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 інтелектуального розпізнавання графічних образів”

Виконав здобувач вищої освіти
IV курсу, групи КІ-20
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Поладов А.
«____» _____ 2024 р.

Керівник проекту
кандидат фізико-математичних наук, доцент
_____ Петренюк В.І.
«____» _____ 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

Поладову Арслану
(прізвище, ім’я, по батькові)

1. Тема роботи *Програмне забезпечення системи інтелектуального розпізнавання графічних образів*

2. Керівник роботи *Петренюк Володимир Ілліч, канд. фіз.-мат. наук, доцент*
(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 інтелектуального розпізнавання графічних образів*

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
_____ Петренюк В.І.
(прізвище та ініціали)

Завдання прийнято до виконання
« 17 » січня 2024 р.

Підпис здобувача
_____ Поладов А.
(прізвище та ініціали)

АНОТАЦІЯ

Поладов А. Програмне забезпечення системи інтелектуального розпізнавання графічних образів. 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 інтелектуального розпізнавання графічних образів.

Метою розробки є програмне забезпечення системи інтелектуального розпізнавання графічних образів.

Результат роботи – програмна реалізація системи інтелектуального розпізнавання графічних образів.

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 Delphi 10.4.

Ключові слова: комп'ютерна інженерія, розпізнавання графічних образів

ABSTRACT

Poladov A. Software for the system of intellectual recognition of graphic images. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2024.

In this final qualification work for the first (bachelor) level of higher education, software is developed, which is intended for the system of intellectual recognition of graphic images.

The purpose of the development is the software of the system of intelligent recognition of graphic images.

The result of the work is the software implementation of the system of intelligent recognition of graphic images.

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 Delphi 10.4 environment.

Keywords: computer engineering, recognition of graphic images

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. Загрози зростають у міру розвитку світу. Тому щодня розробляються передові системи безпеки, щоб протистояти зростаючим ризикам безпеки. Щоб підвищити безпеку в громадських місцях, це дослідження пропонує інтелектуальну систему безпеки, яка об'єднує номерний знак і технологію розпізнавання обличчя. Технологія робить фотографії номерних знаків, аналізує їх за допомогою набору даних, а потім фотографує обличчя людей, аналізуючи їх за допомогою того самого набору даних, коли вони входять і виходять з будівлі. Попередження надсилаються, якщо є розбіжності між набором даних і авторизованими автомобілями та транспортними засобами. Ця технологія забезпечує моніторинг у реальному часі та оперативне реагування на загрози безпеці. Систему можна використовувати в різних місцях, включаючи стоянки, спостереження за громадською безпекою та керування доступом до будівель. Завдяки цьому підходу ризик порушення безпеки знижується, а безпека покращується.

Мета й завдання дослідження. Метою роботи є програмне забезпечення системи інтелектуального розпізнавання графічних образів.

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

– Огляд існуючих систем інтелектуального розпізнавання графічних образів.

– Дослідження системи інтелектуального розпізнавання графічних образів.

– Програмна реалізація системи інтелектуального розпізнавання графічних образів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі інтелектуального розпізнавання графічних образів.

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтелектуального розпізнавання графічних образів, 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

Метод базується на наступних основних кроках:

1. Транспортний засіб наближається до в'їзних воріт, і система розпізнавання номерного знака фіксує зображення номерного знака.

2. Номерні знаки будуть виявлені та проаналізовані та перевірені в наборі даних.

3. Якщо номерний знак розпізнано, тоді активується розпізнавання обличчя, інакше транспортний засіб не буде допущено до приміщення.

4. Обличчя водія фіксується, а потім обробляється, аналізується, а потім порівнюється з набором даних.

5. Якщо обличчя збігається з уповноваженою особою, ворота відчиняються, і тоді транспортний засіб допускається до приміщення, інакше ні.

6. Якщо номерний знак або система розпізнавання обличчя виявляють неавторизований транспортний засіб або особу, співробітники служби безпеки повідомляються.

7. Система постійно відстежує та записує всі транспортні засоби та окремі дії, створюючи повний запис усіх входів та виїздів. Щоб досягти цього, він використовуватиме основні принципи обробки цифрових зображень, системи керування та системи керування наборами даних.

Значення цієї системи полягатиме в тому, щоб переконатися, що правильні співробітники входять до приміщення установи та виключити будь-яких зловмисників.

