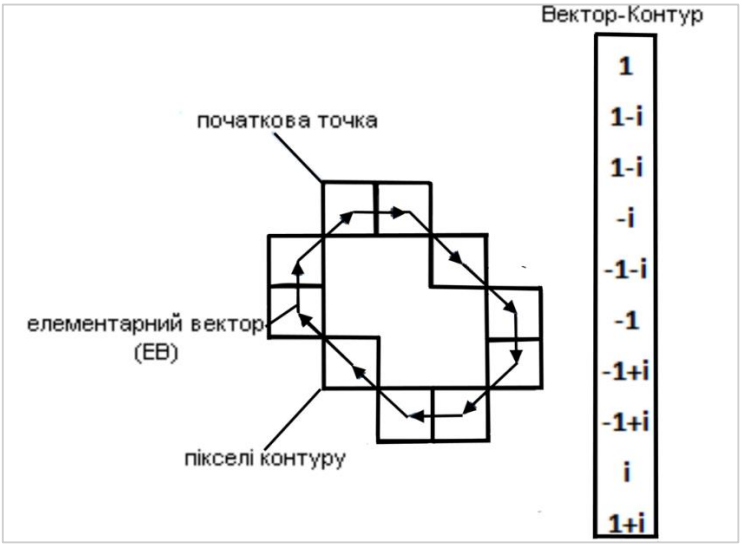


Рисунок 2.1 – Приклад визначення контуру

Через характер тривимірних об’єктів їх контури характеризуються постійною закритістю та не перетинаються між собою. Таким чином, ви можете чітко визначити шлях проходу по контуру. Останній контурний вектор завжди

веде до вихідної точки. Кожен вектор контуру називається елементарним вектором. А послідовність складних чисел - це векторний контур.

Контурний аналіз використовує найскладніше кодування, оскільки операції як по контуру, так і по вектору складних чисел мають відмінні математичні властивості порівняно з іншими методами кодування. Складне кодування наближається до двовимірного кодування, в якому контур визначається як набір елементарних векторів, представлений їх двовимірними координатами. Однак різниця полягає в тому, що робота добутку різна, як для векторів так і для складних чисел, тому це допомагає застосуванні методів контурного аналізу.

Властивості контурів:

1. Сума елементарних векторів із замкнутим циклом дорівнює нулю, це тривіально, тому що елементарні вектори прямують до початкової точки і при цьому їхня сума дорівнює нульовому вектору.

2. Вектор контуру не залежить від паралельної передачі вихідного зображення, так як контур кодується відносно вихідної точки.

3. Повертання зображення на той, чи інший кут так само відповідає повертанню елементарного вектора на той самий кут.

4. Зміна початкової точки призводить до циклічного зсуву контуру вектора, так як елементарні вектори кодуються відносно попередньої точки.

Контурний аналіз має дві групи факторів, які негативно впливають на результати розпізнавання. Перша група факторів пов'язана з проблемою виділення контуру на зображеннях. Контур - строго дискретна структура. Однак багато реальних зображень містять об'єкти, які погано відображаються на навколишньому тлі. Об'єкт може не мати чіткої межі, він може бути ідентичним за яскравістю та кольором фону, це можуть бути шумні перешкоди і т.д. Усі ці фактори призводять до того, що контур або не розрізнений, або не виділений належним чином. Незважаючи на недоліки, методи контурного аналізу привабливі своєю простотою та швидкістю.

Щоб зробити контур більш чітким і легшим розпізнати, перед аналізом контуру необхідно використовувати відповідні фільтри, які виділяють межі зображення та роблять їх однотонними. Після цього процесу контурний пошук виконується більш точно.

Існує кілька загальновживаних алгоритмів виділення границь на зображенні.

Найпоширеніший з цих алгоритмів - оператор Кенні (детектор границь Кенні, алгоритм Кенні). Він був розроблений в 1986 році Джоном Ф. Кенні і використовує багаторівневий алгоритм для виявлення широкого спектра кордонів у зображеннях.

Дж. Кенні дослідив математичну проблему отримання фільтра, який є оптимальним за критеріями вибору, розміщення та мінімізації кількох відповідей з одного краю. Це означає, що детектор повинен реагувати на межі, але в той же час ігнорувати помилкові, точно визначати граничну лінію і один раз реагувати на кожну межу, щоб уникнути сприйняття широких смуг зміни яскравості як набору меж.

Алгоритм включає в себе:

- згладжування - розмиття зображення для видалення шуму;
- пошук градієнтів - межі відзначаються там, де градієнт зображення набуває максимальне значення;
- придушення немаксімумов - тільки локальні максимуми відзначаються як кордони; подвійну порогову
- фільтрацію - потенційні межі визначаються порогами;
- трасування області - неоднозначності підсумкові кордону встановлюються шляхом придушення всіх країв, не пов'язаних з визначеними межами.

Для зменшення чутливості алгоритму до шуму застосовується перша похідна від Гауссіана.

На рис. 2.2 показано застосування оператора Кенні.

Рисунок 2.2 – Зображення застосування оператору Кенні.

Іншим алгоритмом отримання меж зображення є так званий оператор Прюітт – метод вибору ребра при обробці зображення, який обчислює максимальну відповідь на кілька ядер згортання для пошуку локальної орієнтації межі в кожному пікселі. Він був створений доктором Джудіт Прюітт(Judith Prewitt) для визначення кордонів медичних зображень. Для цієї операції використовуються різні ядра.

Граничний детектор Прюітт - це зручний спосіб оцінити розмір та орієнтацію кадру. У той час як детектор диференціального градієнта вимагає трудомісткого розрахунку орієнтації розміру орієнтації у вертикальному та горизонтальному напрямках, тестовий детектор границь вказує напрямок безпосередньо від серцевини, щоб отримати максимальний результат. Набір ядер обмежений 8 можливими напрямками, проте досвід показує, що більшість прямих оцінок орієнтації теж не дуже точні. З іншого боку, набір ядра вимагає 8 витків на піксель, тоді як для градієнтного набору ядер потрібно лише 2: вертикально та горизонтально чутливі. Результат представлення меж обох методів дуже подібний при використанні одних і тих же ядер згортки.

Ще одним є оператор Собеля. По суті, це дискретний диференціальний оператор, який обчислює приблизне значення градієнта яскравості зображення. Результатом використання оператора Собеля в кожній точці зображення є або вектор градієнта яскравості в цій точці, або його норма. Оператор Собеля заснований на згортанні зображення невеликими цілими фільтрами у

вертикальному та горизонтальному напрямках, тому обчислення відносно просте. Градієнт функції двох змінних для кожної точки зображення є двовимірним вектором, компоненти якого виведені горизонтально і вертикально від яскравості зображення. У кожній точці зображення градієнтний вектор вирівнюється у напрямку найбільшого збільшення яскравості і його довжина відповідає величині зміни яскравості. Це означає, що результатом оператора Собеля є нульовий вектор у точці постійного діапазону яскравості, а в точці, що знаходиться на межі областей різної яскравості - вектор, що перевищує межу в напрямку збільшення яскравості.

2.2 Вибір засобів розпізнавання

Теорія розпізнавання образів — розділ кібернетики, що розвиває теоретичні основи й методи класифікації і ідентифікації предметів, явищ, процесів, сигналів, ситуацій і т. п. об'єктів, які характеризуються скінченним набором деяких властивостей і ознак[9].

Враховуючи швидкодію, швидкість розробки та результати для пошуку границь на зображенні було обрано оператор Собеля.

Оператор Собеля - це дискретний диференціальний оператор, який обчислює наближення градієнта яскравості зображення[24]. Даний оператор обчислює градієнт яскравості зображення в кожній точці, тобто це є напрямок найбільшого збільшення яскравості і величина його зміни в цьому напрямку. Результат показує, наскільки сильно або навіть наскільки яскраво змінюється зображення в кожній точці, тобто наскільки ймовірне знайти точку на межі та як орієнтується межа. Таким чином, результатом роботи оператора Собеля є нульовий вектор в діапазоні постійної яскравості та в точці на межі областей різної яскравості, тобто векторі, що перевищує межу що перетинає межу зі збільшенням яскравості яскравості.

Власне кажучи, оператор використовує 3x3 ядра для стиснення зображень та для обчислення приблизних значень похідних по горизонталі та вертикалі. Нехай A - оригінальне зображення, а Gx і Gy - два зображення, кожне з яких

містить приблизні похідні x і y[6]. Таким чином G_x і G_y – це дві матриці які обчислюються наступним чином:

$$G_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ +1 & +2 & +1 \end{bmatrix} * A, \quad (2.1)$$

$$G_x = \begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} * A, \quad (2.2)$$

де знак * означає двомірну операцію згортки.

В матрицях (2.1, 2.2) координата x збільшується «направо», а координата y збільшується «вниз».

В кожній точці зображення наближене значення величини градієнту можна вирахувати, користуючись отриманими наближеними значеннями похідних. Для цього використовується формула:

$$G = \sqrt{G_x^2 + G_y^2}. \quad (2.3)$$

Результуючий алгоритм оператора Собеля складається з наступних етапів:

1) Створити матрицю згортки для координати x та координати y (можна використовувати одну й ту саму матрицю для обох координат);

2) Створити нове зображення такого ж розміру, як і вхідне зображення. В нове зображення буде записано нові значення пікселів;

3) Для кожного пікселя вхідного зображення слід виконати наступні етапи:

3.1) Отримати поточний піксель;

3.2) Отримати значення восьми пікселів навколо поточного (якщо пікселя не має в заданих точках, наприклад верхній лівий край зображення, тобто піксель з координатами (0, 0), замість нього записуються нулі);

3.3) Обчислити G_x та G_y за формулами (2.2) та (2.1) відповідно;

3.4) Обчислити нове значення поточного пікселя за формулою (2.3);

3.5) Записати нове значення пікселя у вихідне зображення.

Результатом використання оператора Собеля є двовимірна карта історії для кожної точки. Її можна обробити і показати як зображення, на якому будуть ділянки білого(з великим значенням градієнта) та чорного(грані повністю чорного кольору) кольору. Нижче показано зображення перед застосуванням(Рис. 2.3) та після застосування(Рис. 2.4) оператора Собеля.

Рисунок 2.3 – Зображення перед застосуванням оператора Собеля

Як вже було визначено вище для пошуку інформації на зображенні будемо використовувати методи контурного аналізу. Для коректного використання цих методів застосовано відповідні терміни.

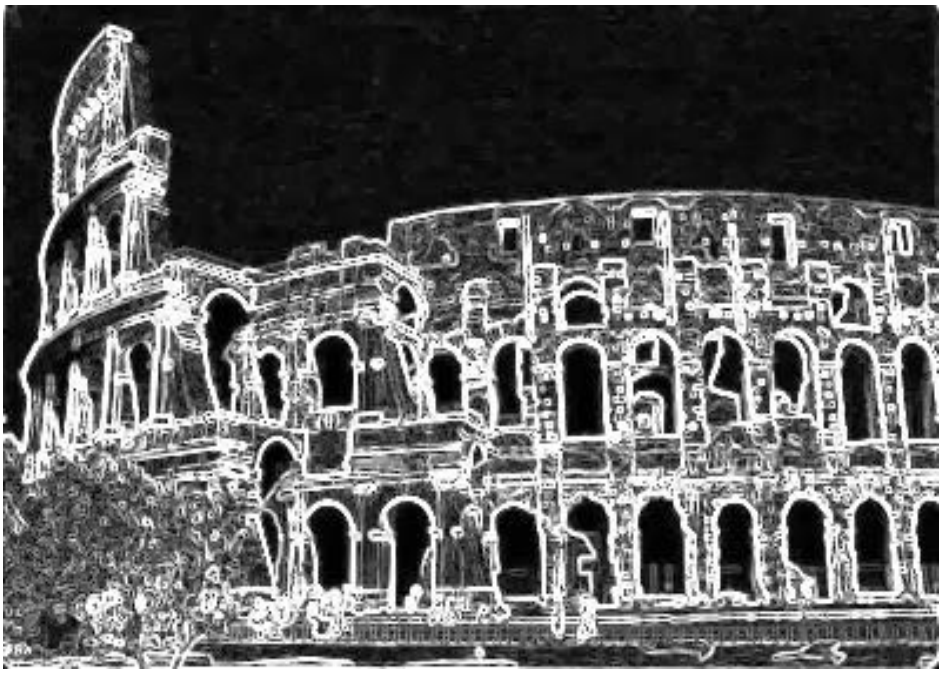


Рисунок 2.4 – Зображення після застосування оператора Собеля

Елементарний вектор γ(n) – вектор, який з’єднує центри чи вузли сусідніх контурних комірок сітчатки, який проводиться в сторону обходу; n – номер цього елементарного вектору, який відраховується від точки a_0 виявлення контуру; n = 0, 1, …, k – 1, k – кількість елементарних векторів в контурі даного зображення. Елементарний вектор показано на рис. 2.5.

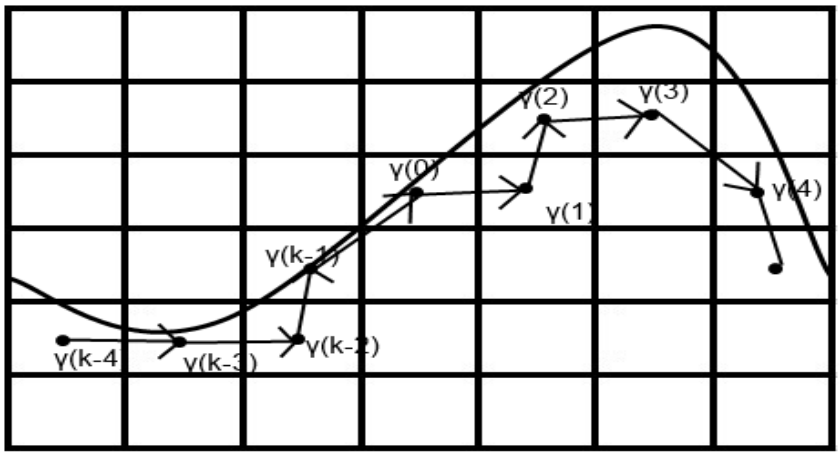


Рисунок 2.5 – Визначення контуру елементарними векторами

Елементарне направлення – аргумент елементарного вектору. Клітинка заповнюється, якщо площа зображення в ній не менше ніж 50% площі клітинки. На прямокутній сітці, показаній на рис. 2.6, відображено можливі вісім різноманітних елементарних векторів.

