Центральноукраїнський національний технічний університет
Механіко-технологічний факультет
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2024 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за першим (бакалаврським) рівнем вищої освіти
на тему
“Програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Дерев`янко О.С.
«____» _____ 2024 р.

Керівник проекту
доктор технічних наук, професор
_____ Коваленко О.В.
«____» _____ 2024 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Освітній ступінь *бакалавр*

Галузь знань ._12 _____*"*Інформаційні технології*"*_____

Спеціальність *123 "*Комп'ютерна інженерія*"*

Освітньо-професійна (освітньо-наукова) програма _*"*Комп'ютерна інженерія*"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 17 » січня 2024 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ПЕРШИМ (БАКАЛАВРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Дерев`янку Олександру Сергійовичу

(прізвище, ім'я, по батькові)

1. Тема роботи *Програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет*

2. Керівник роботи *Коваленко Олександр Володимирович, докт. техн. наук, професор*

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 131-02 від 01.04.2024 року

3. Строк подання студентом роботи до захисту *23.05.2024 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою роботи є розробка програмного забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет*

5. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

1. Призначення та область використання.

2.Перегляд аналогічних існуючих систем.

3. Опис і обґрунтування проектних рішень.

4. Етапи програмування системи.

5. Впровадження системи в промислову експлуатацію.

6. Висновки

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

Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*

7. Дата видачі завдання « 17 » січня 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.03.2024 р.	
2.	Постановка задачі, оформлення ТЗ	15.03.2024 р.	
3.	Розробка моделі компонента	20.03.2024 р.	
4.	Розробка структур даних	25.03.2024 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.03.2024 р.	
6.	Програмування алгоритмів	10.04.2024 р.	
7.	Оформлення ПЗ	17.04.2024 р.	
8.	Попередній захист роботи	23.05.2024 р.	

Дата видачі завдання
« 17 » січня 2024 р.

Підпис керівника
_____ Коваленко О.В.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Дерев`янко О.С.
(прізвище та ініціали)

АНОТАЦІЯ

Дерев`янко О.С. Програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2024.

В даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Метою розробки є програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Результат роботи – програмна реалізація системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

В процесі роботи над програмною моделлю виконано аналіз існуючих апаратних та програмних засобів. В повній мірі описані всі компоненти розробленого програмного забезпечення.

Розроблено зручний інтерфейс користувача. Наведені інструкції по роботі з програмними засобами.

Програма може використовуватися на ПЕОМ з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, структурно-лінгвістичне розпізнавання зображень

ABSTRACT

Derevianko O.S. System software based on the method of structural-linguistic image recognition for Internet search engines. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this graduation thesis for the first (bachelor) level of higher education, software was developed, which is intended for a system based on the method of structural-linguistic image recognition for Internet search engines.

The goal of the development is system software based on the method of structural-linguistic image recognition for search engines on the Internet.

The result of the work is a software implementation of a system based on the method of structural-linguistic image recognition for search engines on the Internet.

In the process of working on the software model, an analysis of existing hardware and software was performed. All components of the developed software are fully described.

A convenient user interface has been developed. Instructions for working with software tools are provided.

The program can be used on a PC with Windows 10/11 OS.

The program was developed in the RAD Studio Delphi 10.4 environment.

Keywords: computer engineering, structural-linguistic image recognition

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

КСМ	–	комп’ютерні системи і мережі
СМК	–	семантична мережа концептів

ВСТУП

Актуальність теми. У зв'язку з сучасним розвитком розподілених інформаційно-телекомунікаційних мереж, зокрема мережі Інтернет, постає питання розробки засобів технологій семантичного пошуку та контекстної обробки як текстової, так і графічної інформації, причому частка графічної інформації в загальному обсягу пошукових запитів є найбільш вагомою. Ці технології, перш за все, повинні бути спрямовані на розробку методів та засобів автоматизованої семантичної обробки множини посилань та пов'язаних з ними зображень, що були знайдені в результаті виконання одного або декількох пошукових запитів з метою формування найбільш повної та релевантної відповіді.

Таким чином, розробка нових методів та технологій збору релевантної графічної інформації (зображень) у структурованих, слабоструктурованих та неструктурованих базах даних на основі концепції семантичної обробки інформації є однією з перспективних науково-технічних задач.

Для побудови опису зображення, що відповідає концепції семантичної обробки інформації та може використовуватися в системах збору релевантних зображень, доцільно використовувати структурно-лінгвістичний підхід до розпізнавання. Існуючі методи не дозволяють отримати подібний опис в наслідок того, що виникає проблема створення універсальних процедур, які здійснюють побудову структур ознак класів розпізнавання інваріантних афінним перетворенням та деформаційним спотворенням контурів зображень. Розробка методу структурно-лінгвістичного розпізнавання, що реалізує процес виділення цих інваріантних ознак на основі перетворення початкового зображення, яке представлено в одному з графічних форматів, в семантичну структуру на основі застосування формалізму семантичних мереж є актуальною задачею та складає напрямок бакалаврського дослідження.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Для досягнення поставленої мети визначена програма дослідження, що складається з наступних завдань:

– Огляд існуючих систем на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

– Дослідження системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

– Програмна реалізація системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначена для дослідження та програмної реалізації методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет. Для цього проведемо аналіз процесів пошуку та збору релевантних зображень у КСМ та запропонуємо шляхи їх вдосконалення. Зазначимо, що незважаючи на певні успіхи, поки не існує задовільних з точки зору користувача програмних засобів, які б дозволяли отримувати результати з високим ступенем відповідності пошуковому запиту. Це пояснюється відсутністю методу однозначного опису зображень та слабким розвиненням математичного апарату для побудови такого опису. Наслідком цього є недостатня проробка загальних методик на основі фундаментальних математичних моделей. Практично всі існуючі розробки базуються на реалізації деяких евристичних алгоритмів.

Виходом з ситуації, що склалася, є застосування методів розпізнавання, які спрямовані на скорочення розмірності вектора інформативних ознак та, як наслідок, на скорочення обчислювальних витрат.

У зв’язку з цим, зроблено порівняльний аналіз сучасних та перспективних методів розпізнавання зображень з метою визначення придатності їх для збору релевантної графічної інформації в КСМ. Вкажемо на доцільність застосування методу, який базується на принципі загальності ознак зображень, що розглядаються, має переваги структурно-лінгвістичного підходу до розпізнавання та відображає особливості інформаційної моделі зору людини.

Також розглянемо методи попередньої обробки зображень в процесі розпізнавання. Зазначимо, що основну увагу слід приділити процедурам сегментації та виділення зовнішніх контурів об’єктів.

1.2 Область застосування

Вибираєте найкращий інструмент пошуку зображень? Ось короткий посібник:

– Споживчі та спеціалізовані: такі інструменти, як Google, TinEye, зручні та відкриті для всіх. Однак деякі вдосконалені інструменти зарезервовано для державних або юридичних установ.

– Варіанти пробної версії або попереднього перегляду: доцільно вибрати платформи, які дозволяють попередньо переглянути принаймні деякі результати або спочатку спробувати послугу. Деякі сервіси, як-от Socialcatfish, вимагають оплати, перш ніж показати будь-які результати. Багато користувачів повідомили про незадоволення моделлю оплати за перегляд.

– Точність і розмір бази даних. Ефективність інструменту зворотного пошуку зображень значною мірою залежить від точності результатів і розміру бази даних. Обирайте інструменти, відомі точними результатами та великою базою даних зображень.

– Простота використання: простий і зрозумілий інтерфейс має велике значення.

– Вартість: деякі інструменти безкоштовні, інші – ні. Зрозумійте їхні ціни, перш ніж зануритися.

– Конфіденційність перш за все: деякі служби можуть зберігати зображення, які ви шукаєте, тоді як інші можуть використовувати їх для аналізу даних або реклами. Переконайтеся, що платформа поважає конфіденційність користувачів.

– Відгуки реальних користувачів: завжди перевіряйте відгуки та відгуки користувачів. Реальний досвід інших користувачів може дати цінну інформацію про надійність, ефективність і потенційні недоліки платформи.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за першим (бакалаврським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

2.1 Огляд існуючих систем, технологій, архітектур, програмних рішень за профілем теми випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

Розглянемо декілька сервісів, які дозволяють шукати інформацію у Інтернет використовуючи зображення.

Google Image

З понад 10 мільярдами проіндексованих зображень Google Images є одним із найнадійніших доступних інструментів зворотного пошуку зображень. Користувачі можуть завантажити зображення або ввести URL-адресу зображення, щоб знайти збіги та подібні візуальні елементи в Інтернеті.

Рисунок 2.1 – Google Images

Як широко поширена пошукова система, Google Images індексує велику базу даних, що значно перевершує більшість альтернатив. Навіть без точного збігу він пропонує пропозиції тематично пов’язаних зображень.

Щоб отримати доступ до пошуку зображень Google, не потрібно реєструвати обліковий запис і користуватися ним можна безкоштовно. Інтерфейс простий і доступний як на комп’ютері, так і на мобільному пристрої, хоча пряме завантаження зараз доступне лише на комп’ютерах.

Хоча Google Images не зосереджений виключно на зворотному пошуку зображень, універсальність, охоплення та простота використання роблять його найкращим вибором для багатьох користувачів. Його можливості та розмір пошукової системи дають йому перевагу у відстеженні незрозумілих зображень і шаблонів.

Ті, хто хоче ідентифікувати або знайти зображення, можуть знайти збіги у величезній базі даних Google. Однак спеціалізовані інструменти зворотного зображення можуть запропонувати розширенішу фільтрацію та можливості. Google Images – хороша відправна точка перед вивченням додаткових послуг.

FaceCheck.ID

FaceCheck.ID – це інноваційна система зворотного пошуку зображень, яка спеціалізується на розпізнаванні облич. Це дозволяє користувачам шукати в Інтернеті зображення та профілі людини, просто завантажуючи фотографію її обличчя.

Завдяки розширеному штучному інтелекту та машинному навчанню FaceCheck.ID сканує мільйони облич у своїй базі даних, щоб знайти профілі в соціальних мережах, новинні статті, фотографії та інші джерела, що містять ті самі чи схожі обличчя.

Інтуїтивно зрозумілий інтерфейс дозволяє будь-кому легко розпочати пошук обличчя за лічені секунди. Користувачі також можуть завантажувати кілька фотографій однієї людини, щоб уточнити результати.

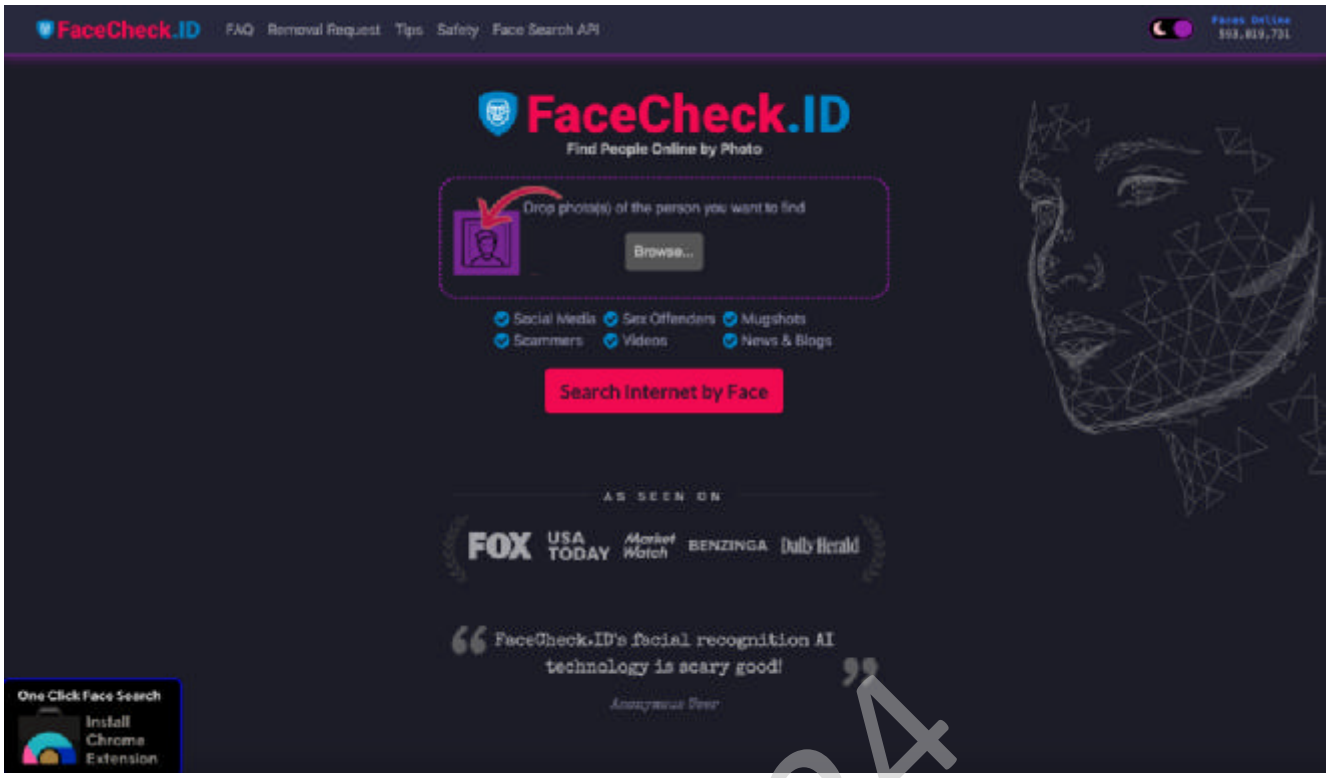

Рисунок 2.2 – FaceCheck.ID

Результати збігів чітко вказують на ймовірність того, що в результаті буде одна й та сама особа, допомагаючи ідентифікувати схожих. Користувачі все одно повинні підтверджувати свою особу за допомогою додаткових підказок.

Для тих, хто хоче додати ім’я до обличчя, знайденого в Інтернеті або на старій фотографії, FaceCheck.ID пропонує розширений пошук. Ми ставимо йому оцінку 4,5 з 5 за зручний інтерфейс, унікальну спеціалізацію та швидкі результати.

PimEyes

PimEyes – це пошукова система розпізнавання облич, яка сканує Інтернет, щоб знайти зображення облич, надіслані користувачами. Завдяки штучному інтелекту та технології зворотного зображення він може ідентифікувати незрозумілі фотографії людей в Інтернеті, навіть якщо їхній зовнішній вигляд або фон змінюються. Він призначений для самостійного пошуку та узгодженого пошуку.

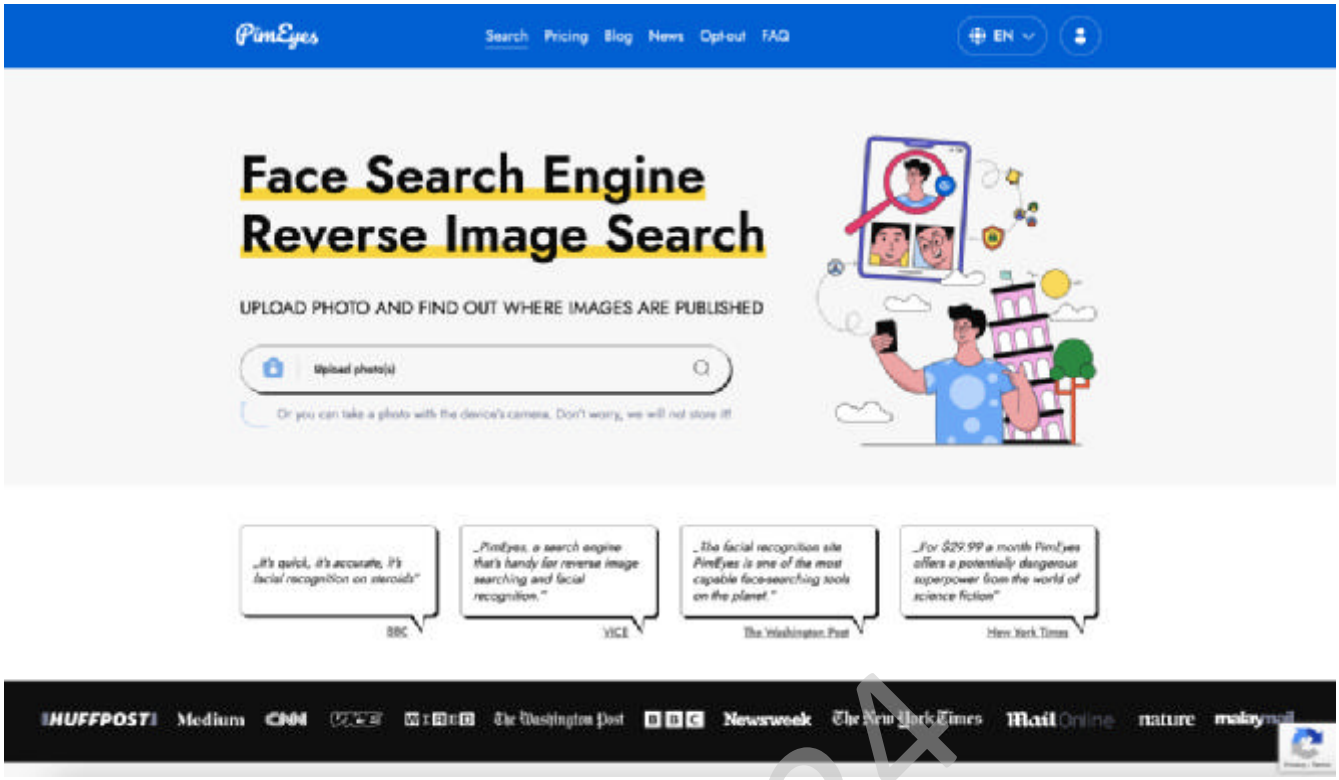

Рисунок 2.3 – PimEyes

Послуга пропонує як безкоштовні, так і платні плани підписки з розширеними функціями, такими як спеціальні сповіщення про нові зображення. Однак PimEyes має мінімальні бар'єри для запобігання пошуку без згоди, що викликає серйозні проблеми щодо конфіденційності.

Незважаючи на те, що PimEyes може допомогти людям контролювати свою присутність в Інтернеті, його потенціал для зловживань як інструменту стеження викликав суперечки. Критики стверджують, що йому бракує гарантій, щоб зупинити переслідування, переслідування та інше зловживання.

У відповідь на це PimEyes додала можливості відмови, заборонила певні пошуки та посилила безпеку. Але залишаються фундаментальні питання щодо необмеженого доступу до таких потужних можливостей розпізнавання обличчя.

TinEye

TinEye – провідна пошукова система зворотного зображення, розроблена компанією Idée Inc. у Канаді. Це був перший, хто використовував технологію ідентифікації зображень, а не ключові слова чи метадані.

TinEye створює унікальний «відбиток» для надісланих фотографій і порівнює його з понад 60 мільярдами проіндексованих зображень, щоб знайти збіги. Він відмінно справляється з пошуком ідентичних або відредагованих версій зображень шляхом зіставлення текстур, об'єктів і візерунків.

Рисунок 2.4 – TinEye

TinEye більше призначена для виявлення візуально схожих зображень, ніж просто для розпізнавання обличчя. Він чудово виявляє копії та різні версії зображення, а не лише ті, які виглядають майже однаково.

Користувачі можуть завантажувати фотографії, вводити посилання на зображення або використовувати додатки браузера TinEye для пошуку. Цей

інструмент зручний для відстеження авторських прав, перевірки фактів і пошуку джерела зображення.

Однак на сьогоднішній день TinEye проіндексував лише частину всіх зображень в Інтернеті. Точність збігу залежить від унікальності деталей зображення. Загальні або розріджені зображення можуть не мати надійних даних відбитків пальців.

Bing Visual Search

Bing Visual Search – це погляд Microsoft на зворотний пошук зображень. Як і Google, він дозволяє завантажувати фотографії або вставляти URL-адреси для сканування своєї бази даних на наявність збігів і пов’язаних зображень.

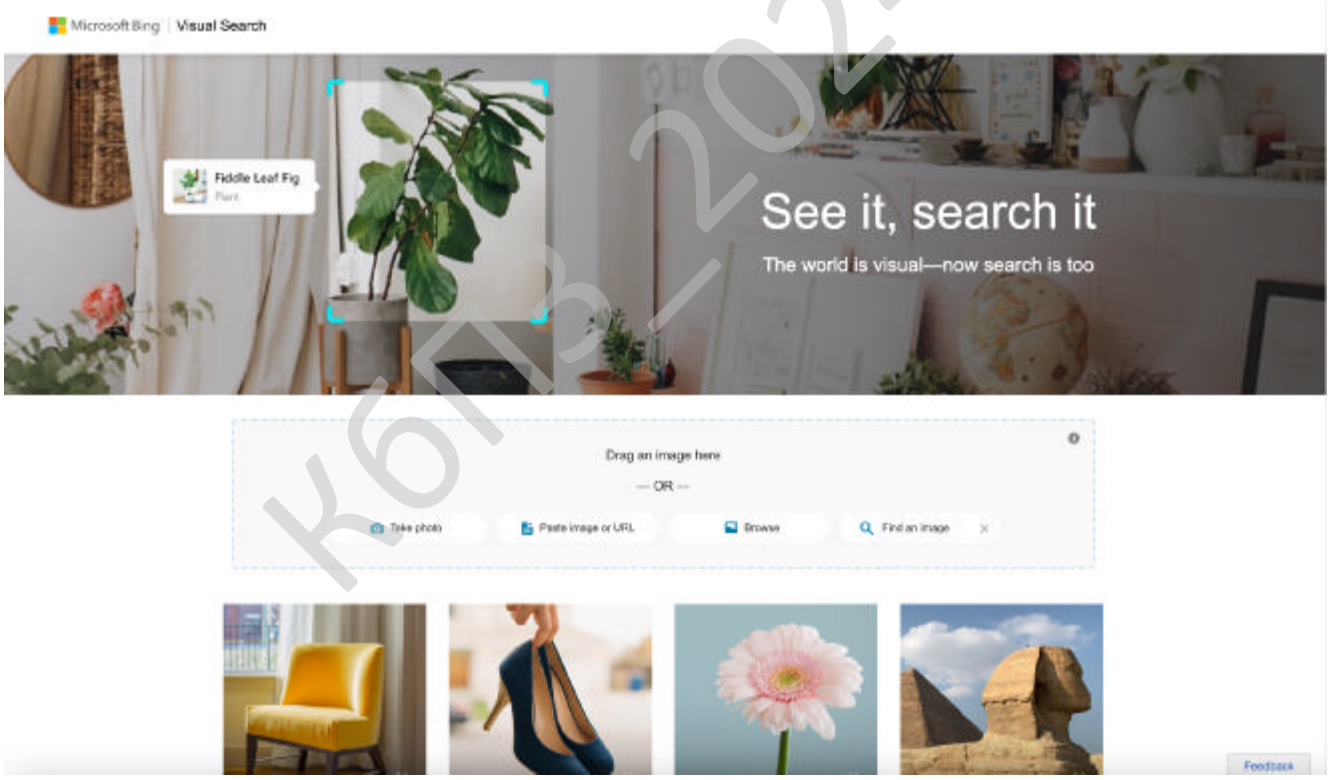

Рисунок 2.5 – Bing Visual Search

Ключовою перевагою Bing є його чистий, візуально привабливий інтерфейс. Результати впорядковано для легкого відсіву, часто з додатковим

контекстом, як-от пов’язані пошуки. Це приносить користь дослідникам і допитливим.

Bing чудово вміє знаходити різні розміри, роздільну здатність і редаговані версії зображень. Це робить його корисним для дизайнерів, фотографів і всіх, хто стежить за своєю роботою в Інтернеті.

Інструмент доступний прямо з домашньої сторінки Bing, без реєстрації. І хоча охоплення Bing може не зрівнятися з Google, воно часто дає унікальні результати, які інші пропускають.

Для тих, хто хоче поєднати потужність і простоту зворотного пошуку, візуальний пошук Bing влучить у ціль. Його поєднання інтуїтивно зрозумілого дизайну, гнучкого підбору та унікальних відкриттів робить його вагомим вибором, який варто дослідити.

Pinterest Lens

Pinterest найбільш відомий своїми візуальними дошками та креативними шпильками, а не розпізнаванням обличчя. Однак його інструмент візуального пошуку може допомогти в обмеженому пошуку облич на основі стилю та естетичної схожості.

Якщо ви помітили інтригуюче обличчя на Pinterest, можливо, завдяки унікальному вигляду чи аксесуару, візуальний пошук допоможе знайти шпильки зі схожими стилями. Він зосереджується на зовнішності, а не на точному зіставленні обличчя.

Щоб скористатися ним, просто завантажте або закріпіть зображення. Pinterest відображатиме пов’язані шпильки зі стилістичними елементами, такими як зачіска, макіяж або модні аксесуари. Однак точність змінюється, оскільки результати підкреслюють творчі тенденції, а не окремі особи.

Для тих, хто цікавиться відкриттями, орієнтованими на стиль, Pinterest пропонує інший підхід до пошуку облич порівняно з інструментами суворого розпізнавання. Його акцент робиться на красі, дизайні та натхненні.

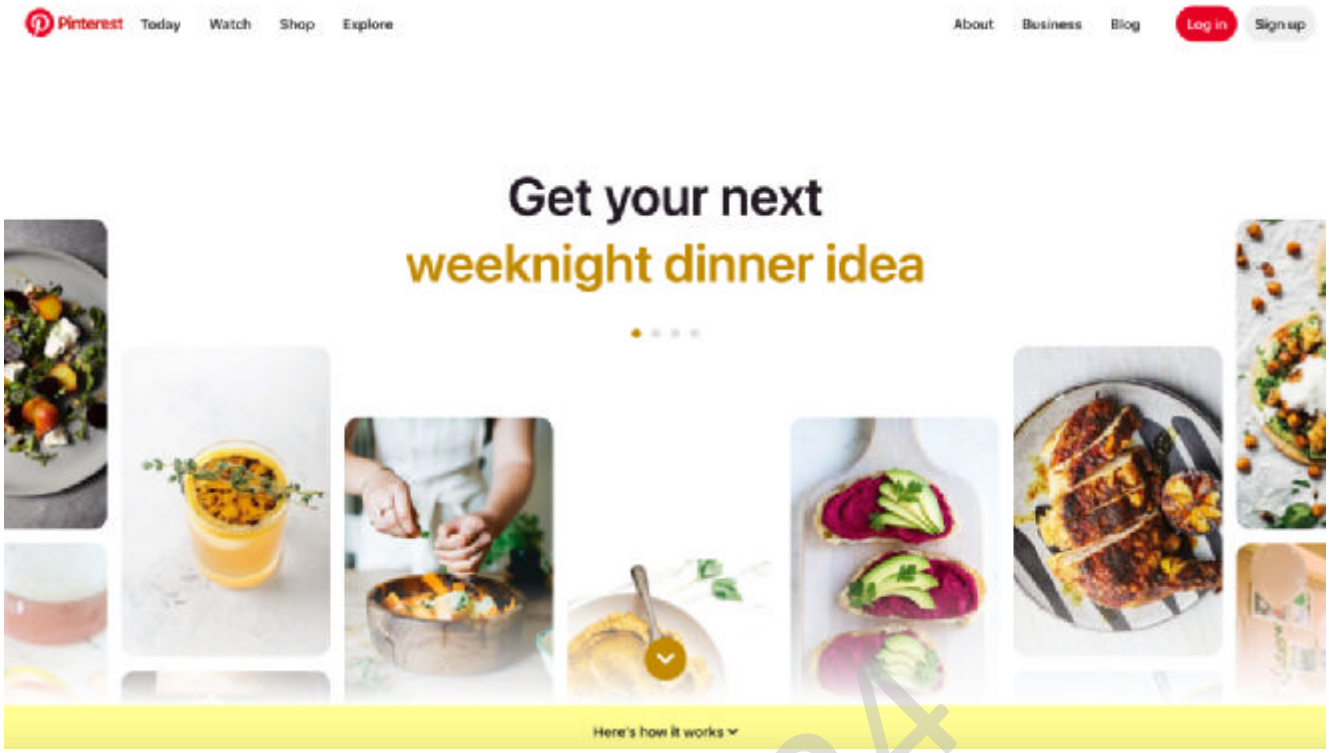

Рисунок 2.6 – Pinterest Lens

Таким чином, Pinterest пропонує скромні можливості пошуку по обличчю через візуальний пошук. Хоча це не платформа розпізнавання облич, вона унікальним чином пов'язує обличчя зі стильовими темами та тенденціями. Поставтеся до нього як до інструменту відкриття, а не як інструменту ідентифікації.

Підсумки

Зворотний пошук зображень став зручним інструментом для ідентифікації облич на фотографіях в Інтернеті. Такі пошукові системи, як Google і Bing, дозволяють завантажувати таємниче обличчя, щоб знаходити відповідні зображення та профілі в Інтернеті. Спеціалізовані сервіси, такі як FaceCheck.ID, використовують технологію розпізнавання облич, щоб зв'язати обличчя з особами.

Кожна платформа пошуку по обличчю має певні сильні сторони – будь то повні бази даних, точне розпізнавання чи виявлення двійників. Вибираючи

послугу, враховуйте такі фактори, як точність, захист конфіденційності та реальні відгуки користувачів.

При етичному використанні ці системи пошуку облич можуть допомогти вам знову знайти старого друга, підтвердити збіг або просто задовольнити цікавість до обличчя, поміченого в Інтернеті. За допомогою правильного інструменту та розумних запобіжних заходів реверсивний пошук зображень надає цінні можливості для дослідження облич у нашому цифровому світі.

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Embarcadero Delphi, раніше Borland Delphi і Codegear Delphi, – інтегроване середовище розробки ПЗ для Microsoft Windows, Mac OS, iOS і Android мовою Delphi (що раніше носила назву Object Pascal), створена спочатку фірмою Borland і на даний момент приналежна й розроблювальна Embarcadero Technologies. Embarcadero Delphi є частиною пакета Embarcadero RAD Studio і поставляється в чотирьох редакціях: Community (поширюється безкоштовно й має обмежену ліцензію на використання в комерційних цілях), Professional, Enterprise і Architect.

Delphi 10.4 Sydney

Випущено 26 травня 2020 року. RAD Studio Delphi 10.4 забезпечує значно поліпшену високопродуктивну нативну підтримку Windows, кращу продуктивність розробки, миттєві підказки code completion, прискорення виконання коду із синтаксисом керованих записів, поліпшення виконання паралельних завдань на сучасних багатоядерних CPU, а також містить більш 1000 виправлень багів, поліпшення продуктивності середовища й бібліотек і багато чого крім того.

Основні можливості Delphi 10.4.1:

− Істотні розширення для Windows: поліпшення для застосунків на моніторах 4K High DPI, інтеграція з новим WebView2 на базі Chromium,

використання розширених title bars, таких же, як в Office, Explorer, Google Chrome.

− Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

− Істотне поліпшення Delphi Code Insight (без можливого блокування IDE – в окремому процесі), що допоможе при роботі з великими проектами.

− Тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання.

− Розширена підтримка бібліотек C++: ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode.

− Відладник Win 64 (на LLDB) і збирач для C++.

− Поліпшення для C++: Включена велика кількість поліпшень STL з Dinkumware.

− Підтримка Metal Driver GPU для macOS і iOS.

− Вбудований Fmxlinux.

− Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API. Реалізація компонента Media Player для macOS тепер використовує Avfoundation. Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.

− Численні поліпшення швидкості й стабільності роботи нашої бібліотеки The Parallel Programming Library (PPL).

− Додані оновлені драйвери для FireBird, PostgreSQL і SQLite.

− Клієнтські бібліотеки HTTP і REST Client розширені застосунковими можливостями роботи з HTTPS. Також були розширені можливості підтримки Amazon AWS services

− У технологію Visual LiveBindings внесена безліч поліпшень, у тому числі швидкодії, що стосуться, застосунків на VCL і FireMonkey

RAD Studio 10.4 Короткий огляд:

– Істотні розширення для Windows. Створення застосунків, що чудово виглядають, із чіткими елементами інтерфейсу на 4k моніторах High DPI за допомогою нової гнучкої підтримки стилів елементів керування на екрані. Інтеграція із сучасними, безпечними web-технологіями від Microsoft – новим WebView2 на базі Chromium. Використання сучасних розширених title bars, таких же, як в Office, Explorer, Google Chrome, у своїх проектах. Істотні поліпшення надійності налагодження в новому відладнику для C++ Windows 64-bit.

– Зросла продуктивність розробки. Ріст продуктивності за рахунок миттєвої реакції підказок code completion у середовищі IDE. Краща сумісність із уже наявною кодовою базою, і спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю. Швидке зв'язування даних і візуальних елементів за допомогою розширеної технології Visual LiveBindings з підвищеною швидкодією. Просте використання розповсюджених бібліотек C++, наприклад, ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode. Оновлена підтримка Amazon AWS cloud.

– Поліпшення швидкодії і якості. Більш 1000 поліпшень швидкодії і якості. Краща ефективність коду за допомогою нового синтаксису custom managed records. Більш швидке виконання паралельних завдань на сучасних багатоядерних CPU. Переконаєтеся в прискоренні відображення на екрані з підтримкою Metal API на macOS і iOS. Краща сумісність із уже наявною кодовою базою й спрощення програмування за рахунок уніфікованої архітектури керування пам'яттю.

Істотне поліпшення Delphi Code Insight

Як найбільше й головне поліпшення інструментів програмування Delphi за багато років, в 10.4 Delphi Code Insight реалізований через Language Server Protocol (LSP). LSP – це технологія генерації результатів для code completion, навігації й інших сервісів в окремому процесі. Це значить, що code completion і Code Insight одержать більш точні результати без блокування IDE. 10.4

забезпечує набагато більш високу продуктивність розроблювачів, які працюють із більшими проектами, що містять мільйони рядків коду.

Delphi Custom Managed Records

Ключове розширення мови Delphi: тип даних Delphi «record» тепер підтримуть довільні ініціалізацію, фіналізацію й операції копіювання. Управляйте тем, як ці структури створюються, копіюються й звільняються з допомогу вашого коду, який буде виконуватися у відповідний момент.

Це розширює потужність конструкцій records в Delphi, які використовуться щоб одержати більшу ефективність у порівнянні із класами.

Єдине керування пам'яттю

Керування пам'яттю в Delphi тепер стандартизоване на всіх підтримуваних платформах – мобільних, настільних і серверних – використовучи класичну реалізацію керування пам'яттю об'єктів.

У порівнянні з Automatic Reference Counting (ARC), це дає кращу сумісність із існучим кодом і спрощує написання компонентів, бібліотек і застосунків.

ARC модель керування пам'яттю model залишилася для керування рядками й посиланнями на тип інтерфейсу на всіх платформах. Для C++ це означає, що при створенні й звільненні Delphi-style класів в C++ використовується звичайне керування пам'яттю, як у будь-якого heap-allocated класу C++, що значно знижує складність коду.

Розширена підтримка бібліотек C++

В 10.4 ми портували багато популярних бібліотек C++ у C++Builder.