1.2 Область застосування

Метою нашого проекту, інтелектуальної системи безпеки, яка використовує номерний знак і розпізнавання обличчя, є посилення заходів безпеки та покращення можливостей спостереження в багатьох громадських місцях, таких як університетські містечка, вокзали, торгові центри й аеропорти

Використовуючи техніку розпізнавання номерних знаків і облич, виявляйте підозрілих осіб і транспортні засоби, а також охоронні сигнали. Це дозволяє персоналу служби безпеки оперативно реагувати на можливі загрози, припиняти злочинну діяльність і підтримувати безпеку та безпеку громадськості та організації. Крім того, ця система може надавати працівникам служби безпеки ключові дані та аналітику/статистику для відстеження та дослідження їхнього руху людей і транспортних засобів, виявлення закономірностей підозрілої поведінки та діяльності, а також створення звітів для подальшого перегляду та розслідування. Загалом ця технологія має потенціал для значного підвищення ефективності заходів безпеки в громадських місцях, створюючи безпечну атмосферу для всіх. Вимога до складної системи безпеки, яка може розпізнавати та забезпечувати вхід лише уповноваженим особам у різноманітних ситуаціях, у тому числі закритих спільнотах, бізнес-будинках і громадських місцях, – це проблема, яку намагається вирішити цей проект. Нездатність поточних заходів безпеки розрізнити авторизованих і неавторизованих осіб може призвести до порушень безпеки. Цей проект має на меті створити інтелектуальну систему безпеки, яка зможе точно ідентифікувати та перевіряти осіб, перш ніж дозволити доступ, забезпечуючи посилені заходи безпеки та знижуючи ризики несанкціонованого доступу. Це досягається шляхом об’єднання технологій розпізнавання номерних знаків і облич. Нижче наведено цілі інтелектуальної системи безпеки, яка використовує технологію розпізнавання облич і номерних знаків:

– Покращена безпека: головна мета цієї системи – підвищити безпеку в громадських місцях шляхом точної ідентифікації та автентифікації людей і

транспортних засобів, які в’їжджають. і залишаючи заклад. Таким чином ви можете зменшити ймовірність провалів у безпеці та гарантувати захист усіх і вашої власності.

– Моніторинг у реальному часі: система повинна забезпечувати можливості моніторингу в реальному часі, щоб дозволити персоналу служби безпеки швидко виявляти загрози безпеці та реагувати на них.

– Зменшення витрат : система має допомогти зменшити потребу в людському персоналі, що може призвести до економії коштів у довгостроковій перспективі.

– Точна ідентифікація: система має точно ідентифікувати та перевіряти осіб і транспортні засоби, використовуючи як номерні знаки, так і технології розпізнавання обличчя, щоб мінімізувати хибно-позитивні та хибно-негативні результати.

– Керування набором даних: система повинна мати ефективну систему керування набором даних для зберігання та керування даними, зібраними з номерних знаків і технологій розпізнавання облич

– Гнучкість: система має бути гнучкою та адаптованою до різних типів громадських місць і сценаріїв безпеки, таких як спостереження.

Загалом цілями цієї системи є надання надійних, ефективних і точних рішень безпеки, які покращують громадську безпеку та зменшують ймовірність порушень безпеки. У цьому проекті ми розробимо систему, яка включає в себе інтеграцію номерних знаків і технологій розпізнавання обличчя для створення розумної системи безпеки, яка зможе точно ідентифікувати та перевіряти осіб і транспортні засоби. Проект передбачає вибір та інтеграцію відповідних апаратних і програмних компонентів для підтримки системи, включаючи камери, датчики та процесори. Система потребує розробки системи керування набором даних для зберігання та керування даними, зібраними з номерних знаків і технологій розпізнавання обличчя. Таким чином, масив буде використовуватися для зберігання всіх авторизованих номерних знаків. Ми будемо розробляти та

впроваджувати алгоритми обробки цифрових зображень для аналізу зображень номерних знаків і облич, знятих камерами. Yolo v4 – це алгоритм, який використовується для виявлення номерних знаків, а OCR використовується для зчитування тексту з номерних знаків. Після цього каскадний класифікатор Хаара для виявлення облич із зображень, а потім алгоритм обличчя Фішера використовується для розпізнавання облич. Після цього буде проведено тестування та оцінку системи, щоб переконатися, що вона відповідає цілям і функціям за призначенням

Таким чином, виходячи з вищеперерахованого, програмне забезпечення системи інтелектуального розпізнавання графічних образів, 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

Проведемо огляд програмного забезпечення, яке використовує розпізнання образів у своїй роботі.

FaceCode

FaceCode – програма розпізнавання особи для контролю доступу ПК. Спеціально розроблена програма, що не дає можливість дітям, членам родини, співробітникам і іншим користувачам робочого місця одержати доступ до вашого комп'ютера. FaceCode – може гарантувати реальну ідентичність користувача.

Використовуючий веб-камеру й не потребуючий введення імені користувача або пароля FaceCode зручний для користувача, прекрасне програмне забезпечення контролю доступу для вашого ПК.

Використовуючи сучасні технології розпізнавання особи, FaceCode розпізнає існуючих користувачів ПК і реєструє їх на їх власну користувальницьку область, поряд з тим перешкоджає неавторизованим користувачам користуватися вашим комп'ютером на те без вашої згоди, навіть якщо вони знають ім'я користувача й пароль.

Ключові переваги продукту:

– Безпечний і надійний.