Для кодування контуру використано метод кодування за допомогою комплексних чисел.

На першому етапі кодування проводиться визначення елементарного вектору контуру в площині квадратної сітки комплексними числами.

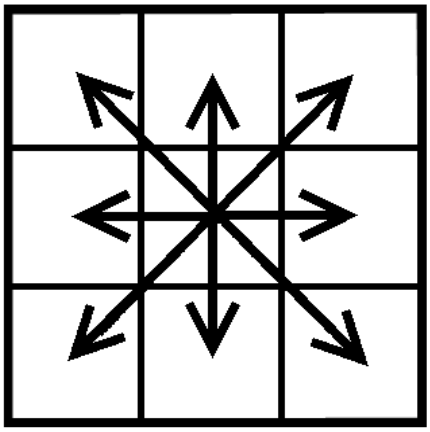


Рисунок 2.6 – Різновиди можливих елементарних векторів на квадратній сітці

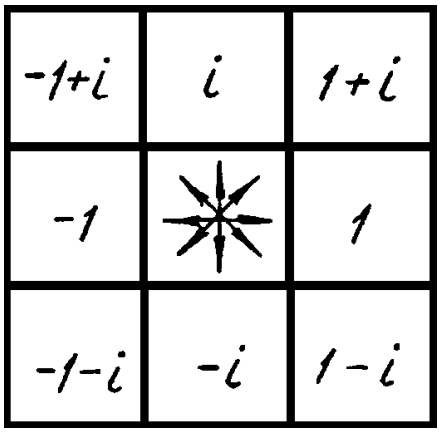


Рисунок 2.7 – Координати елементарного вектора при кодуванні комплексними числами

Було проаналізовано різні бібліотеки для обробки і розпізнавання зображень. Найбільш популярна на сьогодні – бібліотека OpenCV. OpenCV (Open Source Computer Vision Library) – це бібліотека, яка використовується для розпізнавання інформації з будь-яких джерел, що надається з відкритим програмним кодом. Вона зібрала у собі велику кількість алгоритмів для використання технологій розпізнавання та надає своїм користувачам широкі можливості. Після її підключення до свого проекту користувач отримує доступ до величезної кількості функцій(близько 2000), які призначені для вирішення різноманітних завдань. Окрім вирішення завдань розпізнавання функціонал бібліотеки містить різноманітні чисельні алгоритми для обробки зображень. Дана бібліотека реалізована на мовах програмування C / C# / C ++. Однак адаптована і для інших мов. Дана бібліотека може використовуватися на різних операційних системах, до числа яких відносяться OS Windows, Linux, Mac OS, iOS, Android та інші.

Бібліотека OpenCV здатна використовувати багатоядерні процесори.

Головна мета даної бібліотеки – надати легкий у використанні інтерфейс, який полегшить використання технологій розпізнавання в досить складних додатках. Функції, які підтримує OpenCV, охоплюють різноманітні сфери комп'ютерного зору.

Додатково було використано алгоритм під назвою Template Matching [12].

Якщо подивитися на цей алгоритм, то він заснований на швидкому перетворенні Фур'є, також він має більшу швидкість виконання, ніж звичайне перетворення Фур'є. Коли ми розглядаємо, як цей метод застосовується на практиці, отримуємо два зображення: основне та шаблон пошуку. Мета алгоритму - визначити в основному зображенні точку, в якій шаблон пошуку можна відрегулювати до максимуму.

Під час переміщення кожного пікселя зображення по шаблону отримується кількість значень перехресної кореляції, що утворюють матрицю перехресної кореляції. Таким чином, ця матриця дає координати верхнього лівого краю зображення, що нам потрібно. Ця процедура шукає максимум у рядках та

стовпцях перехресної кореляційної матриці. Тому ця точка є (максимальною) і показує максимальну відповідність референтному зображенню (шуканому).

Заповненні матриці зображення-шаблону нулями або середніми значеннями так, щоб воно збільшилось до розмірів основного зображення – це загальна технологія пошуку зображення. Отже є можливість не використовувати області, що оточують пошуковий шаблон, встановивши значення рівні нулю за його межами . Але на практиці виявляється, що заповнення нулями не дає результату. Тому що зображення може мати області, які даватимуть максимум крос-кореляційної матриці у тому місці, яке не відповідає дійсності, а отже отримаємо хибні розрахунки. Шляхом вирішення цієї проблеми є використання нормалізації [12].

Нормалізація дозволить залишити рішення заповнення матриці нулями, а також буде працювати на локальному рівні та на зображенні з більшою роздільною здатністю. Загальний алгоритм нормалізації виглядає як:

1. Знаходження середньої величини відстані між пікселями завдяки використанню низькочастотного фільтру;

2. Віднімання середньої величини від значень відстані між пікселями початкового зображення;

3. Обчислення середньоквадратичного відхилення відстані між пікселями;

4. Нормалізація цього відхилення;

В результаті отримуємо нормалізоване зображення.

2.3 Формування задач розробки

Відповідно до проведеного аналізу, були сформульовані наступні задачі до розробки:

1. Провести детальний аналіз методів та засобів розпізнавання інформації.
2. Розробити методи та інтерфейс, який буде інтуїтивно зрозумілим для користувача.
3. Розробити програмний додаток для розпізнавання інформації.
4. Протестувати розроблений програмний додаток.

2.4 Висновки

В даному розділі було проведено аналіз предметної області застосування розробки, визначено методи, які будуть використані при розробці програмного додатку. Також було обрано засоби розпізнавання. Було сформульовано задачі до розробки.

3 РОЗРОБКА ТА РЕАЛІЗАЦІЯ ПРОГРАМНОГО ДОДАТКУ ДЛЯ РОЗПІЗНАВАННЯ І АНАЛІЗУ ІНФОРМАЦІЇ

3.1 Обґрунтування вибору засобів програмування

Було розглянуто декілька мов програмування – C++, C# та Java.

Усі ці мови близькі за синтаксисом

C # - мова програмування, що поєднує об'єктно-орієнтовані і контекстно-орієнтовані концепції. Розроблено в 1998-2001 роках групою інженерів під керівництвом Андерса Хейлсберга в компанії Microsoft як основна мова розробки додатків для платформи Microsoft .NET. Компілятор з C # входить в стандартну установку самої .NET, тому програми на ньому можна створювати і компілювати навіть без інструментальних засобів на кшталт Visual Studio[13].

C++ - компільована строго типізована мова програмування загального призначення. Підтримує різні парадигми програмування: процедурну, узагальнену, функціональну; найбільшу увагу приділено підтримці об'єктно-орієнтованого програмування. Розробка мови почалася в 1979 році. Метою створення C++ було доповнення C можливостями, зручними для масштабної розробки ПО, зі збереженням гнучкості, швидкості і Портабельная C. Разом з тим творці C ++ прагнули зберегти сумісність з C: синтаксис першого заснований на синтаксисі останнього, і більшість програм на C працюватимуть і як C ++. Спочатку новий мова називалася "C з класами", але потім ім'я було змінене на C ++ - це повинно було підкреслити як його проісходженія від C, так і його перевага над останнім[14].

Java - об'єктно-орієнтована мова програмування, що розробляється компанією Sun Microsystems з 1991 року і офіційно випущений 23 травня 1995 року. Спочатку нову мову програмування називався Oak (James Gosling) і розроблявся для побутової електроніки, але згодом був перейменований в Java і став використовуватися для написання аплетів, додатків і серверного програмного забезпечення. Програми на Java можуть бути трансльовані в байт-код, що виконується на віртуальній java-машині (JVM) - програмою, обробній

байт-код і передавальної інструкції обладнанню, як інтерпретатор, але з тією відмінністю, що байт-код, на відміну від тексту, обробляється значно швидше[15].

Тепер порівняємо ці мови програмування за різними критеріями. За кожну відповідність критерію ставимо 1, за невідвідність – 0.

Таблиця 3.1

Порівняння мов програмування

Критерій	C++	C#	Java
Об’єктно-орієнтована	1	1	1
Прямий доступ до пам'яті	1	1	0
Кросплатформеність	1	1	1
Працює на віртуальній машині	0	1	1
Виразність та простота написання програмного коду	1	1	0
Підтримка бібліотеки OpenCV	1	1	1
Сумарний коефіцєнт	5	6	4

Після порівння, найбільшу кількість балів набрала мова C# – 6, тому було прийнято рішення, що вона найкраще підходить для реалізації поставленої задачі.

Синтаксис C# дуже виразний та простий, як у вивченні так і в розробці, він виконує набагато простіше те, що було складно реалізувати в C++ та дуже складно у Java.

C# підтримує універсальні методи і типи, забезпечуючи більш високий рівень безпеки і продуктивності, а також ітератори, які дозволяють при реалізації

колекцій класів визначати власну поведінку ітерації, яка може легко використовуватися в клієнтському коді[13].

До числа принципово важливих рішень, які реалізовані корпорацією Microsoft у мові програмування C #, можна віднести наступні:

- компонентно-орієнтований підхід до програмування;
- перевантажені оператори;
- обробка подій (оператор try);
- має набір елементів з доступом за номером індексу і однаковою кількістю стовпців і рядків.
- делегати;
- має оператор foreach;
- має indexer - оператори індексу для звернення до елементів класу-контейнера;
- має механізми boxing і unboxing;

та інші.

Врахувавши все вище сказане та взявши до у ваги результати порівняння(табл. 3.1) та все вище сказане для розробки було обрано мову програмування C#.

3.2 Розробка архітектури програмного додатку

Проведений аналіз усіх джерел дозволяє розробити функціонал для роботи програмного додатку.

Основні функції програмного додатку:

1) Вибір файлу зображення, на якому буде проведено пошук, розпізнавання та аналіз заданої інформації;
2) попередня обробка зображення;
3) застосування фільтру виділення границь зображення;
4) застосування фільтрів попередньої обробки зображення для покращення його якості(можливо на зображенні містяться засвіти, цифровий шум та інше);
5) передача обробленого фільтрами зображення в модуль розпізнавання;

6) реалізація контурного аналізу та розпізнання шуканої інформації;
7) пошук та знаходження потрібної інформації на зображенні;
8) виділення контуру;
9) порівняння знайденого об’єкту з шаблоном;
10) виведення на екран сповіщення про вдале знаходження потрібної інформаці.

Виходячи з розроблених функції можемо побудувати модель попередньої обробки вхідних даних(Рис.3.1) та модель розпізнавання(Рис.3.2) за методологією SADT. SADT – це спеціально розроблена методологія для опису штучних систем. Вона успішно застосовувалася і застосовується для опису різних технологічних процесів, процесів управління, планування і проектування інформаційних систем[17].

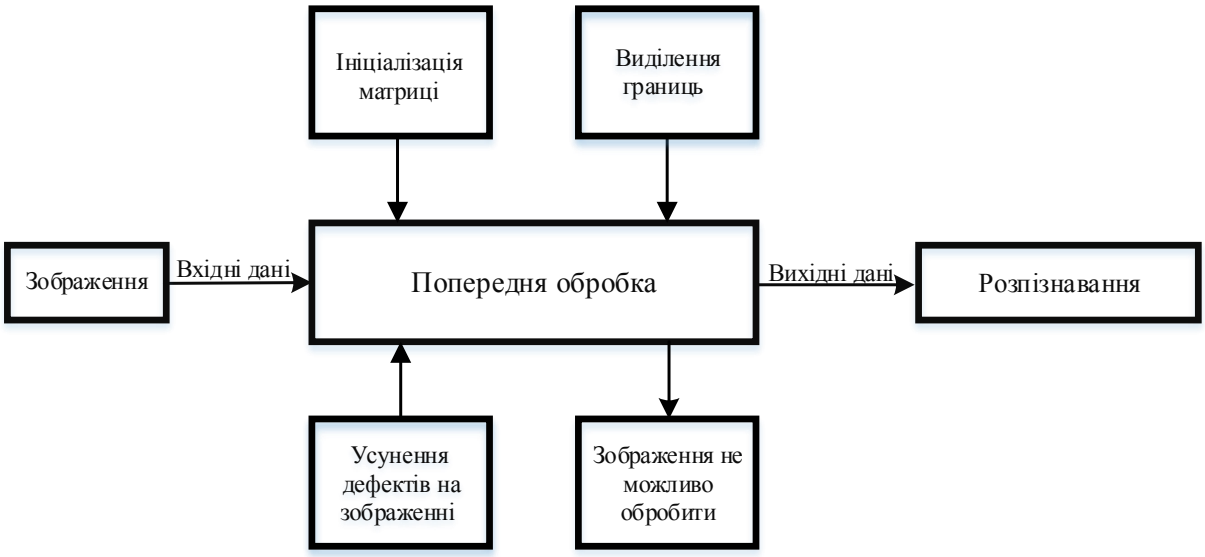


Рисунок 3.1 – Модель попередньої обробки вхідних даних

Опис моделі попередньої обробки зображення:

При отриманні вхідного зображення проводиться ініціалізація матриці.

Далі проводиться виділенні границь зображення шляхом виділення та знаходження градієнта(вектору), який обчислюється як корінь квадратний із суми квадратів сусідніх пікселів, але в даному випадку ми беремо не два сусідніх

пікселя, а сім. Після цього проводиться апроксимація цих точок для спрощення обчислень, більш того, при збільшенні кількості точок апроксимації підвищується стійкість до шуму. Але в результаті збільшуються обчислювальні затрати.

Пікселі, які досягають локального максимуму градієнта у напрямку вектора градієнта стають граничними пікселями, в результаті залишаються пікселі обведені червоним кольором, а інші подавляються.