Забезпечивши оптимізовану підтримку бібліотек ZeroMQ, SDL2, SOCI, libSIMDpp і Nematode, поряд із уже підтримуваними Boost і Eigen, які можуть бути додані за допомогою менеджера пакетів Getit.

Win 64-відладник і збирач для C++

В 10.4 з'явився новий відладник C++ для Windows 64-bit. Відладник заснований на LLDB і показує значне збільшення стабільності при налагодженні

64-bit застосунків поряд з новими відладочними можливостями, такими як перегляд і інспекція типів начебто рядків C++ і Delphi, а також колекцій STL, включаючи std::vector, std::map і інших. Крім того, згенерована для застосунку відладочна інформація має інший внутрішній формат, сприяючи більш стабільному й багатому на можливості процесу налагодження, більш докладним перегляду й інспекції в debug-time.

Підвищення якості й швидкодії інструментів

- Велика кількість поліпшень STL від Dinkumware.
- Поліпшені деякі найважливіші методи й області RTL, на базі поліпшень сумісності з популярними бібліотеками C++.
- Поліпшена підтримка Cmake.
- Велика кількість виправлень для підвищення стабільності і якості.
- Відновлення Windows API – Обновлено й додали безліч декларацій API щоб добитися ще більшої інтеграції із платформою Windows.
- Загальні вдосконалення в бібліотеці доступу до БД FireDAC, включаючи оновлені драйвера для FireBird, PostgreSQL і SQLite. Вибір статичного або динамічного підключення SQLite до застосунку.

Змінені стилі VCL для High DPI

В 10.4, архітектура стилізації VCL була суттєво розширена для підтримки High DPI і 4K моніторів. Тепер усі елементи UI на формі VCL автоматично масштабуться під відповідне до монітора дозвіл для показу форми. Був оновлений API стилізації для підтримки стилів high DPI.

Кожний графічний елемент UI може бути обраний з наборів різних масштабів і масштабований до потрібного DPI, що дає чітке зображення елементів UI на всіх моніторах.

Нові High DPI стилі й стилізація окремих VCL компонент

Обновлено велике число вбудованих і преміальних VCL стилів для підтримки нового режиму стилізації High-dpi. Це дозволяє вам створювати застосунку з відмінним дизайном для всіх моніторів.

Розроблювачі VCL застосунків тепер можуть використовувати трохи VCL стилів на різних формах в одному застосунку або в різних компонентів на одній формі. Це також включає стилізацію компонентів загальною темою для платформи. Крім застосункової гнучкості використання стилів, це дозволяє використовувати нестилізуємі компоненти із зовнішніх бібліотек в VCL застосунках, що використовуть стиль.

Поліпшена кроссплатформеність

- Додана підтримка Metal Driver GPU для macOS і iOS.
- Крім підтримки останнього iOS SDK, в RAD Studio 10.4 розроблювачі можуть задовольнити нові вимоги Apple до набору стартових екранів.
- Реалізований заново стилізуємий FMX компонент TMemo на платформі Windows значно поліпшений і тепер має відмінну підтримку IME.
- Користувачам редакцій Enterprise або Architect доступна повна інтеграція Fmxlinux з IDE для створення клієнтських застосунків Linux з GUI.
- Компонент Twebbrowser для iOS тепер реалізований на Wkwebview API.
- Реалізація компонента Media Player для macOS тепер використовує Avfoundation.

Оновлений менеджер пакетів Getit

Менеджер пакетів Getit в IDE був значно вдосконалений.

Дати випуску релізів пакетів тепер видні, і можливе сортування списку по цих датах; відбір тільки встановлених пакетів, контенту, доступного тільки при наявності підписки, багато чого іншого.

Універсальний інсталятор для установки Online і Offline

В 10.4 включений новий універсальний інсталятор, який використовує технологію на базі Getit. Цей інсталятор підтримує як online, так і offline (з ISO) варіанти установки. Тепер обоє варіанта установки дозволяють вам указати початковий набір можливостей RAD Studio для установки, наприклад, свою комбінацію мов програмування й цільових платформ, мов інтерфейсу, і додавати до нього або видаляти непотрібне в будь-який момент.

2.3 Розгорнута постановка завдання

Згідно з технічним завданням на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

В процесі розробки випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти необхідно виконати наступний обсяг роботи:

а) провести аналіз існуючих систем-аналогів для виявлення їх позитивних і негативних якостей. Результати аналізу врахувати в подальших розробках;

б) вибрати та обґрунтувати методику побудови системи контролю роботи технологічного обладнання на виробництві в автоматизованому режимі. Розробити функціональну та структурну схеми системи;

в) розробити програмне забезпечення системи, що дозволить реалізувати поставлену технічним завданням задачу. Побудувати блок-схеми алгоритмів програми та підпрограми;

г) організувати інтерфейс користувача з метою формування та виводу на екран ЕОМ повідомлень про некоректні дії користувача та нестандартні ситуації в роботі технологічного обладнання;

д) розробити рекомендації по організаційних та методичних заходах, які забезпечать впровадження системи в промислову експлуатацію та її подальшу успішну експлуатацію;

е) провести розрахунки по визначенню економічної ефективності розробленої системи;

ж) розробити заходи по охороні праці при впровадженні та експлуатації системи, а також розробити заходи з цивільного захисту;

з) сформувати висновки про виконаний обсяг робіт та одержані результати.

3 ОПИС І ОБҐРУНТУВАННЯ ПРОЕКТНИХ РІШЕНЬ

3.1 Опис функціонування системи

Розробимо методику верифікації контурних зображень об’єктів, яка виключає ситуації невизначеності та дозволяє забезпечити відповідність певним вимогам, що пред’являються до замкнутого контуру.

Розробимо математичну модель семантичного перетворення контурного зображення в структуру концепту, що основана на побудові вектора структурних перетворень початкового зображення. Кожен етап визначається застосуванням відповідних функцій перетворення відносно введених структурних інваріантів – характеристик структур, що усувають вплив афінних перетворювань (зсув, гомотетія, поворот) та деформаційних спотворювань контуру (стиск, розтягнення, зміна зовнішніх кутів, додавання та видалення існуючих структурних елементів). В результаті ітеративного застосування функцій перетворення формується послідовність структур, які мають інваріантні ознаки класу розпізнавання, а структура вищого рівня визначає структурний концепт.

Замкнутий контур Γ цифрового зображення I тривимірного об’єкту подається як базисна структура $z^{(0)}$, що складається з множини $A^{(0)} = \{a_1^{(0)}, a_2^{(0)}, ..., a_n^{(0)}\}$ непохідних елементів (пікселів) $a_i^{(0)}$ з координатами (x_i, y_i), де x_i – номер рядка, y_i – номер стовпця. Елементи $a_i^{(0)}$ знаходяться між собою у бінарних відношеннях r, де $r = \left\langle A^{(0)}, R \right\rangle$, $R \subseteq A^{(0)} \times A^{(0)}$. Відношення r для довільних елементів $a_i^{(0)}$, $a_j^{(0)}$ та $a_k^{(0)}$ множини $A^{(0)}$ відповідають аксіомам T (рефлексивності $(\forall(a_i^{(0)}), [a_i^{(0)}\, r\, a_i^{(0)}])$, симетричності $(\forall(a_i^{(0)}), \forall(a_j^{(0)}), [a_i^{(0)}\, r\, a_j^{(0)} \Rightarrow a_j^{(0)}\, r\, a_i^{(0)}])$ та транзитивності $(\forall(a_i^{(0)}), \forall(a_j^{(0)}), \forall(a_k^{(0)}),$

$[a_i^{(0)} r\, a_j^{(0)} \wedge a_j^{(0)} r\, a_k^{(0)} \Rightarrow a_i^{(0)} r\, a_k^{(0)}]))$.

Формування множини структур $Z = \{ z^{(0)}, z^{(1)}, \dots, z^{(5)} \}$ відбувається відповідно з множиною функцій перетворення $S = \{ s_1, s_2, \dots, s_5 \}$ шляхом упорядкування просторових елементів на лінійній шкалі порядку $p^{(i)}$ у напрямку обходу контуру відносно «точки захоплення».

З математичної точки зору процес семантичного перетворення контурного зображення в структуру концепту можна представити наступним чином: $z^{(i)}$: $z^{(i)} = \left\langle A^{(i)}, r, T \right\rangle$; $A^{(i)} = s_i(A^{(i-1)})$, $(i = 1, ..., 5)$, де $z^{(i)}$ – структура i-го рівня, $A^{(i)}$ – множина структурних елементів i-го рівня, що формується на основі структурних елементів $(i-1)$-го рівня шляхом застосуванням відповідних функцій перетворення s_i та структурних інваріантів $G^{(i)}$, які мають вигляд:

$$s_1 : a_j^{(1)} = a_j^{(0)} \bigcup a_{j+1}^{(0)}; \qquad G^{(1)} = \underset{\Xi_1}{Inv}(z^{(1)}) = g_\gamma^{(1)}; \qquad (3.1)$$

$$s_2 : a_j^{(2)} = a_j^{(1)} \bigcup a_{j+1}^{(1)} \bigcup \dots \bigcup a_{j+k}^{(1)}; \qquad G^{(2)} = \underset{\Xi_1}{Inv}(z^{(2)}) = g_\gamma^{(2)}; \qquad (3.2)$$

$$s_3 : \hat{a}_j^{(3)} = a_j^{(2)} \bigcup a_{j+1}^{(2)}; \qquad G^{(3)} = \underset{\Xi_2}{Inv}(z^{(3)}) = g_{k\ уз}^{(3)}; \qquad (3.3)$$

$$s_4 : a_j^{(4)} = \underset{R,L}{Cont}(\hat{a}_j^{(3)}, ..., \hat{a}_{j+k}^{(3)}); \qquad G^{(4)} = \underset{\Xi_3}{Inv}(z^{(4)}) = g_{\ell\ уз}^{(4)}; \qquad (3.4)$$

$$s_5 : a_j^{(5)} = \underset{R \to L(L \to R)}{Chan}(a_j^{(4)}, ..., a_{j+k}^{(4)}); \qquad G^{(5)} = \underset{\Xi_3}{Inv}(z^{(5)}) = g_{\ell\ уз}^{(5)}. \qquad (3.5)$$

В (3.1-3.5) $a_j^{(i)}$ – структурні елементи i-го рівня; $Cont$ – функція, що вказує на сталість правого – R (рисунок 3.1, а) або лівого – L (рисунок 3.1, б) розвитку структури на структурних елементах $\hat{a}_j^{(3)}, \dots, \hat{a}_{j+k}^{(3)}$; $Chan$ – функція, що вказує на зміну напрямку розвитку структури (з правого на лівий $R \to L$ або з лівого на правий $L \to R$) на структурних елементах $a_j^{(4)}, ..., a_{j+k}^{(4)}$; $G^{(i)} = \underset{\Xi_s}{Inv}(z^{(i)})$ – структурний інваріант i-го рівня у системі Ξ_s напрямків

розвитку структурних елементів; $g_{\gamma}^{(1)}$, $g_{\gamma}^{(2)}$, $g_{k\,уз}^{(3)}$, $g_{\ell\,уз}^{(4)}$, $g_{\ell\,уз}^{(5)}$ – множини напрямків розвитку структурних елементів для структури кожного рівня відповідно.

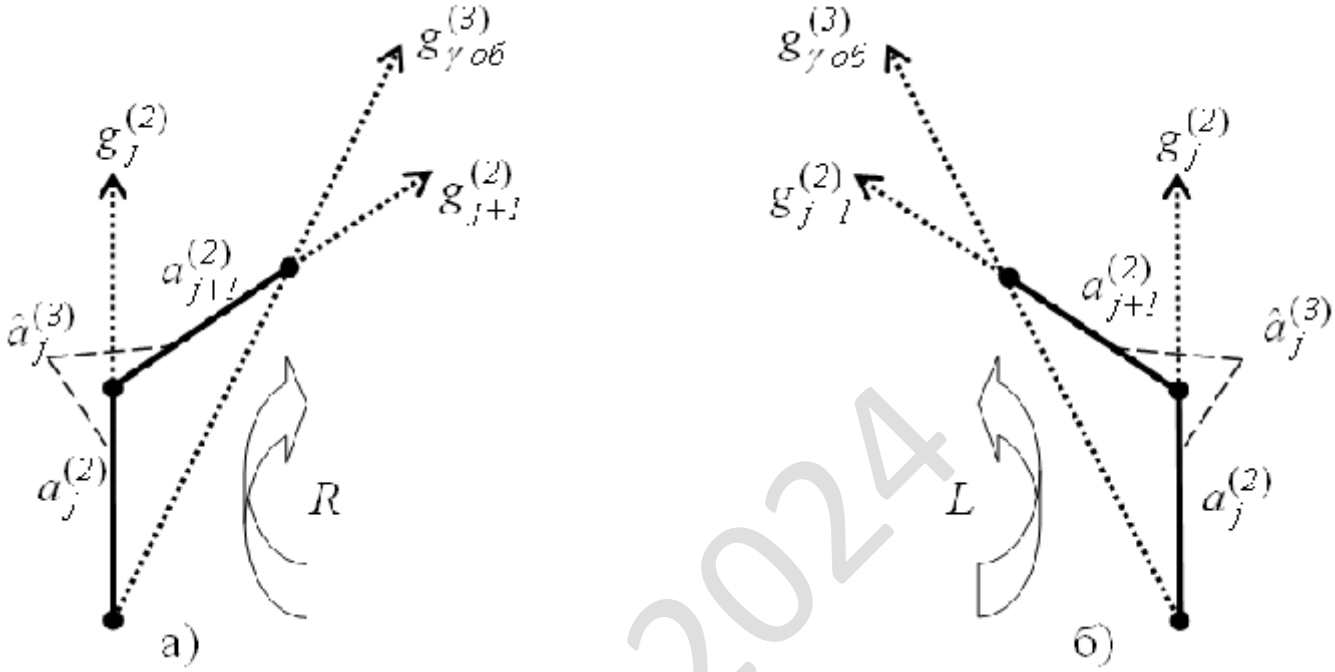

Рисунок 3.1 – Завдання правого (а) та лівого (б) напрямків розвитку структурного елемента $\hat{a}_i^{(3)}$

З практичної точки зору процес формування структурного концепту починається з визначення різнорідних структурних точок у контурі зображення. Спочатку визначаються структурні точки 1-го роду $t_i^{(1)}$ та 2-го роду $t_j^{(2)}$ за принципом, що пояснюється на рисунок 3.2. Потім серед усіх структурних точок 1-го роду $t_i^{(1)}$ знаходяться максимальні структурні точки 1-го роду $t_{i\,\max}^{(1)}$, які є точками максимальної опуклості структури (рисунок 3.3). Нарешті, серед усіх структурних точок 2-го роду $t_j^{(2)}$ знаходяться мінімальні структурні точки 2-го роду $t_{j\,\min}^{(2)}$, які є точками максимальної увігнутості структури (рисунок 3.4).

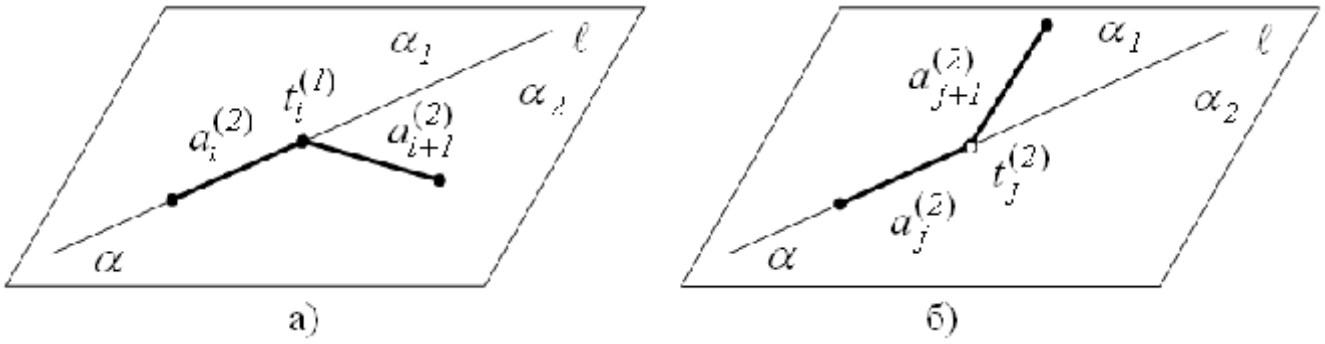

Рисунок 3.2 – Визначення структурних точок 1-го (а) та 2-го (б) роду

Далі максимальні структурні точки 1-го роду $t_{i\max}^{(1)}$ та мінімальні структурні точки 2-го роду $t_{j\min}^{(2)}$ з'єднуються між собою, утворюючи структуру $z^{(4)}$ (рисунок 3.5), яка потім мінімізується до структури $z^{(5)}$ (рисунок 3.6).

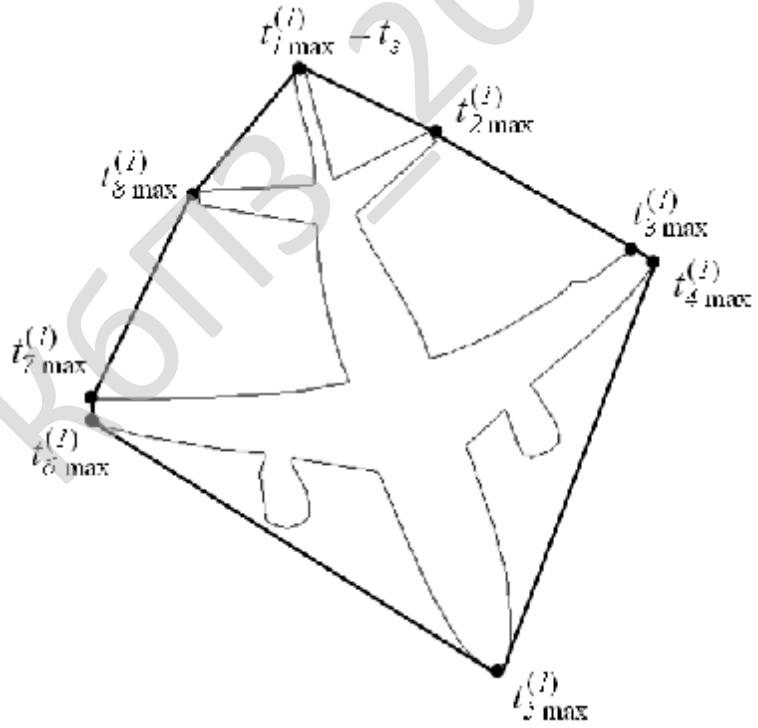

Рисунок 3.3 – Визначення максимальних структурних точок 1-го роду

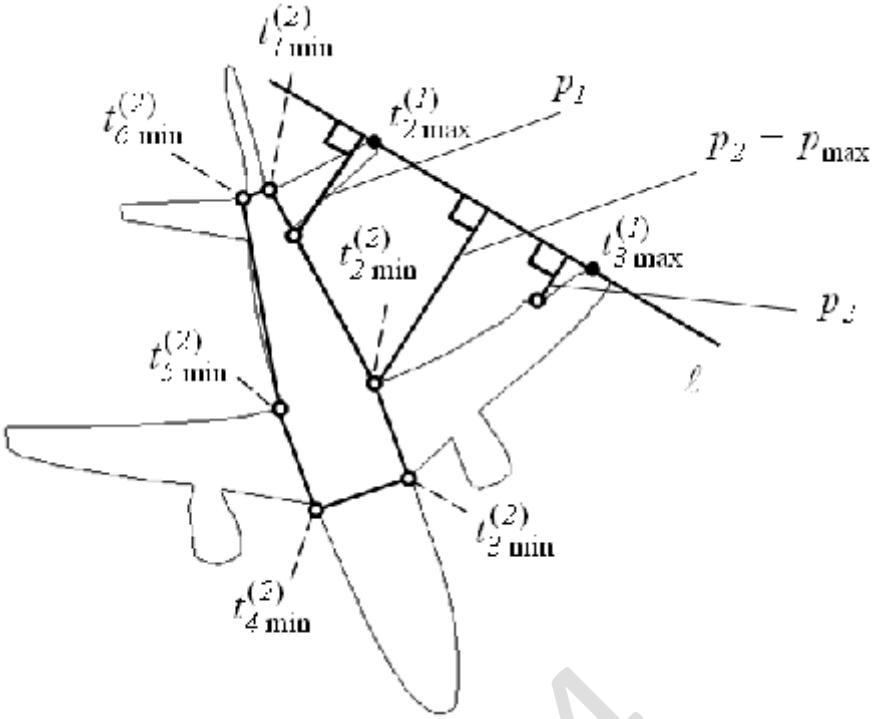

Рисунок 3.4 – Визначення мінімальних структурних точок 2-го роду

Дана структура є заключним етапом семантичного перетворення, має загальні ознаки класу розпізнавання та являє собою структурний концепт

$$Cpt(I) = \bigcup_{i=1}^{6} \left(t^{(1)}_{i\,\max} \wedge t^{(2)}_{i\,\min} \right). \tag{3.6}$$

Подальші перетворення пов'язані з нормалізацією структури $z^{(5)}$. Процес нормалізації відбувається шляхом застосування розробленого способу нормалізації, який передбачає визначення осі нормалізації на основі відповідного алгоритму побудови скелетону (рисунок 3.7), що складається з допоміжних структурних елементів $s1 - s6$.

Структурний концепт розміщується таким чином, щоб ось нормалізації ℓ розташовувалася вертикально. В нашому випадку максимальна структурна точка 1-го роду $t^{(1)}_{4\,\max}$, що визначає допоміжний структурний елемент s_4, буде знаходитись зверху (рисунок 3.8).

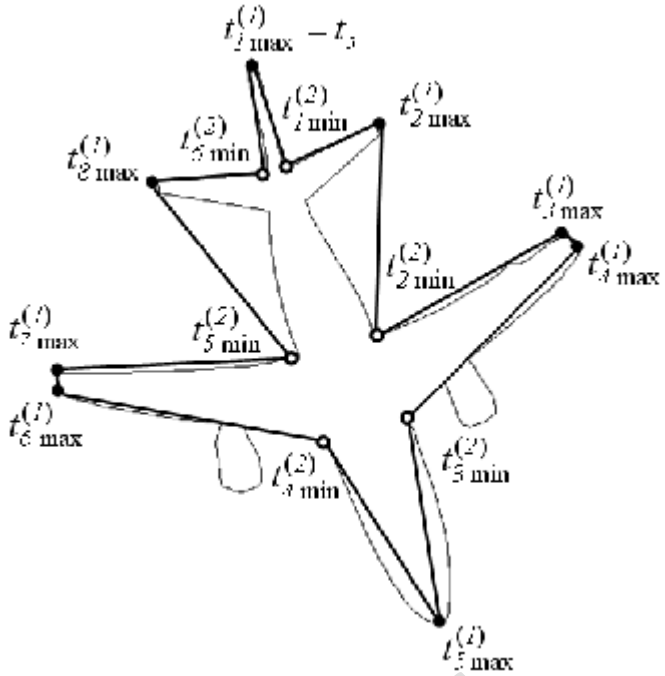

Рисунок 3.5 – Побудова структури $z^{(4)}$

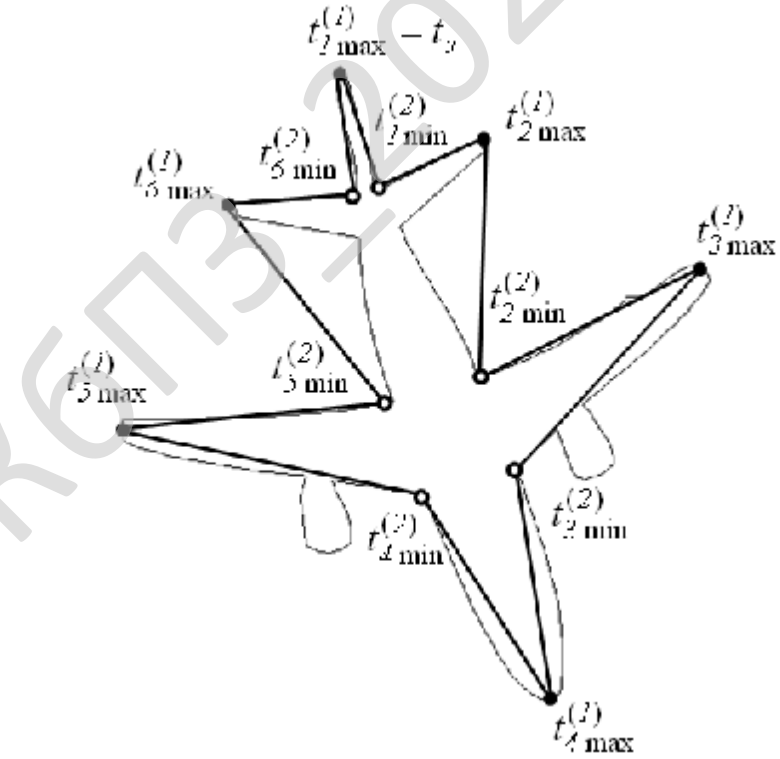

Рисунок 3.6 – Побудова структури $z^{(5)}$

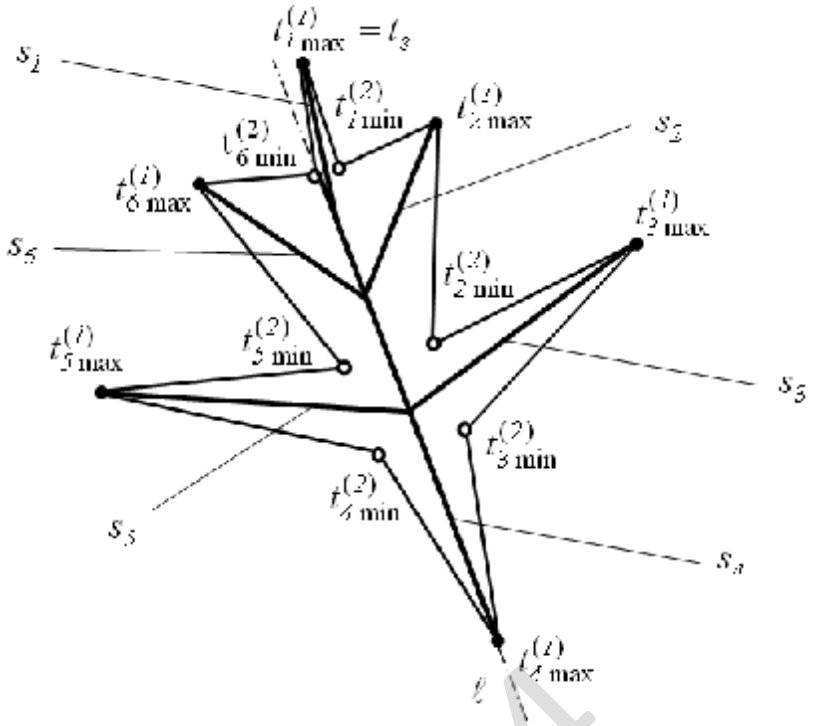

Рисунок 3.7 – Побудова скелетону

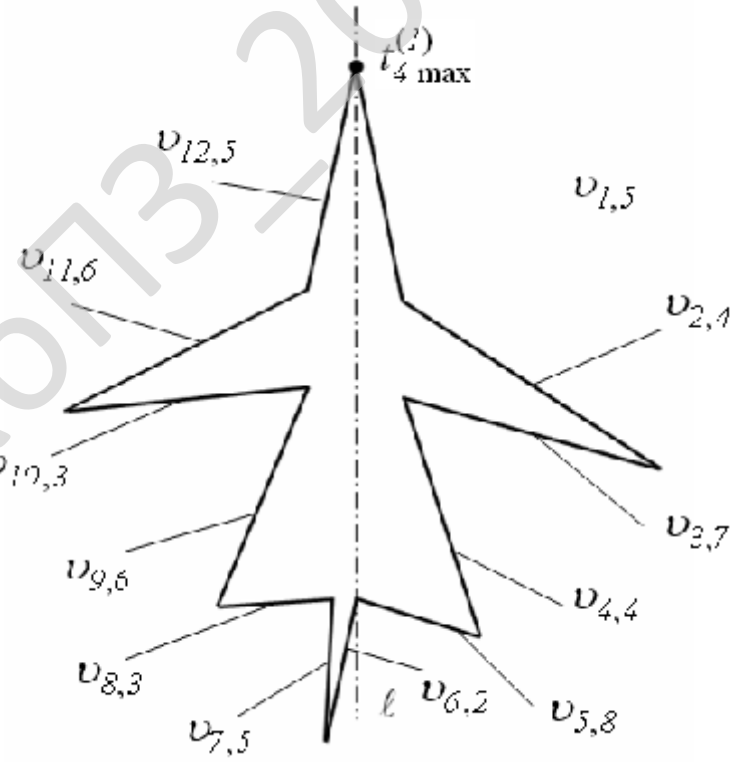

Рисунок 3.8 – Нормалізована структура

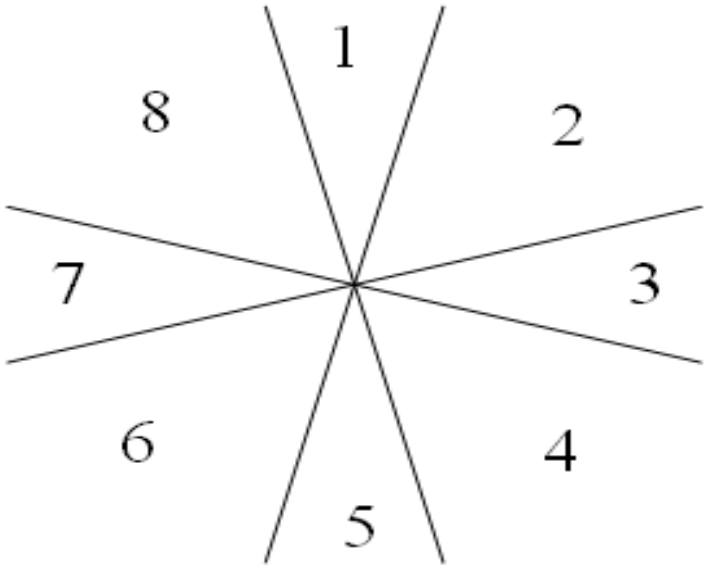

Рисунок 3.9 – Система орієнтації структурних елементів

Наперед задана система напрямків орієнтації структурних елементів (рисунок 3.9), що складається з 8 секторів, дає можливість поставити структурному концепту $Cpt(I)$ його лінгвістичну відповідність $LCpt(I)$ у вигляді конкатенації структурних елементів $\upsilon_{n,\varphi}$, де n – номер структурного елементу, φ – номер сектору, в якому опиниться поточний структурний елемент при зіставленні його з системою напрямків орієнтації. Повний концепт розбивається на лівий $LCpt_L(I)$ та правий $LCpt_R(I)$ структурно-лінгвістичні концепти. Це дозволяє організувати паралельну роботу скінчених автоматів, які реалізують фрагменти СМК в процесі класифікації та ідентифікації. Таким чином, повний концепт $LCpt(I)$ літака (рисунок 3.8) представляється сукупністю лівого $LCpt_L(I)$ та правого $LCpt_R(I)$ структурно-лінгвістичних концептів, що формуються шляхом послідовного зіставлення з системою напрямків орієнтації кожного зі структурних елементів, які розміщуються відповідно справа та зліва

від осі нормалізації ℓ:

$$LCpt_L(I) = \upsilon_{12,5} \wedge \upsilon_{11,6} \wedge \upsilon_{10,3} \wedge \upsilon_{9,6} \wedge \upsilon_{8,3} \wedge \upsilon_{7,5} \wedge \upsilon_{6,2}; \quad (3.7)$$

$$LCpt_R(I) = \upsilon_{1,5} \wedge \upsilon_{2,4} \wedge \upsilon_{3,7} \wedge \upsilon_{4,4} \wedge \upsilon_{5,8}. \quad (3.8)$$

Для $LCpt_L(I)$ структурні елементи розглядаються у зворотному порядку.

3.2 Розробка структурної схеми

Розробимо метод структурно-лінгвістичного розпізнавання контурних зображень тривимірних об'єктів, який оснований на побудові структурно-лінгвістичних концептів та застосуванні СМК.

Шляхом моделювання з використанням пакету 3DS Max для різних кутів α_n відхилення камери від положення в надир (рисунок 3.10) та різних напрямків γ_k камери стосовно об'єкту зйомки (рисунок 3.11) визначено оптимальну кількість узагальнених структурно-лінгвістичних концептів.

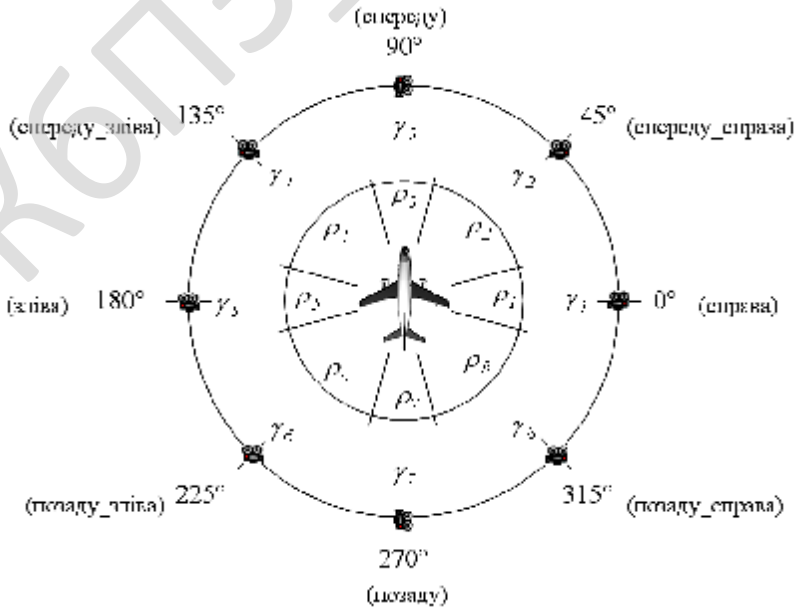

Рисунок 3.10 – Відповідність кутів та напрямків γ_k камери стосовно об'єкту зйомки

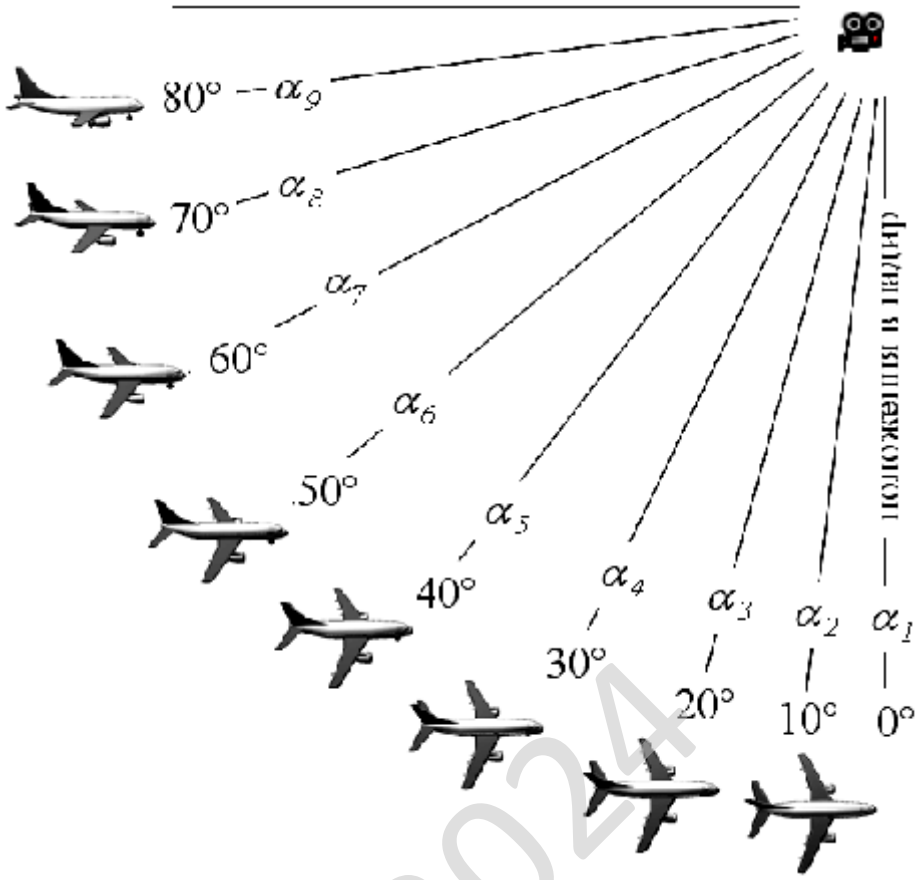

Рисунок 3.11 – Відхилення камери від положення в надир для різних α_n

З'ясовано, що достатньо використовувати один еталонний структурно-лінгвістичний концепт для положення в надир та 8 узагальнених еталонних структурно-лінгвістичних концептів, які враховують всі випадки отримання зображень під час зйомки з секторів $\rho_1, \rho_2, ..., \rho_8$ (рисунок 3.10). У подальшому для зображень, що отримані в даних секторах, використовуються відповідні напрямки $\gamma_1, \gamma_2, ..., \gamma_8$.Узагальнені еталонні структурно-лінгвістичні концепти представляються у вигляді СМК, яка дозволяє описувати об'єкти, явища та поняття предметної області за допомогою мережних структур, що основані на теорії графів.