– Запобігання доступу інших користувачів.

– Автозапуск.

– Штучний інтелект.

– Зручний для користувача.

Рисунок 2.1 – Інтерфейс користувача FaceCode

Особливості:

– Майстер інсталяції, установка й реєстрація.

– Безпечна вхідна реєстрація в Window і Domain Accounts,

використовуючи автентифікацію особи.

– Блокування заставки захищене ознаками особи.

Безпечний вхід у ПК дозволяє тільки зареєстрованим користувачам одержувати доступ до ПК, використовуючи технології розпізнавання особи. Паролі можуть бути дискредитовані. Користувач може визначити рівень безпеки

Блокування заставки можна зняти, показавши особу зареєстрованого користувача.

Заставка автоматично запускається, коли користувач залишає своє місце, використання технології Recognix дозволяє операторові встановлювати параметр “wait” у мінімальний доступний вибір (тільки одна хвилина).

Підтримуються методи автентифікації користувача.

Система розпізнавання автомобільних номерів «Потік» (компанія «Россі»)

Система «Потік» – апаратно-програмний комплекс, призначений для ідентифікації державних реєстраційних знаків автомобілів, створений фірмою РОССІ для використання на великих автомобільних магістралях, невеликих автодорогах, на в'їздах і виїздах на охоронювані території або автостоянки й у багатьох інших випадках.

Система «Потік» дозволяє робити розпізнавання й реєстрацію автомобільних номерів при установці його як на стаціонарному пості, так і при використанні його в складі мобільного комплексу на борті автомобіля, що пересувається. Система дозволяє зберігати найдокладнішу інформацію про характеристики ідентифікуємих нею автомобілів – поточний час виявлення, швидкість і напрямок руху, відеозображення автотранспортного засобу, а також робити пошук одночасно по декількох базах даних на предмет збігу з розпізнаним номером і обмінюватися при цьому інформацією з вилученими клієнтами по мережі. Убудований детектор руху дозволяє реєструвати автомобілі з нерозрізненими (відсутніми) за якимись причинами реєстраційними номерами. Можливість підключення широкого спектра зовнішніх пристроїв (світлофори,

шлагбауми й інші) дозволяє організувати автоматизовані системи обмеження доступу, як у складі комплексних систем безпеки, так і самостійно.

Система «Потік» – це можливість централізованого керування й збору інформації з декількох об'єктів. Система здатна вести протоколювання всіх подій і в потрібний момент сповістити оператора про ту або іншу системну подію, що робить її максимально «дружелюбної» і простий у розумінні й обігу для користувача. Додатково є можливість «програмувати» поводження системи у відповідь на певні події.

Авто-інспектор

Система "Авто-інспектор" – програмно-апаратний комплекс, що забезпечує розпізнавання номерів автомобілів, що рухаються, що надійно працює в широкому діапазоні зовнішніх умов, легко інтегрувальний з охоронним устаткуванням, виконавчими пристроями й зовнішніми базами даних.

Ефективний для рішення завдань реєстрації, ідентифікації й забезпечення безпеки автомобілів, контролю транспортних потоків.

Переваги системи:

– Розпізнавання номерів автомобілів, які переміщуються на швидкості до 130 км/год.

– Висока якість розпізнавання в різних умовах освітленості: у денний і нічний час.

– Розпізнавання реєстраційних номерних знаків різних країн з можливістю адаптації для роботи з новими стандартами.

– Формування бази даних зі збереженням інформації про час, дату, напрямок проїзду, розпізнаному номері автомобіля, його відеозображення.

– Обсяг бази даних обмежується тільки ємністю жорсткого диска.

– Забезпечення пошуку інформації в базі даних по номері, часу, даті проїзду транспортного засобу.

– Миттєвий пошук інформації із заданих параметрів: при обсязі бази до 10 млн. записів не більше 0,2 сек.

– Зіставлення розпізнаного номера з інформацією бази даних (власної або зовнішньої).

– Спеціально розроблена технологія забезпечує високий відсоток розпізнавання, миттєве порівняння з інформацією бази даних, можливість одночасного аналізу декількох реєстраційних номерів автомобілів.

– Синхронне зіставлення з додатковими камерами.

Користувальницькі переваги використання системи "Авто-інспектор":

– Крім функціональних можливостей, які гарантують успішне рішення широкого спектра завдань характерних для різних об'єктів, система надає численні переваги для роботи в складі єдиній системі безпеки об'єкта в результаті інтеграції з необхідним устаткуванням і програмним забезпеченням її складових.

– Використання унікальних технологій дозволяє реалізовувати на платформі "Авто-інспектор" мережні територіально розподілені рішення із багатосерверною архітектурою, здійснювати керування необхідною кількістю відеокамер, передаючи від них дані в єдиний центр аналізу.