Для знаходження границі в потрібній точці використовується дві межі фільтрації: якщо значення пікселя більше верхньої межі, то піксель приймає максимальне значення, якщо менше, тоді даний піксель подавляється. У випадку якщо він має середнє значення, проводиться аналіз, чи з'єднаний цей піксель з виділеними, якщо так, він приєднується до групи пікселів з максимальним значенням, в іншому випадку, даний піксель подавляється.

По завершенні, зображення передається далі, для розпізнавання.

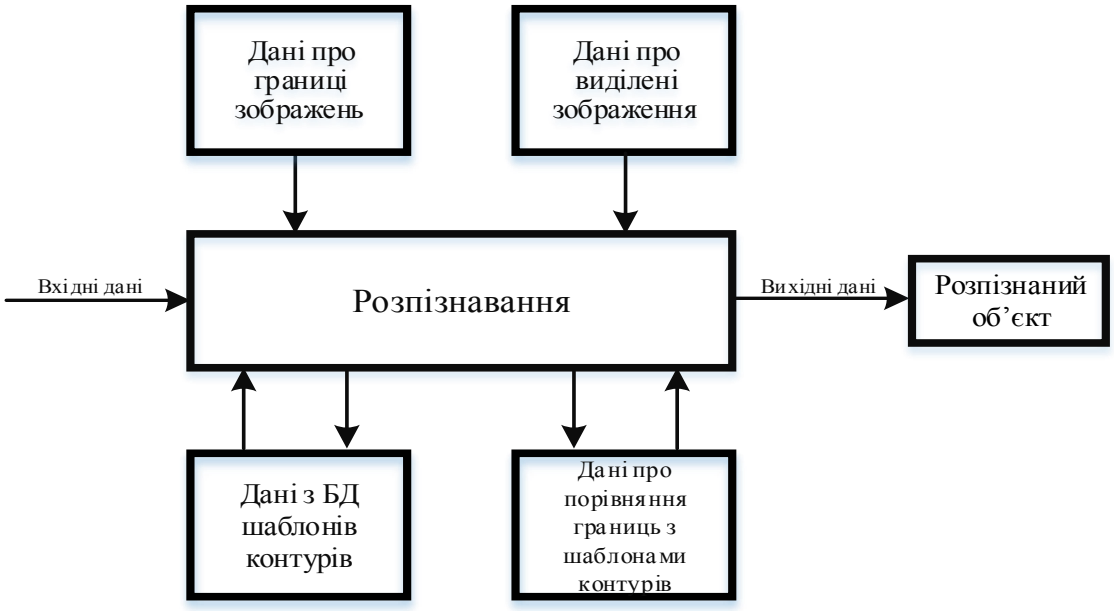


Рисунок 3.2 – Модель розпізнавання

Опис моделі розпізнавання:

Після отримання «обробленого» зображення проводиться первинна фільтрація контурів.

Далі в масиві проводиться пошук точок зі значенням 1, якщо такі є, проводиться порівняння контуру з еталоном, у випадку коли значення дорівнює 0, в якості початкової вибирається наступна точка, якщо точка була останньою порівняння не відбувається, даний контур не зпівпадає з еталонним;

У випадку, якщо не всі контури були перевірені, обирається наступний та знову повторюємо цикл.

У випадку, якщо більше немає контурів для розпізнавання, пошук буде закінчено та виведено результати.

3.3 Розробка та опис модулів

Було розроблено 2 модулі: попередньої обробки зображення та розпізнавання. Далі розглянемо кожен з них:

1. Модуль попередньої обробки.

В даному модулі виконуються наступні функції:

а) Проводиться відкриття зображення з системи.

б) Проводиться конвертація зображення у відтінки сірого.

в) Проводиться процедура згладжування зображення.

г) Проводиться пошук та апроксимація градієнта.

д) Проводиться виділення границь зображення.

е) Проводиться остаточне виділення границь шляхом подавлення всіх країв, які не є зв’язаними з границями.

ж) Проводиться бінаризація зображення для проведення подальшого розпізнавання контурів на ньому.

Далі показана блок-схема алгоритму даного модулю(Рис.3.3).

2. Модуль розпізнавання.

В модулі розпізнавання виконуються наступні фунції:

а) Проводиться виділення контурів шуканих об’єктів.

б) Проводиться первинна фільтрація знайдених контурів.

в) Проводиться адаптація контурів до єдиної довжини, що робить контури інваріантними до масштабу.

г) Виконується пошук максимально схожого шаблону, на знайдений контур(контури).

Далі показана блок-схема роботи даного модулю(Рис.3.4).

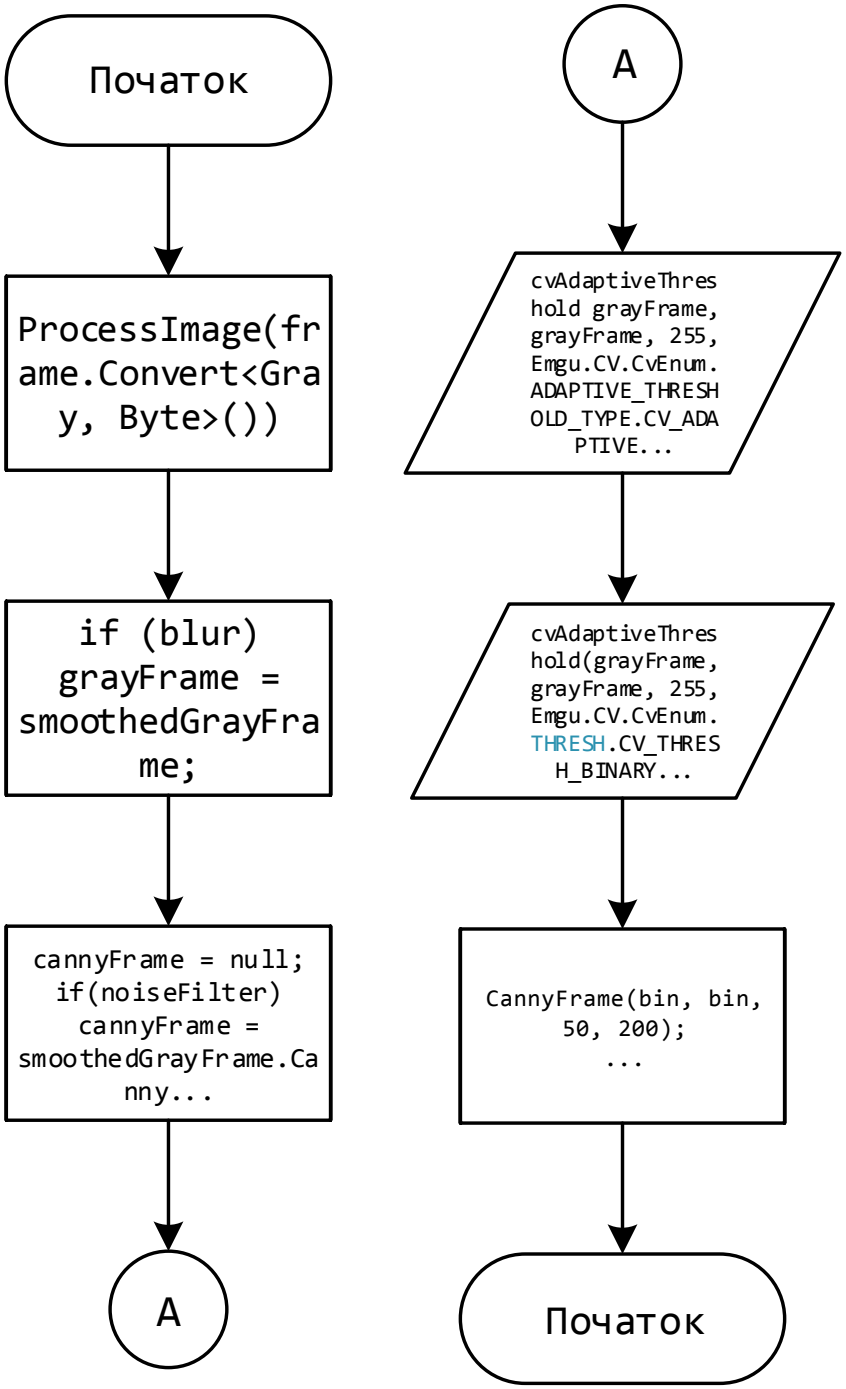


Рисунок 3.3 – Блок-схема модулю попередньої обробки

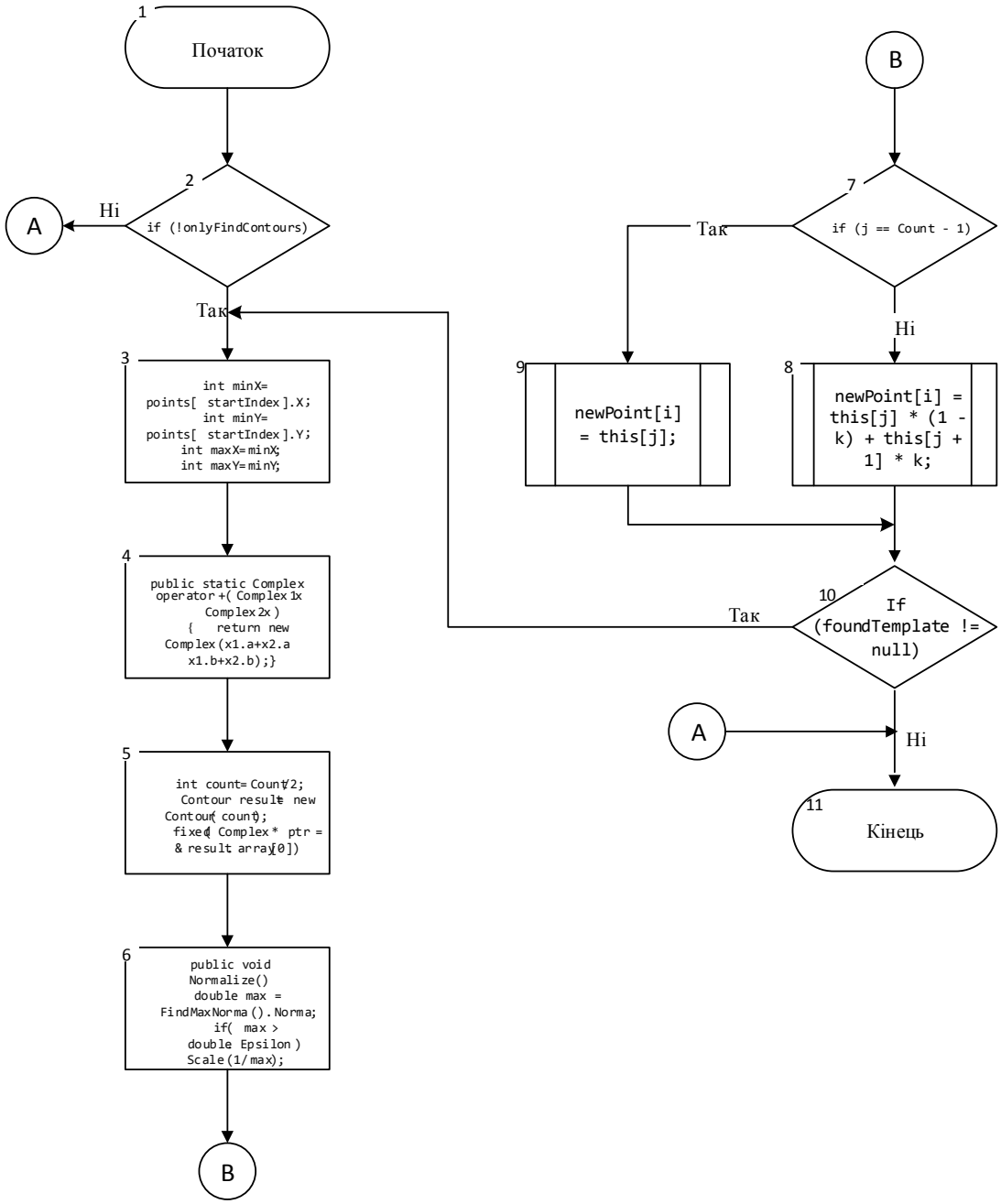


Рисунок 3.4 – Блок-схема модулю розпізнавання

3.4 Розробка інтерфейсу користувача

Інтерфе́йс користувача́ – засіб зручної взаємодії користувача з інформаційною системою. Сукупність засобів для обробки та відбиття інформації, якнайбільше пристосованих для зручності користувача[17].

Задоволення користувача розробленим програмним продуктом у значній мірі залежить від простоти та зручності у використанні інтерфейсу програмного додатку.

Існує декілька типів інтерфейсів користувача:

- Текстовий інтерфейс користувача (англ. Text user interface, TUI) — спосіб взаємодії користувача з комп'ютером з використанням текстового (буквено-цифрового) режиму дисплея або аналогічних — наприклад, командного рядка [18].
- Інтерфейс командного рядка (англ. command-line interface, CLI) — різновид текстового інтерфейсу користувача й комп'ютера, в якому інструкції комп'ютеру можна дати тільки введенням із клавіатури текстових рядків (команд)[19].
- Графі́чний інтерфе́йс кори́стувача́ (англ. GUI, Graphical user interface) – тип інтерфейсу, який дозволяє користувачам взаємодіяти з електронними пристроями через графічні зображення та візуальні вказівки, на відміну від текстових інтерфейсів, заснованих на використанні тексту, текстовому наборі команд та текстовій навігації[20].

Перший та другий типи графічного інтерфейсу не підходять для створення інтерфейсу розроблюваного програмного додатку, так як, вони являють собою інтерфейси, в яких вся інформація вводиться і виводиться тільки через консоль та у вигляді тексту, що є незручним.

Спираючись на те, що розроблюваний програмний додаток має великий набір функцій, а також виведення графічної інформації, буде доцільним використати тип графічного інтерфейсу користувача(GUI).

Графічний інтерфейс містить у собі все необхідне для того, щоб спроектувати інтерфейс додатку зручним та зрозумілим для користувача.