Розглянуто приклад побудови СМК для літаків з нормальної аеродинамічною схемою. Фрагменти СМК для перших трьох рівнів та рівнів 2-5 для напрямку «справа» представлені відповідно на рисунок 3.12 та рисунок 3.13.

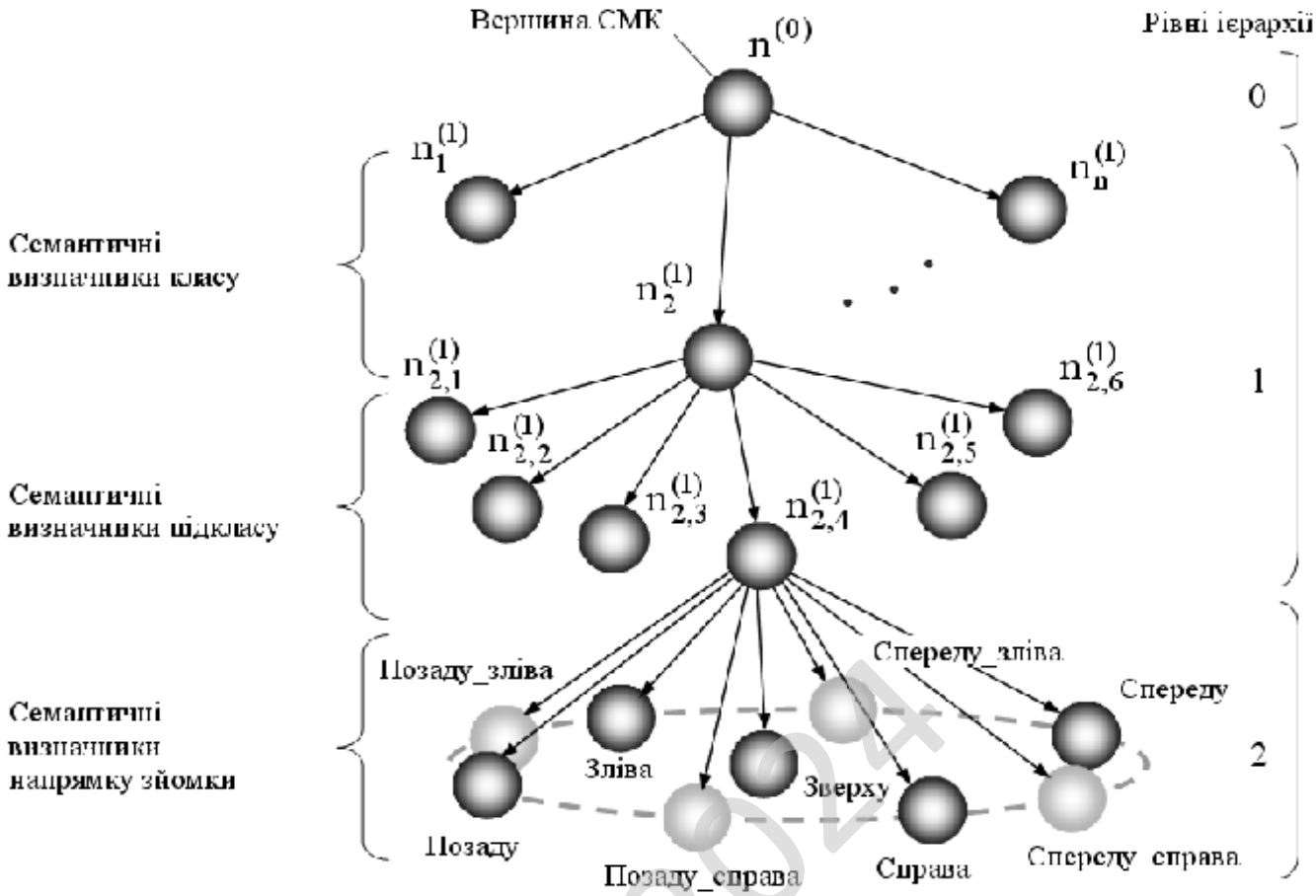

Рисунок 3.12 – Фрагмент СМК для перших трьох рівнів ієрархії

Ідентифікація об’єкта відбувається на основі ознак ідентифікації. Для їх визначення здійснюється декомпозиція об’єкта класифікації. Процес декомпозиції полягає в розбитті нормалізованої структури $z^{(5)}$ на підструктури 1-го рівня вкладеності z'_k, які обмежені сусідніми мінімальними структурними точками 2-го роду. Усі мінімальні структурні точки 2-го роду з'єднуються послідовно відрізками $b_\ell^{(2)}$ (рисунок 3.14), що дозволяє отримати шість підструктур 1-го рівня вкладеності z'_k, а також внутрішню структуру $z^{вн}$, яка обмежена структурними елементами $b_\ell^{(2)}$.

Подальший процес декомпозиції розглянуто на прикладі правого крила літака. Контурне зображення літака розміщується таким чином, щоб структурний

елемент $b_4^{(2)}$, який є спільним для z_2' та $z^{вн}$, був розташований на прямій ℓ (рисунок 3.15).

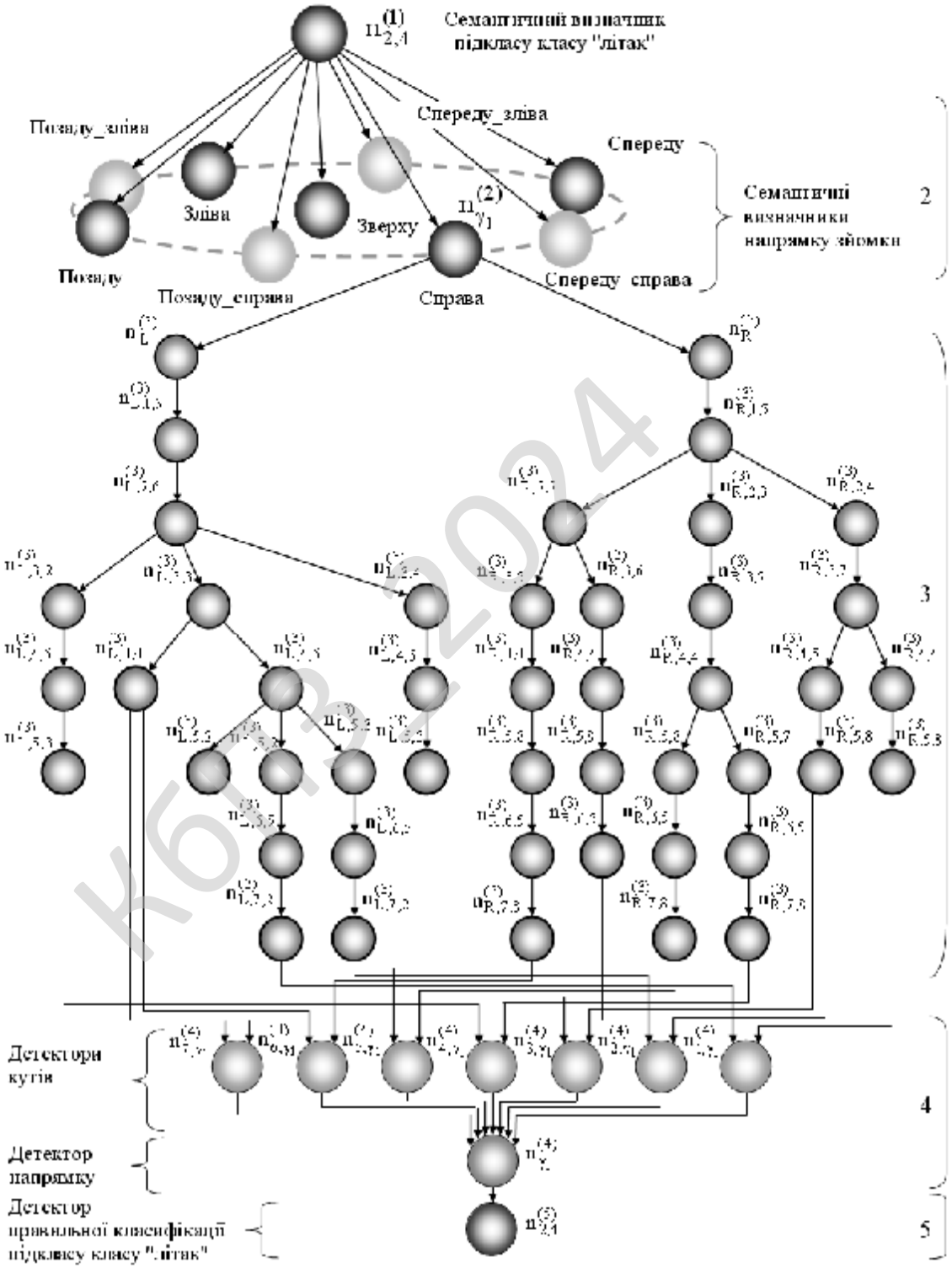

Рисунок 3.13 – Фрагмент СМК з детекторами для напрямку «справа»

Знайдені різнорідні критичні структурні точки підструктури z'_2 з'єднуються між собою. Уточнена підструктура 1-го рівня вкладеності (рисунок 3.16) також містить у собі дві мінімальні структурні точки 2-го роду $t'^{(2)}_{1\min}$ та $t'^{(2)}_{2\min}$, що означає наявність підструктури 2-го рівня вкладеності (рисунок 3.17).

Ієрархічна схема виділення підструктур 1-го та 2-го рівнів вкладеності, що відіграють роль ознак ідентифікації, представлена на рисунок 3.18.

Для кожної підструктури будуються декомпозиційні структурно-лінгвістичні концепти, які також входять до складу СМК. Фрагменти СМК, що містять декомпозиційні структурно-лінгвістичні концепти, відповідають за процес ідентифікації об'єкту. Наприкінці розділу розглянуто питання навчання системи розпізнавання.

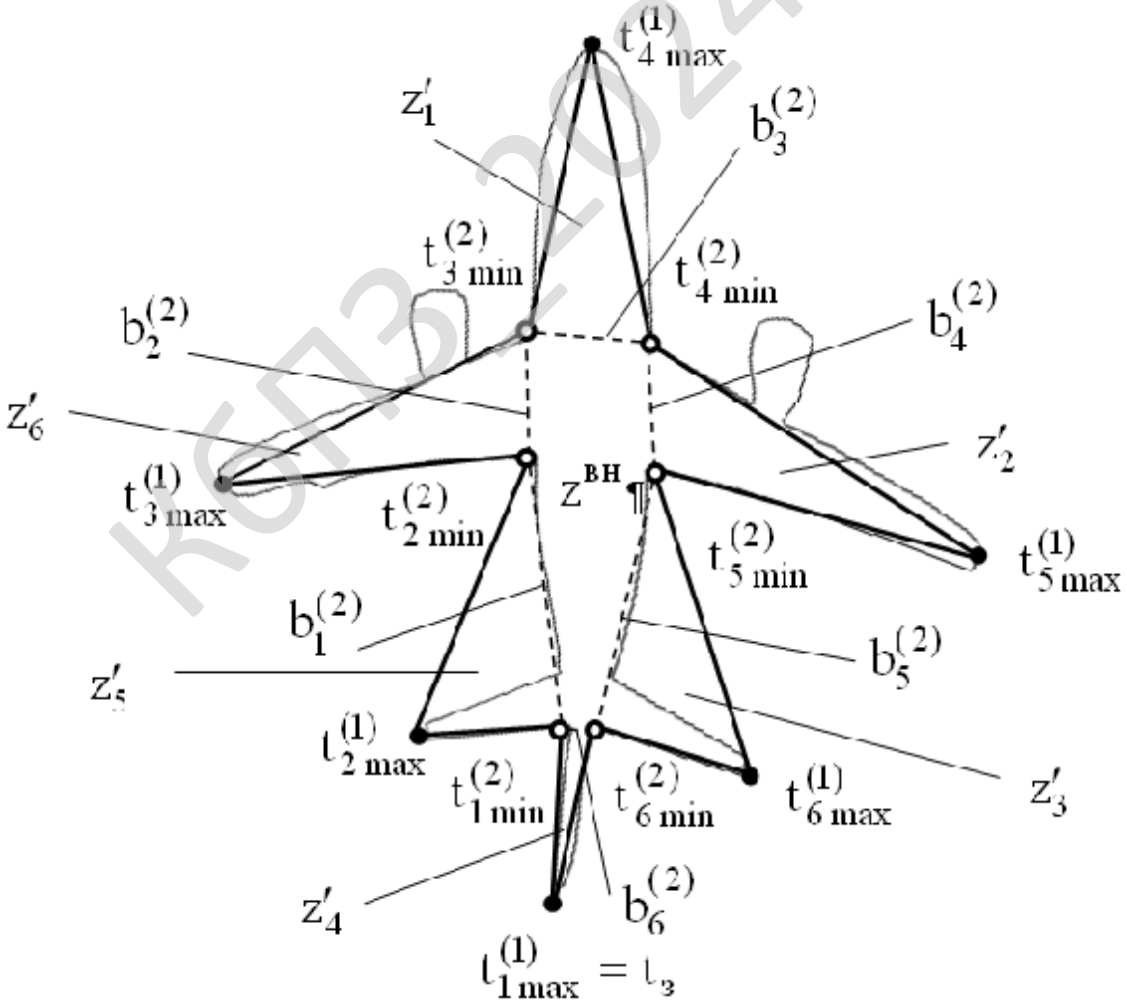

Рисунок 3.14 – Виділення підструктур 1-го рівня вкладеності z'_k

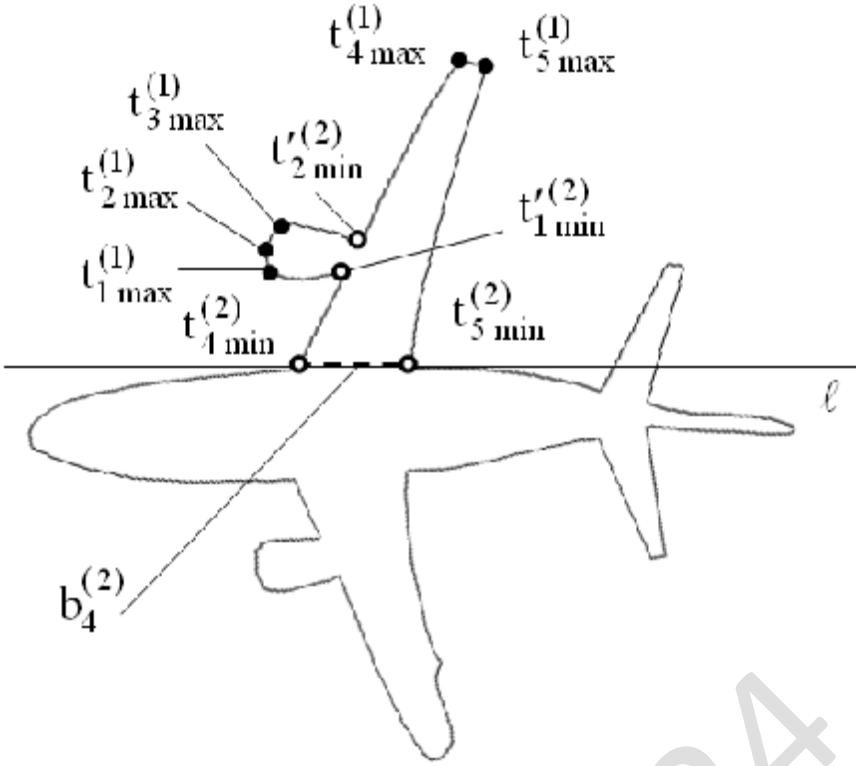

Рисунок 3.15 – Визначення різнорідних критичних точок для правого крила

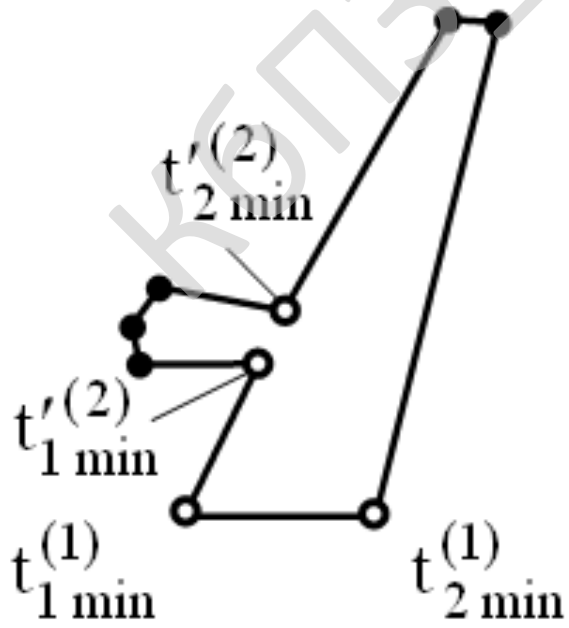

Рисунок 3.16 – Уточнена підструктура 1-го рівня вкладеності

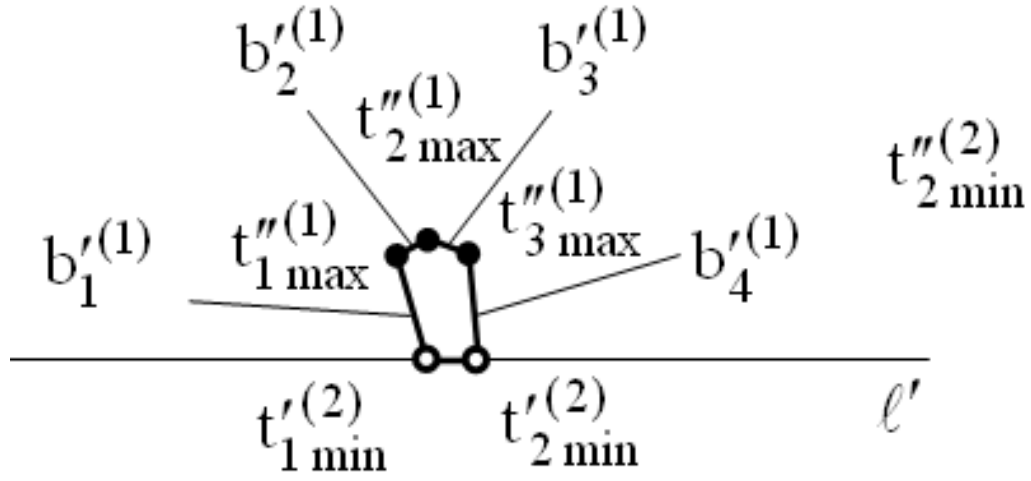

Рисунок 3.17 – Підструктура 2-го рівня вкладеності

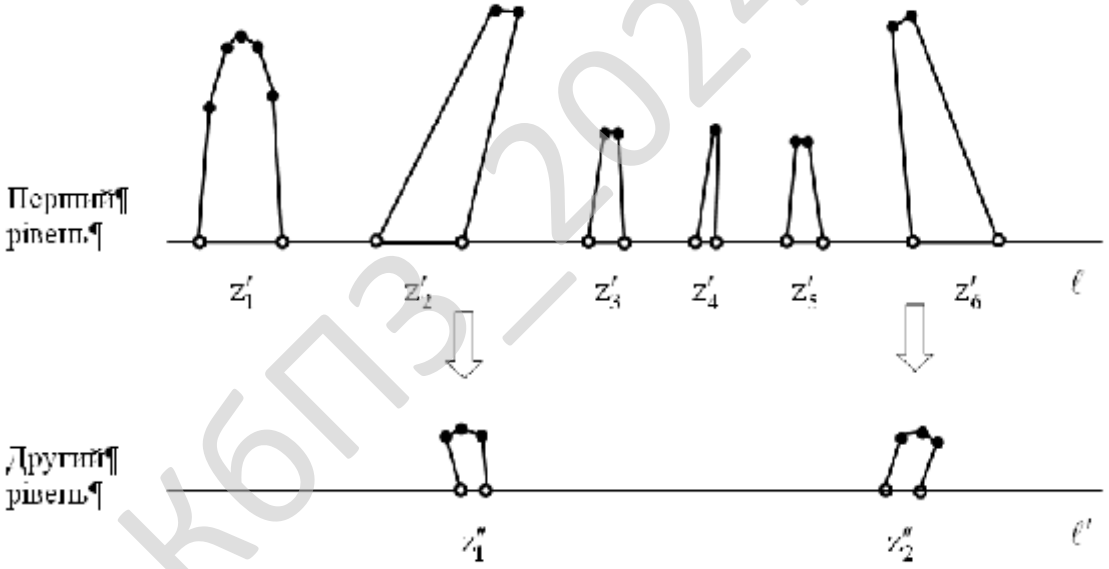

Рисунок 3.18 – Ієрархічна схема виділення підструктур

Структурна схема розробленої системи зображена на рисунку 3.19. На ній показано структуру вирішення поставленого завдання пошуку зображень у мережі Інтернет за допомогою методу структурно-лінгвістичного розпізнавання зображень.

Рисунок 3.19 – Структурна схема системи

3.3 Розробка функціональної схеми

Розробимо спосіб збору релевантної до пошукового запиту графічної інформації в розподілених та телекомунікаційних мережах з застосуванням структурно-лінгвістичного підходу до розпізнавання об’єктів.

Здійснимо програмно-апаратну реалізацію метода структурно-лінгвістичного розпізнавання зображень. Розроблена програма доводить працездатність запропонованого методу. Апаратна частина представлена скінченими автоматами, які були синтезовані на основі сформованих породжуючих граматик для кожного з фрагментів СМК. Для скорочення часу роботи скінчених автоматів рекомендована їх паралельна реалізація.

Проведено експериментальні дослідження щодо оцінки ймовірностей правильної класифікації ($P_{пр.кл} \approx 0,84$) та ідентифікації ($P_{пр.ід} \approx 0,71$), а також ймовірностей помилок 1-го та 2-го роду, отриманих під час класифікації ($P^{кл}_{пом1,2} \approx 0,16$) та ідентифікації ($P^{ід}_{пом1,2} \approx 0,29$). Дані експериментів представлені на рисунках 3.20-3.23.

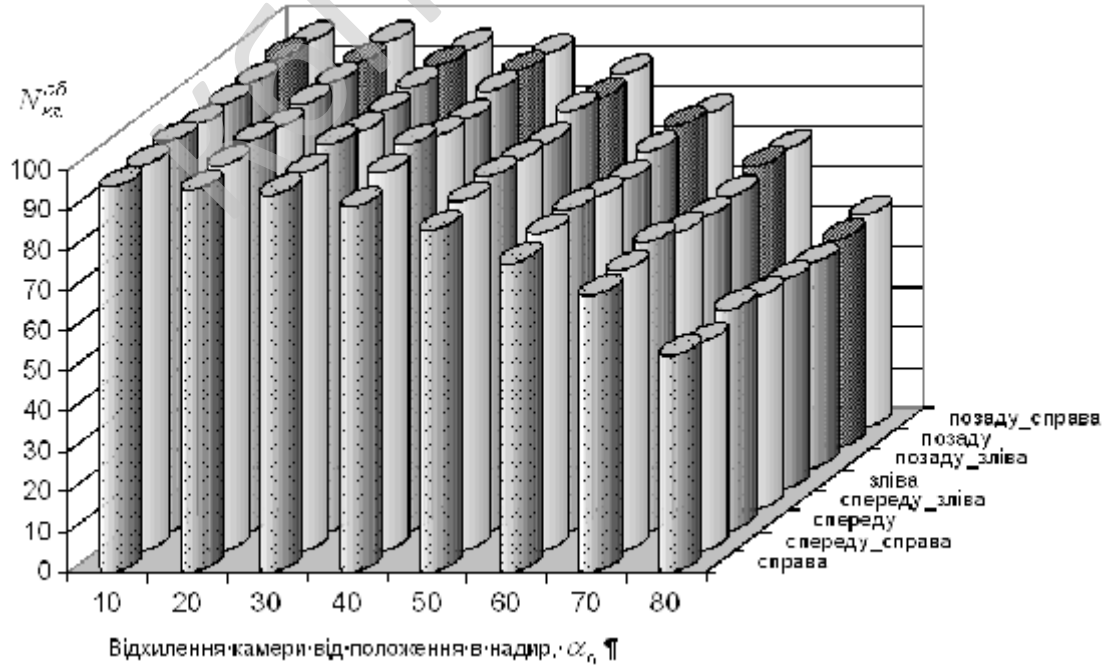

Рисунок 3.20 – Кількість $N^{об}_{кл}$ правильно класифікованих об’єктів

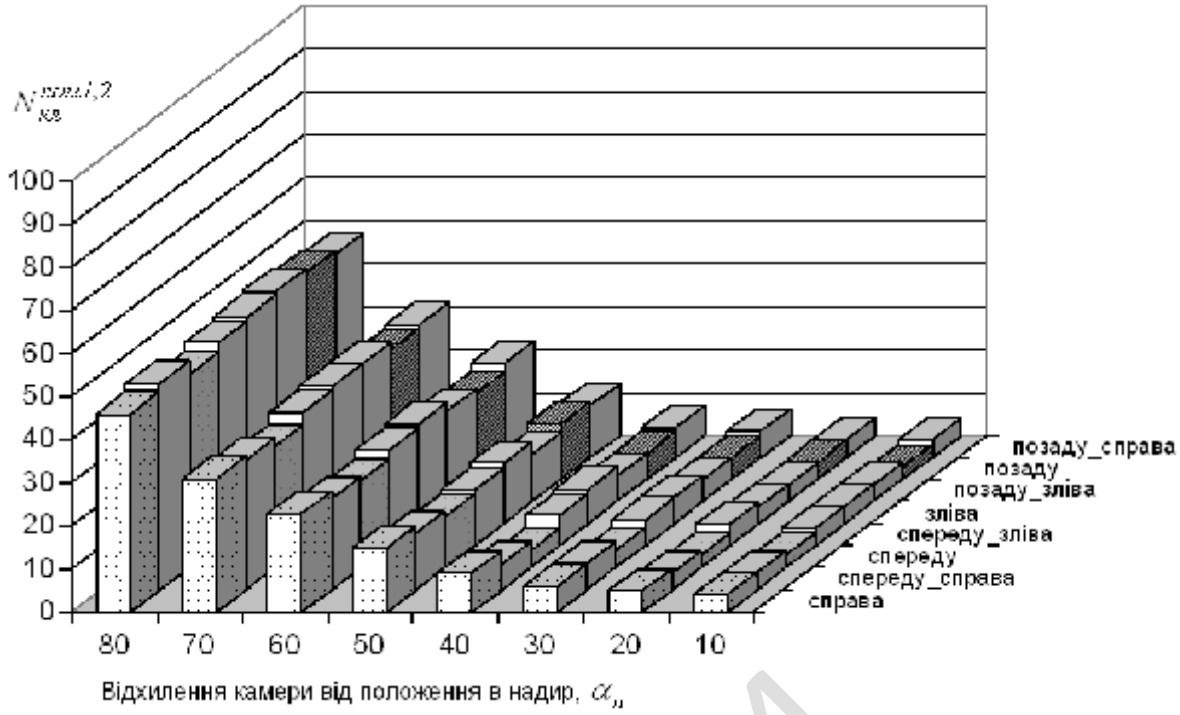

Рисунок 3.21 – Кількість $N_{кл}^{пом1,2}$ помилок 1-го та 2-го роду, отриманих при класифікації

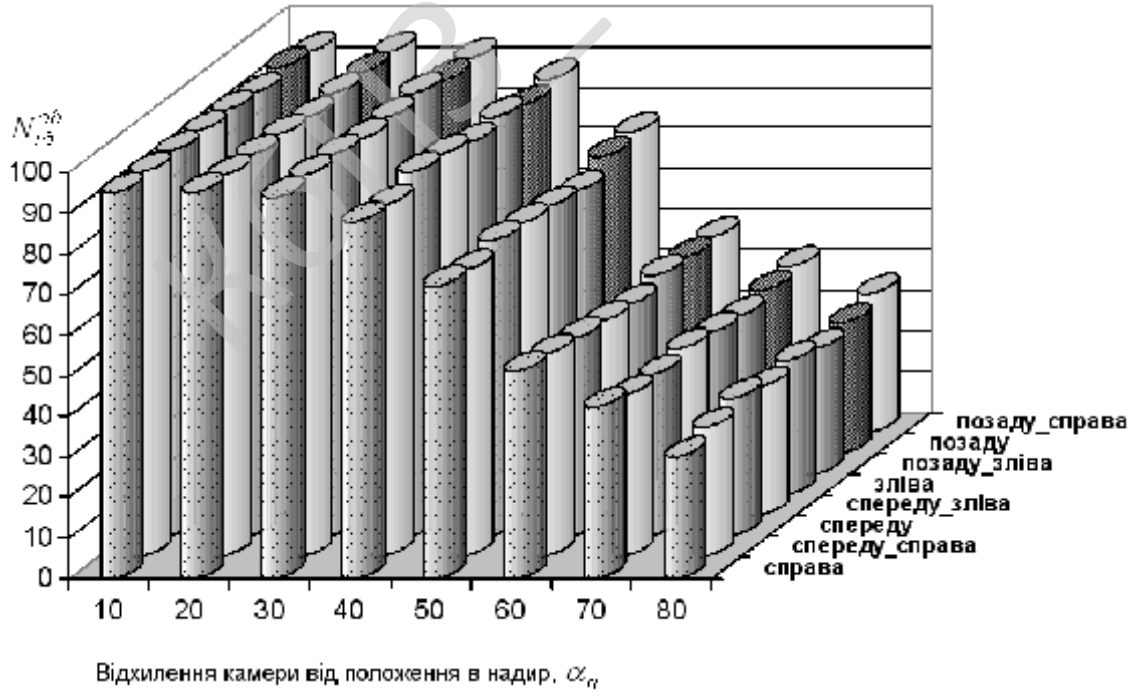

Рисунок 3.22 – Кількість $N_{ід}^{об}$ правильно ідентифікованих об’єктів

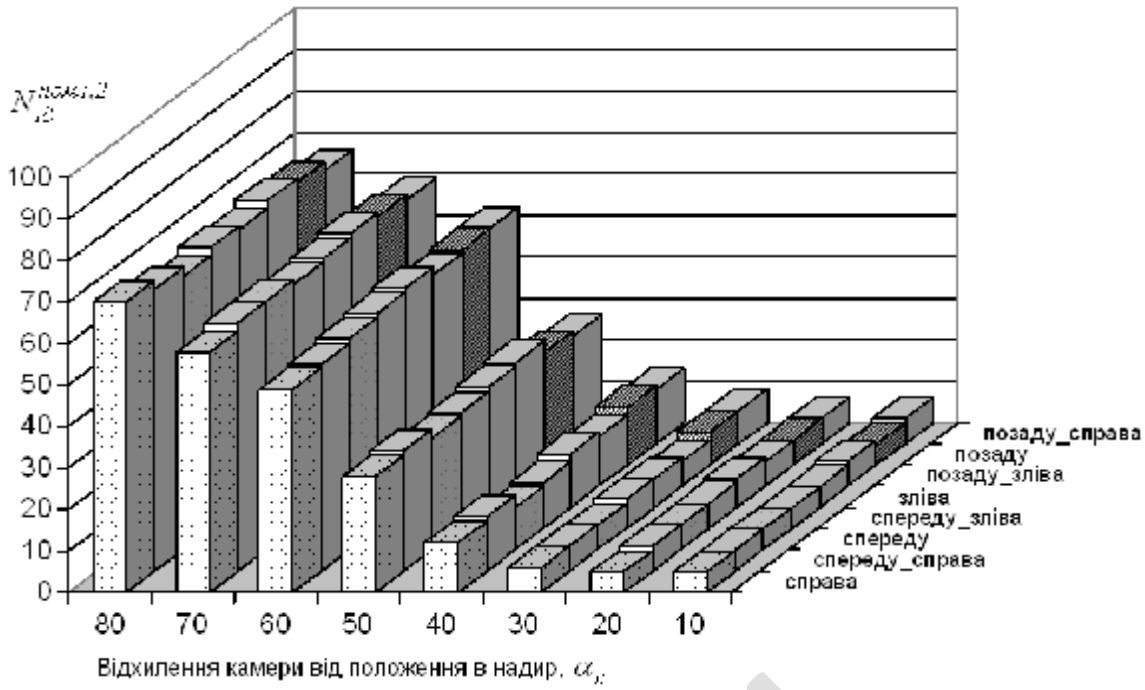

Рисунок 3.23 – Кількість $N_{ід}^{пом1,2}$ помилок 1-го та 2-го роду, отриманих при ідентифікації

Проведена оцінка ефективності використання алгоритмів семантичного перетворення та збору інформації в КСМ на основі застосування методу структурно-лінгвістичного розпізнавання зображень. Показано, що застосування розробленого методу дозволить у 1,5 рази скоротити час збору релевантних зображень в розподілених пошукових системах з децентралізованою архітектурою.

Функціональна схема розробленої системи зображена на рисунку 3.24.

Функціональна схема системи включає в себе наступні функціональні блоки:

Головне вікно програми.