Серед переваг системи "Авто-інспектор", що забезпечують ефективне рішення різноманітних завдань із урахуванням індивідуальних особливостей об'єктів, їхнього масштабу, вимог заказника, можна виділити:

– забезпечення заданої реакції всіх компонентів єдиної системи безпеки об'єкта, виконавчих і інженерно-технічних пристроїв за результатами розпізнавання номерів, їхнього порівняння з інформацією баз даних;

– забезпечення автоматичного оповіщення оператора в результаті реєстрації певних подій, наприклад, при розбіжності результатів розпізнавання номера з інформацією з бази даних, при виявленні в транспортному потоці автомобіля із заданим реєстраційним номером;

– організація необхідної кількості відеоканалів, робочих місць операторів і адміністраторів для одержання даних у режимі реального часу, доступу до архіву й дистанційного керування системою;

– інтеграція із зовнішніми базами даних, у тому числі, для роботи в

режимі заборони/дозволу проїзду по "чорному"/"білому" списку, для реалізації оповіщення при реєстрації в транспортному потоці автомобіля із зазначеним реєстраційним номером;

– інтеграція з технологічним устаткуванням підприємств, наприклад, електронними вагами; інтеграція з автоматичними системами розрахунків;

– висока якість розпізнавання в різних умовах освітленості – не менш 95% у денний і нічний час;

– швидка адаптація для роботи з новими стандартами номерних знаків.

Спеціально розроблена технологія дозволяє досягти близької до 100% імовірності розпізнавання. Такий результат досягається шляхом уточнення результатів розпізнавання окремих символів номера на основі заданих шаблонів номерних знаків. Ця перевага дозволяє уникнути таких помилок як помилкове розпізнавання артефактів, визначення неправильної кількості символів у номері, розпізнавання близьких за написанням символів. Незаперечною перевагою цієї технології є швидка адаптація для роботи з новими шаблонами.

Надійна робота в широкому діапазоні кутів установки відеокамер, унікальна технологія розпізнавання забезпечує надійну роботу системи при більших кутах візування (до 40 градусів по вертикалі й горизонталі), що актуально при використанні системи на складних об'єктах з обмеженнями вибору місця для кріплення відеокамер.

Система "Авто-інспектор" – рішення завдань контролю, реєстрації, ідентифікації й забезпечення безпеки автомобілів на будь-яких об'єктах, для яких характерний транспортний потік різної інтенсивності: від автопаркинга житлового комплексу до міської магістралі.

Система "Авто-інспектор" надає ряд переваг при організації роботи автостоянок: її використання дозволяє зробити процес реєстрації фактів в'їзду й виїзду автомобіля повністю автоматичним. Це знімає необхідність оформлення парковочних талонів, виключивши при цьому махінації персоналу й підвищивши

оперативність і якість обслуговування, а головне гарантувати схоронність транспортних засобів.

Інтеграція системи "Авто-інспектор" з автоматичними системами розрахунків, автоматичними воротами, шлагбаумами, іншим необхідним технічним устаткуванням – додаткові гарантії безпеки автомобілів, захисту їхніх власників, зручність у роботі співробітників стоянки.

На будь-якому підприємстві гостро коштує завдання контролю в'їздів і виїздів з його території, кожний промисловий об'єкт – це завжди зосередження великої кількості автотранспорту.

Система "Авто-інспектор" інтегрована в єдину систему безпеки підприємства забезпечить запобігання несанкціонованого проїзду на об'єкти, контроль за ввозом і вивозом вантажів, їхньою відповідністю супровідної документації, дозволить організувати диференційований в'їзд. Широкі можливості інтеграції й побудови на основі системи розпізнавання автомобільних номерів територіально розподілених рішень, надійна робота в будь-яких зовнішніх умовах дозволяють використовувати неї на об'єктах різного масштабу й галузевої приналежності з обліком.

Забезпечуючи безпеку на дорогах, співробітники дорожньо-патрульних служб вирішують ряд конкретних завдань. Їхнє успішне виконання багато в чому залежить від використання нових високотехнологічних рішень: система "Авто-інспектор" є основою для організації поста ДПС. Система розпізнавання автомобільних номерів на базі стаціонарного або портативного комп'ютера, установленого в патрульному автомобілі дозволяє автоматично контролювати транспортний потік, виявляючи в ньому автомашини з певними номерами (наприклад, база викрадених автомобілів), сповіщаючи про це оператора, формуючи власну, і, працюючи із зовнішніми базами даних.

Використання системи "Авто-інспектор" актуально для організації диференційованого в'їзду в певні зони міста – центр, режимні й особливо охоронювані об'єкти, а також для введення системи платного використання

ділянок доріг і цілих магістралей. Використання модуля дозволить відмовитися від необхідності встаткування й обслуговування спеціальних пропускних пунктів, шлагбаумів. Їх відсутність, крім зниження фінансових витрат, дозволить позбутися від утворення черг на в'їзд у центр міста, на "платні дороги", виключить махінації із пропусками, в'їзними талонами.