Інтерфейс програмного додатку буде містити декілька пунктів випадаючого меню та три робочих області.

Структура випадаючого меню буде містити 3 основних блоки:

1. Вкладка – Файл. У якій знаходяться підблоки:
    - Відкрити (пошук та відкриття файлу або файлів зображення з файлової системи);
    - Експорт БД (додати нову базу даних з контурами);

- Вихід (закриття програмного додатку).

2. Вкладка – Пошук. У якій знаходяться підблоки:
    - Пошук (інціалізувати процедуру пошуку заданих контурів на зображеннях);
    - Пауза (призупинити пошук, з можливістю подальшого його продовження);
    - Стоп (завершити процедуру пошуку та розпізнавання).
3. Вкладка – Довідка. У якій знаходяться підблоки:
    - Довідка (інформація про функції програмного додатку);
    - Про нас (інформація про розробника, посилання на офіційний сайт програмного додатку).

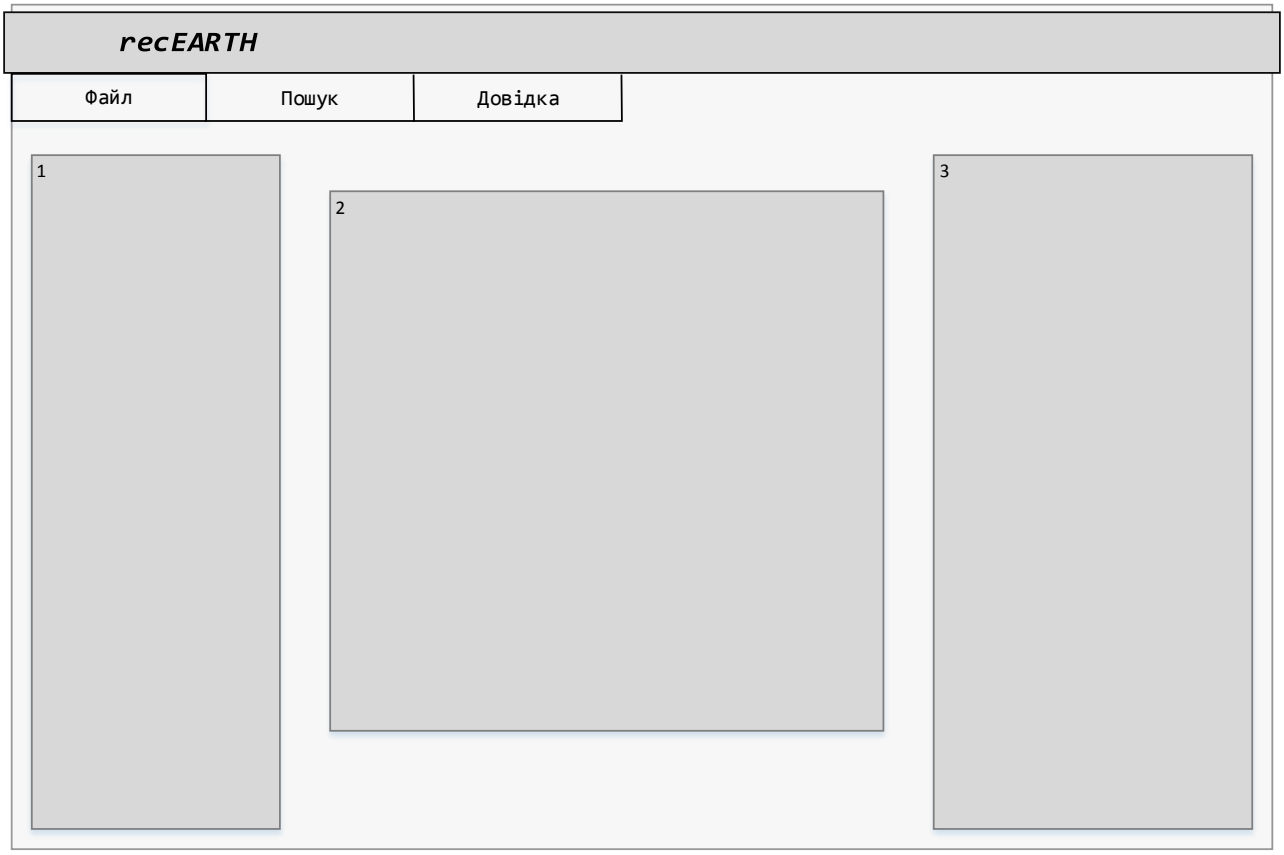


Рисунок 3.5 – Макет інтерфейсу розроблюваного програмного додатку

Під блоками випадаючого меню будуть знаходитися 3 робочих області:

1. Виведення на екран списку завантажених у додаток зображень для проведення розпізнавання та аналізу інформації.

2. Відображає вибране зображення для розпізнавання та виділяє на ньому розпізнану інформацію.

3. Виводить результати роботи програмного додатку після проведення процедури розпізнавання.

Нижче показано макет інтерфейсу розроблюваного додатку(Рис.3.5) та готовий графічний інтерфейс додатку(Рис.3.6):

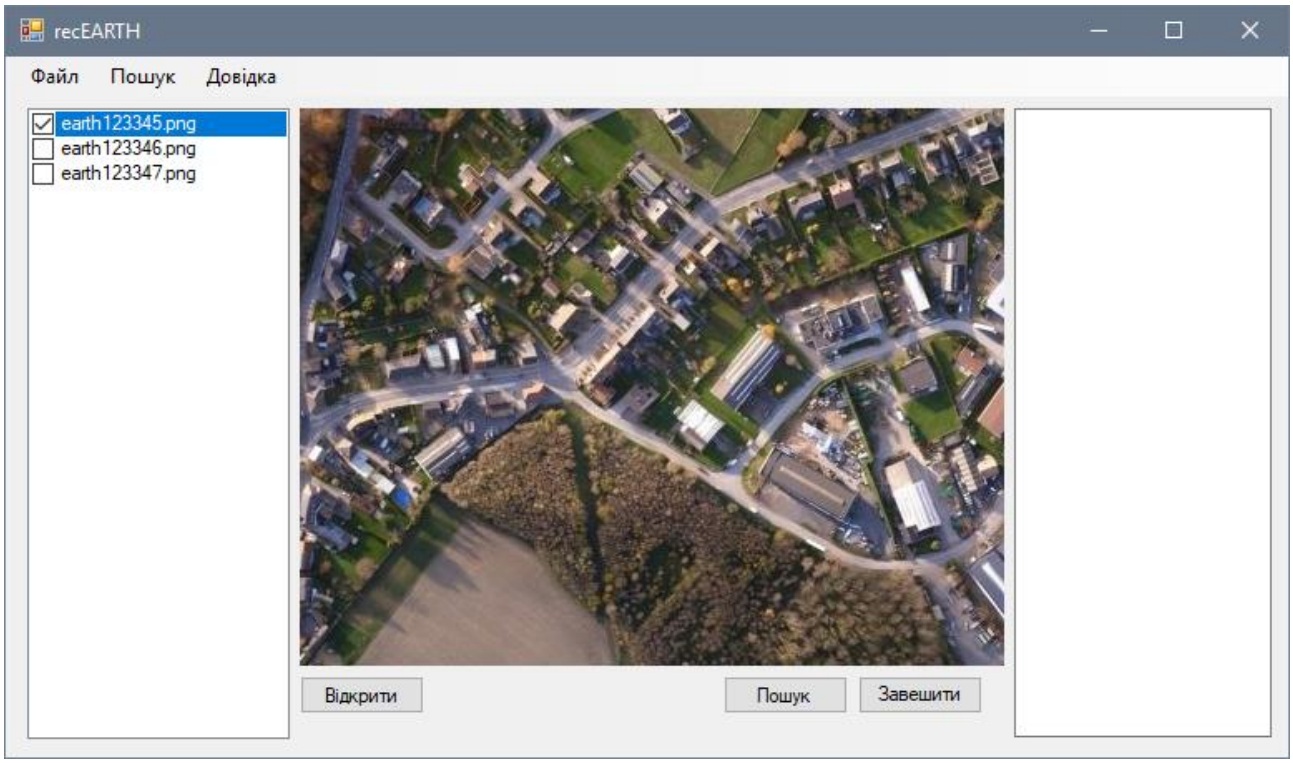


Рисунок 3.6 – Розроблений графічний інтерфейс програмного додатку

3.5 Висновки

В даному розділі було обґрунтовано вибір мови програмування, за допомогою якої буде здійснюватися розробка програмного додатку та наведено її основні переваги. Також була розроблена архітектура та основні модулі програмного додатку, описано їх роботу. Було розроблено простий у використанні та інтуїтивно зрозумілий інтерфейс користувача.

4 ТЕСТУВАННЯ РОБОТИ ПРОГРАМНОГО ДОДАТКУ

4.1 Тестування програмного додатку

Тестування програмного забезпечення - це процес, що використовується для виміру якості розроблюваного програмного забезпечення. Зазвичай, поняття якості обмежується такими поняттями, як коректність, повнота, безпечність, але може містити більше технічних вимог, які описані в стандарті ISO 9126. Тестування - це процес технічного дослідження, який виконується на вимогу замовників, і призначений для вияву інформації про якість продукту відносно контексту, в якому він має використовуватись. До цього процесу входить виконання програми з метою знайдення помилок [21].

Для проведення тестування програмного додатку розроблено набір тест-кейсів(Табл.4.1 та Табл.4.2).

Тест-кейс – це артефакт, що описує сукупність кроків, конкретних умов та параметрів, необхідних для перевірки реалізації системи, що тестується чи її частини. Під тест кейсом мається на увазі наступна структура: Action > Expected Result > Test Result[22].

Таблиця 4.1

Розроблений тест-кейс для тестування інтерфейсу програмного додатку

Назва:	Тест інтерфейсу програмного додатку	
Дія	**Очікуваний результат**	**Результат теста:** • пройдено • провалено
Передумова:		
Запустіть програмний додаток, recEarth.exe.	Програмний додаток успішно запущено.	
Кроки тесту:		
1. Перевірка коректності відображення інтерфейсу. 2. Перевірка доступності функціональних кнопок. 3. Перевірка доступності всіх пунктів випадаючого меню.	1. Інтерфейс відображається коректно. 2. Функціональні клавіші доступні. 3. Всі пункти випадаючого меню доступні.	

Продовження таблиці 4.1

Постумова:		
Закрийте програмний додаток, recEarth.exe.	Програмний додаток коректно завершив роботу.	

Таблиця 4.2

Розроблений тест-кейс для перевірки функціонування програмного додатку

Назва:	Тест функціонування програмного додатку	
Дія	**Очікуваний результат**	**Результат теста:** • пройдено • провалено
Передумова:		
Запустіть програмний додаток, recEarth.exe.	Програмний додаток запущено, все відображається коректно	
Кроки тесту:		
1. Перевірка функціональних клавіш. 2. Перевірка функціонування випадаючого меню. 3. Перевірка функціонування поля «Вибрані зображення». 4. Перевірка функціонування поля відображення вибраного зображення.	1. Після натиснення кожної клавіші запускається відповідний процес. 2. Після вибору та натиснення на будь-який пункт меню запускається відповідний йому процес. 3. Будь-яке зображення зі списку можна видрати для подальшої роботи. 4. Вибране зображення коректно відображається.	
Постумова:		
Закрийте програмний додаток, recEarth.exe.	Програмний додаток коректно завершив роботу.	

Програмний додаток повністю пройшов розроблені тести. Тести показали, що програмний додаток працює вірно.

Нижче показано приклад роботи програмного додатку(Рис.4.1).

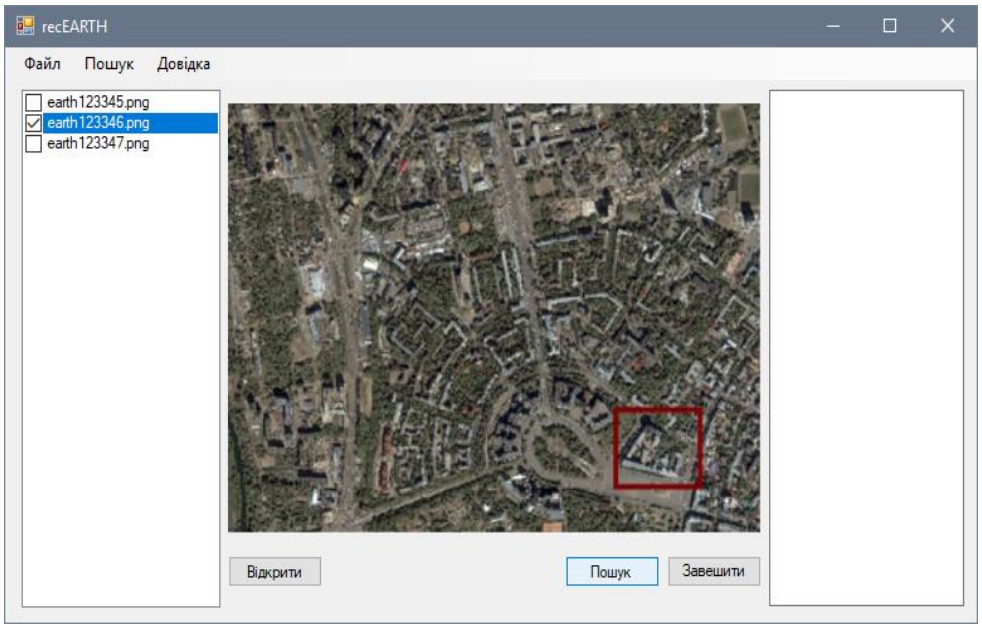


Рисунок 4.1 – Прилад роботи програмного додатку

4.2 Тестування якості та ефективності процесу розпізнавання

Основною проблемою ефективності та точності розпізнавання – є розпізнавання на зображеннях з великою кількістю цифрових шумів, засвітів та інших дефектів, які будуть заважати коректному виконанню поставленої задачі.