Блок розпізнавання образів за допомогою методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Рисунок 3.24 – Функціональна схема системи

Розглянемо ці блоки більш детально.

Блок розпізнавання образів за допомогою методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет включає у себе наступні блоки.

Блок семантичної мережі концептів:

– Семантичні визначники класу.

– Семантичні визначники підкласу.

– Семантичні визначники напрямку зйомки.

Більш детально робота цього блоку описана при роз’ясненні рисунку 3.12.

Блок детектування:

– Детектори кутів.

– Детектор напрямку.

– Детектор правильної класифікації підкласу класу об’єкту, який шукається у мережі Інтернет. Більш детально робота цього блоку описана при роз’ясненні рисунку 3.13.

3.4 Розробка діаграми процесів

Діаграма процесів розробленої системи зображена на рисунку 3.25. При детальному її розгляді можна побачити як саме проходе взаємодія у розробленій системі. Використовується модель проектування, графічне представлення «потоків» даних в інформаційній системі.

Діаграма взаємодії процесів використовується для візуалізації процесів обробки даних (структурне проектування). Для розробника вважається звичним спочатку креслити діаграму взаємодії процесів даних рівня контексту, завдяки чому буде показано взаємодію системи. Ця діаграма в подальшому підлягає уточненню шляхом деталізації процесів та потоків даних з метою показати систему що розробляється.

Діаграми потоків даних містять чотири типи елементів:

– Процеси які являють собою трансформацію даних в рамках описуваної системи.

– Сховища даних (репозиторії).

– Зовнішні по відношенню до системи сутності.

– Потоки даних між елементами трьох попередніх типів.

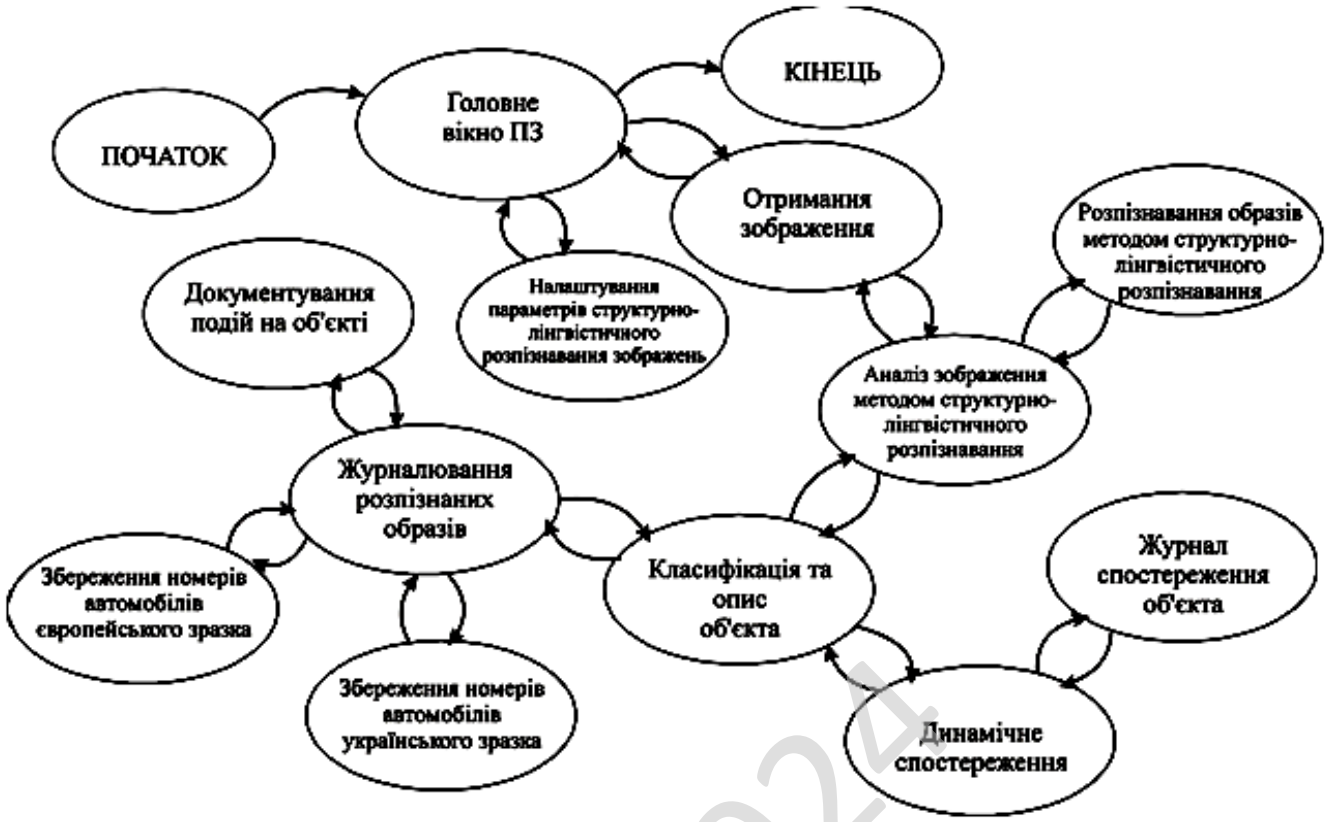

Рисунок 3.25 – Діаграма взаємодії процесів

Таким чином, розглянувши опис системи, структурну, функціональну схеми системи, та діаграму взаємодії процесів перейдемо до опису блок-схем основної програми, та підпрограм, які використовуються, для реалізації системи.

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

Первинною стадією без якої не відбувається розробка програмного забезпечення це звичайно розробка блок-схем. На рисунку 4.1 зображена основна блок-схема програми, на рисунку 4.2 та 4.3 зображено роботу підпрограм.

З яких видно що робота основної програми складається з початкових етапів ініціалізації ПЗ, перевірки наявності ресурсів системи, блоку початку основного циклу з чеканням запиту від користувача в якому відбувається виклик підпрограм та останньої стадії – перевірка поточного стану з завершенням роботи розробленого ПЗ. При роботі підпрограм виконується основний функціонал системи з циклічними послідовностями, перевіркою поточного стану та поверненням в основну програму прапорів стану виконання.

Опис алгоритмів функціонування системи

Блок-схеми є першоджерелами стратегії розвитку ПЗ. Тому від точності і детальної блок-схеми залежить результат всієї програми.

При виборі початкової точки відліку при побудові схем було враховано, що виходячи з вибору мови програмування і інших технічних засобів, програма буде об’єктно-орієнтована що вимагає оптимізації програми високого рівня, також те, що при розробці програми слід надати особливу увагу модулю структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

При складанні блок-схем програмного забезпечення і напрацювання алгоритмів я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою. Для чого були створені додаткові класи, типи даних і константи, що забезпечило вирішення проблем.

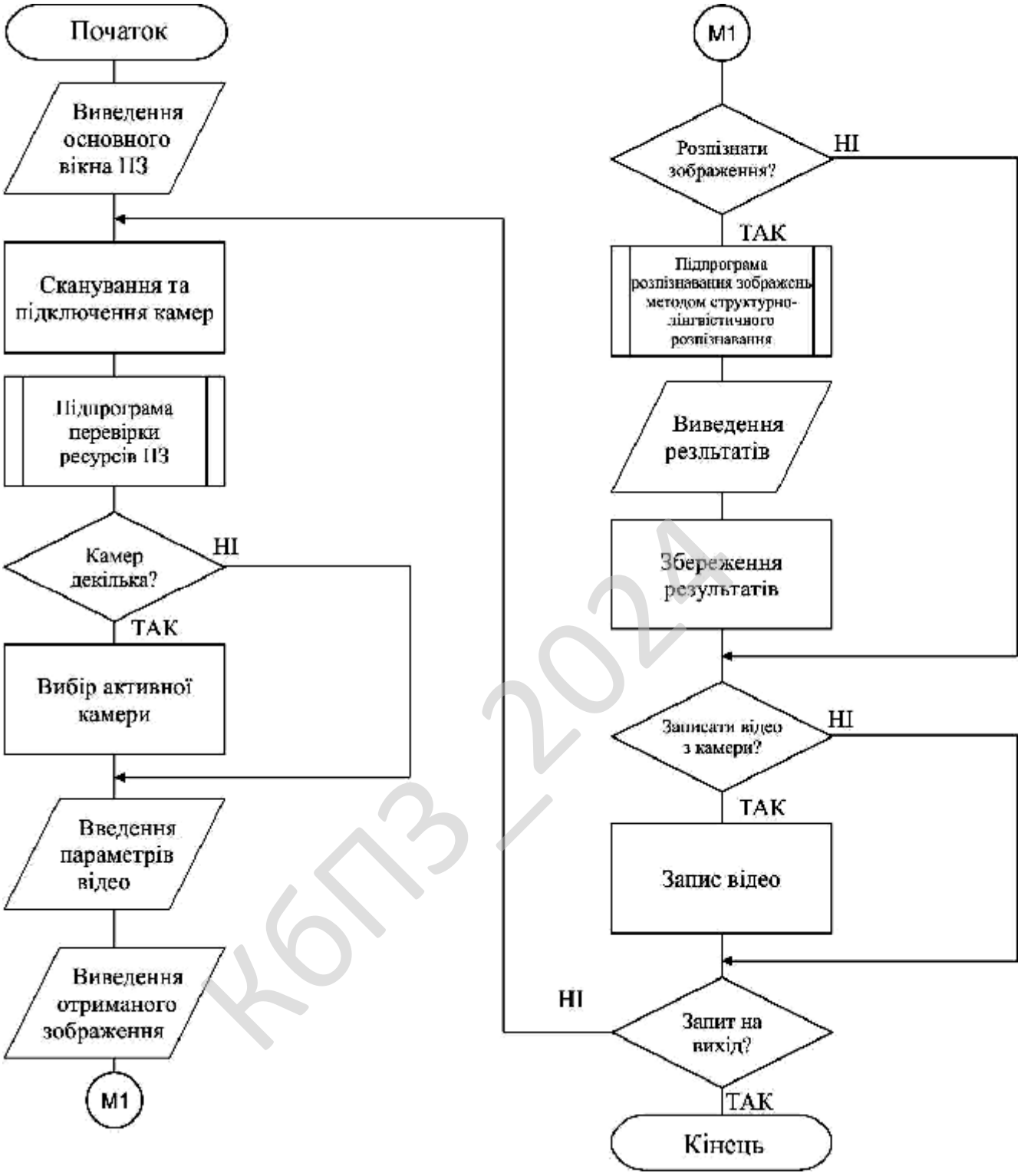

Рисунок 4.1 – Блок-схема основної програми

Було використано підходи з використанням UML, це уніфікована мова моделювання, використовується у парадигмі об'єктно-орієнтованого

програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, називаної UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. UML не є мовою програмування, але в засобах виконання UML-моделей як інтерпретованого коду можлива кодогенерація.

UML може бути застосовано на всіх етапах життєвого циклу аналізу бізнес-систем і розробки прикладних програм. Різні види діаграм які підтримуються UML, і найбагатший набір можливостей представлення певних аспектів системи робить UML універсальним засобом опису як програмних, так і ділових систем.

Діаграми дають можливість представити систему (як ділову, так і програмну) у такому вигляді, щоб її можна було легко перевести в програмний код. Основною причиною використання мови UML є спілкування розробників між собою.

Крім того, UML спеціально створювалася для оптимізації процесу розробки програмних систем, що дозволяє збільшити ефективність їх реалізації у кілька разів і помітно поліпшити якість кінцевого продукту.

UML прекрасно зарекомендувала себе в багатьох успішних програмних проектах. Засоби автоматичної генерації кодів дозволяють перетворювати моделі мовою UML у вихідний код об’єктно-орієнтованих мов програмування, що ще більш прискорює процес розробки. Практично усі CASE-засоби (програми автоматизації процесу аналізу і проектування) мають підтримку UML. Моделі розроблені в UML, дозволяють значно спростити процес кодування і направити зусилля програмістів безпосередньо на реалізацію системи.

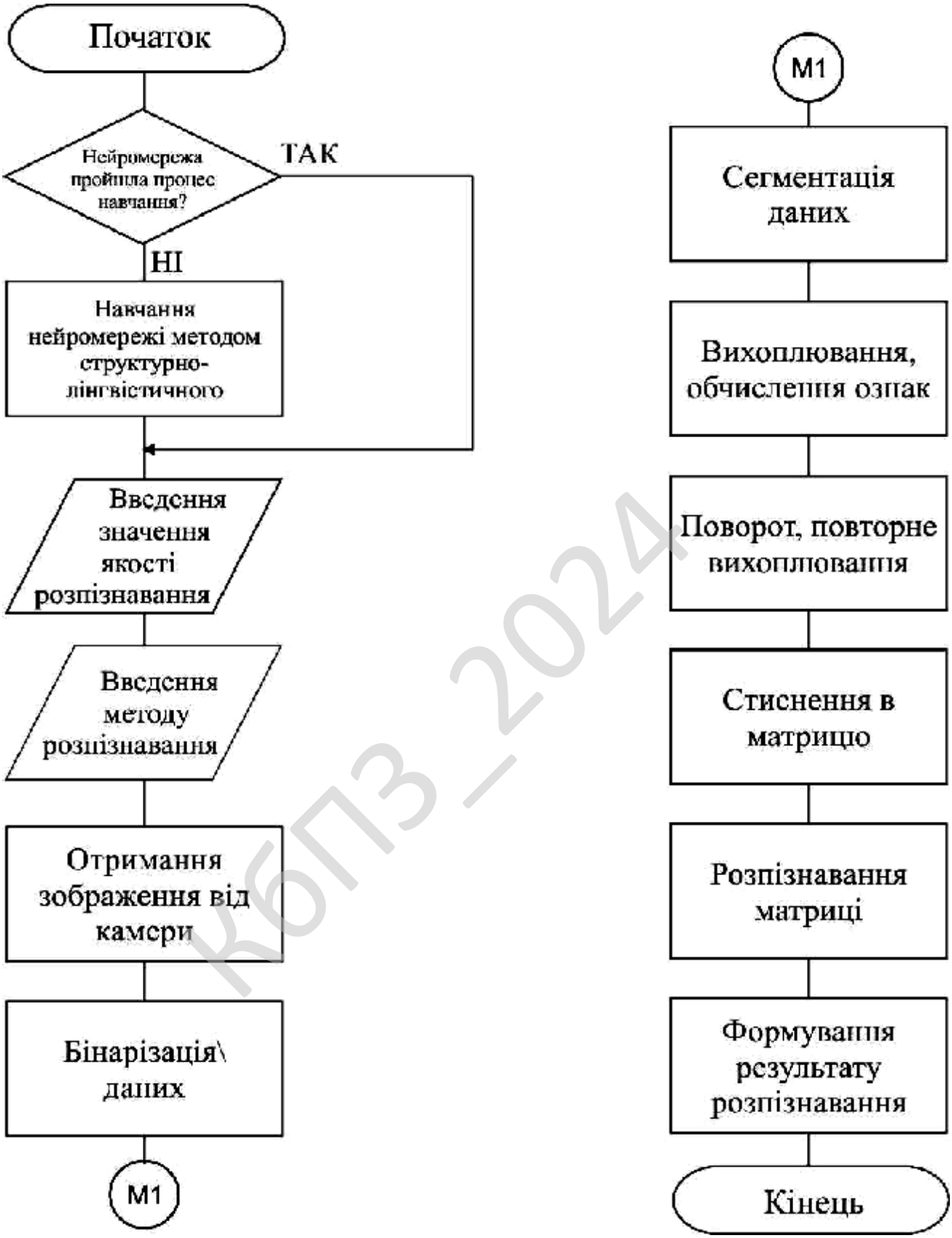

Рисунок 4.2 – Блок-схема роботи підпрограми

Діаграми підвищують супроводжуваність проекту і полегшують розробку документації.

UML необхідний:

– Керівникам проектів, які керують розподілом завдань і контролем за проектом.

– Проектувальникам інформаційних систем які розробляють технічні завдання для програмістів.

– Бізнес-аналітикам, які досліджують реальну систему і здійснюють інжиніринг і реінжиніринг бізнесу компанії.

– Програмістам які реалізовують модулі інформаційної системи.

При модифікації системи об'єктний підхід дозволяє легко включати в систему нові об'єкти і виключати застарілі без істотної зміни її життєздатності. Використання побудованої моделі при модифікаціях системи дає можливість усунути небажані наслідки змін, оскільки вони не ламають структури системи, а тільки змінюють поведінку об'єктів.

Теорія технічного зору існує не перший день, по цьому в літературі можна знайти досить підходів і рішень. Для початку перелічу деякі з них.

Опис алгоритму скелетизації

Коротенько, це якийсь метод розпізнавання одинарних бінарних образів, заснований на побудові кістяків цих образів і виділення з кістяків ребер і вузлів. Далі по співвідношенню ребер, їхньому числу й числу вузлів будується таблиця відповідності образам. Так, наприклад, кістяком кола буде один вузол, кістяком букви П – три ребра й два вузли, причому ребра відносяться як 2:2:1. У програмуванні даний метод має кілька можливих реалізацій.

Опис нейромережних структур

Солідне число нейронів вимагає солідні обчислювальні потужності, які звичайно відсутні на мобільних платформах. Однак треба мати, що нейромережі іноді дають досить цікаві результати, за рахунок своєї нелінійної структури, більше того деякі нейромережі здатні розпізнавати образи інваріантні щодо

повороту без якої або зовнішньої передобробки. Так наприклад мережі на основі неокогнейтронів здатні виділяти деякі характерні риси образів, і розпізнавати їх як би образи не були повернені.

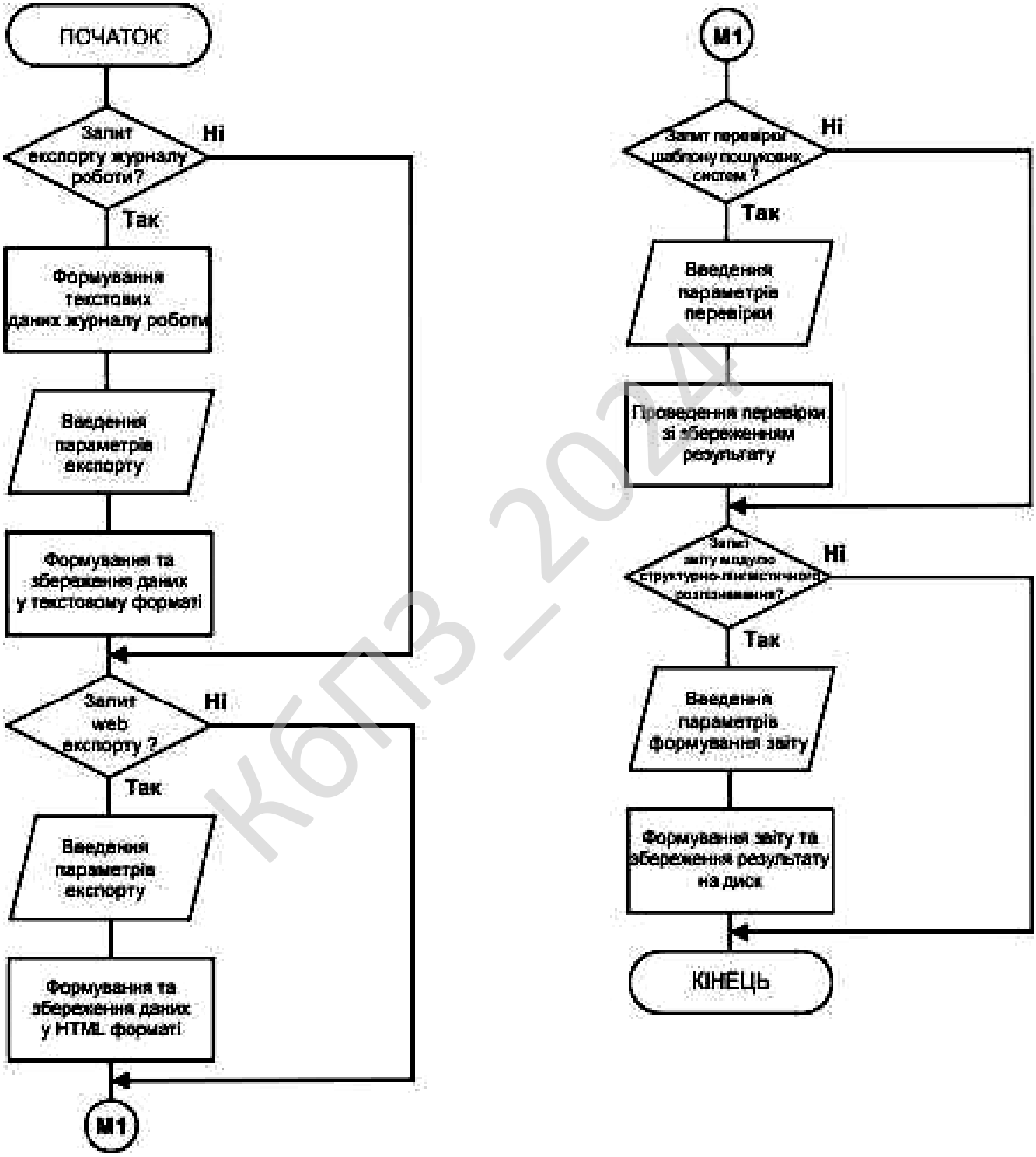

Рисунок 4.3 – Блок-схема роботи підпрограми

Опис інваріантних чисел

З геометрії образів можна виділити деякі числа, інваріантні щодо розміру й повороту образів, далі можна скласти таблицю відповідності цих чисел конкретному образу(майже як в алгоритмі скелетезації). Приклади інваріантних числі – число еллера, екцентриситет, орієнтація (у змісті розташування головної осі інерції щодо чого-небудь теж інваріантного).

Поточечне процентне порівняння з еталоном

Тут повинна бути деяка передобробка, для одержання інваріантності щодо розміру й положення, потім здійснюється порівняння із заготовленою базою еталонів зображень – якщо збіг більше чим якась оцінка, то вважаємо образ розпізнаним.

Практична частина розпізнавання образів

Проаналізувавши всі алгоритми розпізнавання, описані вище, мені здалося, що:

– Вирішувати інваріантність щодо повороту нейромережами (неокогнетронами й т.п) занадто довго на звичайній машині типу PC.

– Скелетезація по описі дуже проста, але як мені здається, базовий алгоритм неоднозначно відтскелетизує приклад із середньою буквою Щ.

– Створювати таблицю інваріантних чисел для образів, а потім дивитися відхилення – варіант непоганий. Однак незрозуміло скільки образів зможе розпізнавати такий алгоритм, наскільки образи можуть бути схожі один на одного, і в теж час вважатися різними. І як перешкоди від бруду й стороннього світла будуть заважати точному обчисленню інваріантних чисел.

Моє рішення відрізняється деякими недоліками (якщо шматок зображення накриє відблиск, то зрушується й центр мас і орієнтація – алгоритм не спрацює, так само не припустимі розриви в об'єкті), але володіє і яке якими плюсами (дрібні помилки повороту, неточності бінаризації й т.п. будуть природно згладжуватися на стадії стискання в матрицю-іконку).

Алгоритм представляє із себе послідовність наступних кроків:

1. Одержання зображення від камери.

2. Бінарізація.

3. Сегментація

4. Вихоплювання, обчислення ознак.

5. Поворот, повторне вихоплювання.

6. Стиск у матрицю.

7. Розпізнавання матриці нейромережею.

Далі буде описаний кожний пункт алгоритму.

Опис та реалізація одержання зображення з камери в ОС Windows

Отже для початку нам потрібна картинка, з якої потім можна працювати. В ОС Windows всі джерела відео мають один шаблон. Сам я навчився працювати з відео за допомогою безкоштовних вихідних кодів DScap.

Є пристрій, що може давати відеопотік (послідовність Bitmap) далі його потрібно настроїти, для цього запросити драйвер про всі можливі діалоги, які він (драйвер пристрою) може дати. Далі руками в додатку налускати потрібні настроювання (яскравість, розрішення дають клацати всі пристрої, у деяких є число каналів, усякі фільтри й т.п). Потім установити число кадрів у секунду – як правило теж задається. І властиво по події захоплення кадру (драйвером якщо завгодно) буде викликатися функція користувача, на вхід іде той, хто її викликав, і ще параметр, властиво захоплений Bitmap у зазначеній палітрі й із зазначеною яскравістю й т.п.

Ще варто згадати про якийсь альтернативний метод спілкування з камерою: як я зрозумів, захоплення так само можливе через API, тобто потрібно слати щось у камеру, а вона щось буде відповідати. Але це, як мені здалося, багато складніше, ніж готовий Direct.

Опис та реалізація проведеної бінарізації

У даній програмі розпізнаються тільки бінарні образи, тому другим етапом після одержання картинки, вона бінарізуется. При роботі з кольоровою камерою

перетворення з кольору в ЧБ іде по стандартній формулі: Y:=0.3*R+0.59*G+0.11*B.

Далі алгоритм досить простий: є деяка планка, якщо колір відтінку сірого вище – він вважається білим, якщо нижче – вважається чорним. Як видно бінарізація дуже проста, однак для серйозного поліпшення якості роботи розпізнавання, і зменшення часу роботи наступних модулів, на цьому місці краще ввести якийсь фільтр, пускай навіть самий простенький. Зі своєї практики можу сказати, що для роботи з відео одним з найпростіших фільтрів є фільтр по контрастності. У своїй програмі я не використовував таку конструкцію, однак місце де вона може бути включена позначене, і на тім місці перебуває більше проста підпрограма, що відслідковує кількість пікселів (білих або чорних) які йдуть підряд, і виключаючих виникнення послідовності в ряді : 01010101 .

У режимі дебага місця зміни кольору, зафіксовані програмою можна побачити по червоному обрамленню образів, що рисується безпосередньо в модулі бінарізації.

Дуже важливо знати, що стандартні модулі DELPHI опитування кольору пікселів в bitmap – річ жахливо повільна. При її використанні ні про який реальний час говорити не доводиться. Для прискорення процесів я використовував зовнішні безкоштовні модулі Qpixels. Після однократного опитування квітів і їх бінарізації, програма працює тільки зі звичайною бінарною матрицею (динамічним двовимірним масивом), тому далі йде вже все швидко.

Опис та реалізація сегментації

Всі описані вище алгоритми розпізнавання образів працюють із єдиним видимим образом, у реальному житті відеокамера(спрямована на підлогу) може бачити відразу кілька об'єктів, спеціально розташованих поруч, або ж у поле зору може попастися який-нитка сторонній об'єкт (нога людини, бруд, потертісь пола та інші прихожі речі). Якщо не передбачати деяку розбивку загального зображення на частині, то жоден з описаних вище алгоритмів не зможе коректно

працювати. Отже розбивка зображення на частині, кожна з яких містить свій унікальних об'єкт називається сегментацією.

Як і в самому розпізнаванні – у сегментації, за час існування науки, було придумано вже досить алгоритмів, кожний з яких має свої достоїнства, і застосовується під конкретну задачу. Для початку помічу ще раз, що в моїй задачі йде робота тільки із чорно-білим зображенням. Під колір існують зовсім інші методи сегментації, ніж будуть описані нижче.

Так само варто помітити, що в сегментації чітко розділяються чорно-білі зображення на бінарні й з відтінками сірого. Тут теж працюють зовсім різні по швидкості й складності алгоритми, однак інтуїтивно зрозуміло, що будь-яке зображення з відтінками сірого можна бінарізовати за деякими правилами.

У моїй задачі з камери надходить пускай і чорно-біла, але цілком реальна картинка з 8бітною палітрою (колір задається від 0 до 255), однак для простоти у своєму алгоритмі я відразу ж бінарізую її.

Як уже було сказано, більшість камер, граберів і інших пристроїв в ОС Windows кодує кольору пікселів в 24 бітовому форматі. Оскільки я працюю в основному зі ЧБ камерами, то в них уміст всіх трьох 8 бітів RGB однакове. Ми маємо деяке зображення, кожний квітів якого змінюється від 0 до 255.

На стадії бінарізації ми повинні перетворити об'єкт зображення в бінарну матрицю даних. Отже, із цього моменту для прискорення процесу починається робота вже не з об'єктом картинки, а з бінарною матрицею.

Наступним пунктом алгоритму повинна бути сегментація. Вона здійснюється Шляхом проходу по матриці зображення ліворуч праворуч, зверху долілиць. Перше що приходить, на розум, щоб раціоналізувати цю частину алгоритму, це завести окремо масив міток, у якому треба прописувати яка мітка на яку посилається, і всі операції при проході по матриці зображення проводити тільки з масивом посилань міток. При цьому наприкінці проходу повинен бути масив міток, що посилаються один на одного, і тільки кілька міток повинні бути унікальними – які властиво й утворюють об'єкти.

Що б одержати унікальні об'єкти, які вже можна направити в модуль розпізнавання потрібно всього лише нормалізувати цей масив посилань і перепризначити всі елементи матриці на унікальні мітки виходячи з даних нормалізованого масиву посилань.

На практиці такий підхід виявився неробочим: на складних об'єктах (що-небудь типу горизонтальної плоскої змійки) відбувався поділ цільного об'єкта на частині, причому в процесі переприсвоєння міток губився зв'язок між частинами об'єкта.

Більше правильним, і не складним рішенням з'явилося рекурентне знаходження унікальних батьків кожної мітки, і далі всю роботу з переприсвоєння робити вже з ними, у такий спосіб зв'язок в об'єкті не може бути загублена, тому що рекурсія дає можливість вихоплювати самі довгі відростки об'єкта.

Термін відростки явно не буде зрозумілий читачеві, що знайомиться з матеріалом перший раз, тому пояснюю: на картинці вище (там де буква К и 1ка) стінки в 1ки ідеально рівні, на практиці ж, розрішення 640х480 у камери, спрямованої на підлогу, дає досить пристойну деталізацію, що може вихоплювати реальні або гадана камера "заусеніци" або перешкоди на рівних місцях образа (пускай навіть роздрукованого на принтері).

У результаті алгоритм працює не з ідеальними картинками, а досить складними, але це відбувається тільки на стадії сегментації, потім ці зайві деталі на більших об'єктах підуть, як буде зрозуміло пізніше.

Так само на цьому етапі даю можливість ознайомиться з виходом програми, у реальних умовах експлуатації. Нижче виведений масив, про який уже йде розмова. У роздруківці масиву третій стовпчик це площа шматків-сегментів. Її досить просто зібрати на цьому етапі роботи програми, потім дані про площу можна використовувати в найпростішому фільтрі, і не пускати дрібне сміття в модуль розпізнавання.

Приклад цей – це реальний вихід програми в текстовий файл, оскільки деякі мітки при проведенні сегментації стали двозначними, то картинка небагато

поповзла, що утрудняє її сприйняття, однак при деякій фантазії й бажанні можна зрозуміти що відбувається.

Опис та реалізація вихоплювання образа та обчислення деяких інваріантних чисел

Після успішного завершення сегментації, кожний сегмент попадає в модуль розпізнавання (як параметр розпізнавання в моїй програмі саме і йде унікальна мітка сегмента).

Треба помітити що ще в модулі сегментації, при фінальному проході по масиві, з переприсвоєнням міток, для кожного образа позначаються його границі й обчислюється фінальна площа. Границі потрібні для прискорення роботи модуля розпізнавання – так як він шукає образ тільки в зазначеному місці.

Для того, що б образи розпізнавалися інваріантно щодо положення й повороту треба прив'язатися до їхньої структури(формі в людському розумінні).

Отже в кожного бінарного образа можна обчислити кілька ознак, що не залежать від його повороту або розміру, наприклад число еллера, екцентрисетет і орієнтація.

Як уже було сказано десь на початку цієї доки, можна зібрати таблицю штук 8мі інваріантних ознак і розпізнавати образи виходячи тільки із цих даних, однак ми підемо другим шляхом.

Із усього множини інваріантних числі ми обчислимо тільки одну орієнтацію, для цього використовуємо алгоритми, схожі на MATLABовські (можливо вони там і використовуються).

Однак опис неточно – як видно при обчисленні З використовується якесь Uxy, що не було уведено раніше, у програмі є - але Uxy обчислюється коректно.

При обчисленні ряду морфометричних ознак використовуються поняття механіки твердого тіла. Зокрема, це ставиться до довжин осей інерції об'єкта. Напрямку в тілі, що збігаються з півосями еліпсоїда інерції, називають головними осями інерції. Для знаходження головних осей інерції, що лежать у площині об'єкта, у функції imfeature використовуються наступні співвідношення.

Нехай N – кількість пікселів, що відносяться до об'єкта. Вся множина пікселів p(x, y), що відносяться до об'єкта, позначимо Q. Тоді координати центра мас об'єкта обчислюються як:

$$x_c = \frac{1}{N} \sum_{p(x,y)\in\Omega} x,$$

$$y_c = \frac{1}{N} \sum_{p(x,y)\in\Omega} y.$$

Обчислимо кілька допоміжних величин:

$$U_x = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y)\in\Omega} (x - x_c)^2,$$

$$U_y = \frac{1}{12} + \frac{1}{N} \sum_{p(x,y)\in\Omega} (y - y_c)^2,$$

$$C = \sqrt{(U_x - U_y)^2 + 4 \cdot U_{xy}{}^2}.$$

Тоді довжини максимальної $A_{\max}$ й мінімальної $A_{\min}$ осей інерції обчислюються як:

$$A_{\max} = 2\sqrt{2} \cdot \sqrt{U_x + U_y + C},$$

$$A_{\min} = 2\sqrt{2} \cdot \sqrt{U_x + U_y - C}.$$

Довжини головних осей інерції використовуються для обчислення ексцентриситету й орієнтації об'єкта.

Ексцентриситет визначається за допомогою співвідношення:

$$E = \frac{2 \cdot \sqrt{(0.5 \cdot A_{\max})^2 - (0.5 \cdot A_{\min})^2}}{A_{\max}}.$$

Орієнтація визначається як кут у градусах між максимальною віссю інерції й віссю X. Якщо $U_y > U_x$, то орієнтація Про обчислюється за допомогою формули:

$$O = \frac{180}{\pi} \cdot arctg\left(\frac{U_y - U_x + C}{2 \cdot U_{xy}}\right),$$

у противному випадку Про обчислюється як

$$O = \frac{180}{\pi} \cdot arctg\left(\frac{2 \cdot U_{xy}}{U_x - U_y + C}\right).$$

Опис та реалізація повороту образа з повторним вихоплюванням

Отже, знайдена орієнтація зображення, що унікальна для кожного образа.

Зроблено це для того, що б тепер образ можна було повернути щодо центра мас, так що б його орієнтація була паралельна осі X. Властиво цей прийом і дає інваріантність щодо початкового повороту.

Ну й звичайно деяка демонстрація роботи алгоритму обчислення орієнтації й повороту бінарного образа.