Постійний відеоконтроль за пересуванням транспорту усередині контрольованої зони забезпечить реєстрацію будь-якого порушення встановлених правил в'їзду.

Overseer Parking

Система Overseer Parking призначена для оптимізації й автоматизації роботи стоянок і парковочних комплексів.

Інтерфейс системи оптимізований для роботи із чутливими до натискання дисплеями, що значно спрощує навчання роботі із системою, адже це найбільш прогресивний і інтуїтивний механізм роботи з комп'ютером. Практика показала, що людина жодного разу не зіштовхувався з Overseer Parking уже буквально через пару годин міг виконувати більшість операторських функцій, не прибігаючи до інструкції користувача.

Окремої уваги заслуговує система тарифікації: гнучкий механізм створення тарифів представляє можливість призначати різні вартості залежно від часу доби, дня, рівня завантаження паркування й безлічі інших параметрів. Також, різні тарифи можуть призначатися як окремо взятим клієнтам так і їхнім групам.

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 інтелектуального розпізнавання графічних образів.

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

Система призначення для розпізнавання графічних образів з телекамери для системи безпеки установи. Тобто розробляється програмне забезпечення для системи відеоспостереження.

Розрізняють наступні стани системи відеоспостереження:

– стан тривоги – є результатом реагування системи на тривожну подію;

– стан спостереження – система виконує функції, достатні для перегляду сцени оператором, або ручного супроводу цілі;

– стан охорони – система виконує функції, достатні для автоматичного й при необхідності ручного супроводу цілі.

Огляд основних функцій

Сумісність – необхідний, але недостатній фактор, що поєднує різні пристрої в систему. Набір більш-менш випадково розташованих відеокамер ще не є системою. Ефективна система відеоспостереження повинна будуватися на основі ретельно продуманої концепції захисту об'єкта. У ній повинні бути чітко визначені завдання, які покликане вирішувати система відеоспостереження при забезпеченні безпеки. Серед типових завдань можна виділити наступні:

1. Оперативне спостереження за охоронюваною територією, будинками й приміщеннями. Найпростіша функція "системи відеоспостереження в стані спостереження" – так це називається за ДСТ Р 51558-2000. Відеокамери можуть установлюватися потай або відкрито, залежно від розв'язуваного завдання. Виявлення порушника покладене на оператора.

2. Оцінка сигналу тривоги. Відеокамера використовується разом з технічним засобом охорони для підтвердження факту спрацьовування останнього.

Тут треба звернути увагу на одну обставину: виявлення й оцінка – дві різні речі. Виявлення – це повідомлення про можливу подію, критичну з погляду забезпечення безпеки. Оцінка – заходи щодо з'ясування того, чи дійсно відбувся напад або має місце фіктивна тривога. Крім того, буває корисна будь-яка додаткова інформація "з місця події": як виглядають порушники, скільки їх, як вони поводяться.

Проводилися спеціальні дослідження, які показали, що люди краще вирішують завдання оцінки, ніж виявлення. Тому на невеликому й не дуже важливому об'єкті застосування людей для виявлення цілком припустимо, але для забезпечення високого "рівня безпеки" потрібна функція автоматичної оцінки подій.

3. Телебачення може використовуватися разом із системою керування доступом для підвищення ефективності контрольно-пропускних функцій. Наприклад, при проході через КПП із низьким трафіком і відсутністю оператора можна дистанційно встановлювати особистість людини по фотографії, що зберігається в базі даних.

4. Психологічний вплив на порушника. Відеокамери, навіть непрацюючі, можуть робити "відлякуючу" дію, виконуючи в такий спосіб послужливо-профілактичну функцію.

5. Документування подій на об'єкті. Матеріал відеоархівів може виявитися корисним як доказова база при розслідуванні несанкціонованих дій.

Це найпростіші функції телебачення в системі охорони, що вимагають присутності людини-оператора й/або постійного запису. До "інтелектуальних" функцій відносяться такі, у яких телебачення бере на себе функцію автоматичної оцінки обстановки або ж виступає в ролі технічного засобу виявлення. У їхньому числі:

– Виявлення переміщення в зоні спостереження (відеодетекція). Такі пристрої часто вбудовуються в стандартні мультиплексори. При цьому оператор може задавати зону на екрані монітора, рух у якій викликає сигнал тривоги.

– Розпізнавання (класифікація) об'єктів. Більше складна функція. Система повинна не тільки виявити динамічний об'єкт, але й правильно віднести його до якого-небудь класу, відрізнити людини від тварини й від хитання віток дерев. Це дозволяє різко підвищити завадостійкість відеодетектора, що діє в складній перешкодній обстановці, наприклад на відкритому повітрі. Основними параметрами, по яких відбувається розпізнавання графічних образів, є просторові характеристики об'єктів: габаритні розміри, периметр, площа й т.д.

– Динамічне спостереження за порушником. Системи динамічної цілевказівки аналізують зміни координат характерних точок об'єкта, наприклад центра ваги, кольору.