Цифровий шум – дефект зображення, що вноситься фотосенсором та електронікою пристроїв, які їх використовують. Цифровий шум помітний на зображенні у вигляді накладеної маски з пікселів випадкового кольору і яскравості. На камерах з масивом кольорових фільтрів колірний шум зазвичай має візуально більш великі зерна, ніж пікселі на зображеннях[23].

Для оцінки результатів роботи програмного додатку було проведено дослідження, суть якого полягала у штучному спотворенні зображення цифровим шумом та ділянками з «засвіченнями». Для цього було використано програмний засіб Adobe Photoshop.

Дослідження впливу цифрового шуму на якість розпізнавання шуканої інформації показало, що наявність незначного цифрового шуму не несе значного пониження ефективності розпізнавання. За еталон було взято 10 правильно розпізнаних об’єктів при зашумленості зображення 0-5%. Щільність шуму, яка не перевищує 15% не несе знаного погіршення знаходження шуканих об’єктів,

якщо щільність цифрового шуму більше 20% якість розпізнавання стрімко падає. На рис. 4.2 зображено графік залежності кількості правильно розпізнаних об’єктів від відсотку зашумленості зображення.

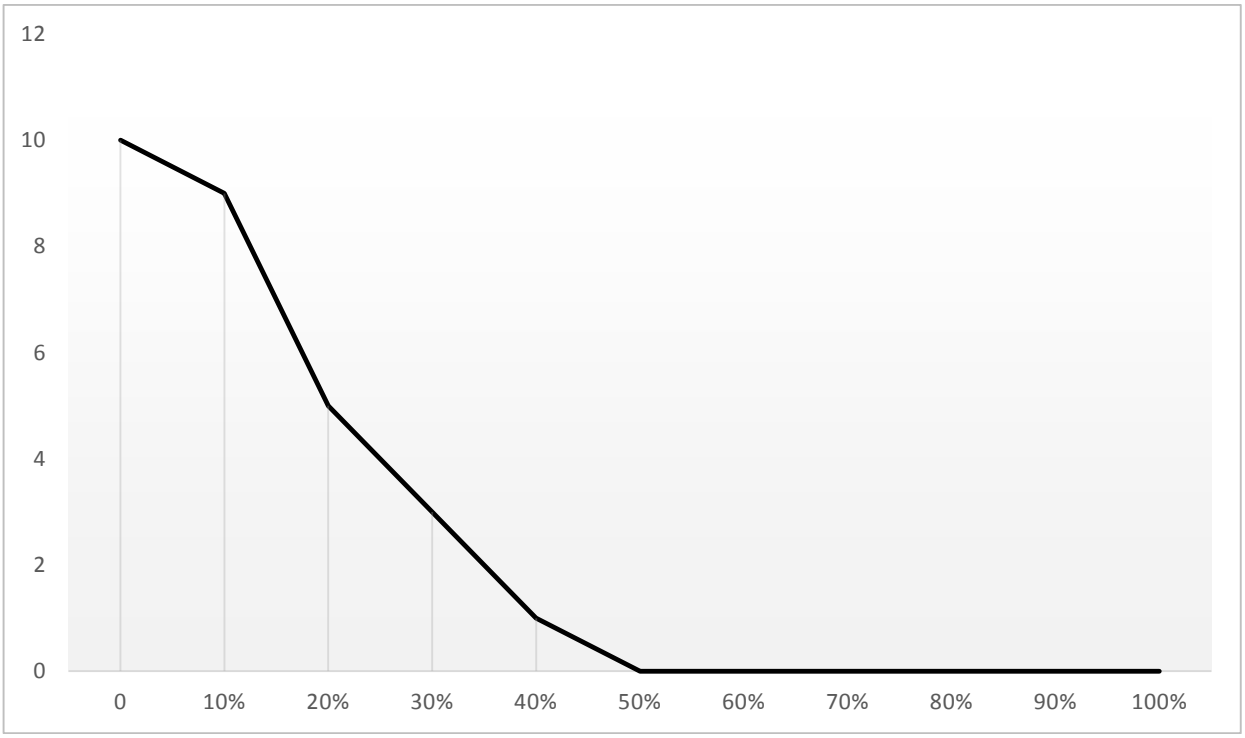


Рисунок 4.2 – Графік залежності кількості правильно розпізнаних об’єктів від щільності цифрового шуму на зображенні

Відповідно було проведено дослідження впливу «засвічення» на якість розпізнавання. В цьому випадку за еталон візьмемо 10 правильно розпізнаних об’єктів при «засвічені» 0-7%. Відповідно було побудовано графік залежності кількості правильно визначених об’єктів від відсотку «засвічення» зображення(Рис.4.3).

З наведених вище графіків можна сказати, що на зображеннях з великою кількістю цифрових дефектів процес розпізнавання відбувається з великою кількістю помилок. Для того, щоб помилок було якомога менше якість зображення має бути досить високою.

Також було проведено тестування швидкості роботи додатку на персональних комп’ютерах з новим та застарілим апаратним забезпечення.

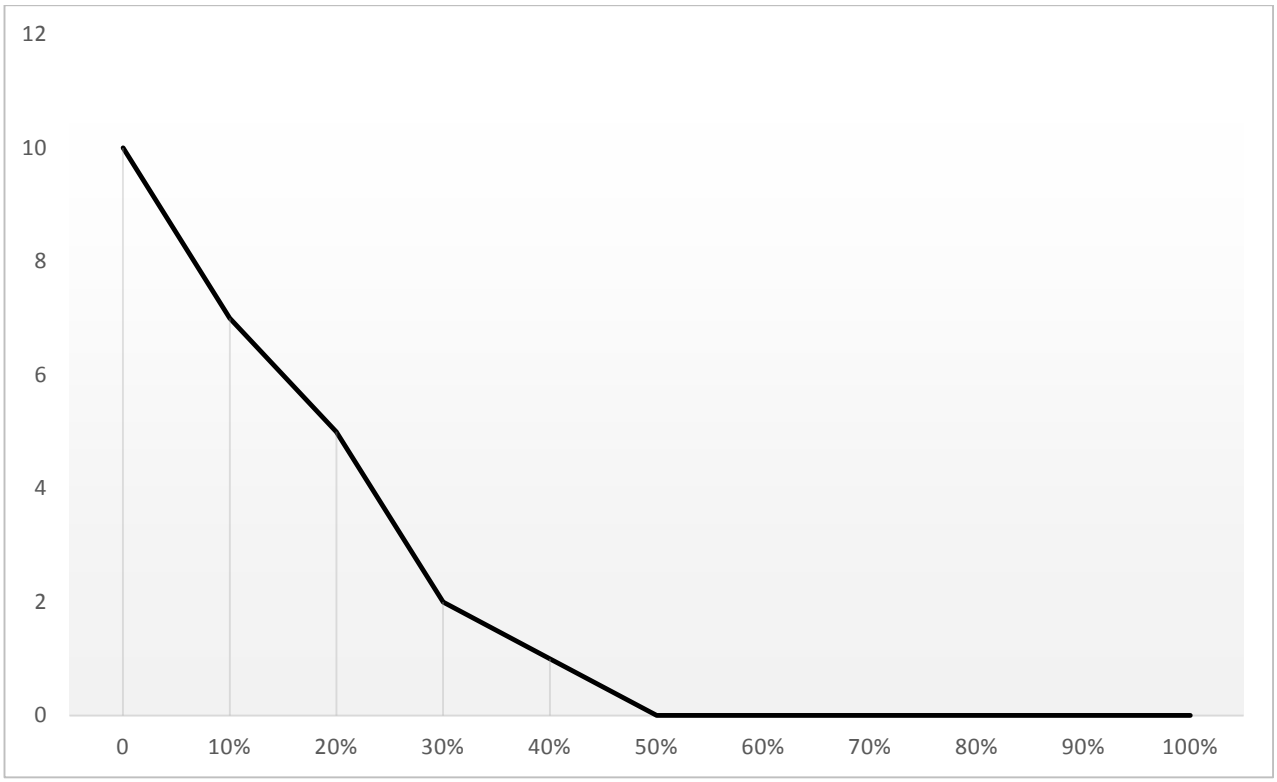


Рисунок 4.3 – Графік залежності кількості правильно розпізнаних об’єктів від відсотку ділянок з «засвіченями» зображення

Проведено тестування роботи додатку при великому навантаженні на центральний процесор, паралельній роботі декількох програмних систем. Тестування показало, що даний програмний додаток працює при будь-якому навантаженні на операційну систему однаково, процес розпізнавання проходить достатньо швидко, близько 0,1-0,3 секунди на персональних комп’ютерах з новим апаратним забезпечення та 0,27-0,4 секунди з застарілим. Точність розпізнавання складає 93%.

В цілому програмний додаток працює вірно. Тестування показало повну відповідність поставленому технічному завданню.

4.3 Висновки

У даному розділі було проведено тестування програмного додатку різними методами, яке показало повну її працездатність та відповідність поставленому технічному завданню.

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Оцінювання комерційного потенціалу розробки

Метою проведення технологічного аудиту є оцінювання комерційного потенціалу розробки. Для проведення технологічного аудиту було залучено 2-х незалежних експертів. Такими експертами будуть д.т.н, професор Романюк О.Н. та к.т.н., доцент Рейда О.М.

Оцінювання комерційного потенціалу здійснюємо за 12-ма критеріями, які наведені в таблиці 5.1, та 5-ти бальною шкалою.

Таблиця 5.1

Критерії оцінювання комерційногопотенціалу розробки та їх відповідна оцінка

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздат-ність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гір-ші, ніж в аналогів	Технічні та споживчі властивості проду-кту на рівні аналогів	Технічні та споживчі властивості продукту трохи кра-щі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крите р.	0	1	2	3	4
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуа-таційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позити-вною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з ко-мерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на на-вчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фі-нансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріа-лів	Потрібні матеріали, що ви-користовуються у військовопромис ловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використову-ються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвест. більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвест. більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвест. від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвест. менше 3-х років

Продовження таблиці 5.1

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Крите р.	0	1	2	3	4
12	Необхідна розробка регламентни документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення від-повідним органам про виробництво та реалізацію продукту	Відсутні будьякі регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу розробки наведено в таблиці 5.2.

Таблиця 5.2

Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта	
	1. Романюк О.Н., д.т.н., професор кафедри ПЗ	2. Рейда О.М., к.т.н., доцент кафедри ПЗ
	Бали, виставлені експертами:	
1	4	4
2	4	4
3	3	3
4	4	4
5	3	3
6	3	4
7	4	3
8	3	4
9	4	4
10	4	4
11	3	4
12	3	4
Сума балів	$СБ_1 = 43$	$СБ_2 = 45$
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_{1}^{3} СБ_i}{2} = 44$	

Отже, з отриманих даних таблиці 5.2 видно, що розробка має високий рівень комерційного потенціалу.

Існуючі реалізації для ідентифікації та класифікації особи за зображенням мають коефіцієнт спрацювання більше вісімдесяти відсотків.

Основними конкурентами на ринку є непрямі аналоги нової розробки, так як їх функціонал не має певних функцій, які виконує розроблюваний програмний додаток. Наведемо декілька прикладів систем розпізнавання:

1. ABBYY FineReader — це система оптичного розпізнавання текстів. Вона призначена для конвертування відсканованих документів в різні формати. Точність розпізнавання тексту приблизно 91%.

2. Різного роду нейронні мережі – це системи з автоматичним навчанням або з «учителем» для розпізнавання заданої інформації. Точність розпізнавання інформації даними системами близько 92%, але вони програють у швидкості розпізнавання.

Основними недоліками аналогів є не повна реалізація розпізнавання та ідентифікації. У розроблюваному програмному додатку має точність розпізнавання близько 92,3%, також було проведено дослідження, яке показало, що швидкість розпізнавання інформації була менше на 0,7 секунд та 1,03 секунди відповідно ніж у аналогів.

Функціональність нової розробки дозволяє вносити зміни і розширювати можливості системи для покращення роботи.

Простий для та інтуїтивно зрозумілий інтерфейс дозволяє мінімізувати зусилля користувача при використанні.

Більш помітна економія ресурсів дозволяє перемикатися на іншу програму, всі операції можуть виконуватися в розроблюваному додатку.

Можна досить просто виконувати підтримку нового продукту так як розробка має гнучку структуру та дає можливість внесення змін не змінюючи основні аспекти.

Відмінними рисами нового програмного продукту є: простота й зручність у використанні, висока швидкість та точність розпізнавання.

Програмний продукт планується реалізовувати шляхом створення власного веб-сайту, з якого буде вестися продаж та підтримка програмного продукту.

Даний програмний продукт буде корисний у аеро-космічній зйомці земної поверхні, за рахунок впровадження покращених методів розпізнавання.

На даний час програмний продукт готовий і знаходиться на дослідного зразка.

З метою комерціалізації необхідно провести пошук потенційних партнерів та інвесторів.

Новому програмному продукту не погрожує конкуренція, тому що не було знайдено точних аналогів.

Для зацікавлення споживачів потрібно провести рекламну кампанію в мережі Інтернет, на спеціалізованих веб-сайтах зондування земної поверхні, адже дана продукція не буде цікавою для більшості споживачів і рекламувати її деінде не доцільно. Також, необхідно приділити найсерйознішу увагу наданню високоякісних послуг з технічної підтримки і супроводу програмного продукту, який буде поставлятися користувачам.

5.2 Прогнозування витрат на виконання науково-дослідної роботи та конструкторсько–технологічної роботи.

Для розробки нового програмного продукту необхідні: .

Основна заробітна плата для розробників визначається за формулою (5.1):

$$З_о = \frac{М}{Т_р} \cdot t, \qquad (5.1)$$

де М- місячний посадовий оклад конкретного розробника;

$Т_р$ - кількість робочих днів у місяці, $Т_р = 22$ дні;

t - число днів роботи розробника, t = 66 днів.

Розрахунки заробітних плат наведені в табл. 5.3.