Вище білі букви – це те, що бачить камера, без усяких фільтрів(тільки невелика гра яскравості й контрасту). Червоний хрестик позначає центр мас, обчислений по формулах вище. Зелені букви, повернені після обчислення орієнтації по формулах вище. У процесі написання програми на цьому етапі в мене ще не була налагоджена сегментація, тому задача розпізнавання небагато полегшена – на екрані присутній свідомо один образ, далі буде складніше

Опис та реалізація стиску образу у матрицю заданого розміру

Як уже помітно з картинок вище, в оболонці програми з'явилася якась бінарна матриця, у яку стискається кожний окремо сегментований образ. Розмір матриці задається по потребам – тобто якщо потрібна більша деталізація, то краще використовувати матриці більшого розміру, ніж у моїй поточній версії програми. Це дасть додаткові обчислення на стадії розпізнавання, але по ідеї й підвищить якість процесу. Однак важливо розуміти**,** що малому розрішенні розпізнаваної матриці дозволяє виправити деякі можливі помилки при повороті зображення(обчислення орієнтації образа), так як незначне відхилення в 5 градусів, не буде помітно після стиску образа.

На практиці такі помилки обов'язково будуть проскакувати, коли якусь частину образа закриє перешкода або відблиск – центр мас зміщається, орієнтація можливо теж – поворот буде не цілком коректний. Якщо розмова пішла про відблиски, то треба вже накручувати оптикові, і ставити можливі джерела проти висвітлення, але тут важливо розуміти, що в реальних умовах при розпізнаванні, образ ніколи не буде видний камерою як повинен бути на 100%. Тобто в русі хоча б одна точка образа буде гарантовано перевранa камерою, тому чим менше розпізнавана матриця (тобто чим грубіше стискання до її розмірів) тим більше можливих помилок буде незамічено, але тем менше рівень можливої деталізації.

Властиво, тут я намагався пояснити, що для кожної задачи розмір матриці треба вибирати відповідний. І не завжди чим більше тим краще

Я працював із двома розмірами: 32x32 і 16x16, останній розмір, для розпізнавання образів, приклади яких можна бачити на картинках мені сподобався більше.

Опис та реалізація розпізнавання образа нейромережею

Споконвічно для перевірки передобробки, і всіх описаних вище алгоритмів, як метод розпізнавання був втілений алгоритм процентного порівняння з еталонами. Тобто споконвічно програма фотографувала й передоброблювала тими ж алгоритмами якісь навчальні образи, після чого, коли було потрібно розпізнавати щось нове, вона це нове знову ж таки передоброблювала до матриці заданого розміру, і потім цю матрицю порівнювала з усім запам’ятовуваними матрицями. У загальному-те простий алгоритм, що справно працював, тому я його залишив у програмі, а що б використовувати треба тільки розкоментувати деякі шматки.

Структура обраної мною мережі досить стандартна. Багатошарова нейромережа зворотного поширення помилки, займається тим же, що й процентний алгоритм порівняння матриці, однак за рахунок своєї нелінійної структури розпізнає вона на 10-30% краще поточечного порівняння двох матриць.

Для людей що представляють скажу, що використовував безкоштовні

модулі NeuralBase. Модулі мені дуже сподобалися, досить прості у використанні, можна взагалі не знати що таке нейромережі, використовуючи їх, але це й не дуже цікаво.

Були проведені експерименти структурою нейроних мереж. Мною були створені дві сіточки – одна 256-6, друга 256-40-6. Перший шар в 256 нейронів це вхід матриці 16x16, останній шар це вхід – розпізнаємо наприклад 6 букв.

На практиці, нейромережа із трьох шарів (з якоюсь внутрішньою обробкою – в 40 нейронів) навчалася із роботою, постійно ловлячи локальні мінімуми (висновок зроблений з поводження середньоквадратичної помилки, виведеної в реальному часі при навчанні). Якість же розпізнавання візуально не зросла.

Тому я вирішив, що зайві обчислення безглузді й зупинився на простій структурі в 256-X (де X – число заучених образів), що навчається досить нетривалий час, поводиться стабільно й показує гарні результати.

За результат розпізнавання був узятий вихід нейрона, більший 1-K, причому жоден з інших виходів не може бути більше K. У моїй задачі K береться рівне 0.2, тобто можна провести деяку аналогію із процентним порівнянням (за істину береться більше 80% збігів), з тією лише різницею, що нейромережа "порівнює" нелінійно.

4.2 Захист розробленого програмного забезпечення

Для захисту розробленого програмного забезпечення запропоновано використовувати алгоритм ДСТУ 8845:2019 – алгоритм симетричного потокового перетворення. В основі ДСТУ 8845:2019 лежить класична схема підсумовуючого генератора подібна генератору SNOW 2.0, SNOW 3.0 та SNOW V. В основі ДСТУ 8845:2019 збережені всі базові операції шифрів сімейства SNOW, а раунд шифрування AES замінений на функцію нелінійної підстановки T, що реалізовує

перестановку елементів скінченного поля GF(2^{64}) за допомогою компонентів національного стандарту симетричного криптоперетворення ДСТУ 7624:2014.

ДСТУ 8845:2019 використовує 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K і забезпечує високий та надвисокий рівень стійкості із врахуванням можливого застосування квантового криптографічного аналізу. При розробці алгоритму ДСТУ 8845:2019 орієнтувалися на сучасні 64-бітні обчислювальні системи, тому розмір слова обрано рівним 8 байт. запису байтів застосовують подання від старшого до молодшого. Генератор ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру схематично приведено у стандарті.

Як випливає з генератора ключових потоків ДСТУ 8845:2019 у режимі генерації гами шифру основними компонентами генератору є регістр зсуву з лінійним зворотнім зв'язком та скінчений автомат на базі якого виконується нелінійне перетворення T. Вхідні дані (ключ шифрування K та вектор ініціалізації IV) використовуються для ініціалізації змінної стану $S_i (i \geq 0)$, яка складається із двох компонент до складу яких входять[12]:

– 16 змінних $s^{(i)}$ – комірок регістра зсуву з лінійним зворотнім зв'язком: $s^{(i)} = (s_{15}^{(i)}, s_{14}^{(i)}, s_{13}^{(i)}, s_{12}^{(i)}, s_{11}^{(i)}, s_{10}^{(i)}, s_9^{(i)}, s_8^{(i)}, s_7^{(i)}, s_6^{(i)}, s_5^{(i)}, s_4^{(i)}, s_3^{(i)}, s_2^{(i)}, s_1^{(i)}, s_0^{(i)})$;

– Два регістри скінченного автомату $r^{(i)}$: $r^{(i)} = (r_2^{(i)}, r_1^{(i)})$. На виході отримуємо ключовий потік (гамма), який формується з 8-байтних слів Z_i.

З рисунку слідує, що відводи регістра зсуву з лінійним оберненим зв'язком побудовані за примітивним над полем GF(2^{64}) поліномом: $f(x) = x^{16} + x^{13} + \alpha^{-1} x^{11} + \alpha$, де α є коренем примітивного над полем GF(2^8) поліному $gz(z) = z^8 + \beta^{170} z^7 + \beta^{166} z^6 + \beta^2 z^5 + \beta^{224} z^4 + \beta^{70} z^3 + \beta^2$. Поле GF(2^8) як і в ДСТУ 7624:2014 побудовано за примітивним на полем GF(2) поліномом $p(y) = x^8 + x^4 + x^3 + x^2 + 1$, а коефіцієнти $g(z)$ подаються через ступінь примітивного елементу β поля GF(2^8), тобто β корінь поліному $p(y)$. Тобто, у нас є вежа полів: $GF(2) \subset GF(2^8) \subset GF(2^{64}) \subset GF(2^{1024})$, де:

– поле $GF(2^{1024})$ задається відводами зворотного зв'язку як фактор кільце $GF(2^{64})[x]/(f(x))$;

– поле $GF(2^{16424})$ задається як фактор кільце $GF(2^8)[z]/(g(z))$;

– поле $GF(2^{1024})$ задається як фактор кільце $GF(2)[y]/(p(y))$.

З вищезазначеного, слідує що період вихідної послідовності становить 2^{1024}.

Структурно в алгоритмі симетричного потокового перетворення ДСТУ 8845:2019 виділяють три основні функції:

– функція ініціалізації, яка приймає в якості вхідних даних 256-бітний вектор ініціалізації IV та 256-бітний або 512-бітний секретний ключ K, і виробляє початкове значення змінної стану $S_0 = (s^{(0)}, r^{(0)})$;

– функція наступного стану Next, яка приймає на вхід зміну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє наступне значення змінної стану $S_{i+1} = (s^{(i+1)}, r^{(i+1)})$;

– функція ключового потоку Strm, що приймає на вході змінну стану $S_i = (s^{(i)}, r^{(i)})$ та виробляє на виході 64-бітний ключовий потік Z^i.

Також функція Next може виконуватися в двох режимах, в залежності від способу виконання ітерації, як частина реалізації ініціалізації алгоритму ДСТУ 8845:2019 або як частина функції ключового потоку Strm.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено інтерфейс програмного забезпечення, розробленого у результаті виконання бакалаврської роботи.

Розроблене програмне забезпечення структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет складається з наступних функціональних блоків:

– Вікна виведення образу для розпізнавання.

– Функції представлені у графічному вигляді (іконки).

– Функціональний блок параметрів розпізнавання образів.

– Вікно виведення результату роботи системи.

– Навігаційного меню яке визивається натисканням правої клавіші маніпулятора миші.

Рисунок 5.1 – Головне вікно розробленого ПЗ

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку авторського права, після чого на екрані з’явиться вікно показане на рисунку 5.2.

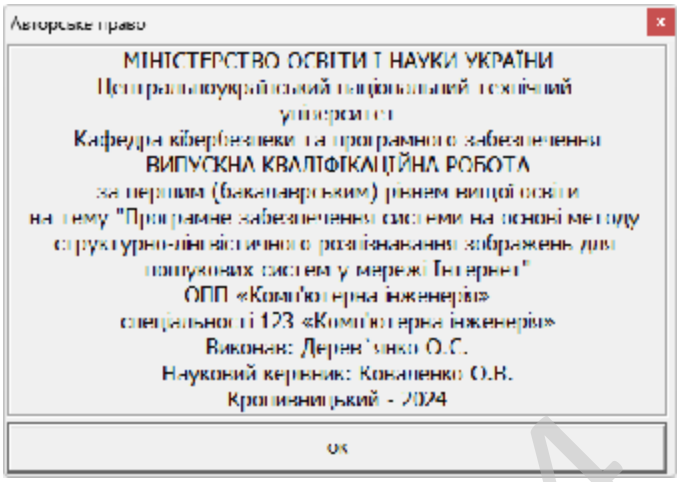

Рисунок 5.2 – Вікно розробника ПЗ

Під час роботи над програмою було проведено тестування програмного забезпечення, тобто технічне дослідження, призначене для виявлення інформації про якість продукту відносно контексту, в якому воно має використовуватись.

Тестування включає як процес пошуку помилок або інших дефектів, так і випробування програмних складових з метою їх оцінки.

Проводилась оцінка:

– відповідності поставленим вимогам;

– правильна відповідь для усіх можливих вхідних даних;

– виконання функцій за прийнятний час;

– практичність;

– сумісність з ОС та стороннім ПЗ.

Оскільки число можливих тестів для програмних компонент практично нескінченне, тому стратегія тестування полягала в тому, щоб провести всі можливі тести з урахуванням наявного часу та ресурсів.

Як результат ПЗ тестувалось стандартним виконанням програми з метою виявлення помилок або інших дефектів.

Проводилось тестування форматом чорної скриньки

Основне місце програми тестів «чорної скриньки» – інтерфейс ПЗ. Відомі: функції програми. Досліджується: робота кожної функції на всій області визначення.

Ці тести демонструють:

– Як виконуються функції програми.

– Як приймаються вихідні дані.

– Як виробляються результати.

– Як зберігається цілісність зовнішньої інформації.

При тестуванні «чорної скриньки» розглядаються системні характеристики програм, ігнорується їхня внутрішня логічна структура. Вичерпне тестування, як правило, неможливе.

Наприклад, якщо в програмі 10 вхідних величин і кожна приймає по 10 значень, то кількість тестових варіантів становитиме 10^10. Тестування «чорної скриньки» не реагує на багато особливостей програмних помилок.

Тестування «чорної скриньки» (функціональне тестування) дозволяє отримати комбінації вхідних даних, які забезпечують повну перевірку всіх функціональних вимог до програми.

Програмний виріб тут розглядається як «чорна скринька», чию поведінку можна визначити тільки дослідженням його входів та відповідних виходів. При такому підході бажано мати:

– Набір, утворений такими вхідними даними, які призводять до аномалій у поведінці програми (назвемо його ITc).

– Набір, утворений такими вхідними даними, які демонструють дефекти програми (назвемо його OT).

Будь-який спосіб тестування «чорної скриньки» повинен:

– Виявити такі вхідні дані, які з високою ймовірністю належать набору ITc.

– Сформулювати такі очікувані результати, які з високою імовірністю є елементами набору ОТ.

Принцип «чорної скриньки» не альтернативний принципу «білої скриньки». Скоріше це доповнює підхід, який виявляє інший клас помилок.

Тестування «чорної скриньки» забезпечує пошук наступних категорій помилок:

– Некоректних чи відсутніх функцій.

– Помилок інтерфейсу.

– Помилок у зовнішніх структурах даних або в доступі до зовнішньої бази даних.

– Помилок характеристик (необхідна ємність пам'яті і т.д.).

– Помилок ініціалізації та завершення.

Обрано умови розповсюдження – proprietary software.

Програмне забезпечення, на яке зберігаються як немайнові, так і майнові авторські права. Отримавши або придбавши таке програмне забезпечення, користувач отримує обмежені права користування ним: може бути заборонено або закрито доступ до коду (вивчення), внесення змін, тиражування, розповсюдження та перепродаж. Програмне забезпечення вважається власницьким, якщо наявне хоча б одне з перелічених обмежень.

Найчастіше основним методом захисту майнових прав на власницьке ПЗ, поза ліцензійною угодою, власник обирає закриття сирцевого коду, захищаючи свій продукт від модифікації і вбудовуючи системи обмеження користування через авторизацію.

Таке програмне забезпечення називається закритим. Проте, код власницького продукту може бути і відкритим, але власник може обмежити права користувача умовами користувацької ліцензії.

Власницьке програмне забезпечення та комерційне програмне забезпечення не є синонімами – власницьким може бути і безплатне (тобто, некомерційне) програмне забезпечення.

На противагу власницькому ПЗ існує вільне програмне забезпечення, автори і власники якого дозволяють вивчати, модифікувати і поширювати свій продукт.

Саме визначення власницького програмного забезпечення виникло в результаті діяльності громадського руху вільного програмного забезпечення (представленого Фондом вільного програмного забезпечення та іншими організаціями) і осмислення умов свободи користування програмами. Визначенням власницького програмного забезпечення є не невідповідність хоча б одній з базових умов вільного програмного забезпечення.

Сама назва власницьке ПЗ підкреслює визначальне значення власника у способі використання і можливостях розвитку цього програмного забезпечення.

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

В межах України в недостатній мірі представлені вітчизняні розробки в цій області.

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

– Досліджена система на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

– На основі отриманих результатів досліджень створена програмна реалізація системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

Розроблене програмне забезпечення має простий, дружній та зручний інтерфейс користувача, що забезпечує легкість у освоєнні роботи програмного продукту, зручність у використанні, і не потребує особливих спеціальних знань.

При створені програмного забезпечення було використано об’єктно-орієнтований підхід, що відповідає сучасним тенденціям у галузі розробки комерційних програмних систем.

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет. Це дозволило мінімізувати строк розробки програмного забезпечення, і, як слід, зменшити витрати на його розробку. Запропоноване програмне забезпечення ділиться на загальне програмне забезпечення, що поставляється із засобами обчислювальної техніки й спеціальне програмне забезпечення, що спеціально розроблене для даної конкретної системи й включає програми, що реалізують її функції.

Програма призначена для виконання під управлінням багатозадачної операційної системи Windows 10/11.

Даються необхідні рекомендації з установки розробленого програмного забезпечення.

Для підвищення рівня безпеки запропоновано застосовувати алгоритм ДСТУ 8845:2019.

В цілому створене програмне забезпечення підтверджує правильність використаних проектних рішень та повністю відповідає вимогам технічного завдання. Створене програмне забезпечення має потенційну можливість для подальшого вдосконалення і застосування у різних галузях.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, Clifford Stein. Introduction to Algorithms. The MIT Press. 2022 1677 p.

2. Will Grant. 101 UX Principles. Packt Publishing. 2022. 432 p.

3. Nathan Metzler. Kotlin Programming for Beginners. Independently published. 2021. 158 p.

4. Henry Lloyd. Interactive Computer Graphics. States Academic Press. 2022. 247 p.

5. Ranjan Parekh. Fundamentals of Image, Audio, and Video Processing Using MATLAB®With Applications to Pattern Recognition. CRC Press. 2021. 406 p.

6. Alasdair McAndrew. A Computational Introduction to Digital Image Processing. Chapman & Hall. 2021. 560 p.

7. PeterShirley, SteveMarschner. Fundamentals of Computer Graphics. 2009

8. Михайло Пічугін, Іван Канкін, Володимир Воротніков Комп'ютерна графіка. Навчальний посібник / Центр навчальної літератури 346 с. 2019р.

9. Маценко В.Г. Комп’ютерна графіка: Навчальний посібник. – Чернівці: Рута, 2009 – 343 с.

10. Інженерна комп’ютерна графіка: підручник / В.В. Проців [та ін.] / М-во освіти і науки України, Нац. гірн. унт-т. – Дніпро: НГУ, 2017. – 247 с.

11. Проців В.В. Прикладна комп'ютерна графіка [Текст]: Навч. посібник / В.В. Проців, К.А. Зіборов, К.М. Бас, Г.К. Ванжа; М-во освіти і наук, Нац. гірн. ун-т. - Д.: НГУ, 2016. - 187 с.

12. Kopf, Johannes and Lischinski, Dani. Depixelizing Pixel Art (англ.) // ACM Trans. Graph. – 2011. – Vol. 30, no. 4. – P. 99:1--99:8.

13. Giachetti, Andrea and Asuni, Nicola. Real-Time Artifact-Free Image Upscaling (англ.) // Trans. Img. Proc.. – 2011. – Vol. 20, no. 10. – P. 2760—2768.

14. Al-Mudhafar Aqeel, A.M., Smirnova, T., Buravchenko, K., Smirnov, O. «The method of assessing and improving the user experience of subscribers in software-configured networks based on the use of machine learning». *Advanced Information Systems*, 2023, 7(2), pp. 49-56.

15. Smirnov, O., Sydorenko, V., Aleksander, M., Zhyharevych, O., Yenchev, S. «Simulation of the cloud IoT-based monitoring system for critical infrastructures». *CEUR Workshop Proceedings,* Volume 3530, 2023, pp. 256-265.

16. Smirnov, O., Odarchenko, R., Smirnova, T., Bondar, S., Volosheniuk, D. «Optimal Structure Construction of Private 5G Network for the Needs of Enterprises». Lecture Notes on Data Engineering and Communications Technologies, 2023, 178, pp. 208–223.

17. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022,

18. Smirnov O., Smirnova T., Anas M. Al-Oraiqat, Drieiev O., Polishchuk L., Sheroz Khan, Yassin M. Y. Hasan, Aladdein M. Amro, Hazim S. AlRawashdeh «Method for Determining Treated Metal Surface Quality Using Computer Vision Technology». Sensors (Basel, Switzerland) Volume 22, Issue 16, 6223, 2022.

19. Smirnov O., Kuznetsov A., Kryvinska N., Kiian A., Kuznetsova K. «Full Non-Binary Constant-Weight Codes». SN Computer Science, Vol 2, 337, 2021. https://doi.org/10.1007/s42979-021-00739-w

20. Smirnov O., Kuznetsov A., Zhora V., Onikiychuk A., Pieshkova O. «Hiding Messages in Audio Files Using Direct Spread Spectrum». 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS 2021, Cracow, Poland, 22-25 September 2021. P. 414-418.

21. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE

International Conference on Advanced Information and Communication Technologies (AICT) - 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

22. Smirnov, O., Kuznetsov, A., Potii, O., Poluyanenko, N., Stelnyk, I., Mialkovsky, D. «Combining and filtering functions in the framework of nonlinear-feedback shift register». International Journal of Computing; 2020, Volume 19, Issue 2 – Research Institute for Intelligent Computer Systems – 2020. – P. 247-256.

23. Smirnov O., Kuznetsov A., Kiian A., Kuznetsova T. «Non-binary constant weight coding technique». CEUR Workshop Proceedings. Volume 2740, 2020, Pages 102-114.

24. Smirnov O., Alimseitova Zh., Adranova A., Akhmetov B., Lakhno V., Zhilkishbayeva G. «Models and algorithms for ensuring functional stability and cybersecurity of virtual cloud resources». Journal of theoretical and applied information technology Vol.98. No 21, 2020, P. 3334-3346.

25. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New technique for data hiding in cover images using adaptively generated pseudorandom sequences». CEUR Workshop Proceedings Volume 2654, 2020, Pages 1-14.

26. Smirnov O., Kuznetsov A., Onikiychuk A., Makushenko T., Anisimova O., Arischenko A. «Adaptive pseudo-random sequence generation for spread spectrum image steganography». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 161-165.

27. Smirnov O., Kuznetsov A., Kiian A., Cherep A., Kanabekova M., Chepurko I. «Testing of code-based pseudorandom number generators for post-quantum application». 2020 IEEE 11th International Conference on Dependable Systems, Services and Technologies (DESSERT), Ukraine, Kyiv, May 14-18. 2020. P. 172-177.

28. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and

Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

29. Smirnov, O., Drieieva, H., Drieiev, O., Polishchuk, Y., Brzhanov, R., Aleksander, M. «Method of fractal traffic generation by a model of generator on the graph». CEUR Workshop Proceedings Volume 2616, 2020, Pages 366-379.

30. Smirnov, O., Drieieva, H., Drieiev, O., Simakhin, V., Bondar, S., Odarchenko, R. «Managing multifractal properties of the binary sequence generated with the Markov chains», CEUR Workshop Proceedings Volume 2608, 2020, Pages 633-645.

31. Smirnov, O., Kuznetsov, A., Gorbacheva, L., Babenko, V., «Hiding data in images using a pseudo-random sequence», CEUR Workshop Proceedings Volume 2608, 2020, Pages 646-660.

32. Zhurakovskyi, B., Tsopa, N., Batrak, Y., Odarchenko, R., Smirnova, T «Comparative analysis of modern formats of lossy audio compression». Workshop Proceedings, 2020, 2654, стр. 315-327.

33. Smirnov O. Kuznetsov A., Zaicenko Yu., Pastukhov M., Oleshko O., Kuznetsova K., «Formation of Discrete Signals with Special Correlation Properties». International Conference on Information and Telecommunication Technologies and Radio Electronics, UkrMiCo 2019; Odessa; Ukraine; 9-13 September 2019. P.22-28.

34. Smirnov, O., Kuznetsov, A., Kolovanova, I., Kuznetsova, T., «Noise immunity of the algebraic geometric codes». International Journal of Computing; 2019, Volume 18, Issue 4 – Research Institute for Intelligent Computer Systems – 2019. – P. 393-407.

35. Smirnov, O., Kuznetsov, A., Reshetniak, O., Ivko, N., Katkova, T., Kuznetsova, T., «Generators of Pseudorandom Sequence with Multilevel Function of Correlation». 2019 IEEE International Scientific-Practical Conference Problems of Infocommunications, Science and Technology (PIC S&T), Kyiv, Ukraine, 8 – 11 October 2019 . P.517-522.

36. Smirnov, O., Krasnobayev, V., Yanko, A., Kuznetsova, T. «Methods of nulling numbers in the system of residual classes». CEUR Workshop Proceedings, Vol 2588, P. 90-106, 2019.

37. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Averchev, A., Pastukhov, M., Kuznetsova, K., «Formation of Pseudorandom Sequences with Special Correlation Properties», 2019 3rd International Conference on Advanced Information and Communications Technologies, AICT -2019/ Lviv, Ukraine, 2-6 July, 2019, P. 395-399.

38. Smirnov, O., Kuznetsov, A., Kavun, S., Babenko, B., Nakisko, O., Kuznetsova, K., «Malware Correlation Monitoring in Computer Networks of Promising Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 347-352.

39. Smirnov, O., Kuznetsov, A., Kovalchuk, D., Pastukhov, M., Kuznetsova, K., Prokopovych-Tkachenko, D., «Discrete Signals with Special Correlation Properties», CEUR Workshop Proceedings Volume 2353, CEUR Workshop Proceedings 2019, Pages 618-629.

40. Smirnov A.A., Kuzne tsov A.A., Danilenko D.A., Berezovsky A., «The statistical analysis of a network traffic for the intrusion detection and prevention systems», Telecommunications and Radio Engineering. – Volume 74, Issue 1. – Begel House Inc. – 2015. – P. 61-78.

41. Smirnov O., Kuznetsov A., Kovalchuk D., Kuznetsova T. «New Technique for Hiding Data in Cover Images Using Adaptively Generated Pseudorandom Sequences». CEUR Workshop Proceedings Volume 2732, 2020, Pages 214-227.

42. Батрак О., Смірнова Т., Гнатюк В., Одарченко Р., Смірнов О. «Дослідження показників ефективності функціонування та перспектив розвитку систем IP-телефонії». *Підводні технології,* 2024, № 13, с. 28-35.

43. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду

абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». *Сучасні інформаційні системи,* 2023, том 7, № 2, С. 49-56.

44. Т.В. Смірнова, О.М. Дрєєв, О.А. Смірнов «Хмарна інформаційна система оцінювання шорсткості з використанням дискретного частотного аналізу макрофотографій». IV міжнародна науково-практична конференція "Інформаційна безпека та комп'ютерні технології", м. Кропивницький. 15-16 квітня 2021р. – Кропивницький: ЦНТУ. – 2021. – С. 30.

45. О.А. Смірнов, П.С. Усік, «Дослідження перспектив використання технологічних рішень в мережах 5G» у Кібербезпека та інформаційні технології: монографія. – Х. : ТОВ «ДІСА ПЛЮС», 2020.С. 122-135.

46. О.А.Смірнов, Т.В.Смірнова, Л.І. Поліщук, К.О. Буравченко, А.О.Макевнін, «Дослідження хмарних технологій як сервісів», Кібербезпека: освіта, наука, техніка. № 3(7). С. 43-62. 2020.

47. Смірнов О.А., Дрєєва Г.М., Дрєєв О.М., Смірнова Т.В. «Фрактальний аналіз генератора самоподібного трафіку на основі ланцюга Маркова». Центральноукраїнський науковий вісник. Технічні науки. № 2(33). с. 161-172, 2019.

48. О. Смірнов, Є. Деменко, О. Онікійчук, А. Арищенко, Л. Горбачова, «Формування псевдовипадкових послідовностей для приховування даних в зображеннях» Комп'ютерні науки та кібербезпека. № 4. С. 30-37. 2019.

49. Смірнов О.А., Конопліцька-Слободенюк О.К., Смірнов С.А., Буравченко К.О., Смірнова Т.В. Поліщук Л.І. Проектування комп'ютерних систем та мереж. Навчальний посібник – Кропивницький: вид. Лисенко В.Ф. 2019. – 264 с.

50. Смірнов О.А., Дрєєва Г.М. Метод генерування фрактального трафіку за допомогою моделі генератора на графі. Монографія: Інформаційна безпека та інформаційні технології : монографія / за заг. ред. В. С. Пономаренка. – Х. : Вид. Рожко С.Г. 2019. С. 123-139

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за першим (бакалаврським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 131-02 від 01.04.2024 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є розробка програмного забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти є стосовна до теми література і існуючі аналоги.

5 Технічні вимоги

5.1 Склад продукції

Складниками розробки є:

– вибір і обґрунтування методів реалізації проекту;

– розробка програмної частин системи, а також розробка взаємодії системи з ОС та з користувачем;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет;

– цілісність даних у процесі роботи та при зберіганні;

– простий, інтуїтивно зрозумілий інтерфейс.

5.3 Вимоги до функціональних характеристик

Розроблене програмне забезпечення не повинно мати обмежень на версію драйверів та операційної системи.

5.4 Вимоги до архітектури

Компонент, що розробляється повинен використовувати системні засоби та апаратні засоби, що на даному етапі розвитку обчислювальної техніки найбільше поширені.

5.5 Вимоги до надійності

Програмні модулі написані по всім правилам, які стосуються стандартних викликів процедур, функцій, методів і форм, визначених технічною документацією на середовище розробки.

5.6 Умови експлуатації

Робочі місця користувачів ПЗ повинні задовольняти наступним умовам експлуатації:

– температура повітря: 19-20 град. по Цельсію;

– відносна вологість повітря до 80%;

– атмосферний тиск 107 кПа.

5.7 Вимоги до складу та параметрів технічних засобів

Програмне забезпечення повинно бути реалізоване на ПЕОМ, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

Переносність програмного забезпечення повинна бути забезпечена за рахунок його реалізації стандартного інтерфейсу взаємодії з ОС, що працюють під управлінням ОС Windows 10/11.

5.8.1 Обладнання

Комп’ютер Intel® Celeron/8 Mb/1.2 Gb/SVGA 14” 1Mb або сумісні з ним.

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

Програмна продукція повинна бути представлена у виді опису структури даних, схем та опису алгоритму, а також текстів вихідних модулів програмного забезпечення згідно ЄСПД .

7 Перелік документів, що розробляються

- Структурна схема системи – 1 аркуш.
- Функціональна схема системи – 1 аркуш.
- Діаграма процесів – 1 аркуш.
- Блок-схема алгоритму роботи програми – 2 аркуша.
- Пояснювальна записка – 74 аркуші.

8 Етапи розробки

8.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти (складання ТЗ).

8.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти.

8.3 Розробка функціональних схем, блок схем алгоритмів роботи програмного забезпечення.

8.4 Побудова схем взаємодії даних.

8.5 Створення прототипу ПЗ.

8.6 Віднаходження ПЗ, аналіз отриманих результатів.

8.7 Оформлення пояснювальної записки і виконання робіт по графічній частині.

9 Порядок контролю та приймання

9.1 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на попередній захист 23.05.2024 р.