Переваги використання засобів охоронного телебачення як технічних засобів виявлення:

По-перше, відпадає необхідність пристрою контрольно-слідової смуги в заборонній зоні. Однак, якщо відеокамера включається по факту спрацьовування, а порушник буде переборювати рубіж біля відеокамери, то він буде в поле зору менше секунди й оператор може не встигнути його помітити. Тому таку схему бажано застосовувати при постійному спостереженні (запису) обстановки в заборонній зоні. Крім того, буває дуже корисно влаштовувати телеспостереження за ділянками, віддаленими від розміщення підрозділу служби безпеки.

По-друге, пасивний принцип дії забезпечує маскуємість рубежу спостереження. Апаратура телеспостереження працює у видимому (інфрачервоному) діапазоні електромагнітних хвиль, виявлення технічними засобами розвідки досить утруднено.

По-третє, висока швидкодія роботи. Сигнал від відеокамери являє собою растровий просторово-часовий сигнал. Цифрові методи обробки дозволяють у реальному часі провести експрес-аналіз відеосигналу в завданнях виявлення й розпізнавання об'єктів у зоні спостереження, що особливо важливо при швидкому розвитку конфліктної ситуації в системі "охорона – порушник".

Областю застосування розробляємого програмного забезпечення є відеодетектори систем безпеки установ.

Відеодетектор є невід'ємною частиною будь-якого сучасного відеореєстратора. Принцип дії детектора полягає в розпізнаванні по відеосигналу динамічних об'єктів. Як правило, у неспеціалізованих системах детектори не володіють "інтелектом" і спрацьовують на будь-який рух у кадрі.

Основні характеристики детектора – це його стійкість до помилок першого й другого порядку.

Помилки першого порядку (помилкові спрацьовування) – фактор, що робить відеодетектори в більшості встановлених цифрових систем відеоспостереження неефективними. Можливості системи по усуненню помилкових спрацьовувань – важливий показник якості відеодетектора.

Помилки другого порядку (пропуск об'єкта, що рухається) характеризуються просторовою чутливістю, тобто мінімальним рухом, на яке детектор зреагує, і динамічною чутливістю, тобто мінімальною необхідною для спрацьовування швидкістю руху об'єкта.

Відповідно необхідно розробити таке програмне забезпечення, яке б унеможливило як помилки першого роду, так й помилки другого роду

Перш, ніж приступитися до основних методів розпізнавання графічних образів, приведемо кілька необхідних визначень.

Отже, перша частина нашого проекту – це розпізнавання номерних знаків. Для цього є дві основні речі, які потрібно зробити для його розпізнавання. По-перше, ми повинні правильно локалізувати його, а потім точно прочитати з нього текст. Розпізнавання номерних знаків складається з наступних кроків:

– Зйомка зображення: зображення номерного знака транспортного засобу буде зафіксовано добре розміщеною та збалансованою камерою.

– Обробка зображення: Потім зображення обробляється для виявлення області номерного знака за допомогою алгоритму Yolov4, про який буде детально обговорено далі.

– Сегментація символів: після локалізації номерного знака символи на ньому будуть сегментовані за допомогою алгоритму OCR. Окремі символи відокремлені один від одного.

– Розпізнавання символів: після сегментації символи розпізнаються за допомогою технології OCR.

– Набір даних: після розпізнавання символів їх буде порівняно з дозволеними номерними знаками. Якщо він збігається, ми переходимо до наступного кроку, який є розпізнаванням обличчя, а якщо ні, тоді вхід буде відхилено.

3.2 Розробка структурної схеми

Структурна схема системи наведена на рисунку 3.1. Структурно система складається з наступних частин:

1. Відеокамера спостереженняв.
2. Система розпізнання образів, яка включає в себе:

– Блок читання картинки з відеокамери.

– Блок аналізу та виділення ознак.

– Блок класифікації та опису об’єкта.

3. База даних журналювання розпізнаних образів.
4. База даних образів облич.
5. База даних образів цифр та букв на номерах автомобілів.

Система виділення й розпізнавання осіб

Апаратно-програмний комплекс «Система некооперативного виділення й розпізнавання осіб» призначений для автоматичного виділення зображень осіб з панорамного відеопотоку і їхнього наступного розпізнавання.