Розрахуємо додаткову заробітну плату:

$$З_{дод} = 0{,}1 \cdot 12504{,}54 = 1250{,}45 \text{ (грн.)}$$

Таблиця 5.3

Розрахунки основної заробітної плати

Працівник	Оклад М, грн.	Оплата за роб. день, грн.	Число днів роботи, t	Витрати на оплату праці,грн.
Науковий керівник	7500	340,90	5	1704,54
Інженер-програміст	3600	163,63	66	10800
Всього:				12504,54

Нарахування на заробітну плату операторів $Н_{зп}$ розраховується як 37,5…40% від суми їхньої основної та додаткової заробітної плати:

$$Н_{зп} = (З_о + З_р) \cdot \frac{\beta}{100}, \quad (5.2)$$

$$Н_{зп} = (12504{,}54 + 1250{,}45) \cdot \frac{36{,}3}{100} = 4993{,}06 \text{ (грн.)}.$$

Таблиця 5.4

Розрахунок амортизаційних відрахувань

Найменування програмного забезпечення	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань,грн
Процесор та комплектуючі	20000	25	3	1250
Всього:				1250

Розрахунок амортизаційних витрат для програмного забезпечення виконується за такою формулою:

$$А = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12}, \quad (5.3)$$

де Ц – балансова вартість обладнання, грн;

$Н_а$ – річна норма амортизаційних відрахувань % (для програмного забезпечення 25%);

Т – Термін використання (Т=3 міс.).

Розрахуємо витрати на комплектуючі. Витрати на комплектуючі розрахуємо за формулою:

$$K = \sum_1^n Н_і \cdot Ц_і \cdot К_і, \quad (5.4)$$

де n – кількість комплектуючих;

Ні - кількість комплектуючих і-го виду;

Ці – покупна ціна комплектуючих і-го виду, грн;

Кі – коефіцієнт транспортних витрат (приймемо $К_і$ = 1,15).

Таблиця 5.5

Витрати на комплектуючі, що були використані для розробки ПЗ

Найменування матеріалу	Одиниці виміру	Ціна, грн.	Витрачено	Вартість витрачених матеріалів, грн.
Пачка паперу	уп.	130	1	130
Ручка	шт.	15	1	15
Всього з урахуванням транспортних витрат				166,75

Витрати на силову електроенергію розраховуються за формулою:

$$В_е = В \cdot П \cdot Ф \cdot К_п ; \quad (5.5)$$

де В – вартість 1кВт-години електроенергії (В=1,68 грн/кВт);

П – установлена потужність комп’ютера (П=0,4кВт);

Ф – фактична кількість годин роботи комп’ютера (Ф=528 год.);

$К_п$ – коефіцієнт використання потужності ($К_п$< 1, $К_п = 0,9$).

$$В_е = 1,68 \cdot 0,4 \cdot 528 \cdot 0,9 = 319,33 \text{ (грн.)}$$

Розрахуємо інші витрати $В_{ін}$.

Інші витрати $І_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$В_{ін} = (1..3) \cdot (З_о + З_р). \quad (5.6)$$

Отже, розрахуємо інші витрати:

$$В_{ін} = 1 \cdot (12504{,}54 + 1250{,}45) = 13754{,}99 \text{ (грн)}.$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини роботи:

$$В = З_о + З_д + Н_{зп} + А + К + В_е + В_{ін}$$

$$В = 12504{,}54+1250{,}45+4993{,}06+1250+166{,}75+319{,}33+13754{,}99$$

$$= 34239{,}12 \text{ (грн.)}$$

Розрахуємо загальну вартість наукової роботи $B_{заг}$ за формулою:

$$В_{заг} = \frac{В_{ін}}{\alpha} \quad (5.7)$$

де α – частка витрат, які безпосередньо здійснює виконавець даного етапу роботи, у відн. одиницях = 1.

$$В_{заг} = \frac{34239{,}12}{1} = 34239{,}12$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta} \quad (5.8)$$

де β – коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Отже, розрахуємо загальні витрати:

$$ЗВ = \frac{34239,12}{0,9} = 38043,46 \text{ (грн.)}$$

5.3 Прогнозування комерційних ефектів від реалізації результатів розробки.

Спрогнозуємо отримання прибутку від реалізації результатів нашої розробки. Зростання чистого прибутку можна оцінити у теперішній вартості грошей. Це забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності .

Оцінка зростання чистого прибутку підприємства від впровадження результатів наукової розробки. У цьому випадку збільшення чистого прибутку підприємства $\Delta П_i$ для кожного із років, протягом яких очікується отримання позитивних результатів від впровадження розробки, розраховується за формулою:

$$\Delta П_i = \sum_1^n (\Delta Ц_0 \times N + Ц_0 \times \Delta N)_i \times \lambda \times \rho \times \left(1 - \frac{\upsilon}{100}\right) \qquad (5.9)$$

Де $\Delta Ц_0$ – покращення основного оціночного показника від впровадження результатів розробки у даному році. Зазвичай таким показником може бути ціна одиниці нової розробки;

N – основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN – покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$Ц_0$ – основний оціночний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n – кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки;

λ – коефіцієнт, який враховує сплату податку на додану вартість. Ставка податку на додану вартість встановлена на рівні 20%, а коефіцієнт $\lambda = 0,9945$;

ρ – коефіцієнт, який враховує рентабельність продукту. Рекомендується приймати ρ = 0,2...0,3;

υ – ставка податку на прибуток – 18%.

В результаті впровадження результатів наукової розробки була покращена якість продукту, що дозволяє підвищити ціну його реалізації на 200 грн. Кількість одиниць реалізованої продукції збільшиться: протягом першого року – на 400 шт., протягом другого року – ще на 280 шт., протягом третього року – ще на 125 шт.

Реалізація продукції до впровадження результатів наукової розробки складала 1200 шт., а її ціна – 520 грн.

Спрогнозуємо збільшення чистого прибутку від впровадження результатів наукової розробки у кожному році відносно базового.

Отже, збільшення чистого продукту $\Delta П_1$ за формулою 5.9 протягом першого року складатиме:

$$\Delta П_1 = [200 \times 1 + (520 + 200) \times 400] \times 0{,}9945 \times 0{,}25 \times 0.82 = 58756{,}05 \text{ грн.}$$

Протягом другого року:

$$\Delta П_2 = [200 \times 1 + (520 + 200) \times (400 + 280)] \times 0{,}9945 \times 0{,}25 \times 0.82 = 99856{,}75 \text{ грн.}$$

Протягом третього року:

$$\Delta П_3 = [200 \times 1 + (520 + 200) \times (400 + 280 + 125)] \times 0{,}9945 \times 0{,}25 \times 0.82 = 118205{,}27 \text{грн.}$$

5.4 Розрахунок ефективності вкладених інвестицій та період їх окупності

Визначимо абсолютну і відносну ефективність вкладених інвестором інвестицій та розрахуємо термін окупності.

Абсолютна ефективність $E_{абс}$ вкладених інвестицій розраховується за формулою:

$$\text{Е}_{\text{абс}} = (\text{ПП} - PV), \qquad (5.10)$$

де $\Delta\text{П}_\text{i}$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, 3 роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки 2, 3,4.

Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, рисунок 5.1.

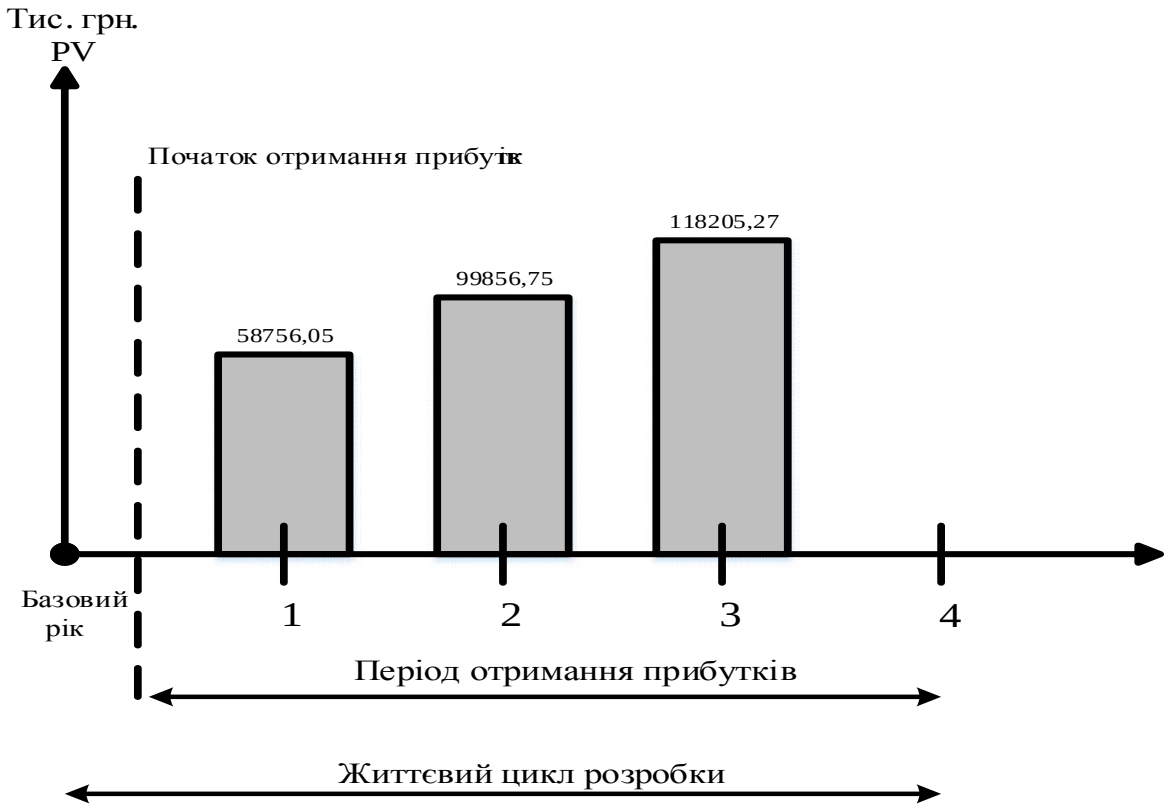


Рисунок 5.1 – Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

Розрахуємо вартість чистих прибутків за формулою:

$$ПП = \sum_1^T \frac{\Delta П_i}{(1+\tau)^t} \quad (5.11)$$

де $\Delta П_i$ – збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

т – період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ – ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t – період часу (в роках) від моменту отримання чистого прибутку до точки.

Отже, розрахуємо вартість чистого прибутку:

$$ПП = \frac{58756,05}{(1+0,1)^1} + \frac{99856,75}{(1+0,1)^2} + \frac{118205,27}{(1+0,1)^3} = 224750,19 \text{ (грн.)}$$

Тоді розрахуємо $E_{абс}$:

$$Е_{абс} = 224750,19 - 38043,46 = 186706,73 \text{ грн.}$$

Оскільки $E_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів НДДКР буде доцільним.

Розрахуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$ за формулою:

$$Е_в = \sqrt[T]{1 + \frac{Е_{абс}}{PV}} - 1 \quad (5.12)$$

де $Е_{абс}$ – абсолютна ефективність вкладених інвестицій, грн;

PV – теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ – життєвий цикл наукової розробки, роки.

Тоді будемо мати:

$$E_в = \sqrt[3]{1 + \frac{186706,73}{38043,46}} - 1 = \ 0,807 \text{ або } 80,7\ \%$$

Далі, розраховану величина $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$\tau = d + f, \tag{5.13}$$

де d – середньозважена ставка за депозитними операціями в комерційних банках; в 2019 році в Україні d = 0,2;

f – показник, що характеризує ризикованість вкладень, величина f = 0,1.

$$\tau = 0,2 + 0,1 = 0,3$$

Оскільки $E_в$ = 80,7% > $\tau_{мін}$ = 0,3 = 30%, то у інвестор буде зацікавлений вкладати гроші в дану наукову розробку.

Термін окупності вкладених у реалізацію наукового проекту інвестицій. Термін окупності вкладених у реалізацію наукового проекту інвестицій $T_{ок}$ розраховується за формулою:

$$T_{ок} = \frac{1}{E_в} \tag{5.14}$$

$$T_{ок} = \frac{1}{0,807} = 1,23 \text{ року}$$

Обрахувавши термін окупності даної наукової розробки, можна зробити висновок, що фінансування даної наукової розробки буде доцільним.

5.5 Висновки

В даному розділі було виконано оцінювання комерційного потенціалу програмного засобу розпізнавання та аналізу інформації про властивості матеріалів зондування земної поверхні. Було визначено, що рівень комерційного

потенціалу розробки на високому рівні. Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти, що підтверджує її перспективність. Програмний додаток має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Згідно із розрахунками всіх витрат, загальні витрати на розробку складають 38043,46 грн. Розрахована абсолютна ефективність вкладених інвестицій в сумі 186706,73 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту. Термін окупності вкладених у реалізацію проекту інвестицій становить 1,35 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВОК

У магістерській дипломній роботі було проаналізовано методи та засоби розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні.

На основі проведеного аналізу для розпізнавання було обрано метод контурного аналізу, який є одним з кращих за показниками ефективності розпізнавання та швидкість роботи.

Було розроблено архітектуру програмного додатку.

Вдосконалено метод розпізнавання об’єктів за допомогою контурного аналізу та удосконалено алгоритм Собеля для пошуку границь на зображенні.

Було розроблено архітектуру програмного додатку, модулі попередньої обробки зображень та розпізнавання. Розроблено інтерфейс, а також протестовано програмний додаток на відповідність поставленим вимогам та задачам.

Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти, що підтверджує її перспективність. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Існуючі переваги нової розробки дозволять зробити висновки про швидке поширення її на ринку.

Розроблена комп’ютерна система може існувати як самостійно, так і служити основою для подальших більш складних розробок.

СПИСОК ЛІТЕРАТУРИ

1. Н. Новикова Структурное распознавание образов – Воронеж: Издательско-полиграфический центр Воронежского государственного университета, 2006.