9.2 Подання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти на захист 14.06.2024 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти

_____ Коваленко О.В.

Програмне забезпечення системи на основі методу структурно-лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет

Лістинг програми

Код документу 12

Носій: CD/DVD-диск / USB-флеш-накопичувач

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2024 року

Файл UMain.pas - основна програма

```
unit UMain;

{---------Головний модуль програми керування--------}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, VCap,VCapStrings ,StdCtrls,DirectShow,
ExtCtrls,ComCtrls,ComObj,Active,Speech,
  UPreview, //модуль із компонентами для захоплення відео
  UGraphConfig, //модуль інтерфейс, що реалізує, настроювання камер
  USingleFrame, //модуль перегляд, що реалізує, окремих кадрів
  UTypeConst, //модуль утримуючі загальні типи й константи
  Can_Dll, //модуль імпортує фукції з Can.dll
  UCommunication, //модуль із фукціями для повідомлення по Кан'у
  URazvertka, //модуль циклічного розгорнення
  UCamTimers, //таймери камер
  UMoving, //руху
  URazvertkaConfig, //форма настроювання розгорнення
  URoboRootServer, //робосервер
  URoboServerConfig, //форма настроювання робосервера
  USystemCoordinat, //система координат
  URisovalka, //рисовалка
  UCommunicationForm, //форма висновку інформації про маяк
  UCharAnalyz, //аналіз букв
  UQPixels, //швидкі пікселі
  UAddSample,
  URestaran,
  UDebug,
  UEmul,
  UDialog, BCPort,
  about;

const port = 'COM2';

type
  TMainForm = class(TForm)
    Button3: TButton;
    MoveForLine: TButton;
    CamsListBox: TListBox;
    Label7: TLabel;
    RefreshCamsListButton: TButton;
    CamConfigButton: TButton;
    CamsGroupBox: TGroupBox;
    DialogListBox: TListBox;
    Label1: TLabel;
    VideoModeComboBox: TComboBox;
    Label2: TLabel;
    RisovalkaButton: TButton;
    CommunicationButton: TButton;
    AnalyzFrameButton: TButton;
    RazvertkaButton: TButton;
    OnFlyDebugButton: TButton;
    Button1: TButton;
    Button2: TButton;
    TLabel: TLabel;
    AddSampleButton: TButton;
    RestaranButton: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Label3: TLabel;
    Timer1: TTimer;
    Edit1: TEdit;
```

```
    Edit2: TEdit;
    Label4: TLabel;
    Button7: TButton;
    NeuroCount: TEdit;
    Button8: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure MovingFormButtonClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure MoveForLineClick(Sender: TObject);
    procedure RefreshCamsListButtonClick(Sender: TObject);
    procedure CamConfigButtonClick(Sender: TObject);
    procedure CamsListBoxClick(Sender: TObject);
    procedure DialogListBoxDblClick(Sender: TObject);
    procedure VideoModeComboBoxChange(Sender: TObject);
    procedure CommunicationButtonClick(Sender: TObject);
    procedure RisovalkaButtonClick(Sender: TObject);
    procedure AnalyzFrameButtonClick(Sender: TObject);
    procedure RazvertkaButtonClick(Sender: TObject);
    procedure OnFlyDebugButtonClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure RestaranButtonClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);

  private
    { Private declarations }
  public
    { Public declarations }
    {Для розгорнення}
    //настроїти параметри циклічного розгорнення
    procedure ConfigRazvertka(CamNum: Cardinal);
  end;



function Send:integer; //запис даних у буфер КАН'а для відправлення

type
  mass = array [1..250] of byte;
  reg_array = ^smallint;


function mbConnect(const port: string ; speed: integer;parity: integer;
stopbits: integer;flow: integer): integer;
 cdecl external 'MODBUS.dll' ;
function mbDisconnect(): integer;
 cdecl external 'MODBUS.dll';
function mbExecuteProgramFile(dev: cardinal; filename: string): integer;
 cdecl external 'MODBUS.dll';
function mbReadHoldingRegisters  (dev: cardinal; dest: reg_array; address:
cardinal; count: cardinal): integer;
 cdecl external 'MODBUS.dll';
FUNCTION mbReportDeviceID(dev: CARDINAL;  dest: mass; max_len: cardinal;
actual_len : Pointer): integer;
 cdecl external 'MODBUS.dll';
function mbReset( dev: cardinal): integer;
 cdecl external 'MODBUS.dll';
function mbSetLogDetails(log_errors: boolean;log_messages: boolean;log_data:
boolean): integer;
 cdecl external 'MODBUS.dll';
```

```
function mbWriteHoldingRegisters(dev: integer; from: reg_array; address:
cardinal; count: cardinal): integer;
 cdecl external 'MODBUS.dll';




var
  MainForm: TMainForm; //головна форма

  //масив настроювань камер
  GraphConfigExArr: TGraphConfigExArr;

  //об'єкт головного потоку робосервера
  RoboRootServerManageThread: TRoboRootServerManagThread;

   //прапор "рух по смузі"
  MoveForLineFlag: boolean = false;

  //камера для висновку відео
  ActiveCamIndex: Integer = -1;

  {для TextToSpeech}
  {Центральний інтерфейс,через який виробляються всі дії з мовою}
  fITTSCentral: ITTSCentral;

  {Інтерфейс для зв'язку з аудіопристроєм}
  fIAMM: IAudioMultimediaDevice;

  {Інтерфейс для перебору движків}
  aTTSEnum: ITTSEnum;

  {Покажчик на параметри движка}
  fpModeInfo: PTTSModeInfo;

  NumFound : DWord;
  ModeInfo : TTSModeInfo;

  QPixels: TQuickPixels;
  timercnt: Cardinal = 0;


   i: integer;
   dest1: mass;



{процедура виводить в ListBox
список доступних діалогів у компоненті VideoCapture}
procedure ShowAvialableDialogs(ListBox: TListBox; VideoCapture: TVideoCapture);

{ функція зберігає відео настроювання у файл зі змінних
У разі удачі повертає true, інакше false}
function SaveGraphConfig(var GraphConfigExArr: TGraphConfigExArr): boolean;

{ функція ініціалізує камери
VideoDeviceList - список всіх доступних камер
у випадку удачі функція повертає true, інакше false}
function InitCams(var GraphConfigExArr: TgraphConfigExArr;
                        VideoDeviceList: TStringList): boolean;

//процедура виводить в ComboBox доступні відео режими на компоненті VideoCapture
procedure ShowVideoModes(ComboBox: TComboBox; VideoCapture: TVideoCapture);

//процедура ініціалізує відео настроювання для камери CamName за замовчуванням
procedure DefaultInitCam(CamName: String; var GraphConfigEx: TGraphConfigEx);

//---------------------------------i
```

```
//процедура, що вимовляє текст
procedure SayText(Text: String);

function GetVideoDevicesListEm(): TStringList;

implementation

uses Math;

{$R *.dfm}

function GetVideoDevicesListEm(): TStringList;
begin
  Result := TStringList.Create;
  Result.Add('Samsung SuperShit Cam');
  Result.Add('Hitachi MegaSlow WebCam');
  Result.Add('Електроніка КХ - 1');
end;


{Допоміжні процедури, що не входять у клас форми}

//процедура, що вимовляє текст
procedure SayText(Text: String);
var
  SData: TSData;
begin
  {Цей текст буде прочитаний}
  SData.dwSize := length(Text) + 1;
  SData.pData := pChar(Text);

  fITTSCentral.TextData(CHARSET_TEXT, 0, SData, nil, IID_ITTSBufNotifySink);
end;
//---------------------------------i

{процедура виводить в ListBox
список доступних діалогів у компоненті VideoCapture}
procedure ShowAvialableDialogs(ListBox: TListBox; VideoCapture: TVideoCapture);
var
  d: TCaptureDialog; //всі можливі ідентифікатори діалогів
begin
  ListBox.Clear; //очищення ListBox
  //додавання в ListBox всіх доступних діалогів
  for d := Low(TCaptureDialog) to High(TCaptureDialog) do
  if d in VideoCapture.Dialogs then
    ListBox.Items.AddObject(DialogTitles[d],TObject(d));

end;
//---------------------------------i

{ функція зберігає відео настроювання у файл зі змінних,
у разі удачі повертає true, інакше false}
function SaveGraphConfig(var GraphConfigExArr: TGraphConfigExArr): boolean;
var
  GraphConfigFile: TextFile; //файлова змінна
  GraphConfigStr: String; //рядок з відео настроюваннями
  i: integer; //лічильник
begin
  Result := true;

  //відкриття файлу
  AssignFile(GraphConfigFile,GraphConfigFileName);
  {$ I-I-}
  Rewrite(GraphConfigFile);

  for i := 0 to High(GraphConfigExArr) do
  begin
```

```
    //одержання настроювань у строковому форматі
    GraphConfigStr := GraphConfigExArr[i].GraphConfig.SaveGraph;
    //запис у файл
    writeln(GraphConfigFile,GraphConfigStr);
    writeln(GraphConfigFile,GraphConfigExArr[i].ShowPreview);
    writeln(GraphConfigFile,GraphConfigExArr[i].CamFunc);
    writeln(GraphConfigFile,GraphConfigExArr[i].TimerDelay);
  end;

  //закриття файлу
  CloseFile(GraphConfigFile);
  {$I+}
  if IOResult <> 0 then Result := false;
end;
//---------------------------------i

{ функція ініціалізує камери
VideoDeviceList - список всіх доступних камер
у випадку удачі функція повертає true, інакше false}
function InitCams(var GraphConfigExArr: TGraphConfigExArr;
                        VideoDeviceList: TStringList): boolean;
var
  GraphConfigFile: TextFile; //файлова змінна
  CurLine: String; //рядок файлу
  i: integer; //лічильник
  CamsInitStat: array of boolean; //стан ініціалізованості камер
  TmpConfig: TGraphConfig;//тимчасове зберігання настроювань

begin
  Result := true;

  //розміри масивів
  SetLength(GraphConfigExArr,VideoDeviceList.Count);
  SetLength(CamsInitStat,VideoDeviceList.Count);

  //заповнюємо
  for i := 0 to High(CamsInitStat) do
    CamsInitStat[i] := false;

  //якщо є файл із настроюваннями відео - зчитуємо з нього рядка настроювань
  if FileExists(GraphConfigFileName) then
  begin
    //ініціалізація
    TmpConfig := TGraphConfig.Create;

    //присвоєння файлу
    AssignFile(GraphConfigFile,GraphConfigFileName);

    //проходимо за списком доступних камер і шукаємо їхнього настроювання у
файлі
    for i := 0 to VideoDeviceList.Count - 1 do
    begin
      {$ I-I-}
      Reset(GraphConfigFile);

      //поки не скінчився файл або поки не знайшли настроювання поточної камери
      while (not EOF(GraphConfigFile)) and (not CamsInitStat[i]) do
      begin
        //читаємо рядок з головними настроюваннями
        readln(GraphConfigFile,CurLine);
        TmpConfig.RestoreGraph(CurLine);

        //якщо ці настроювання для поточної камери, те привласнюємо їх їй
        if TmpConfig.VCapSource = VideoDeviceList.Strings[i] then
        begin
          GraphConfigExArr[i].GraphConfig := TGraphConfig.Create;

          GraphConfigExArr[i].GraphConfig.RestoreGraph(CurLine);
```

```
          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].ShowPreview := StrToInt(CurLine);

          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].CamFunc := StrToInt(CurLine);

          readln(GraphConfigFile,CurLine);
          GraphConfigExArr[i].TimerDelay := StrToInt(CurLine);

          //ця камера проиниціалазирована!
          CamsInitStat[i] := true;
        end
        else
        begin
          //перелистуємо 3-и рядка з іншими настроюваннями
          readln(GraphConfigFile);
          readln(GraphConfigFile);
          readln(GraphConfigFile);
        end;
      end;

      //закриття файлу
      CloseFile(GraphConfigFile);
      {$I+}
      if IOResult <> 0 then
      begin
        Result := false;
        Exit;
      end;
    end;
  end;
  //якщо залишилися не проініціалізовані камери, ініціалізуємо їх за
замовчуванням
  for i := 0 to High(CamsInitStat) do
  begin
    if not CamsInitStat[i] then
      DefaultInitCam(VideoDeviceList.Strings[i],GraphConfigExArr[i]);
  end;
end;
//---------------------------------

//процедура виводить в ComboBox доступні відео режими на компоненті VideoCapture
procedure ShowVideoModes(ComboBox: TComboBox; VideoCapture: TVideoCapture);
var
  i: integer; //лічильник
  CurVideoMode: TVCapMode; //поточний відео режим
begin
  ComboBox.Clear; //очищення
  CurVideoMode := VideoCapture.VCapMode; //запам'ятовуємо тек. відео режим

  //у циклі виводимо доступні й визначаємо поточний відео режими
  for i := 0 to VideoCapture.VCapModeCount - 1 do
  begin
    ComboBox.Items.Add(GetModeString(VideoCapture.VCapModes[i]));
    if IsEqualModes(CurVideoMode,VideoCapture.VCapModes[i]) then
      ComboBox.ItemIndex := i;
  end;
end;
//---------------------------------

//процедура ініціалізує відео настроювання для камери CamName за замовчуванням
procedure DefaultInitCam(CamName: String; var GraphConfigEx: TGraphConfigEx);
begin
 //створення об'єкта основних настроювань камери
 GraphConfigEx.GraphConfig := TGraphConfig.Create;

 with GraphConfigEx.GraphConfig do
  begin
    //ім'я камери
```

```
    VCapSource := CamName;

    //аудіо пристрою не використовуємо
    ACapSource := '';

    //аудіо компресори не використовуємо
    AComp := '';

    //імена файлів для запису відео
    CaptureFileName := 'Capture.avi';
    TempCaptureFileName := 'TempCapture.avi';
    PreallocFileSize := 0;

    //що хочемо одержати
    WantCapture := false;
    WantPreview := false;
    WantBitmaps := false;

    //аудіо не потрібно
    WantAudio := false;
    WantDVAudio := false;
    WantAudioPreview := false;

    //тимчасовий файл
    UseTempFile := true;
    DoPreallocFile := false;
    PreallocFileSize := 0;

    //формат пікселя
    PixelFormat := pfDevice;
    //???
    DVResolution := dvrDontWorry;

    //настроювання відео режиму
    with VCapMode do
    begin
      MediaType := MEDIATYPE_Video;
      MediaSubType := MEDIASUBTYPE_RGB24;
      Width := 640;
      Height := 480;
      BitCount := 24; //???
      FrameRate := 30.000;
      MinFrameRate := 30.000;
      MaxFrameRate := 30.000;
    end;
  end;
  GraphConfigEx.ShowPreview := 0; //потрібно чи показувати форму із зображенням
  GraphConfigEx.CamFunc := CamFuncNone; //камера не функціональна
  GraphConfigEx.TimerDelay := 65; //частота обробки 65 мс
end;
//---------------------------------

{допоміжні процедури, що входять у клас форми
 у всіх процедурах: CamNum - номер камери,
 над якою виробляється дія}

{Для розгорнення}
//настроїти параметри циклічного розгорнення
procedure TMainForm.ConfigRazvertka(CamNum: Cardinal);
begin
  //набудовуємо перше розгорнення
  URazvertkaConfig.LocalRazvertkaConfig := @RazvertkaArr[CamsListBox.ItemIndex];
  RazvertkaConfigForm.ShowModal;
  RazvertkaArr[CamsListBox.ItemIndex].SetRazvertkaConfig(LocalRazvertkaConfig^);
end;
//---------------------------------

{Оброблювачі подій}
```

```
//Подія - перед створенням форми
{{Тут відбувається ініціалізація настроювань компонентів і змінних}
procedure TMainForm.FormCreate(Sender: TObject);
var
  VideoDeviceList: TStringList; //список відео пристроїв
  Res1,Res2: HRESULT;
begin
  {для text to speech}
  {Ініціалізація аудіопристрою}
  Res1 := CoCreateInstance(CLSID_MMAudioDest, Nil,
CLSCTX_ALL,IID_IAudioMultiMediaDevice, fIAMM);
  {Створення об'єкта, що перераховується, для перебору всіх движків у системі за
допомогою інтерфейсу ITTSEnum}
  Res2 := CoCreateInstance(CLSID_TTSEnumerator, Nil,
CLSCTX_ALL,IID_ITTSEnum,aTTSEnum);

  if (Res1 = S_OK) and (Res2 = S_OK) then
  begin
    aTTSEnum.Reset;//Скидаємо на перший
    {Одержуємо другий движок}
    aTTSEnum.Next(1, ModeInfo, @NumFound);
    aTTSEnum.Next(1, ModeInfo, @NumFound);
    aTTSEnum.Select(ModeInfo.gModeID, fITTSCentral, IUnknown(fIAMM));
    {Одержуємо інші}
    {While NumFound > 0 do
     begin
       ComboBox1.Items.Add(String(ModeInfo.szModeName));
       aTTSEnum.Next(1, ModeInfo, @NumFound);
     end;}
  end;
  //ініціалізація модуля спілкування
  Communication := TCommunication.Create;
  Communication.Start;

  //створюємо об'єкт системи координат
  SystemKoordinat := TSystemKoordinat.Create;
  SystemKoordinat.Start;

  //одержуємо список доступних камер
  VideoDeviceList := GetVideoDevicesList();

  //запуск головного потоку робосервера
  if RoboRootServerActive then
    RoboRootServerManageThread := TRoboRootServerManagThread.Create(false);

  //якщо є хоча б одна камера
  if VideoDeviceList.Count > 0 then
    URoboRootServer.VideoExist := true;

  //потрібно проініціалізувати настроювання камер
  InitCams(GraphConfigExArr,VideoDeviceList);

  //заповнюємо список камер
  CamsListBox.Items.Assign(VideoDeviceList);

end;
//---------------------------------

//Подія - перед відображенням форми
//Тут активуються компоненти й настроюються керуючі елементи
//(кнопки, форми й т.д.)
procedure TMainForm.FormShow(Sender: TObject);
var
  i: integer; //лічильник
begin
  //створення масиву компонентів для роботи з камерами
  SetLength(VideoCaptureArr,Length(GraphConfigExArr));
  for i := 0 to High(VideoCaptureArr) do
  begin
```

```
    VideoCaptureArr[i] := TVideoCapture.CreateParented(PreviewWindow.Handle);
    VideoCaptureArr[i].RestoreGraph(GraphConfigExArr[i].GraphConfig);
  end;

  //завдання розміру масиву оброблювачів захоплення кадру
  SetLength(BitmapGrabEventArr,Length(GraphConfigExArr));

  //створення об'єктів оброблювачів кадрів
  for i := 0 to High(GraphConfigExArr) do
  begin
    BitmapGrabEventArr[i] := TBitmapGrabEvent.Create(i);
    VideoCaptureArr[i].OnBitmapGrabbed := BitmapGrabEventArr[i].AnalyzBitmap;
  end;

  //завдання розміру масиву таймерів камер
  SetLength(CamTimerArr,Length(GraphConfigExArr));

  //створення й запуск таймерів для потрібних камер
  for i := 0 to High(GraphConfigExArr) do
  begin
    CamTimerArr[i] := TCamTimer.Create(i);
    if GraphConfigExArr[i].GraphConfig.WantBitmaps then
    begin
      CamTimerArr[i].StartTimer(GraphConfigExArr[i].TimerDelay);
    end;
  end;

  //задаємо розмір масивам розгорнення
  SetLength(RazvertkaArr,Length(GraphConfigExArr));
  SetLength(RazvertkaConfigArr,Length(GraphConfigExArr));

  //задаємо настроювання для розгорнень
  for i := 0 to High(RazvertkaArr) do
  begin
    with RazvertkaConfigArr[i] do
    begin
      a := VideoCaptureArr[i].VCapMode.Width div 2;
      a_corr := 5;
      b := VideoCaptureArr[i].VCapMode.Height div 2;
      y_offset := 0;
      klaster_size := 5;
      porog := 50;
      step := 3;
      fi_step := 0.005;
      RazvertkaArr[i] := TRazvertka.Creat(RazvertkaConfigArr[i]);
    end;
  end;
  //кнопка для руху
  if not Communication.PortEn then
  begin
   // RisovalkaButton.Enabled := false;
  end;
end;
//---------------------------------


//Подія - перед закриттям форми
procedure TMainForm.FormClose(Sender: TObject; var Action: TCloseAction);
var
  i: integer;
begin
  //зупинка системи координат
  SystemKoordinat.Stop;
  SystemKoordinat.Destroy;


  //зупинка каналу
  Communication.Destroy;
```

```
  //зупинка робосервера
    if RoboRootServerActive then
    RoboRootServerManageThread.Terminate;



  //зупинка таймерів камер
  for i := 0 to High(CamTimerArr) do
  begin
    if CamTimerArr[i] <> nil then
      CamTimerArr[i].Destroy;
  end;
  SetLength(CamTimerArr,0);

  //знищення оброблювачів події захоплення кадру з камер
  for i := 0 to High(BitmapGrabEventArr) do
  begin
    if BitmapGrabEventArr[i] <> nil then
      BitmapGrabEventArr[i].Destroy;
  end;
  SetLength(BitmapGrabEventArr,0);

  //зупинка й знищення об'єктів для захоплення відео
  for i := 0 to High(VideoCaptureArr) do
  begin
    if VideoCaptureArr[i].Capturing then
      VideoCaptureArr[i].StopCapture;
    if VideoCaptureArr[i].Previewing then
      VideoCaptureArr[i].StopPreview;
    VideoCaptureArr[i].Destroy;
  end;
  SetLength(VideoCaptureArr,0);

  //знищення розгорнень і їхніх настроювань
  for i := 0 to High(RazvertkaArr) do
  begin
    RazvertkaArr[i].Destroy;
  end;
  SetLength(RazvertkaArr,0);
  SetLength(RazvertkaConfigArr,0);
end;
//-----------------------------------

//Подія - натискання на "Рухи"
procedure TMainForm.MovingFormButtonClick(Sender: TObject);
begin
  MovingForm.ShowModal;
end;
//-----------------------------------

//Подія - Натискання "Сервер"
procedure TMainForm.Button3Click(Sender: TObject);
begin
  RoboServerConfigForm.ShowModal;
end;
//-----------------------------------

//Подія - Натискання на "Рух по смузі"
procedure TMainForm.MoveForLineClick(Sender: TObject);
begin
  ForwSpeed := 30;
  MoveForLineFlag := not MoveForlineFlag;
end;
//-----------------------------------

//Подія - натискання на "Обновити" (список камер)
procedure TMainForm.RefreshCamsListButtonClick(Sender: TObject);
var
```

```
  VideoDeviceList: TStringList; //список камер
  i: integer; //лічильник
begin
  //одержуємо список камер
  VideoDeviceList := GetVideoDevicesList(true);

  //ініціалізуємо камери заново
  InitCams(GraphConfigExArr,VideoDeviceList);

  //знищення об'єктів для вже неіснуючих камер
  for i := VideoDeviceList.Count to High(VideoCaptureArr) do
  begin
    CamTimerArr[i].StopTimer;
    CamTimerArr[i].Destroy;
    BitmapGrabEventArr[i].Destroy;
  end;

  //установлюємо нові розміри масивів
  SetLength(VideoCaptureArr,VideoDeviceList.Count);
  SetLength(BitmapGrabEventArr,VideoDeviceList.Count);
  SetLength(CamTimerArr,VideoDeviceList.Count);
  //у циклі створюємо потрібні об'єкти
  for i := 0 to High(VideoCaptureArr) do
  begin
    //об'єкти для захоплення відео
    if VideoCaptureArr[i] = nil then
    begin
      VideoCaptureArr[i] := TVideoCapture.CreateParented(PreviewWindow.Handle);
      VideoCaptureArr[i].RestoreGraph(GraphConfigExArr[i].GraphConfig);
    end;
    //події захоплення кадру
    if BitmapGrabEventArr[i] = nil then
    begin
      BitmapGrabEventArr[i] := TBitmapGrabEvent.Create(i);
      VideoCaptureArr[i].OnBitmapGrabbed := BitmapGrabEventArr[i].AnalyzBitmap;
    end;
    //таймери для захоплення кадрів
    if (CamTimerArr[i] = nil) then
    begin
      CamTimerArr[i] := TCamTimer.Create(i);
      if GraphConfigExArr[i].GraphConfig.WantBitmaps then
        CamTimerArr[i].StartTimer(GraphConfigExArr[i].TimerDelay);
    end;
  end;

  //заповнюємо список на формі
  CamsListBox.Clear;
  for i := 0 to VideoDeviceList.Count - 1 do
    CamsListBox.Items.Add(VideoDeviceList.Strings[i]);
end;
//-----------------------------------

//Подія - натискання на "Настроювання" (обраної камери)
procedure TMainForm.CamConfigButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    //передаємо індекс настроювань обраної камери в модуль настроювання камери
    UGraphConfig.GraphConfigExIndex := CamsListBox.ItemIndex;
    //виводимо форму настроювань камери в модальному режимі
    if UGraphConfig.GraphConfigForm.ShowModal = mrOK then
    begin
      //зберігаємо й відновлюємо настроювання камери
      SaveGraphConfig(GraphConfigExArr);

VideoCaptureArr[CamsListBox.ItemIndex].RestoreGraph(GraphConfigExArr[CamsListBox
.ItemIndex].GraphConfig);

      //запускаємо або перезапускаємо або зупиняємо таймери якщо потрібно
```

```
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;

CamTimerArr[CamsListBox.ItemIndex].StartTimer(GraphConfigExArr[CamsListBox.ItemI
ndex].TimerDelay);
      end
      else
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;
      end;

      //якщо потрібно показувати зображення на екрані
      if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
      begin
        ActiveCamIndex := CamsListBox.ItemIndex;
        PreviewWindow.ShowEx;
      end
      else
      begin
        PreviewWindow.Hide;
        ActiveCamIndex := -1;
      end;
      //обновляємо діалоги й відео режими

ShowAvialableDialogs(DialogListBox,VideoCaptureArr[CamsListBox.ItemIndex]);
      ShowVideoModes(VideoModeComboBox, VideoCaptureArr[CamsListBox.ItemIndex]);

      //кнопки
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        AnalyzFrameButton.Enabled := true;
        OnFlyDebugButton.Enabled := true;
      end
      else
      begin
        AnalyzFrameButton.Enabled := false;
        OnFlyDebugButton.Enabled := false;
      end;
    end;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//подія - натискання на Списку камер
procedure TMainForm.CamsListBoxClick(Sender: TObject);
begin
  //виводимо картинку якщо потрібно
  if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
  begin
    ActiveCamIndex := CamsListBox.ItemIndex;
    PreviewWindow.ShowEx;
  end
  else
  begin
    PreviewWindow.Hide;
    ActiveCamIndex := -1;
  end;
  //виводимо діалоги й відео режими для обраної камери
  ShowAvialableDialogs(DialogListBox,VideoCaptureArr[CamsListBox.ItemIndex]);
  ShowVideoModes(VideoModeComboBox, VideoCaptureArr[CamsListBox.ItemIndex]);

  //ім'я камери
  PreviewWindow.Caption :=
GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapSource;
```

```
  //кнопки
  if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
  begin
    AnalyzFrameButton.Enabled := true;
    OnFlyDebugButton.Enabled := true;
  end
  else
  begin
    AnalyzFrameButton.Enabled := false;
    OnFlyDebugButton.Enabled := false;
  end;
end;
//-----------------------------------

//подія - подвійний натискання на списку діалогів камери
procedure TMainForm.DialogListBoxDblClick(Sender: TObject);
var
  i: integer; //лічильник
begin
  //шукаємо виділений рядок, щоб викликати діалог, ім'я якого в ній написано
  for i := 0 to DialogListBox.Count - 1 do
    if DialogListBox.Selected[i] then
    begin
      MainForm.Enabled := false;

UPreview.VideoCaptureArr[CamsListBox.ItemIndex].ShowDialog(TCaptureDialog(Dialog
ListBox.Items.Objects[i]));
      MainForm.Enabled := true;
    end;
  //зберігаємо новий відео режим
  GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapMode :=
VideoCaptureArr[CamsListBox.ItemIndex].VCapMode;
  SaveGraphConfig(GraphConfigExArr);
  //виводимо сталий режим у списку відео режимів

ShowVideoModes(VideoModeComboBox,UPreview.VideoCaptureArr[CamsListBox.ItemIndex]
);
end;
//-----------------------------------

//Подія - зміна в списку відео режимів
procedure TMainForm.VideoModeComboBoxChange(Sender: TObject);
begin
  //Установлюємо новий відео режим

VideoCaptureArr[CamsListBox.ItemIndex].SetVCapMode(VideoModeComboBox.ItemIndex);
  //змінюємо настроювання
  GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapMode :=
    VideoCaptureArr[CamsListBox.ItemIndex].VCapMode;
  //зберігаємо настроювання
  SaveGraphConfig(GraphConfigExArr);
  //якщо потрібно, те возобнавляем висновок на екран
  if GraphConfigExArr[CamsListBox.ItemIndex].ShowPreview = 1 then
    VideoCaptureArr[CamsListBox.ItemIndex].StartPreview;
end;
//-----------------------------------

//Подія - натискання на "Маяки"
procedure TMainForm.CommunicationButtonClick(Sender: TObject);
begin
  CommunicationForm.Show;
end;
//-----------------------------------

//Подія - натискання на "Рисовалка"
procedure TMainForm.RisovalkaButtonClick(Sender: TObject);
begin
 OKRightDlg.Visible:=true;
```

```
// MainForm.Hide;
//  URisovalka.RisovalkaForm.Visible := true;
end;
//------------------+-----------------i

//Подія - Натискання на "Аналіз Кадру"
procedure TMainForm.AnalyzFrameButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].AnalyzFrame := true;
    FrameForm.ShowModal;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//Подія - натискання на "Розгорнення"
procedure TMainForm.RazvertkaButtonClick(Sender: TObject);
begin
  //виводимо форму настроювань розгорнення
  if CamsListBox.ItemIndex >= 0 then
  begin
    URazvertkaConfig.LocalRazvertkaConfig :=
@RazvertkaConfigArr[CamsListBox.ItemIndex];
    RazvertkaConfigForm.ShowModal;

RazvertkaArr[CamsListBox.ItemIndex].SetRazvertkaConfig(URazvertkaConfig.LocalRaz
vertkaConfig^);
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------

//Подія - натискання на "дебаг на льоту"
procedure TMainForm.OnFlyDebugButtonClick(Sender: TObject);
begin
  //якщо обрано камеру
  if CamsListBox.ItemIndex >= 0 then
  begin
    //установлюємо прапорець для дебага на льоту й виводимо форму
    BitmapGrabEventArr[CamsListBox.ItemIndex].OnFlyDebug := true;
    FrameForm.ShowModal;
  end
  else
    ShowMessage('Не обрана камера!');
end;
//-----------------------------------


procedure TMainForm.Button1Click(Sender: TObject);
begin
  Communication.stopsignal := 1;
end;

procedure TMainForm.Button2Click(Sender: TObject);
begin
  Communication.stopsignal := 0;
end;
//-----------------------------------

//Подія - натискання "Семпли"
procedure TMainForm.AddSampleButtonClick(Sender: TObject);
begin
  if CamsListBox.ItemIndex >= 0 then
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].NeedAddSample := true;
```

```
  end
  else
    UAddSample.AddSampleForm.ShowModal;
end;

procedure TMainForm.RestaranButtonClick(Sender: TObject);
begin
  Restaran := TRestaran.Create;
  Restaran.Start(1);
end;

procedure TMainForm.Button4Click(Sender: TObject);
begin
  UDebug.DebugForm.Show;
end;

procedure TMainForm.Button6Click(Sender: TObject);
begin
 //mbDisconnect();
 //MainForm.Label3.Caption:='Порт закритий';
end;



function Send:integer;
begin
   result:=mbWriteHoldingRegisters(1,@(Communication.RecvBuff),0,3);
 // mbWriteHoldingRegisters(1,@SendBuff,0,4);
 //result:=SENDMSG(SendMsNb, CANSendBuff)
end;


procedure TMainForm.Button5Click(Sender: TObject);
var
  len_written: integer;
  speeds: array [0..1] of smallint;
begin
  //mbSetLogDetails(true,true,true);
  mbSetLogDetails(false,false,false);
  i:=mbConnect(port,115200,1,0,3);

  if i=0 then
  begin
   MainForm.Label3.Caption:='не вдалося відкрити порт';
  end
  else
  begin
   MainForm.Label3.Caption:='порт відкритий';


   mbReset(1);
   sleep(10);

   mbExecuteProgramFile(1,'image.raw');

   sleep(10);
   mbReportDeviceID(1,dest1,250,(@len_written));

   Communication.PortEn:=true;
  end;



   MainForm.Timer1.Enabled:=true;

   //Speeds[0]:=100;
   //Speeds[1]:=-100;
```

```
  // Communication.RecvBuff.w1:=100;
  // Communication.RecvBuff.w2:=-100;
  // Communication.RecvBuff.w3:=0;

   // Send;
  // mbWriteHoldingRegisters(1,@(Communication.RecvBuff),0,3);
end;




procedure TMainForm.Timer1Timer(Sender: TObject);
begin
//ghgf
end;

procedure TMainForm.Button7Click(Sender: TObject);
var i1,count,tmp,i:integer;
n_debug:textfile;
begin

    assignfile(n_debug,neuro_debug_file_name);
    append(n_debug);

    count:=0;
    tmp:=AddSampleForm.NeuralNetBP1.LayerCount-1;
    NeuroCount.Text:='';

    //    for i1:=0 to length(xInputVector) do
    //    write(n_debug,inttostr(round(xInputVector[i1])));
    //    writeln(n_debug,'');

    addsampleform.NeuralNetBP1.Compute(xInputVector);


    //addsampleform.NeuralNetBP1.
    for i := 0 to length(xOutputVector)-1 do
    begin

xOutputVector[i]:=AddSampleForm.NeuralNetBP1.LayersBP[tmp].Neurons[i].Output;
    NeuroCount.Text:=NeuroCount.Text+'-'+floattostr(xOutputVector[i]);
    end;
    //if AddSampleForm.Layers[1].Neurons[i].Output = 1 then
      //Count:=count+1;

  // Нейрона мережа Хопфілда
    {addsampleform.NeuralNetHopf1.Calc;

  for i := 0 to IconSize* IconSize-1 do
    if AddSampleForm.NeuralNetHopf1.Layers[1].Neurons[i].Output = 1 then
      Count:=count+1;

  NeuroCount.Text:=inttostr(count);
 }
    closefile(n_debug)
end;

procedure TMainForm.Button8Click(Sender: TObject);
begin
Form5.Show;
end;

end.
```

Файл UPreview.pas – модуль роботи з камерами

```
unit UPreview;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, VCap,USingleFrame,UTypeConst,URazvertka, UCamFunc,URoboRootServer,
  ExtCtrls,UMoving,UCharAnalyz,UQPixels,UAddSample;

type
  //захоплений кадр
  TCapturedBitmap = Vcap.TCapturedBitmap;

  TBitmap = Graphics.TBitmap;

  //клас який описує подію захоплення кадру з камери
  TBitmapGrabEvent = class
    private
      EventIndex: Integer; //індекс події (номер камери)
    public
      AnalyzFrame: boolean; //аналіз кадру
      OnFlyDebug: boolean; //дебаг на льоту
      NeedAddSample: boolean; //потрібно додати семпл
      constructor Create(Index: Cardinal); //конструктор
      destructor Destroy; override; //деструктор
      //аналіз захопленого кадру
      procedure AnalyzBitmap(Bitmap: TCapturedBitmap);
  end;

  //тип масиву оброблювачів захоплення кадру з камери
  TBitmapGrabEventArr = array of TBitmapGrabEvent;


  TPreviewWindow = class(TForm)
    Video: TImage;
    procedure VideoCaptureDeviceLost(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
    procedure ShowEx;
  end;

var
  PreviewWindow: TPreviewWindow; //форма для висновку зображення з камери
  VideoCaptureArr: TVideoCaptureArr; //масив компонентів для захоплення відео
  BitmapGrabEventArr: TBitmapGrabEventArr;  //масив оброблювачів захоплення
кадру з камери
  QP: TQuickPixels;

  UgliPolosi: TUgliRazrivov; //кути можливих напрямків руху
  Flag: boolean = false;

  implementation

uses
  UMain,UCamTimers,Urestaran;

var
  F: TextFile;
//  BitmapArr: TByteArr;

{$R *.dfm}

//конструктор
```

```
constructor TBitmapGrabEvent.Create(Index: Cardinal);
begin
  inherited Create;
  //індекс оброблювача
  EventIndex := Index;
  //аналіз кадру
  AnalyzFrame := false;
  //дебаг на льоту
  OnFlyDebug := false;
end;
//---------------------------------

//деструктор
destructor TBitmapGrabEvent.Destroy;
begin
  inherited Destroy;
end;
//---------------------------------

//аналіз захопленого бітмапа
procedure TBitmapGrabEvent.AnalyzBitmap(Bitmap: TCapturedBitmap);
var
  RazvertkaTlumitsya: TRazvertka;
  RazvertkaTlumitsyaConfig: TRazvertkaConfig;
  razriv_cnt,razriv_cnt_tlumitsya: integer; // кількість розривів
  UgliRazrivov,UgliRazrivovTlumitsya: TUgliRazrivov; //кути розривів
//  BitmapArr: TByteArr;
  y: Cardinal;
begin
  //якщо потрібно давати відео робосерверу
  if URoboRootServer.VideoInUse and (ActiveCamNum = EventIndex) then
  begin
    if not KadrSending then
    begin
      KadrFormating := true;
      Kadr.Assign(Bitmap);
      //KadrSizeWH := Bitmap.Width;
      KadrFormating := false;
    end;
  end;

  //аналіз одного кадру з виводом інформації про кластери
  if AnalyzFrame and not OnFlyDebug then
  begin
    USingleFrame.FrameForm.DebugFrame(Bitmap,true);
    AnalyzFrame := false;
  end;

  //аналіз потоку кадрів без висновку інформації про кластери
  if OnFlyDebug and not AnalyzFrame then
  begin
    USingleFrame.FrameForm.DebugFrame(Bitmap);
  end;

  //якщо потрібно взяти семпл
  if NeedAddSample then
  begin
    UAddSample.AddSampleForm.SampleImage.Picture.Assign(Bitmap);
    NeedAddSample := false;
    with AddSampleForm do
    begin
      SampleImage.Width := Bitmap.Width;
      SampleImage.Height := Bitmap.Height;
      {
      RadioTSample.Top := SampleImage.Height;
      RadioNotTSample.Top := SampleImage.Height;
      AddSampleButton.Top := SampleImage.Height + RadioTSample.Height;
      BrowseButton.Top := SampleImage.Height + RadioTSample.Height;
      }
```

```
      //height:=GroupBox1.Height+SampleImage.Height;
      //ObjectImage.Left:=SampleImage.Width;
      ObjectImage.Left:=0;
      GroupBox1.top:=SampleImage.Height;  //AddSampleForm.height
      //AddSampleForm.ClientWidth := SampleImage.Width+;
      if not(visible) then ShowModal;
    end;
  end;

  //якщо обрано камеру подія якого відбулося
  if (EventIndex = ActiveCamIndex) then
  begin
    PreviewWindow.ClientWidth := Bitmap.Width;
    PreviewWindow.ClientHeight := Bitmap.Height;
    PreviewWindow.Video.Picture.Bitmap.Assign(Bitmap);

   // UMain.MainForm.TLabel.Caption := 'немає!';
  end;

  //рух по полігону, обробляємо обрану камеру
  if GraphConfigExArr[EventIndex].CamFunc = CamFuncPolygonForw then
  begin
    //розгорнення
    razriv_cnt := RazvertkaArr[EventIndex].Klaster_Analiz(UgliRazrivov,Bitmap);
    //смуга
    Polosa(razriv_cnt,UgliRazrivov,UgliPolosi);
    //може бути T
    //if razriv_cnt = 2 then


    if recognition_run then
    begin
      QP.Attach(Bitmap);

      SetLength(BitmapArr,QP.Height);
      for y := 0 to High(BitmapArr) do
        SetLength(BitmapArr[y],QP.Width);
      ConvertBitmapToMonoChrome(QP,BitmapArr);
      //segment(BitmapArr);

      //TSampleRule(BitmapArr);

        RobotRecognition.BitmapReady:=true;
    end;
  end;

  //тлумиться
  if Restaran <> nil then
  if Restaran.tlumitsya = 1 then
  begin
    with RazvertkaTlumitsyaConfig do
    begin
      a := RazvertkaConfigArr[EventIndex].a;
      a_corr := RazvertkaConfigArr[EventIndex].a_corr;
      b := 10;
      y_offset := RazvertkaConfigArr[EventIndex].y_offset;
      klaster_size := RazvertkaConfigArr[EventIndex].klaster_size;
      porog := RazvertkaConfigArr[EventIndex].porog;
      step := RazvertkaConfigArr[EventIndex].step;
      fi_step := RazvertkaConfigArr[EventIndex].fi_step;
    end;

    RazvertkaTlumitsya := TRazvertka.Creat(RazvertkaTlumitsyaConfig);
    razriv_cnt_tlumitsya :=
RazvertkaTlumitsya.Klaster_Analiz(UgliRazrivovTlumitsya,Bitmap);

    if razriv_cnt_tlumitsya = 2 then
      Restaran.ugol_tlumitsya := (UgliRazrivov[0] + UgliRazrivov[1]) / 2;
```

```
    RazvertkaTlumitsya.Destroy;
  end;

  //простий рух по смузі
  if MoveForLineFlag then
  begin
    razriv_cnt := RazvertkaArr[EventIndex].Klaster_Analiz(UgliRazrivov,Bitmap);
    Polosa(razriv_cnt,UgliRazrivov,UgliPolosi);
    MoveForLine(UgliPolosi);
  end;
end;
//-----------------------------------

//Показати форму
procedure TPreviewWindow.ShowEx;
begin
  if GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Width <= 640 then
    PreviewWindow.ClientWidth :=
GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Width
  else
    PreviewWindow.ClientWidth := 640;
  if GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Height <= 480 then
    PreviewWindow.ClientHeight :=
GraphConfigExArr[ActiveCamIndex].GraphConfig.VCapMode.Height
  else
    PreviewWindow.ClientHeight := 480;

  Left := Screen.Width - Width;
  Top := 0;

  Show;
end;
//-----------------------------------

//Подія - зв'язок з камерою загублена
procedure TPreviewWindow.VideoCaptureDeviceLost(Sender: TObject);
begin
  ShowMessage('Зв'язок з відео пристроєм загублена!');
end;

initialization
  //QP2 := TQuickPixels.Create;
  QP := TQuickPixels.Create;
  AssignFile(F,'Preview.txt');
  Rewrite(F);

finalization
  QP.Destroy;
  CloseFile(F);

end.
```