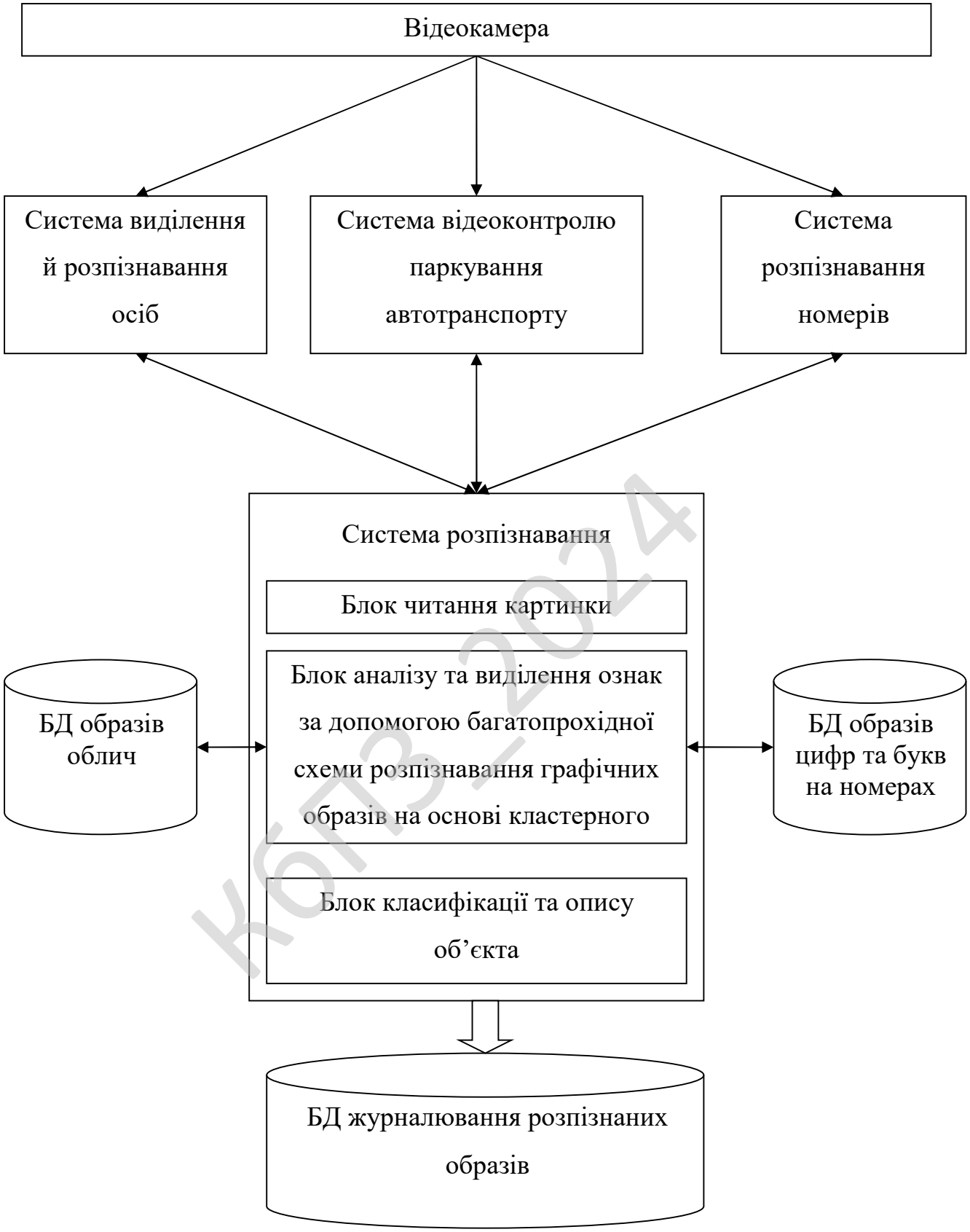

Рисунок 3.1 – Структурна схема системи

«Некооперативність» розпізнавання – відмінна риса системи Система некооперативного виділення й розпізнавання осіб, що працює с використанням камер високого дозволу. Системі «некооперативного» розпізнавання для виділення зображення особи не потрібно спеціального позиціонування людини в поле зору відеокамери. Більше того, людина навіть може й не припускати, що перебуває в зоні відеоспостереження.

Можливості системи некооперативного виділення й розпізнавання осіб:

1. Автоматичне виділення й супровід осіб у поле зору відеокамери.

2. Автоматичне формування бази фотографій осіб, що потрапили в поле зору камери.

3. Система виділення зображень осіб може працювати у двох режимах:

– збереження відеозображення особи протягом усього часу знаходження в області видимості;

– збереження для кожної особи одного найкращого скриншоту.

4. Автоматичне розпізнавання осіб за результатами порівняння з базою еталонних зображень.

5. Трансляція відеозображень по мережі.

Області застосування:

– Місця масового перебування людей: стадіони, кінотеатри, торгові центри.

– Транспортні вузли: аеропорти, вокзали, автостанції.

– Прикордонні паспортновізові контрольні пункти.

– Прохідні й контрольно-перепускні пункти.

– Суспільні установи.

Переваги

Висока якість відеоматеріалів

Основною якісною перевагою системи є висока якість реєструємих зображень осіб. Це досягається за рахунок використання камер з розрішенням від 1,3 до 5,1 мегапікселів, що дає наступні переваги:

– високий дозвіл системи відеозахвату;

– прогресивне розгорнення (відсутність ефекту гребінки при русі);

– відеоаналіз на базі некомпресованого потоку відеоданих;

– компресія даних виробляється після відеоаналізу.