2. Х. П. Дуда Р., Распознавание образов и анализ сцен – Москва: Издательство «МИР», 1976.

4. А.К. Чудовская Возможности распараллеливания алгоритмов выделения контура по технологии CUDA / Чудовская А.К. Сб. докл. IV Межд. науч.-практич. конф. «Современная информационная Украина: информатика, экономика, философия». Донецк, 2010, с. 67–70.

5. J.F. Canny Finding edges and lines in images. Master's thesis. / Canny J.F. MIT, Cambridge, USA, 1983, pp. 50–67.

6. Фисенко В.Т. Компьютерная обработка и распознавание изображений: уч. пособие / В. Т. Фисенко, Т.Ю. Фисенко – СПб.: СПбГУ ИТМО, 2008. – 192 с.

7. Кепстр [Електронний ресурс] – Режим доступу: https://essuir.sumdu.edu.ua/bitstream/123456789/6056/1/40.pdf.

8. В.В. Вершинина Организация базы знаний семантической сети на основе XML-формата. / Вершинина В.В., Паламарь И.Н. Тез. докл. IV ВНТК «Информационные технологии в науке, проектировании и производстве». Нижний Новгород, МВВО АТН РФ, 2002, с. 23.

9. Гомоморфна фільтрація [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Гомоморфна_фільтрація.

10. C. S. A. T. D. Erhan, «Neural Information Processing Systems» – Deep Neural Networks for Object Detection, 2013.

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

12. Template matching algorithm – Режим доступу: http://docs.opencv.org/doc/tutorials/imgproc/histograms/template_matchng/template_matching.html?highlight=matchtemplate.

13. C Sharp – Режим доступу: http://progopedia.ru/language/csharp/

14. C++ – Режим доступу: http://progopedia.ru/language/c-plus-plus/

15. Java – Режим доступу:http://progopedia.ru/language/java/

16. Діаграма потоків даних [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Діаграма_потоків_даних.

17. Методологія функцонального моделювання SADT – Режим доступу: https://stud.com.ua/87183/ekonomika/metodologiya_funktsionalnogo_modelyuvannya_sadt

18. Текстовий інтерфейс користувача – Режим доступу: https://uk.wikipedia.org/wiki/Текстовий_інтерфейс_користувача

19. Інтерфейс командного рядка – Режим доступу: https://uk.wikipedia.org/wiki/Інтерфейс_командного_рядка

20. Графічний інтерфейс користувача – Режим доступу: https://uk.wikipedia.org/wiki/Графічний_інтерфейс_користувача

21. Тествання ПЗ – Режим доступу: http://lib.mdpu.org.ua/e-book/vstup/L11.htm

22. Тестовий випадок – Режим доступу: https://uk.wikipedia.org/wiki/Тестовий_випадок.

23. Цифровий шум зображення – Режим доступу: http://znaimo.com.ua/Цифровий_шум_зображення

24. Оператор Собеля – Режим доступу^ uk.wikipedia.org/wiki/Оператор_Собеля.

ДОДАТКИ

Додаток А. Технічне завдання на магістерську кваліфікаційну роботу

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп’ютерної інженерії

ЗАТВЕРДЖУЮ
Завідувач кафедри ПЗ
Романюк О.Н.
“____”_____20__року

Технічне завдання

на магістерську кваліфікаційну роботу за спеціальністю

121 - Інженерія програмного забезпечення

Керівник бакалаврської роботи:
к.т.н., доцент кафедри ПЗ
_____О.М. Рейда
"____"_____2019 р.

Виконав:
студент гр.2ПІ-18м
_____Б.П. Воловик
"____"_____2019 р.

Вінниця ВНТУ 2019

1.1 Найменування та галузь застосування

Розробка має назву «recEarth» і виконується в рамках магістерської кваліфікаційної роботи.

Згідно отриманого завдання кінцевий програмний продукт може використовуватись офіційною організацією.

1.2 Підстава для розробки

Підставою для розробки даної магістерської кваліфікаційної роботи є рішення засідання кафедри програмного забезпечення (протокол №___ від «___»_____ 20__ року).

1.3 Мета та призначення розробки

Метою роботи є підвищення продуктивності розпізнавання та аналізу інформації матеріалів зондування земної поверхні за рахунок покращення існуючих алгоритмів розпізнавання.

Для досягнення поставленої мети в роботі необхідно вирішити такі завдання:

- провести аналіз існуючих методів і засобів розпізнавання для підвищення їх продуктивності;
- запропонувати покращені методи та засоби для підвищення швидкості та точності розпізнавання інформації;
- розробити відповідну систему розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні на основі запропонованих методів;
- провести експериментальні дослідження розроблених методів та засобів.

1.4 Технічні вимоги

Вимоги до складових *частин комплексу технічних засобів.*

Для нормальної роботи програми, необхідний персональний комп’ютер з наступною мінімальною конфігурацією:

1. Процесор з частотою не менше 1,44 Гц;

2. Оперативна пам’ять ємністю не менше 4 Гб;

3. Запам’ятовуючий пристрій ємністю 500 Gb.

4. Операційна система Windows 7/8/8.1/10.

Вимоги до програмного *забезпечення.*

Програмний продукт розробляється на мові програмування C# з використанням платформи Microsoft .NET Framework.

1.5. Перелік технічної документації, що пред’являється по закінченню робіт

– пояснювальна записка;

– технічне обґрунтування доцільності розробки;

– лістинги програми.

1.6 Стадії і етапи розробки

Завдання на проектування видане _____ 2019 року. Проектування та дослідження повинно бути завершеним до _____2019 року.

Аналіз сучасних методів розпізнавання	07.10.2019 – 16.10.2019
Аналіз засобів розробки	17.10.2019 – 24.10.2019
Розробка та реалізація програмного додатку для розпізнавання і аналізу інформації	25.10.2019 – 10.11.2019
Тестування роботи програмного додатку	10.11.2019 – 14.11.2019
Економічна частина	15.11.2019 – 27.11.2019

1.7 Порядок контролю і приймання

Порядок контролю і приймання роботи регламентується відповідними документами ВНТУ і державними стандартами.

Додаток Б. Ілюстративний матеріал

ІЛЮСТРАТИВНИЙ МАТЕРІАЛ ДО ЗАХИСТУ МАГІСТЕРСЬКОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Завідувач кафедри ПЗ, д. т. н., професор _____ О. Н. Романюк

Науковий керівник, к. т. н., доцент кафедри ПЗ_____ О.М. Рейда

Рецензент, к.т.н, доцент кафедри КН_____ О. К. Колесницький

Нормоконтроль, к.т.н., доцент кафедри ПЗ _____В. В. Войтко

Виконавець, студент групи 2ПІ-18м _____ Б. П. Воловик

Міністерство освіти і науки України
Вінницький Національний Технічний Університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра програмного забезпечення

Магістерська кваліфікаційна робота
на тему:

Методи та програмні засоби розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні

Виконав студент
Воловик Б.П.
Науковий керівник
к.т.н., доц. Рейда О.М.

Метою роботи є підвищення продуктивності розпізнавання та аналізу інформації про властивості матеріалів зондування земної поверхні.

Об'єкт дослідження – процес розпізнавання інформації із зображення шляхом покращення існуючих аналогів.

Предмет дослідження – методи та засоби розпізнавання і аналізу інформації матеріалів зондування земної поверхні.

Постановка завдання

- провести аналіз існуючих методів і засобів розпізнавання для підвищення їх продуктивності;
- запропонувати покращені методи та засоби для підвищення швидкості та точності розпізнавання інформації;
- розробити відповідну систему розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні на основі запропонованих методів;
- провести експериментальні дослідження розроблених методів та засобів.

3

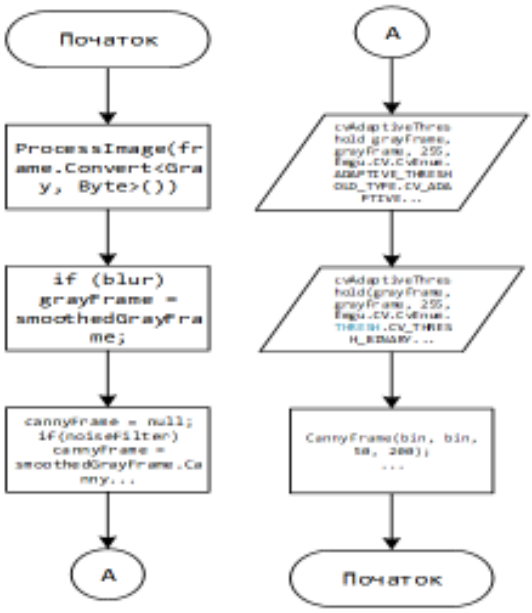
Наукова новизна отриманих результатів.

- Вперше запропоновано модель розпізнавання на основі контурного аналізу, яка при відповідному виборі лінійного простору для подання контуру, дозволяє зменшити трудомісткість операцій обробки та розпізнавання зображення, що в свою чергу, підвищує швидкість виконуваної операції(розпізнавання).
- Вперше запропоновано модель попередньої обробки зображення, на основі оператора Собеля, який базується на більш м'якій апроксимації градієнта, що дозволяє збільшити точність розпізнавання інформації.

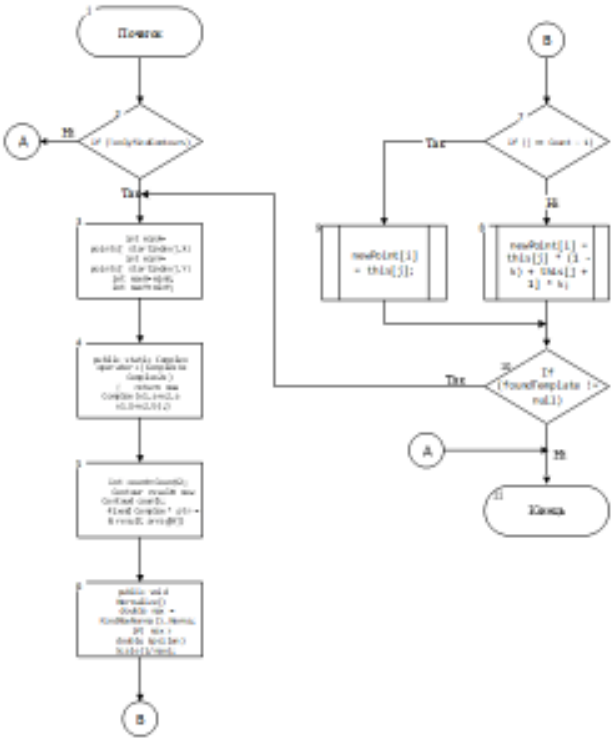
Основні функції програмного додатку:

1) Вибір файлу зображення, на якому буде проведено пошук, розпізнавання та аналіз заданої інформації;
2) попередня обробка зображення;
3) застосування фільтру виділення границь зображення;
4) застосування фільтрів попередньої обробки зображення для покращення його якості(можливо на зображенні містяться засвіти, цифровий шум та інше);
5) передача обробленого фільтрами зображення в модуль розпізнавання;
6) реалізація контурного аналізу та розпізнання шуканої інформації;
7) пошук та знаходження потрібної інформації на зображенні;
8) виділення контуру;
9) порівняння знайденого об'єкту з шаблоном;
10) виведення на екран сповіщення про вдале знаходження потрібної інформаці.

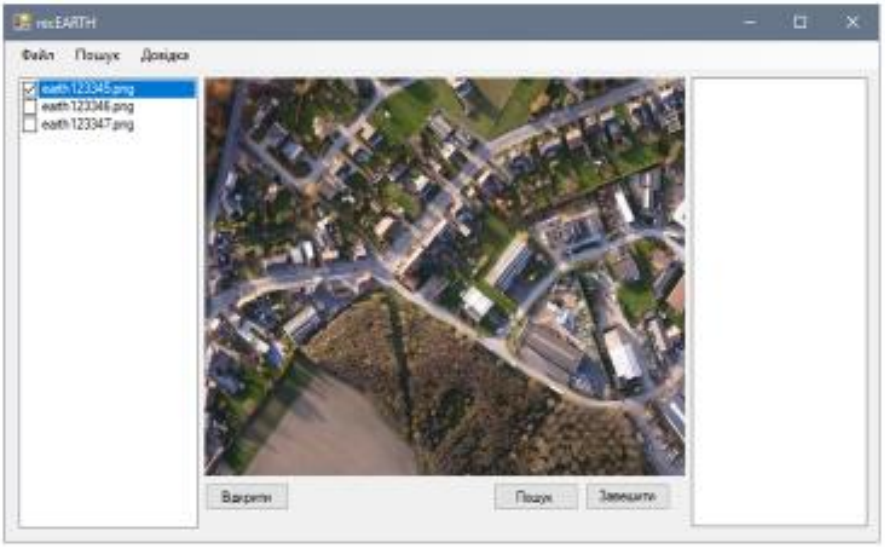
Блок-схема функціонування модулю попередньої обробки