Файл UCamTimers.pas – модуль встановлення таймерів камер

```
unit UCamTimers;

interface

uses
  Windows,URazvertka, UPreview, UTypeConst, SysUtils;

type
   //тип процедури спрацьовування таймера як метод класу
   //TTimeProc = procedure(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD) of object;

  //клас таймера камери
  TCamTimer = class
    private
      uidtimer: UINT; //ідентифікатор таймера
      TimerDelay: Cardinal; //період спрацьовування таймеа (мс)
      TimerIndex: Cardinal; //індекс таймера
    public
      Active: boolean; // чиактивний таймер
      Constructor Create(Index: Cardinal);
      Destructor Destroy(); override;
      procedure StartTimer(Delay: Cardinal);
      procedure StopTimer();
  end;

  //масив таймерів камер
  TCamTimerArr = array of TCamTimer;

var
  CamTimerArr: TCamTimerArr;//масив таймерів камер

//оброблювач таймерів
procedure TimeProc(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD); stdcall;

implementation

Constructor TCamTimer.Create(Index: Cardinal);
begin
  inherited Create;
  TimerIndex := Index;
  Active := false;
end;
Destructor TCamTimer.Destroy();
begin
  StopTimer();
  inherited Destroy();
end;
procedure TCamTimer.StartTimer(Delay: Cardinal);
begin
  TimerDelay := Delay;
  uidtimer := timeSetEvent(TimerDelay,1,@TimeProc,TimerIndex,1);
  Active := true;
end;
procedure TCamTimer.StopTimer();
begin
  timeKillEvent(uidtimer);
  Active := false;
end;
//оброблювач таймерів камер
procedure TimeProc(uID, uMsg: UINT; dwUser, dw1, dw2: DWORD); stdcall;
begin
  VideoCaptureArr[dwuser].CaptureFrame;
end;

end.
```

Файл UGraphConfig.pas – модуль налаштування камер

```
unit UGraphConfig;

{Модуль настроювання камери}

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, VCap, Buttons, ExtCtrls, ComCtrls, UTypeConst;


const
  MMInit_WrongDeviceNum = -1; //неправильний номер пристрою
  MMInit_WrongCompNum = -1; //неправильний номер компресора


type
  TGraphConfigForm = class(TForm)
    Label3: TLabel;
    VideoCompBox: TListBox;
    OKBitBtn: TBitBtn;
    GraphConfigGroupBox: TGroupBox;
    WantPreviewCheckBox: TCheckBox;
    PixelFormatComboBox: TComboBox;
    Label4: TLabel;
    CaptureFileNameEdit: TLabeledEdit;
    WantCaptureCheckBox: TCheckBox;
    CamFuncComboBox: TComboBox;
    Label1: TLabel;
    WantBitmapsCheckBox: TCheckBox;
    TimerDelayEdit: TLabeledEdit;
    procedure OKBitBtnClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure WantBitmapsCheckBoxClick(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  GraphConfigForm: TGraphConfigForm; //форма

  //індекс елемента масиву настроювань
  GraphConfigExIndex: Cardinal;

implementation

uses
  UMain;

{$R *.dfm}

//---------------------------------i

{при натисканні кнопки ОК відбувається зчитування настроювань,
заданих користувачем, і присвоєння їх  переданої змінної}
procedure TGraphConfigForm.OKBitBtnClick(Sender: TObject);
var
  i: integer; //лічильник
  SelVCompNum: integer; //номер обраного відео компресора
begin
  //заповнюємо змінну стану камери
  with GraphConfigExArr[GraphConfigExIndex].GraphConfig do
  begin
```

```
    //заповнюємо поле "відео компресор" ім'ям обраного відео компресора
    SelVCompNum := MMInit_WrongCompNum;
    for i := 0 to VideoCompBox.Count - 1 do
      if VideoCompBox.Selected[i] then
        SelVCompNum := i;

    //якщо не один компресор не обраний
    if SelVCompNum = MMInit_WrongCompNum then
      VComp := ''
    else
      VComp := VideoCompBox.Items[SelVCompNum];

    //потрібно записувати відео?
    if WantCaptureCheckBox.Checked then
      WantCapture := true
    else
      WantCapture := false;

    //за замовчуванням не потрібно зображення
    WantPreview := false;
    WantBitmaps := false;

    //потрібно показувати зображення
    if WantPreviewCheckBox.Checked then
    begin
      GraphConfigExArr[GraphConfigExIndex].ShowPreview := 1;
      WantPreview := true;
    end
    else
    begin
      GraphConfigExArr[GraphConfigExIndex].ShowPreview := 0;
    end;

    //потрібні кадри
    if WantBitmapsCheckBox.Checked then
    begin
     if WantPreview = false then
       WantPreview := true;
     WantBitmaps := true;
    end
    else
      WantBitmaps := false;

    //ім'я файлу в який записується відео
    CaptureFileName := CaptureFileNameEdit.Text;

    // функція камери
    GraphConfigExArr[GraphConfigExIndex].CamFunc := CamFuncComboBox.ItemIndex;

    //частота таймера
    GraphConfigExArr[GraphConfigExIndex].TimerDelay :=
StrToInt(TimerDelayEdit.Text);
  end;

  //вибираємо формат подання пікселя
  with PixelFormatComboBox do
  begin
    if Text = 'Визначається пристроєм' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pfDevice
    else if Text = '1 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf1bit
    else if Text = '4 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf4bit
    else if Text = '8 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf8bit
    else if Text = '15 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf15bit
    else if Text = '16 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf16bit
```

```
    else if Text = '24 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf24bit
    else if Text = '32 біт' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pf32bit
    else if Text = 'Налаштовуваний' then
GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat := pfCustom
    else GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat :=
pfDevice;
  end;
end;
//--------------------------------i

{Подія - перед появою форми
виводимо списки доступних
компресорів для стиску відео й відновлюємо галочки у відповідності
с переданими відео настроюваннями}
procedure TGraphConfigForm.FormShow(Sender: TObject);
var
  i: integer; //лічильник
  VideoCompList: TStringList; //аркуш доступних відео компресорів
begin
  Caption := 'Настроювання відео для ' +
GraphConfigExArr[GraphConfigExIndex].GraphConfig.VCapSource;

  //одержуємо список відео кодеків
  VideoCompList := GetVideoCompressorsList(true);

  //виводимо в компоненти отриману інформацію
  VideoCompBox.Items.Assign(VideoCompList);

  //жодна рядок не виділений
  VideoCompBox.ItemIndex := -1;

  //відновлюємо номер відео компресора
  for i := 0 to VideoCompBox.Count - 1 do
  begin
    if VideoCompBox.Items[i] =
GraphConfigExArr[GraphConfigExIndex].GraphConfig.VComp then
      VideoCompBox.ItemIndex := i;
  end;

  //відновлюємо галочки "Потрібно зображення"
  if GraphConfigExArr[GraphConfigExIndex].ShowPreview = 1 then
    WantPreviewCheckBox.Checked := true;
  if GraphConfigExArr[GraphConfigExIndex].ShowPreview = 0 then
    WantPreviewCheckBox.Checked := false;

  //відновлюємо галочку "Потрібно записувати відео"
  if GraphConfigExArr[GraphConfigExIndex].GraphConfig.WantCapture then
    WantCaptureCheckBox.Checked := true
  else
    WantCaptureCheckBox.Checked := false;

  //відновлюємо поле "потрібні кадри"
  if GraphConfigExArr[GraphConfigExIndex].GraphConfig.WantBitmaps then
    WantBitmapsCheckBox.Checked := true
  else
    WantBitmapsCheckBox.Checked := false;

  //відновлюємо формат пікселя
  case GraphConfigExArr[GraphConfigExIndex].GraphConfig.PixelFormat of
    pfDevice : PixelFormatComboBox.Text := 'Визначається пристроєм';
    pf1bit : PixelFormatComboBox.Text := '1 біт';
    pf4bit : PixelFormatComboBox.Text := '4 біт';
    pf8bit : PixelFormatComboBox.Text := '8 біт';
    pf15bit : PixelFormatComboBox.Text := '15 біт';
    pf16bit : PixelFormatComboBox.Text := '16 біт';
    pf24bit : PixelFormatComboBox.Text := '24 біт';
    pf32bit : PixelFormatComboBox.Text := '32 біт';
```

```
    pfCustom : PixelFormatComboBox.Text := 'Налаштовуваний';
  end;

  //відновлюємо ім'я файлу для запису відео  CaptureFileNameEdit.Text :=
GraphConfigExArr[GraphConfigExIndex].GraphConfig.CaptureFileName;


  // фукції камери
  CamFuncComboBox.ItemIndex := GraphConfigExArr[GraphConfigExIndex].CamFunc;

  //частота таймера
  TimerDelayEdit.Text :=
IntToStr(GraphConfigExArr[GraphConfigExIndex].TimerDelay);

end;

//подія - натискання на "потрібні кадри"
procedure TGraphConfigForm.WantBitmapsCheckBoxClick(Sender: TObject);
begin
  if WantBitmapsCheckBox.Checked then
  begin
    WantPreviewCheckBox.Enabled := true;
  end
  else
  begin
    WantPreviewCheckBox.Checked := false;
    WantPreviewCheckBox.Enabled := false;
  end;
end;

end.
```

Файл UCharAnalyz.pas - модуль розпізнавання образів з телекамери за допомогою нейроної мережі

```
unit UCharAnalyz;
// один з модулів розпізнавання образів, містить:
// -повну функція сегментації й допоміжні процедури
// -підготовку й запуск розпізнавання нейромережею
// -процентне розпізнавання й супутні функції


interface

uses Graphics,SysUtils,Math,UTypeConst,Windows,UQPixels,NeuralBaseComp,Classes,
NeuralBaseTypes;

const
  Icon = 0;
  IconNot = 1;
  min_size=10;

type

  TByteArr = array of array of integer;

  TRobotRecognition = class(TThread) //Потік розпізнання
    private


    public
      //BitmapArr: TByteArr;
      BitmapReady:boolean;
      RecType:integer; // вибір параметрів розпізнання
      procedure Execute(); override; //запуск потоку
  end;


  TBitmap = Graphics.TBitmap;


  //масив байт Nx
  //TByteArr = array of array of byte;


  //семпл
  TIcon = record
    Icon: TByteArr;
    IconName:string;
    IconType: Cardinal;

  end;


type
    //Tlables =array of byte;
    Tlables =array of integer;
    Tlables_cnt=array of word;
    // інформація про сегменти образів
    Tsegments = record
    x:integer; // геометричне положення сегмента
    y:integer;
    top:integer;
    buttom:integer;
    left:integer;
    right:integer;
    segment_type:integer; // тип об'єкта - присвоюється підпрограмою розпізнання
    size:integer;// площа об'єкта (для фільтра)
    end;
```

```
    //масив іконок
    TIconArr = array of TIcon;
    TsegmentArr=array of Tsegments;
var
  BitmapArr: TByteArr;
  RobotRecognition:TRobotRecognition;
  leter_types:array [0..6] of char;
  sort_segments:Tsegments;
  segments:TsegmentArr;
  IconArr: TIconArr; //масив семплів
  //параметри перетворення в семпл
  IconSize: Cardinal = 16;
  IConSize: Cardinal = 16;
  MinBPixelsPercent: Cardinal = 80;
  IconTypes:Cardinal = 0;
  F: TextFile;
  left,right,top,buttom:integer;
  neuro_answer:integer;
  lowerest:integer;

procedure segment(BitmapArr: TByteArr);

procedure reader(letters_count:integer);

procedure ConvertBitmapToMonoChrome(var QPSource: TQuickPixels; var BitmapArr:
TByteArr);

//процедура перетворить масив BitmapArr Nx байт (монохромний бітмап) у масив
//IconAtrr[0..IconSize - 1][0..IConSize - 1], розбиваючи на осередки масив
//BitmapArr, MinBPixelsPercent - мінімальне відсоток чорних пікселів в осередку,
//достатніх, щоб зробити чорним елемент меншого масиву IConArr
procedure BitmapToIcon(var BitmapArr: TByteArr; var Icon: TByteArr;
                       IconSize:Cardinal; IconSize: Cardinal;
                       MinBPixelsPercent: Cardinal);

// функція порівнює два масиви байт Nx і видає відсоток співпавших байт
function CompareIcons(Arr1: TByteArr; Arr2: TByteArr): Cardinal;

//додати семпл зазначеного типу
procedure AddSample(var BitmapArr: TByteArr; IconType: Cardinal);

//семпл букви Т рулить
function TSampleRule(var BitmapArr: TByteArr): boolean;

implementation

uses
  UMain,UAddSample,upreview;



// головна функція нитки розпізнавання образів:
// розпізнає весь екран цілком, як один образ
// або б'є по сегментах, і розпізнає кожний окремо

// Вона сама просить зробити їй масив бітмепа на наступній події захоплення
// зображення камерою, як тільки обробить  попередні дані
// тут же вважається й FPS
procedure TRobotRecognition.execute();
begin
  FreeOnTerminate := true;
while not(RobotRecognition.Terminated) do
begin
  if BitmapReady then
    begin

      recognition_run:=false;
      //debug:=false;
      BitmapReady:=false;
```

```
      if rectype =1 then
      begin
        segment(BitmapArr);

      end;
      if rectype =0 then
      begin

        TSampleRule(BitmapArr);

      end;
              tm_tick:=tm_tick+1;
    recognition_run:=true;
    end;
end;
end;

procedure ConvertBitmapToMonoChrome(var QPSource: TQuickPixels; var BitmapArr:
TByteArr);
var
  i,x,y: Integer;
  Pixel: Cardinal;
  Pixel, Pixel, Pixel: byte;
  PixelRes,oldPix,oldPixRes: Cardinal;
  oldPixSum:real;
  oldPixCnt:integer;
  Canvas: TCanvas;

begin

oldPixCnt:=0;
OldPix:=0;
oldPixRes:=1;

  Canvas:=AddSampleForm.SampleImage.Canvas;


  left:=QPSource.Width - 1;
  right:=0;
  top:=QPSource.Height - 1;
  buttom:=0;

  for y := 0 to QPSource.Height - 1 do
  begin

    for x := 0 to QPSource.Width - 1 do
    begin
      Pixel := QPSource.GetPixels24(x,y);
      {
      Pixel := Pixel shr 23;
      Pixel := Pixel shr 15;
      Pixel := Pixel shr 7;
      }
      Pixel := Pixel shr 23;
      Pixel:= Pixel shr 15;
      Pixel := Pixel shr 7;
      //PixelRes := (Pixel and Pixel) and Pixel;
      PixelRes := Pixel;// and Pixel ;

      BitmapArr[y,x] := PixelRes;

     if PixelRes=1 then
      begin
        if x>right then right:=x;
        if y>buttom then  buttom:=y;
        if x<left  then left:=x;
        if y<top  then top:=y;
      end;
```

```
      // що- те на подобі простенького фільтра

      //BitmapArr[y,x] := PixelRes and oldPixRes;

      //BitmapArr[y,x] := round(oldPixSum) and 1;
      if debug then if BitmapArr[y,x]<>OldPix then canvas.Pixels[x,y]:=ClRed;

      //oldPixSum:=abs(oldPixSum+( PixelRes-OldPix)/10);

      OldPix:=PixelRes;


      //oldPixRes:=round((oldPixRes+PixelRes)/10);
      //oldPixRes:=PixelRes;

      //QPDest.SetPixels1(j,i,PixelRes {* clWhite});
    end;

  end;


end;


// обчислення даних нейромережею
// і зняття даних
procedure NeuroCompute(Arr1: TByteArr);
  var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  cnt:integer;
  totle_layers,tmp,i:integer;
  true_exit:boolean;
  buff:string;
  Param,Param2:integer;

begin

  // вхідний параметр зняття даний з нейромережі
  Param:= 100-AddSampleForm.TrackBar1.Position;
  Param2:=AddSampleForm.TrackBar1.Position;

  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);


  // робимо вхідний вектор нейромережі з іконки (у модулів такий формат)
  cnt:=0;
  for x := 0 to Width - 2 do
    begin
      for y := 0 to Height - 2 do
        begin
      // багатошаровий
      if Arr1[y,x]=1 then  xInputVector[cnt] := 0 else xInputVector[cnt] := 1;
      //xInputVector[cnt] := Arr1[y,x];
      //Нейрона мережа Хопфілда
      //addsampleform.NeuralNetHopf1.Layers[1].Neurons[cnt].Output := Arr1[y,x];
            cnt:=cnt+1;
        end;

    end;

    neuro_answer:=0;
    // нейромережа робить прогін у прямому напрямку
    addsampleform.NeuralNetBP1.Compute(xInputVector);

    totle_layers:=AddSampleForm.NeuralNetBP1.LayerCount-1;
    buff:='';
```

```
    // знімаємо даний з нейромережі
    for i := 0 to length(xOutputVector)-1 do
    begin

//xOutputVector[i]:=AddSampleForm.NeuralNetBP1.LayersBP[tmp].Neurons[i].Output;

tmp:=round(AddSampleForm.NeuralNetBP1.LayersBP[totle_layers].Neurons[i].Output*1
00);



    // суть алгоритму така - якщо якийсь із вихідних нейронів більше якогось
    // числа (наприклад 0.9, а всі інші вектора менше 0.1 кожний
    // те тоді відповіддю вважається той нейрон, що 0.9
    // ці параметри задаються з форми плазуючої
    // числа 0.9 і 0.1 актуальні по експериментах, але некритично
    // збільшити до 0.8 і 0.2, далі буде розпізнавати всі підряд.

    // подумав ще небагато: можна взагалі шукати максимальна відповідь,
    // як це зроблено у відсотках = тоді буде краще небагато
    // хоча міняти параметр нижче 0.8 однаково небезпечно - значить щось не так
    // на вхід іде.
    if tmp>Param then neuro_answer:=i+1; // собственно 0.9
    buff:=buff+' '+inttostr(tmp)
    end;
    true_exit:=true;
    for i := 0 to length(xOutputVector)-1 do
    begin
    if (neuro_ answer-1)<>i then
      begin

tmp:=round(AddSampleForm.NeuralNetBP1.LayersBP[totle_layers].Neurons[i].Output*1
00);
           if tmp>Param2 then true_exit:=false; // а це 0.1
      end;
    end;
    if not(true_exit) then neuro_answer:=0; // якщо не зустріли нічого іншого
              // більше чим 0.1 те даємо відповідь про успішний розпізнання,
              // інакше говоримо про те, що цей образ провалився.

    //mainform.NeuroCount.Text:=inttostr(neuro_answer);

end;



// це один з інстументів переприсвоєння міток,
// друга частина написана в правилах

function get_parent(var lables:Tlables;var lables_cnt:Tlables_cnt;
new_cnt:integer;test:integer):integer;
begin
  if (lables[new_cnt]<>0)  then result:=get_parent(lables,lables_cnt,
lables[new_cnt],test)
  else result:= new_cnt;
end;


function normalize(var lables:Tlables;var lables_cnt:Tlables_cnt;
index_cnt:integer;new_cnt:integer):integer;
var i,j:integer;
begin
      if lables[new_cnt]<>0 then
//      if lables[index_cnt]
      //lables[index_cnt]<>0
//      if lables[index_cnt]<>0 then
normalize(lables,lables_cnt,lables[index_cnt],index_cnt);
      //lables[index_cnt]:=new_cnt;
```

```
//      lables[new_cnt]:=index_cnt;
      //lables_cnt[index_cnt]
      //lables_cnt[new_cnt]:=lables_cnt[new_cnt]+lables_cnt[index_cnt];
      //lables_cnt[index_cnt]:=0;
      {
      for i:=1 to index_cnt do
        if lables[i]<>0 then
          begin
          for j:=1 to index_cnt do
            begin
            if lables[lables[i]]<>0 then
              begin
              lables[i]:=lables[lables[i]]; // і перепризначуємо влучні
              end;
            end;
          end;
        }
//result:= lables;
end;

// сегментація масиву
// у коментариях: L,m -lables
// цей же алгоритм застосовується в матлабе BWLABEL,
// реалізація цілком може бути іншою BWLABEL

// по суті - прохід по рядах і присвоєння міток за правилами,
// зазначеним нижче, якщо мітка вже привласнена комусь а треба НЕЮ привласнити
// поточну, то пошук її батька, і присвоєння ім'я батька.
// так само захист від присвоєння батька на самого себе.

// У результаті роботи алгоритму - після першого ж проходу масиву
// всі мітки посилаються або один на одного або на батька - він посилається на 0
// нумерація міток починається з 2 (т.до споконвічно масив містить тільки 1 і 0)

procedure segment( BitmapArr: TByteArr);
var i,ii,j :integer;
    width,height,x,y:integer;
    index_cnt:integer;
    lables:Tlables; // мітки, що розставляються спочатку
    lables_cnt:Tlables_cnt; // площа міток
    real_lables:array of integer; // кожна мітка  - унікальний масив
    //debug:boolean;
    seg_debug:textfile;
    seg_debug_file_name:string;
    flag:boolean;
    middel:integer;

begin


  setlength(real_lables,0);
  seg_debug_file_name:='data\seg.txt';
  //debug:=true;
  index_cnt:=1;
  setlength(lables,2);
  setlength(lables_cnt,2);
  height:=length(BitmapArr);
  width:=length(BitmapArr[0]);


  // прохід по масиві
  // первинна установка міток по нижченаписаним правилах
  for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin

        if BitmapArr[i,j]>=1 then
            begin
```

```
            //   0       0
            // 0 1 ->  0 L

            if  (BitmapArr[i, j-1]=0) and
                (BitmapArr[ i-1,j]=0) then
                  begin
                  setlength(lables,length(lables)+1);
                  setlength(lables_cnt,length(lables_cnt)+1);
                  index_cnt:=index_cnt+1;
                  lables[index_cnt]:=0;
                  lables_cnt[index_cnt]:=1;
                  BitmapArr[i,j]:=index_cnt;
                  end;

            //   0       0
            // L 1 ->  L L

            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[ i-1,j]=0) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[i, j-1];
                  lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  end;

            //   L       L
            // 0 1 ->  0 L

            if  (BitmapArr[i, j-1]=0) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  //BitmapArr[i,j]:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j]);
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];
                  lables_cnt[index_cnt]:=lables_cnt[index_cnt]+1;
                  //lables_cnt[BitmapArr[i,j]]:=lables_cnt[BitmapArr[i,j]]+1;
                  end;

            //   L       L
            // L 1 ->  L L

            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]=BitmapArr[ i-1,j]) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];
                  lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  end;


            //   L       L
            // M 1 ->  M L (M:=L)
            // якщо L не вказує на 0, то пошук її самого далекого родича
            // якщо M це далекий родич L те не призначити M лінк на саму себе

            if  ((BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]<>BitmapArr[ i-1,j]) and
               //(lables[BitmapArr[i, j-1]]=0) and
                (BitmapArr[ i-1,j]>1)) then
                  begin
                  if (lables[BitmapArr[ i-1,j]]<>BitmapArr[i, j-1])then
                  begin
                  middel:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j],BitmapArr[i, j-1]);
                  if (BitmapArr[i, j-1]<>middel) then
                  begin
                  lables[BitmapArr[i, j-1]]:=middel;
```

```
                  end;
                  BitmapArr[i,j]:=middel;//BitmapArr[ i-1,j];
                  end
                  //set_parent(lables,lables_cnt,index_cnt,BitmapArr[ i-1,j]);
                  else
                  begin
                  BitmapArr[i,j]:=get_parent(lables,lables_cnt,BitmapArr[ i-
1,j],BitmapArr[i, j-1]);
                  end;

                  end;

            //   L       L   M<>0
            // M 1 ->  M L (L:=M)
            end; // if BitmapArr[i,j]=1 then

//            assignfile(seg_debug,seg_debug_file_name);
//            append(seg_debug);

//             write(seg_debug,inttostr(BitmapArr[i,j]));
//            closefile(seg_debug);
      end; // for j
//      assignfile(seg_debug,seg_debug_file_name);
//      append(seg_debug);

//      writeln(seg_debug,'');
//    closefile(seg_debug);
    end; //for i

    // отже нормалізуємо масив посилань лейблів один на одного
    // у результаті повинні  вийти набагато менше лейблів, але кожний буде
    // унікальним об'єктом, і піде в підпрограму розпізнання зі своїм номером
    if debug then
      begin
      assignfile(seg_debug,seg_debug_file_name);
      rewrite(seg_debug);
          for i:=1 to height-2 do
            begin
            for j:=1 to width-2 do
              begin
              //if BitmapArr[i,j]>1 then  write(seg_debug,inttostr(1)) else
write(seg_debug,inttostr(0))
              write(seg_debug,inttostr(BitmapArr[i,j]));
              end;
              writeln(seg_debug,'');
            end;
      writeln(seg_debug,'----------');

      for i:=1 to index_cnt do
        writeln(seg_debug,inttostr(i)+' '+inttostr(lables[i])+'
'+inttostr(lables_cnt[i]));
      writeln(seg_debug,'----------');

      end;


      for i:=1 to index_cnt do
        if lables[i]<>0 then
          begin
          for j:=1 to index_cnt do
            begin
            if lables[lables[i]]<>0 then
              begin

              lables[i]:=lables[lables[i]]; // і перепризначуємо влучні
              end;
            end;
          end;
```

```
    // можна сполучити з попереднім
    //
    // !!!!!!!!!!
    // на цьому ж етапі краще шукати границі кожного образа, що б потім
    // не проходити масив заданого стільки разів ,скільки образів знайдене

    // Плюси : кількість повних проходів скоротитися ґрунтовно:
    // Мінуси...м.. напевно їх немає.. так що треба буде зробити обов'язково.
    for i:=0 to  index_cnt do
      begin
        if lables[i]<>0 then
         begin
          lables_cnt[lables[i]]:=lables_cnt[lables[i]]+lables_cnt[i];
//підсумуємо ваги
         end;
      end;

       // Заглушка алгоритму нормалізації - тимчасово!
{
  for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
        if BitmapArr[i,j]>1 then
            begin
            if  (BitmapArr[i, j-1]>1) and
                (BitmapArr[i, j-1]<>BitmapArr[ i-1,j]) and
                //(lables[BitmapArr[i, j-1]]=0) and
                (BitmapArr[ i-1,j]>1) then
                  begin
                  BitmapArr[i,j]:=BitmapArr[ i-1,j];

                  //lables[BitmapArr[ i-1,j]]:=BitmapArr[i, j-1];
                  //lables_cnt[BitmapArr[i, j-1]]:=lables_cnt[BitmapArr[i, j-
1]]+1;
                  // спробуємо навпаки
                  lables[BitmapArr[i, j-1]]:=BitmapArr[ i-1,j];
                  lables_cnt[BitmapArr[ i-1,j]]:=lables_cnt[BitmapArr[ i-
1,j]]+1;

                  end;

            end;
      end;

 }


    if debug then
      begin
      for i:=1 to index_cnt do
        writeln(seg_debug,inttostr(i)+' '+inttostr(lables[i])+'
'+inttostr(lables_cnt[i]));
      writeln(seg_debug,'----------');

      end;

    // тепер зробимо прохід по масиві ще раз і перепишемо мітки
    // на нормалізовані, не хочеться витрачати час на ще один прохід

    for i:=1 to height-2 do
    begin
    for j:=1 to width-2 do
      begin
        if BitmapArr[i,j]<>0 then
          begin
           if lables[BitmapArr[i,j]]<>0 then
            begin
```

```
              BitmapArr[i,j]:=lables[BitmapArr[i,j]];

            end;
          end;

      end;
    end;

    if debug then
//        if true then
      begin
//             assignfile(seg_debug,seg_debug_file_name);
//      rewrite(seg_debug);

          for i:=1 to height-2 do
            begin
            for j:=1 to width-2 do
              begin
              write(seg_debug,inttostr(BitmapArr[i,j]));
              end;
              writeln(seg_debug,'');
            end;
//                              closefile(seg_debug);
      end;

    // SetLength(IconArr,Length(IconArr) + 1);

    // дивимося які мітки були унікальними
    if debug then
      begin
      write(seg_debug,'----дивимося які мітки були унікальними-');
      end;

      for i:=2 to index_cnt do
        if lables[i]=0 then
         begin
          if length(real_lables)>0 then
            begin
            // перевірка мітки на унікальність
            flag:=true;
            for j:=0 to (length(real_lables)-1) do
              begin
                if real_lables[j]=i then flag:=false;
              end;
              // якщо так, те унікальна
              if flag then
                begin
                  setlength(real_lables,length(real_lables)+1);
                  real_lables[length(real_lables)-1]:=i;
                end;
            end
            else
            begin
              setlength(real_lables,length(real_lables)+1);
              real_lables[length(real_lables)-1]:=i;
            end
         end;

      if debug then
         begin

         writeln(seg_debug,'--Унікальні мітки--');
           for i:=0 to length(real_lables)-1 do
           begin
             writeln(seg_debug,inttostr(real_lables[i])+'
'+inttostr(lables_cnt[real_lables[i]]));
           end;
```

```
     end;


// ну от, переходимо до суті - властиво до розпізнавання образів:
// на цьому етапі вже є матриця, у якій всі помічено не одиничками,
// а цифрами, до якому сигменту все це ставиться

// тепер треба чи подивитися достатня площа об'єкта
// (або навпаки занадто великувата)
// і послати його розпізнаватися, заодно заготовивши структуру з його даними


char_cnt:=1;
setlength(segments,0);
j:=0;
i:=0;
if length(real_lables)>0 then
begin
for i:=0 to length(real_lables)-1 do
 begin
  if  lables_cnt[real_lables[i]]>min_size then
  begin

        left:=Width - 2;
        right:=0;
        top:=Height - 2;
        buttom:=0;

      for y := 0 to height-2 do
        begin

        for x := 0 to Width - 2 do
          begin

          if BitmapArr[y,x]=real_lables[i] then
            begin
            if x>right then right:=x;
            if y>buttom then  buttom:=y;
            if x<left  then left:=x;
            if y<top  then top:=y;
            end;
          end;
        end;

  //writeln(seg_debug,inttostr(real_lables[i]));
  //top:=0;buttom:= height-2;left:=0;right:= width-2;

  if (right<>0) and (buttom<>0) then
  begin
    setlength(segments,(length(segments)+1));


    // пропускаємо через підпрограму розпізнання
    imFeatures(BitmapArr,real_lables[i]);

    // і так само пропускаємо  через нейромережу
    NeuroCompute(BitmapArr3);
    segments[j].x:=segment_X;
    segments[j].y:=segment_Y;
    segments[j].size:=lables_cnt[real_lables[i]];
    segments[j].segment_type:=neuro_answer;
    j:=j+1;
  end;
  end;
 end;

 // що -те робимо з типами образів, наприклад читаємо або
 // виставляємо прапорці для інший програми робота.
 if (( j-1)>=0)and(j=length(segments)) then   reader( j-1);
```

```
     end;



    // дебаг закінчився
    if debug then
      begin
      closefile(seg_debug);
      end;

end;


procedure reader(letters_count:integer);
var i,j,ii:integer;
begin
          // прочитаний текст - для початку опустошаємо усе з попереднього кроку
          AddSampleForm.reader.Text:='';
         // сортуємо букви
{    if letters_count>1 then
    begin
    for i := letters_count downto 0 do
      begin
   //   if debug then writeln(F,inttostr(segments[i].x)+'
'+inttostr(segments[i].segment_type));
      for ii := 0 to i do
      if segments[ii].x > segments[ii+1].x then
      begin
       sort_segments := segments[ii];
       segments[ii] := segments[ii+1];
       segments[ii+1] := sort_segments;
       end;
      end;
    end;

 }

     for i:=0 to letters_count do
     begin
     if (segments[i].segment_type<length(leter_types))and
(segments[i].segment_type>0) then

AddSampleForm.reader.text:=AddSampleForm.reader.Text+leter_types[segments[i].seg
ment_type];
     end;


     if AddSampleForm.ComboBox2.ItemIndex=1 then
       SayText(AddSampleForm.reader.Text);

end;


//семпл букви Т рулить
// стара функція для розпізнання
// порівнює дві матриці (одна з екрана, інша з масиву еталонів)
// результат дає якщо кількість що збіглися пікселей більше якогось відсотка
// а так само якщо воно найбільше із всіх що збіглися

// по експериментах - нейромережа працює луше від 10% до 30% по цьому
// далі використовувати процентрое порівняння
// практично ні до чого
function TSampleRule(var BitmapArr: TByteArr): boolean;
var
  Icon: TByteArr;
  y,i,j: Integer;
  MaxCompareTPercent: Cardinal;
  MaxCompareNotTPercent: Cardinal;
```

```
  CurComparePercent: Cardinal;
  iconFind:integer;
begin
  Result := false;
  SetLength(Icon,IConSize);
  for y := 0 to IconSize - 1 do
    SetLength(Icon[y],IconSize);


    // якась заглушка.. чи не знаю потрібна чи зараз ні
    if  right<>0 then
  begin
  // перетворимо ВЕСЬ екран в іконку (з першого білого, до останнього білого
  // пікселя.
 imFeatures(BitmapArr,1);

  if length(BitmapArr3)>1 then
 Icon:=BitmapArr3;


  // для процентного порівняння
  MaxCompareTPercent := 0;
  MaxCompareNotTPercent := 0;
  CurComparePercent := 0;


  {
  writeln(F,'--Бітмап--');
  for i := 0 to IconSize - 1 do
  begin
    for j := 0 to IconSize - 1 do
      write(F,Icon[i,j]);
    writeln(F);
  end;
  writeln(F);
  }

  //Порівнюємо сіткою
  NeuroCompute(Icon);


  // порівнюємо по пікселям що співпали
  iconFind:=0;
  for y := 0 to High(IconArr) do
  begin
    CurComparePercent := CompareIcons(Icon,IconArr[y].Icon);
    {
    if (IconArr[y].IconType = Icon) and (CurComparePercent > MaxCompareTPercent)
then
      MaxCompareTPercent := CurComparePercent;
    if (IconArr[y].IconType = IconNot) and (CurComparePercent >
MaxCompareNotTPercent) then
      MaxCompareNotTPercent := CurComparePercent;
    }
     if CurComparePercent > MaxCompareTPercent then
     begin
     MaxCompareTPercent := CurComparePercent; iconFind:=IconArr[y].IconType;
     end;

  end;
  if MaxCompareTPercent < 80 then iconFind:=0;
  //MainForm.Caption := IntToStr(iconFind);
  //MainForm.Caption := IntToStr(MaxCompareTPercent);
  //MainForm.Caption := '-no-';
//  if (MaxCompareTPercent > MaxCompareNotTPercent) and (MaxCompareTPercent >
80) then
 //   begin {MainForm.Caption := 'OK';} Result := true; end;
 end;
end;
```

```
// генеруємо іконку заданого розміру з масиву
procedure AddSample(var BitmapArr: TByteArr; IconType: Cardinal);
var
  Icon: TByteArr;
  x,y: Integer;
  canvas:tcanvas;
begin
  SetLength(IconArr,Length(IconArr) + 1);

  SetLength(Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Icon[y],IconSize);

  SetLength(IconArr[High(IconArr)].Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(IconArr[High(IconArr)].Icon[y],IconSize);

  //BitmapToIcon(BitmapArr,Icon,IconSize,IconSize,MinBPixelsPercent);

  //1canvas:=AddSampleForm.cellImage.canvas;

  for y := 0 to High(Icon) do
    for x := 0 to High(Icon[Low(Icon)]) do
    begin

      IconArr[High(IconArr)].Icon[y,x] := BitmapArr3[y,x];
      if Icon[y,x]=1 then


    end;

  IconArr[High(IconArr)].IconType := IconType;
end;

// функція порівнює два масиви байт Nx і видає відсоток співпавших байт
function CompareIcons(Arr1: TByteArr; Arr2: TByteArr): Cardinal;
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  cnt:integer;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);


//  cnt:=0;

  if  debug  then
  begin
  for x := 0 to Width - 2 do
    begin
      for y := 0 to Height - 2 do
         write(F,Arr1[y,x]);
      write(F,' ');
      for y := 0 to Height - 2 do
        write(F,Arr2[y,x]);
      writeln(F,' ');
    end;
        writeln(F,' ');
  end;
  for y := 0 to Height - 2 do
    for x := 0 to Width - 2 do
    begin
      if Arr1[y,x]=1 then
      //xInputVector[cnt] := 1
      //Нейрона мережа Хопфілда
```

```
      //addsampleform.NeuralNetHopf1.Layers[1].Neurons[cnt].Output := 1
      else
      //xInputVector[cnt] := 0;
      // Нейрона мережа Хопфілда
      //addsampleform.NeuralNetHopf1.Layers[1].Neurons[cnt].Output := -1;

      cnt:=cnt+1;

      if Arr1[y,x] = Arr2[y,x] then
        SamePixels := SamePixels + 1;
    end;
    //addsampleform.NeuralNetHopf1.Calc;
    Result := Round(SamePixels * 100 / (Width * Height));
end;


// Ця процедура використовувалася раніше
// зараз уже не потрібна! - ніде не викликається

//процедура перетворить масив BitmapArr Nx байт (монохромний бітмап) у масив
//IconAtrr[0..IconSize - 1][0..IConSize - 1], розбиваючи на осередки масив
//BitmapArr, MinBPixelsPercent - мінімальне відсоток чорних пікселів в осередку,
//достатніх, щоб зробити чорним елемент меншого масиву IConArr
procedure BitmapToIcon(var BitmapArr: TByteArr; var Icon: TByteArr;
                           IconSize: Cardinal ; IconSize: Cardinal ;
                           MinBPixelsPercent: Cardinal );


var
  i,j,x,y,bx,by,ix,iy: Integer; //Лічильники
  CSize, CSize: Integer; //Розміри осередку
  BPorog: Cardinal; //поріг чорного
  BPixelsInCell: Cardinal; // кількість чорних пікселей в осередку
  xLeft, xRight, yTop, yBottom : integer;           // абс. коорд. образа
  line:string;
  const color:integer = 0;
begin


  // перетворимо трохи пікселей на ділянці в один осередок, зчитуємо колір
  CSize := Floor(Length(BitmapArr) / IconSize);
  CSize := Floor(Length(BitmapArr[Low(BitmapArr)]) / IconSize);

  Bporog := Round(CSize * CSize / 100 * MinBPixelsPercent);

  bx := 0;
  by := 0;
  BPixelsInCell := 0;

  for iy := 0 to IconSize - 1 do
    for ix := 0 to IconSize - 1 do
    begin
      for y := by to by + CSize - 1 do
        for x := bx to bx + CSize - 1 do
          if BitmapArr[y,x] = 0 then
            BPixelsInCell := BPixelsInCell + 1;
      if BPixelsInCell > BPorog then
        Icon[iy,ix] := 0
      else
        ICon[iy,ix] := 1;
//        line:=line+inttostr(ICon[iy,ix]);
      BPixelsInCell := 0;
      bx := bx + CSize;
      if Round(bx / CSize) >= IConSize then
      begin
        bx := 0;
        by := by + CSize;
      end;
```

```
    end;

end;

initialization
  AssignFile(F,'Sampledebug.txt');
  Rewrite(F);
  leter_types[0]:=' ';
  leter_types[1]:='П';
  leter_types[2]:='И';
  leter_types[3]:='М';
  leter_types[4]:='Л';
  leter_types[5]:='О';
  leter_types[6]:='Д';

finalization
  CloseFile(F);


end.
```