Блок-схема функціонування модулю розпізнавання

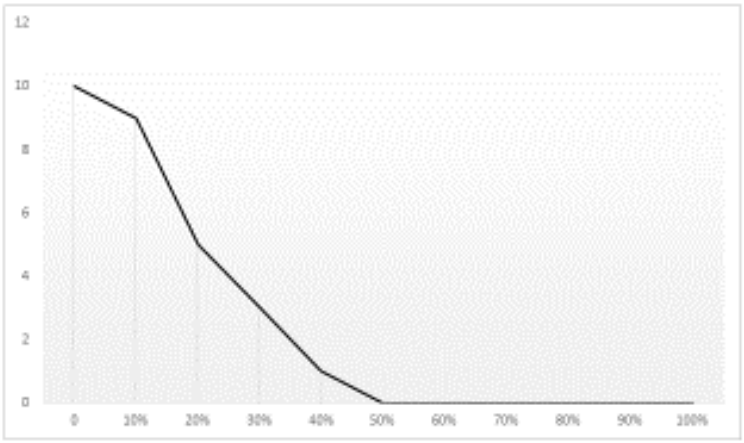


Розроблений інтерфейс системи



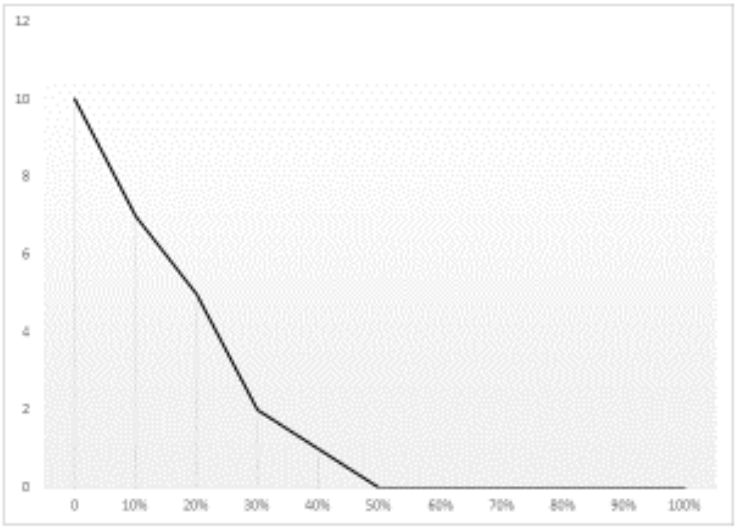
Тестування системи розпізнавання об'єктів при відеоспостереженні

Графік залежності кількості правильно розпізнаних об'єктів від щільності цифрового шуму на зображенні

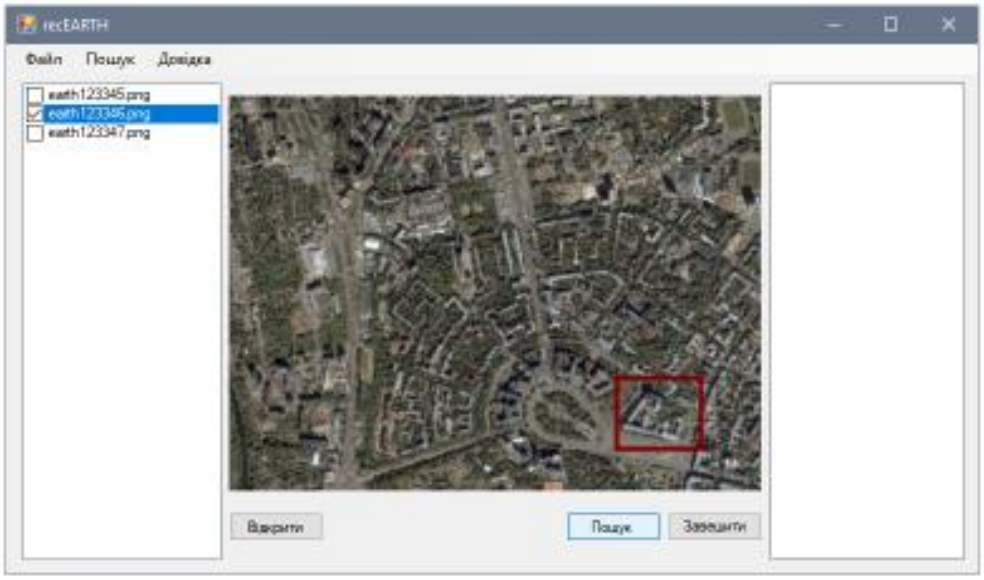


Тестування системи розпізнавання об'єктів при відеоспостереженні

Графік залежності кількості правильно розпізнаних об'єктів від відсотку ділянок з «засвіченими» зображення



Прилад роботи програмного додатку



Висновки

- У магістерській дипломній роботі було проаналізовано методи та засоби розпізнавання і аналізу інформації про властивості матеріалів зондування земної поверхні.
- На основі проведеного аналізу для розпізнавання було обрано метод контурного аналізу, який є одним з кращих за показниками ефективності розпізнавання та швидкість роботи.
- Було розроблено архітектуру програмного додатку.
- Вдосконалено метод розпізнавання об'єктів за допомогою контурного аналізу та удосконалено алгоритм Собеля для пошуку границь на зображенні.
- Було розроблено архітектуру програмного додатку, модулі попередньої обробки зображень та розпізнавання. Розроблено інтерфейс, а також протестовано програмний додаток на відповідність поставленим вимогам та задачам.
- Аналіз комерційного потенціалу розробки показав, що програмний продукт за своїми характеристиками випереджає аналогічні програмні продукти, що підтверджує її перспективність. Він має кращі функціональні показники, а тому є конкурентоспроможним товаром на ринку. Існуючі переваги нової розробки дозволять зробити висновки про швидке поширення її на ринку.
- Розроблена комп'ютерна система може існувати як самостійно, так і служити основою для подальших більш складних розробок.

Додаток В. Лістинг програми

```
using System;
using System.Collections.Generic;
using System.Drawing;
using System.Windows.Forms;
using Emgu.CV;
using Emgu.CV.Structure;
using ContourAnalysisNS;
using System.Runtime.Serialization.Formatters.Binary;
using System.IO;

namespace ObjectAnalyzerDemo
{
    public partial class MainForm : Form
    {
        private Capture _capture;
        Image<Bgr, Byte> frame;
        ImageProcessor processor;
        Dictionary<string, Image> AugmentedRealityImages = new
Dictionary<string, Image>();

        bool captureFromCam = true;
        int frameCount = 0;
        int oldFrameCount = 0;
        bool showAngle;
        int camWidth = 640;
        int camHeight = 480;
        string templateFile;

        public MainForm()
        {
            InitializeComponent();
            //create image preocessor
            processor = new ImageProcessor();
            //load default templates
            templateFile = AppDomain.CurrentDomain.BaseDirectory +
"\\Tahoma.bin";
            LoadTemplates(templateFile);

            StartCapture();
            //apply settings
            ApplySettings();

            Application.Idle += new EventHandler(Application_Idle);
        }

        private void LoadTemplates(string fileName)
        {
            try
            {
                using(FileStream fs = new FileStream(fileName, FileMode.Open))
```

```
                    processor.templates = (Templates)new
BinaryFormatter().Deserialize(fs);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void SaveTemplates(string fileName)
        {
            try
            {
                using (FileStream fs = new FileStream(fileName,
FileMode.Create))
                    new BinaryFormatter().Serialize(fs, processor.templates);
            }
            catch (Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void StartCapture()
        {
            try
            {
                _capture = new Capture();
                ApplyCamSettings();
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void ApplyCamSettings()
        {
            try
            {

_capture.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.CV_CAP_PROP_FRAME_WIDTH,
camWidth);

_capture.SetCaptureProperty(Emgu.CV.CvEnum.CAP_PROP.CV_CAP_PROP_FRAME_HEIGHT,
camHeight);
                cbCamResolution.Text = camWidth + "x" + camHeight;
            }
            catch (NullReferenceException ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        void Application_Idle(object sender, EventArgs e)
        {
```

```
            ProcessFrame();
        }

        private void ProcessFrame()
        {
            try
            {
                if (captureFromCam)
                    frame = _capture.QueryFrame();
                frameCount++;
                //
                processor.ProcessImage(frame);
                //
                if(cbShowBinarized.Checked)
                    ibMain.Image = processor.binarizedFrame;
                else
                    ibMain.Image = frame;
            }
            catch (Exception ex)
            {
                Console.WriteLine(ex.Message);
            }
        }

        private void tmUpdateState_Tick(object sender, EventArgs e)
        {
            lbFPS.Text = (frameCount - oldFrameCount) + " fps";
            oldFrameCount = frameCount;
            if (processor.contours!=null)
                lbContoursCount.Text = "Contours: "+processor.contours.Count;
            if (processor.foundTemplates != null)
                lbRecognized.Text = "Recognized contours: " +
processor.foundTemplates.Count;
        }

        private void ibMain_Paint(object sender, PaintEventArgs e)
        {
            if (frame == null) return;

            Font font = new Font(Font.FontFamily, 24);//16

            e.Graphics.DrawString(lbFPS.Text, new Font(Font.FontFamily, 16),
Brushes.Yellow, new PointF(1, 1));

            Brush bgBrush = new SolidBrush(Color.FromArgb(255, 0, 0, 0));
            Brush foreBrush = new SolidBrush(Color.FromArgb(255, 255, 255,
255));
            Pen borderPen = new Pen(Color.FromArgb(150, 0, 255, 0));

            if(cbShowContours.Checked)
            foreach (var contour in processor.contours)
                if(contour.Total>1)
                e.Graphics.DrawLines(Pens.Red, contour.ToArray());
            lock (processor.foundTemplates)
            foreach (FoundTemplateDesc found in processor.foundTemplates)
            {
```

```
                if (found.template.name.EndsWith(".png") ||
found.template.name.EndsWith(".jpg"))
                {
                    DrawAugmentedReality(found, e.Graphics);
                    continue;
                }

                Rectangle foundRect = found.sample.contour.SourceBoundingRect;
                Point p1 = new Point((foundRect.Left + foundRect.Right)/2,
foundRect.Top);
                string text = found.template.name;
                if (showAngle)
                    text +=
string.Format("\r\nangle={0:000}°\r\nscale={1:0.0}", 180 * found.angle /
Math.PI, found.scale);
                e.Graphics.DrawRectangle(borderPen, foundRect);
                e.Graphics.DrawString(text, font, bgBrush, new PointF(p1.X + 1
- font.Height/3, p1.Y + 1 - font.Height));
                e.Graphics.DrawString(text, font, foreBrush, new PointF(p1.X -
font.Height/3, p1.Y - font.Height));
            }
        }

        private void DrawAugmentedReality(FoundTemplateDesc found, Graphics gr)
        {
            string fileName = Path.GetDirectoryName(templateFile) + "\\" +
found.template.name;
            if (!AugmentedRealityImages.ContainsKey(fileName))
            {
                if (!File.Exists(fileName)) return;
                AugmentedRealityImages[fileName] = Image.FromFile(fileName);
            }
            Image img = AugmentedRealityImages[fileName];
            Point p = found.sample.contour.SourceBoundingRect.Center();
            var state = gr.Save();
            gr.TranslateTransform(p.X, p.Y);
            gr.RotateTransform((float)(180f * found.angle / Math.PI));
            gr.ScaleTransform((float)(found.scale), (float)(found.scale));
            gr.DrawImage(img, new Point(-img.Width/2, -img.Height/2));
            gr.Restore(state);
        }

        private void cbAutoContrast_CheckedChanged(object sender, EventArgs e)
        {
            ApplySettings();
        }

        private void ApplySettings()
        {
            try
            {
                processor.equalizeHist = cbAutoContrast.Checked;
                showAngle = cbShowAngle.Checked;
                captureFromCam = cbCaptureFromCam.Checked;
                btLoadImage.Enabled = !captureFromCam;
                cbCamResolution.Enabled = captureFromCam;
```

```
                processor.finder.maxRotateAngle = cbAllowAngleMore45.Checked ?
Math.PI : Math.PI / 4;
                processor.minContourArea = (int)nudMinContourArea.Value;
                processor.minContourLength = (int)nudMinContourLength.Value;
                processor.finder.maxACFDescriptorDeviation =
(int)nudMaxACFdesc.Value;
                processor.finder.minACF = (double)nudMinACF.Value;
                processor.finder.minICF = (double)nudMinICF.Value;
                processor.blur = cbBlur.Checked;
                processor.noiseFilter = cbNoiseFilter.Checked;
                processor.cannyThreshold = (int)nudMinDefinition.Value;
                nudMinDefinition.Enabled = processor.noiseFilter;
                processor.adaptiveThresholdBlockSize =
(int)nudAdaptiveThBlockSize.Value;
                processor.adaptiveThresholdParameter =
cbAdaptiveNoiseFilter.Checked?1.5:0.5;
                //cam resolution
                string[] parts = cbCamResolution.Text.ToLower().Split('x');
                if (parts.Length == 2)
                {
                    int camWidth = int.Parse(parts[0]);
                    int camHeight = int.Parse(parts[1]);
                    if (this.camHeight != camHeight || this.camWidth !=
camWidth)
                    {
                        this.camWidth = camWidth;
                        this.camHeight = camHeight;
                        ApplyCamSettings();
                    }
                }
            }
            catch(Exception ex)
            {
                MessageBox.Show(ex.Message);
            }
        }

        private void btLoadImage_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "Image|*.bmp;*.png;*.jpg;*.jpeg";
            if (ofd.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
                try
                {
                    frame = new Image<Bgr,
byte>((Bitmap)Bitmap.FromFile(ofd.FileName));
                }
                catch (Exception ex)
                {
                    MessageBox.Show(ex.Message);
                }
        }

        private void btCreateTemplate_Click(object sender, EventArgs e)
        {
            if(frame!=null)
```

```
                new ShowContoursForm(processor.templates, processor.samples,
frame).ShowDialog();
        }

        private void btNewTemplates_Click(object sender, EventArgs e)
        {
            if (MessageBox.Show("Do you want to create new template database?",
"Create new template database", MessageBoxButtons.OKCancel) ==
System.Windows.Forms.DialogResult.OK)
                processor.templates.Clear();
        }

        private void btOpenTemplates_Click(object sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "Templates(*.bin)|*.bin";
            if (ofd.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
            {
                templateFile = ofd.FileName;
                LoadTemplates(templateFile);
            }
        }

        private void btSaveTemplates_Click(object sender, EventArgs e)
        {
            SaveFileDialog sfd = new SaveFileDialog();
            sfd.Filter = "Templates(*.bin)|*.bin";
            if (sfd.ShowDialog(this) == System.Windows.Forms.DialogResult.OK)
            {
                templateFile = sfd.FileName;
                SaveTemplates(templateFile);
            }
        }

        private void btTemplateEditor_Click(object sender, EventArgs e)
        {
            new TemplateEditor(processor.templates).Show();
        }

        private void btAutoGenerate_Click(object sender, EventArgs e)
        {
            new AutoGenerateForm(processor).ShowDialog();
        }
    }
}
```