Файл UAddSample.pas – обробка розпізнаного образу

```
unit UAddSample;
// один з модулів розпізнавання образів, містить:
// -підготовку образа до розпізнання, з обчисленням інваріантів і поворотом
// -читання, запис та інші операції з Базою даних образів
// -навчання нейромережі
// -збереження й запис вагових коефіцієнтів нейромережі
// -автоматичний підбор коефіцієнтів нейромережі (не впевнений що це не марення)
// -захоплення й додавання нового образа в БД


// навчання нейромережі може проиходить досить довгий час
// змінювана цифра - це середньоквадратична помилка:
// на практиці,  якщо вона скакає кілька мінут в однієї й тої ж величини
// значить пійманий локальний мінімум

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, ExtCtrls,UCharAnalyz,UQPixels, Grids, ValEdit,math,
  NeuralBaseComp, NeuralBaseTypes, Spin, ComCtrls;

type
  TAddSampleForm = class(TForm)
    SampleImage: TImage;
    SampleOpenDialog: TOpenDialog;
    SampleSaveDialog: TSaveDialog;
    GroupBox1: TGroupBox;
    Label2: TLabel;
    Label3: TLabel;
    matrix_size: TEdit;
    matrix_size: TEdit;
    NumIcons: TEdit;
    Button1: TButton;
    IconTypeSet: TComboBox;
    Label1: TLabel;
    AddSampleButton: TButton;
    BrowseButton: TButton;
    SaveSampleButton: TButton;
    ObjectImage: TImage;
    cellImage: TImage;
    Button2: TButton;
    Label4: TLabel;
    Label5: TLabel;
    shir: TLabel;
    hjkhjk: TLabel;
    Celx: TLabel;
    sdf: TLabel;
    vis: TLabel;
    Label6: TLabel;
    cely: TLabel;
    Label7: TLabel;
    Label8: TLabel;
    ComboBox1: TComboBox;

    NeuralNetHopf1    : TNeuralNetHopf;
    NeuralNetBP1: TNeuralNetBP;
    prbEpoch: TProgressBar;
    speEpochCount: TSpinEdit;
    Label9: TLabel;
    sttError: TLabel;
    Button3: TButton;
    neroIMp: TEdit;
    neroAlfa: TEdit;
    neroRate: TEdit;
```

```
    Label10: TLabel;
    Button4: TButton;
    NeuralNetExtended1: TNeuralNetExtended;
    Button5: TButton;
    Button6: TButton;
    Button7: TButton;
    Button8: TButton;
    Button9: TButton;
    reader: TEdit;
    ComboBox2: TComboBox;
    recType: TComboBox;
    TrackBar1: TTrackBar;
    param: TLabel;
    Timer1: TTimer;
    lFPS: TLabel;
    Label11: TLabel;
    Label12: TLabel;
    Label13: TLabel;
    Label14: TLabel;
    Label15: TLabel;
    procedure BrowseButtonClick(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure SaveSampleButtonClick(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure Button2Click(Sender: TObject);
    procedure NeuralNetBP1EpochPassed(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure NeuralNetExtended1EpochPassed(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);
    procedure Button9Click(Sender: TObject);
    procedure TrackBar1Change(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;
      procedure imFeatures(BitmapArr: TByteArr; pixeltype:integer);

var
  AddSampleForm: TAddSampleForm;
  sample_db,n_debug:textfile;
  sample_db_file_name:string ='data\config.db';
  neuro_debug_file_name:string ='data\debug.txt';
  neuro_teach_log_file_name:string ='data\log.txt';
  neuro_weight_file_name:string ='data\neuro.db';
  BitmapArr3:TByteArr;
  recognition_run:boolean=false;
  xVector: TVectorInt;
  xOutputVector: TVectorFloat;
  xInputVector: TVectorFloat;   // Вхідний вектор
  segment_X,segment_Y:integer;
  char_cnt,x_,y_:cardinal;
  debug:boolean =false;
  tm_tick:integer=0;

  implementation

uses upreview,umain;
{$R *.dfm}


// щось для роботи з картинками без камери
procedure TAddSampleForm.BrowseButtonClick(Sender: TObject);
```

```
var
  Bitmap: Tbitmap;
begin
  if SampleOpenDialog.Execute then
  begin
    Bitmap := Tbitmap.Create;
    Bitmap.LoadFromFile(SampleOpenDialog.FileName);
    SampleImage.Width := Bitmap.Width;
    SampleImage.Height := Bitmap.Height;
  //  RadioTSample.Top := SampleImage.Height;
  //  RadioNotTSample.Top := SampleImage.Height;
    //AddSampleButton.Top := SampleImage.Height ;//+ RadioTSample.Height;
    BrowseButton.Top := SampleImage.Height;// + RadioTSample.Height;
    SampleImage.Picture.Assign(Bitmap);
    Bitmap.Destroy;
  end;
end;



// функція обробки бінарного масиву
// 1. обчислення центра мас об'єкта
// 2. обчислення орієнтації об'єкта
// 3. поворот об'єкта
// 4. вихоплювання об'єкта
// 5. вжимання об'єкта до іконки
// багато частин можна прискорити.
procedure imFeatures(BitmapArr: TByteArr;pixeltype:integer);
var x,y,i,j,xc,yc:integer;
    N,sumx,sumy:cardinal;
    buff:integer;
    max_x,max_y:integer;
    canvas: tcanvas;
    tmp,O,SumUx,SumUy,SumUxy,Ux,Uy,Uxy,C:real;
    r:single;
    s,ss:extended;
    BitmapArr4,BitmapArr2: TByteArr;
    al:real;
    x_new,y_new:integer;
    actual_min_x,actual_min_y,actual_max_x,actual_max_y:integer;
    Bporog,BPixelsInCell,ix,iy,new_size_x,new_size_y,by,bx:integer;
    dx,dy,celx,cely:real;
    line:string;
    longest:integer;
    offset:cardinal;

const MinBPixelsPercent =10;

begin
  max_x:=Length(BitmapArr[0])-1;max_y:=Length(BitmapArr)-1;
  sumx:=0;sumy:=0;N:=0;



      // малюємо зелененьку рамочку
    if debug then
    begin
      if AddSampleForm.visible then
        begin
        Canvas:=AddSampleForm.SampleImage.Canvas;
        Canvas.Pen.Color := clGreen;
        Canvas.Pen.Width := 1;

        Canvas.MoveTo(Round(left),Round(top));
        Canvas.LineTo(Round(right),Round(top));
        Canvas.LineTo(Round(right),Round(buttom));
        Canvas.LineTo(Round(left),Round(buttom));
        Canvas.LineTo(Round(left),Round(top));
//        Canvas.MoveTo(Round(x - 10),Round(y - 10));
```

```
        end;
    end;


  // Алгоритм обчислення центра мас образа
   for y:=top to buttom do
    for x:=left to right do
      begin
      //BitmapArr2[y,x]:=0;
      buff:=BitmapArr[y,x];
      if buff=pixeltype then buff:=1 else buff:=0;
      sumx:=sumx+( x-left)*buff;
      sumy:=sumy+( y-top)*buff;
      N:=N+buff;
      end;


  // одержали координати центра мас образа
  xc:=left+round(sumx/N);
  yc:=top+round(sumy/N);

  // для сегментації
  segment_X:=xc;
  segment_Y:=yc;

    // малюємо хрестик там, де визначився центр мас
    if debug then
    begin
      if AddSampleForm.visible then
        begin
        Canvas:=AddSampleForm.SampleImage.Canvas;
        x:=xc;
        y:=yc;
        Canvas.Pen.Color := clRed;
        Canvas.Pen.Width := 2;

        Canvas.MoveTo(Round(x - 10),Round(y + 10));
        Canvas.LineTo(Round(x + 10),Round(y - 10));
        Canvas.MoveTo(Round(x - 10),Round(y - 10));
        Canvas.LineTo(Round(x + 10), Round(y + 10));
        end;
    end;


    // обчислення декількох допоміжних величин
    SumUx:=0;SumUy:=0;
    for y:=top to buttom do
      for x:=left to right do
        begin
        buff:=BitmapArr[y,x];
        if buff<>pixeltype then buff:=0 else buff:=1;
        SumUx:=SumUx+buff*sqr( x-xc);
        SumUy:=SumUy+buff*sqr( y-yc);
        SumUxy:=SumUxy+buff*( y-yc)*( x-xc);
      end;
    Ux:=1/12+SumUx*1/N; Uy:=1/12+SumUy*1/N; Uxy:=1/12+SumUxy*1/N; C:=sqrt(sqr(
Ux-Uy)+4*sqr(Uxy));

    //Обчислюємо орієнтацію об'єкта
    if Uy>Ux then
      begin
      O:=180/pi*ArcTan(( Uy-Ux+C)/(2*Uxy));
      //tmp:=( Uy-Ux+C)/(2*Ux*Uy);
      end
      else
      begin
      O:=180/pi*ArcTan((2*Uxy)/( Ux-Uy+C));
      //tmp:=(2*Uxy)/( Ux-Uy+C);
```

```
      end;


      // малюємо напрямок орієнтації об'єкта
  if debug then
    begin
        Canvas.Pen.Color := clRed;
        Canvas.Pen.Width := 5;

        Canvas.MoveTo(xc,yc);
        x:= trunc(( right-xc) * cos(O/180*pi)+xc);
        y:= trunc(( buttom-yc) * sin(O/180*pi)+yc);
        Canvas.LineTo(x,y);


    end;

   // Повертаємо об'єкт щодо точки центра мас, на кут орієнтації
   // попутно визначаємо точне розташування повернутого образа

   actual_min_x:=max_x;
   actual_min_y:=max_y;
   actual_max_x:=0;
   actual_max_y:=0;


    // оскільки невідомо - де в об'єкта що стирчить - споконвічно робимо
    // квадратну матрицю - довгої з діагональ первісної
    if ( right-left)>( buttom-top) then
      begin
      longest:=round(1.5*( right-left));

      end else
      begin
      longest:=round(1.5*( buttom-top));
      end;

      SetLength(BitmapArr2,round(longest)+1);
      for y := 0 to High(BitmapArr2) do
        SetLength(BitmapArr2[y],round(longest)+1);


   O:=-O*pi/180;
   for y := top to buttom do
   begin
     for x := left to right do
     begin
       if BitmapArr[y,x]=pixelType then
         begin
         al:=arctan2((y - yc), (x - xc));
         r := sqrt(sqr(x - xc) + sqr(y - yc));
         //x_new:= trunc(xc + r * cos(al + O));
         //y_new:= trunc(yc + r * sin(al + O));
         x_new:= trunc(r * cos(al + O)+longest/2);
         y_new:= trunc(r * sin(al + O)+longest/2);

         if x_new<actual_min_x then actual_min_x:=x_new;
         if y_new<actual_min_y then actual_min_y:=y_new;
         if x_new>actual_max_x then actual_max_x:=x_new;
         if y_new>actual_max_y then actual_max_y:=y_new;
         if (y_new<longest)and(x_new<longest) and(y_new>0) and (x_new>0) then
BitmapArr2[y_new,x_new]:=1;
         //Canvas.Pixels[x_new,y_new]:= clGreen;
         end;
     end;
   end;
```

```
    // прощай квадратна матриця, довгої з діагональ початкової- вихоплюємо
    // заново повернений образ
    new_size_x:=actual_max_ x-actual_min_x;
    new_size_y:=actual_max_ y-actual_min_y;



    SetLength(BitmapArr4,new_size_y+1);
    for y := 0 to High(BitmapArr4) do
      SetLength(BitmapArr4[y],new_size_x+1);

    if Debug then
    begin
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=new_size_y;
      AddSampleForm.ObjectImage.Width :=new_size_x;

    end;

   // перетворюємо уже повернений масив із квадратного в прямокутний по границі
   // потім це можно буде зробити в наступному кроці, поки однаково він
   // щось малює - те можна й залишити тут

    for y := 0 to new_size_y do
      begin
      for x := 0 to new_size_x do
        begin

          ///// !!!!!!!!!!!!!!!!!!!!!!!!

          if ((y+actual_min_y)<(longest)) and ((x+actual_min_x)<(longest)) and
((y+actual_min_y)>0) and ((x+actual_min_x)>0)  then
          BitmapArr4[y,x]:=BitmapArr2[y+actual_min_y,x+actual_min_x];
          if debug then if BitmapArr4[y,x]>0 then Canvas.Pixels[x,y]:=clRed else
Canvas.Pixels[x,y]:= clWhite;
        end;
      end;

    if debug then
    begin
     //AddSampleForm.ObjectImage.Width :=new_size_x;
     //AddSampleForm.ClientWidth := 640;
    end;


    SetLength(BitmapArr3,IConSize+1);
    for y := 0 to High(BitmapArr3) do
      SetLength(BitmapArr3[y],IconSize+1);


    // знаходимо відповідність між осередком іконки масивом
    if new_size_x> IconSize then
      celx:=new_size_x/IconSize else celx:=1;

    if new_size_y> IconSize then
      cely:=new_size_y/IconSize else cely:=1;


      // якась інформація про образ
    if debug then
      begin
       AddSampleForm.shir.Caption:=inttostr(new_size_x);
       AddSampleForm.celx.Caption:=inttostr(round(celx));
       AddSampleForm.vis.Caption:=inttostr(new_size_y);
       AddSampleForm.cely.Caption:=inttostr(round(cely));
      end;

    if debug then
      begin
```

```
      offset:=round((6*IconSize+1)*(char_ cnt-1)); //
      AddSampleForm.cellImage.width:=(6*IconSize+1)*5;//(6*IconSize+1) +offset;
      AddSampleForm.cellImage.height:=(6*IconSize+1);

      end;


    if debug then
      begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;//Canvas.FillRect(Canvas.ClipRect);
      for x_:=0 to IconSize+offset do

       begin
       Canvas.MoveTo(Round(6*x_),Round(0));
       Canvas.LineTo(Round(6*x_),Round(6*IconSize));
      end;
      for y_:=0 to IconSize do
       begin
       Canvas.MoveTo(round(0),Round(6*y_));
       Canvas.LineTo(Round(6*IconSize+offset),Round(6*y_));
      end;
      end;

      // Алгоритм зменшення зображення  - всі параметри плаваючі

          if debug then    Canvas.Brush.Color := clBlack;

    Bporog := Round(celx * cely / 100 * MinBPixelsPercent);


    dy:=0;
    dx:=0;
    BPixelsInCell := 0;


    for iy := 0 to IconSize - 1 do
      begin

      for ix := 0 to IconSize - 1 do
        begin
         dy:=0;dx:=0;
         while (dy+iy*cely)<((iy+1)*cely) do
          begin
          while (dx+ix*celx)<((ix+1)*celx) do
            begin
                  if (round(dy+iy*cely)<new_size_y) and
(round(dx+ix*celx)<new_size_x) then
                  if BitmapArr4[round(dy+iy*cely),round(dx+ix*celx)] = 1 then
BPixelsInCell := BPixelsInCell + 1;
                  dx:=dx+1;
            end;
            dy:=dy+1;dx:=0;;
          end;

        if BPixelsInCell > BPorog then
          begin  BitmapArr3[iy,ix] := 0; end
          else
          begin  BitmapArr3[iy,ix] := 1; if debug then
canvas.FloodFill(ix*6+3+offset,iy*6+3,canvas.pixels[ix*6+3+offset,iy*6+3],fsSurf
ace);  end;
         BPixelsInCell:=0;


        end;
      end;

    // ну от і все - іконка готова - тепер або розпізнаємо її або
    // додаємо в БД.
```

```
    if debug then
    begin
    for y := 0 to IconSize - 1 do
    begin
      for x := 0 to IconSize - 1 do
         write(F,BitmapArr3[y,x]);
      writeln(F,' ');
    end;
        writeln(F,'--------------');
    end;

end;

// додавання нового семпла - перетворення його в іконку, додавання в масив
procedure TAddSampleForm.AddSampleButtonClick(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  Icon: TByteArr;
  y: Cardinal;
  IconType: Cardinal;

begin
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);

  SetLength(BitmapArr,QP.Height);
  for y := 0 to High(BitmapArr) do
    SetLength(BitmapArr[y],QP.Width);

    IconType:=IconTypeSet.ItemIndex;
{  if RadioTSample.Checked then
    IconType := Icon
  else
    IconType := IconNot;
 }
  //IconTypes:=IconTypes+1;
  ConvertBitmapToMonoChrome(QP,BitmapArr);
  imFeatures(BitmapArr,1);
  AddSample(BitmapArr,IconType); // реально використовується BitmapArr3

  //Close;
end;

//подія - натискання "Зберегти"
procedure TAddSampleForm.SaveSampleButtonClick(Sender: TObject);
begin
  if SampleSaveDialog.Execute then
  begin
    SampleImage.Picture.Bitmap.SaveToFile(SampleSaveDialog.FileName);
  end;
end;
// читання іконки з файлу
function ReadIcon:TByteArr;
var
  x,y: Integer;
  Arr1:TByteArr;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  buff:string;
  buff2:char;
  cnt:integer;

begin
  SamePixels := 0;
  Width := IconSize;
  Height := IconSize;
```

```
  SetLength(Arr1,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Arr1[y],IconSize);

  SetLength(IconArr[High(IconArr)].Icon,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(IconArr[High(IconArr)].Icon[y],IconSize);

  cnt:=0;
  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        read(sample_db,buff2);
        Arr1[x,y]:=strtoint(buff2);
            if Arr1[x,y] = 1 then
            begin

            xVector[cnt] := 2;
            xInputVector[cnt] := 0;
            end
            else
            begin
            xVector[cnt] := -1;
            xInputVector[cnt] := 1;
            end;
          cnt:=cnt+1;
        //write(F,Arr1[x,y]);
      end;
      //WRITELN(f);
      readln(sample_db,buff);
    end;

  //NeuralNetHopf1.AddPattern(xVector);
  result:=Arr1;

end;


// подія відкриття форми, завантаження семплів з файла, ініціалізація й т.п.
procedure TAddSampleForm.FormShow(Sender: TObject);

  var i1,i,j,jj:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff:string;
    buff_file:textfile;
    max_icon_type:integer;

begin
  AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);


  matrix_size.text:=inttostr(IconSize);
  matrix_size.text:=inttostr(IconSize);



  if FileExists(sample_db_file_name) then
    begin
      assignfile(sample_db,sample_db_file_name);
      assignfile(n_debug,neuro_debug_file_name);
      rewrite(n_debug);
      reset(sample_db);

      //IconTypes:=2;
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
```

```
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconTypes:=strtoint(buff);//eIconTypes.Text:=buff;
      readln(sample_db,buff);
SetLength(IconArr,strtoint(buff)+1);NumIcons.Text:=buff;
      // вхідний вектор Нейрона мережа Хопфілда - нейронів стільки ж , скільки й
точок в іконці
      SetLength(xVector, IconSize * IconSize);
      // вихідний вектор багатошарової - дорівнює числу образів
      //IconTypes:=3;
      SetLength(xOutputVector, IconTypes);

      // вхідний вектор багатошарової - дорівнює числу нейронів
      SetLength(xInputVector, IconSize * IconSize);


      //  SetLength(xInputVector, 5);
      //  SetLength(xOutputVector, 1);

      // настроювання нейроної мережі Хопфілда

      AddSampleForm.NeuralNetHopf1.LayerCount:=1;
      AddSampleForm.NeuralNetHopf1.InputNeuronCount:=IconSize * IconSize;

      //NeuralNetExtended1.InputFieldCount:=IconSize * IconSize;
      {
      NeuralNetExtended1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNetExtended1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNetExtended1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      }



      // настроювання багатошарової
      //NeuralNetBP1.LayerCount:=2;
      //NeuralNetBP1.LayersBP[0].NeuronCount:=IconSize * IconSize;
      //NeuralNetBP1.LayersBP[1].NeuronCount:=IconSize * IconSize;
      //NeuralNetBP1.LayersBP[1].NeuronCount:=IconTypes;
      //NeuralNetBP1.LayersBP[2].NeuronCount:=IconTypes;

      NeuralNetBP1.ResetPatterns;




      for i:=0 to High(IconArr) do
        begin
        readln(sample_db,buff); IconArr[i].IconType:=strtoint(buff)+1;


        for j:=1 to IconTypes do
          begin
          if j<>IconArr[i].IconType then xOutputVector[ j-1]:=0 else
xOutputVector[ j-1]:=1;
          end;



        IconArr[i].Icon:=ReadIcon;
        // додаємо вектор навчання нейроної мережі Хопфілда
        AddSampleForm.NeuralNetHopf1.AddPattern(xVector);

        // Додаємо вектор багатошарової мережі
        //SetLength(xInputVector, 100);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        for i1:=0 to length(xInputVector) do
        write(n_debug,inttostr(round(xInputVector[i1])));
        writeln(n_debug,'');
        // NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
```

```
        //SetLength(xInputVector, 256);
        readln(sample_db,buff);
        end;
        writeln(n_debug,'-----');

        {
        readln(sample_db,buff); IconArr[0].IconType:=strtoint(buff);
        xOutputVector[0]:=0;  xOutputVector[1]:=1;
        IconArr[0].Icon:=ReadIcon; // SetLength(xInputVector, 256);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);

        //SetLength(xInputVector, 256);
        readln(sample_db,buff); IconArr[1].IconType:=strtoint(buff);
        xOutputVector[0]:=1;  xOutputVector[1]:=0;
        IconArr[1].Icon:=ReadIcon; //SetLength(xInputVector, 256);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);
        }
        // Ініціалізувати ваги Нейроної мережі Хопфілда
        //NeuralNetHopf1.InitWeights;



      closefile(n_debug);
      closefile(sample_db);
    end;
  //NeuralNetBP1.ResetPatterns;

end;


// запис іконки у файл
procedure WriteIcon(Arr1: TByteArr);
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);

  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        write(sample_db,Arr1[x,y]);
      end;
    writeln(sample_db,'');
    end;
end;

// закриття форми
procedure TAddSampleForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
//  recognition_run:=not(recognition_run);
  if recognition_run then
    begin
    RobotRecognition.Terminate;
```

```
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    recognition_run:=not(recognition_run);
    end;

  //IconTypes:=strtoint(eIconTypes.text);

  {
  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;

  }
end;

// відновлення картинки для збереження семпла
procedure TAddSampleForm.Button2Click(Sender: TObject);
var canvas:tcanvas;
begin
if debug then
begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;Canvas.FillRect(Canvas.ClipRect);
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=0;
      AddSampleForm.ObjectImage.Width :=0;

end;
upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
end;

procedure TAddSampleForm.NeuralNetBP1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNetBP1.TeachError);
  Application.ProcessMessages;

end;


// навчання нейромережі із зазначеними параметрами
procedure TAddSampleForm.Button3Click(Sender: TObject);
begin


      NeuralNetBP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNetBP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNetBP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      NeuralNetBP1.Init;
 // Учимо багатошарову
 NeuralNetBP1.EpochCount := speEpochCount.Value;
 prbEpoch.Max := NeuralNetBP1.EpochCount;
 prbEpoch.Position := 0;

 // Запуск процесу навчання (offline)
 //NeuralNetExtended1.TeachOffLine;
 NeuralNetBP1.TeachOffLine;
```

```
end;


// перебір параметрів навчання, навчання, записати помилки в лог
// для дослідження  - яка нейромережа краще, і як не потрапити в локальний
// мінімум при навчанні.
procedure TAddSampleForm.Button4Click(Sender: TObject);
  var log:textfile;
  var i,j:integer;
  var cnt:integer;
begin


  NeuralNetBP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
  NeuralNetBP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
  NeuralNetBP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
  NeuralNetBP1.EpochCount := 5000;


  prbEpoch.Max := NeuralNetBP1.EpochCount;
  prbEpoch.Position := 0;
      assignfile(log,neuro_teach_log_file_name);
      append(log);
      writeln(log,'---------------------------');
      closefile(log);
  cnt:=400;
  for i:=1 to 20 do
    begin
    NeuralNetBP1.Alpha:=NeuralNetBP1.Alpha-0.01;
    for j:=1 to 20 do
      begin
      prbEpoch.Position := 0;
      NeuralNetBP1.TeachRate:=NeuralNetBP1.TeachRate-0.005;
      NeuralNetBP1.TeachOffLine;
      assignfile(log,neuro_teach_log_file_name);
      append(log);

      writeln(log,'помилка='+floattostr(NeuralNetBP1.TeachError)+'
альфа='+floattostr(NeuralNetBP1.Alpha)+'
шаг навчання =',floattostr(NeuralNetBP1.TeachRate));
      closefile(log);
      cnt:= cnt-1;
      sttError.Caption := inttostr(cnt);
      //prbEpoch.Position := prbEpoch.Position + 1;
      Application.ProcessMessages;
      end;
    end;

end;

// подія кінця епохи навчання , зрушення процес бара, відображення помилки
procedure TAddSampleForm.NeuralNetExtended1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNetExtended1.TeachError);
  Application.ProcessMessages;

end;


// дозвіл розпізнавання
procedure TAddSampleForm.Button5Click(Sender: TObject);
begin
  debug:=false;
  recognition_run:=not(recognition_run);


  if recognition_run then
    begin
```

```
    RobotRecognition := TRobotRecognition.Create(false);
    RobotRecognition.RecType:=AddSampleForm.recType.ItemIndex;
    AddSampleForm.Button5.Caption:='СТОП'
    end
    else
    begin
    //RobotRecognition.
    RobotRecognition.Terminate;
    //RobotRecognition.Destroy;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    end;

end;



// збереження образів у файл
procedure TAddSampleForm.Button6Click(Sender: TObject);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
  //IconTypes:=strtoint(eIconTypes.text);


  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;


end;


// збереження ваг нейромережі у файл
procedure TAddSampleForm.Button7Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
begin
  AssignFile(neuro,neuro_weight_file_name);
  rewrite(neuro);
  writeln(neuro,floattostr(NeuralNetBP1.TeachRate));
  writeln(neuro,floattostr(NeuralNetBP1.Momentum));
  writeln(neuro,floattostr(NeuralNetBP1.Alpha));

  for i:=0 to NeuralNetBP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNetBP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to  length(NeuralNetBP1.Layers[i].Neurons[j].FWeights)-1 do
          begin

writeln(neuro,floattostr(NeuralNetBP1.Layers[i].Neurons[j].Weights[w]));
          end;
```

```
      end;
    end;
  closefile(neuro);
end;



// завантаження ваг нейромережі з файла
procedure TAddSampleForm.Button8Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
  buff:string;
begin
  AssignFile(neuro,neuro_weight_file_name);
  reset(neuro);


  readln(neuro,buff);NeuralNetBP1.TeachRate:=StrToFloat(buff);
  readln(neuro,buff);NeuralNetBP1.Momentum:=StrToFloat(buff);
  readln(neuro,buff);NeuralNetBP1.Alpha:=StrToFloat(buff);
  NeuralNetBP1.Init;
  for i:=0 to NeuralNetBP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNetBP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to  length(NeuralNetBP1.Layers[i].Neurons[j].FWeights)-1 do
          begin
            readln(neuro,buff);
            NeuralNetBP1.Layers[i].Neurons[j].Weights[w]:=strtofloat(buff);
          end;
      end;
    end;
  closefile(neuro);

end;

procedure TAddSampleForm.Button9Click(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  y: Cardinal;

begin
  debug:=true;
 // upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);


  SetLength(BitmapArr,QP.Height);
  for y := 0 to High(BitmapArr) do
    SetLength(BitmapArr[y],QP.Width);

  ConvertBitmapToMonoChrome(QP,BitmapArr);

  //ConvertBitmapToMonoChrome(QP2,BitmapArr);
  if length(BitmapArr)>0 then
  segment(BitmapArr);
  debug:=false;
end;

procedure TAddSampleForm.TrackBar1Change(Sender: TObject);
begin
AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);
end;
```

```
procedure TAddSampleForm.Timer1Timer(Sender: TObject);
begin
AddSampleForm.lFPS.Caption:=floattostr(round(tm_tick/5*10)/10);
tm_tick:=0;
end;

end.
```

Файл About.pas - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TAboutForm = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutForm: TAboutForm;

implementation

{$R *.dfm}

procedure TAboutForm.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('БАКАЛАВРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Програмне забезпечення системи на основі методу структурно-
лінгвістичного розпізнавання зображень для пошукових систем у мережі Інтернет');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Коваленко О.В.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Дерев`янко Олександр Сергійович');
Memo1.Lines.Add('                    гр. КІ-20');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2024');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;

procedure TAboutForm.Button1Click(Sender: TObject);
begin
AboutForm.Close;
end;

end.
```