Висока вірогідність розпізнавання

Висока вірогідність розпізнавання досягається за допомогою застосування нових технологій:

– високе розрішення отриманих зображень осіб;

– розпізнавання не по одному, а по серії зображень;

– попередня цифрова обробка зображень: нормалізація яскравості, контрасту, усунення паразитних відблисків і контрового засвітчення;

– використання стереозору й можливість створення 3D-моделей осіб;

– робота в інфрачервоному спектральному діапазоні, у тому числі з ІК-підсвічуванням.

Висока продуктивність

Застосування сучасних комп'ютерних платформ, технологій багатоядерних, багатопоточних і кластерних обчислень дають нові можливості для обробки більших масивів відеоінформації. Алгоритми цифрової обробки відеоданих оптимізуються для високопродуктивних процесорів: Intel, GPU Nvidia/AMD-ATI.

У результаті досягаються унікальні характеристики системи по швидкодії й вірогідності одержуваних результатів.

Робота при несприятливих умовах висвітлення

У відеокамерах використовуються сенсори з розрядністю АЦП 10 і 12 біт. Крім цього, технології подвійного експонування, просторового й часового нагромадження сигналу дозволяють істотно розширити динамічний діапазон вихідних відеоданих. Тому система добре працює при несприятливих умовах висвітлення, у тому числі при зустрічному засвічуванні.

Відповідність вимогам біометричних стандартів

Сьогодні система некооперативного виділення й розпізнавання осіб – це одна з систем виділення осіб, що задовольняє вимогам стандарту ДСТ ІСО/МЕК 19794-5-2006 «Автоматична ідентифікація. Ідентифікація біометрична. Формати обміну біометричними даними. Дані зображення особи» на якість цифрових відеоданих, придатних для проведення автоматичної біометричної ідентифікації особистості по зображенню особи. Даний біометричний стандарт повністю гармонізований з аналогічним міжнародним стандартом ISO. Важливо відзначити, що ДСТ висуває тверді вимоги до дозволу зображення, яким, зокрема, не задовольняють зображення, одержувані із традиційних камер стандарту PAL/NTSC.

Інтеграція із системами біометричної ідентифікації

Для підвищення вірогідності ідентифікації особистості система може працювати з різними модулями розпізнавання осіб: «Cognitec», «BARS», « Asia-Soft» і ін.

Обчислювальний модуль

Як обчислювальна платформа можуть використовуватися вуличні сервери або стандартні високопродуктивні сервери з архітектурою x86.

Состав системи:

– Програмний модуль виділення зображень осіб.

– Програмний модуль розпізнавання осіб.

– База еталонних зображень і настановних даних розшукуваних осіб.

Система розпізнавання номерів

Призначення системи розпізнавання номерів – цілодобове автоматизоване розпізнавання автомобільних номерів.

Система зберігає інформацію із прив'язкою вчасно й місцю про розпізнаний державний реєстраційний знак (ДРЗ), типі, швидкості й інших характеристиках руху транспортного засобу у внутрішній базі даних. Зберігаються також зображення ТЗ, зображення його ДРЗ і панорамний знімок

зони контролю в момент проїзду ТЗ. При підключенні до Системи сторонніх баз ДРЗ ТЗ, автоматично визначаються настановні дані власника транспортного засобу, проводиться перевірка розпізнаного знака по базах розшуку.

Записана інформація може транслюватися по цифрових каналах передачі даних. Система розпізнавання номерів також може транслювати зображення безпосередньо в реальному часі, при необхідності сигналізуючи про різні події.

Система розпізнавання номерів може бути використана, як:

– система розпізнавання номерів що в'їжджають і виїжджають автомобілів на стоянках, паркуваннях, пропускних пунктах на охоронювану територію (наприклад, в аеропортах), на платних дорогах;

– для збору статистичних даних про транспортні потоки й шляхово-транспортну обстановку.

Система розпізнавання номерів вирішує наступні завдання:

– розпізнавання автомобільних номерів. Система автоматично розпізнає однорядкові номерні знаки різної державної приналежності[1] як у латинської, так і кириличному кодуванню для нерухливих і що рухаються, що наближаються й віддаляються автомобілів. При візуально помітних номерних знаках імовірність розпізнавання не нижче 90% як у світле, так і в темний час доби;

– вимір швидкості руху транспортного засобу. Для виміру швидкості використовуються сертифіковані засоби виміру: радіолокаційні (мікрохвильові радари) або лазерні вимірники швидкості руху ТЗ;

– класифікація транспортних засобів по наступних типах: легкові, вантажні, автобуси, мотоцикли;

– збереження в архіві знімків і транспортних засобів і розпізнаних ДРЗ по кожному ідентифікованому транспортному засобі.

Система відеоконтролю паркування автотранспорту

Коло розв'язуваних завдань можна розділити на дві категорії. Перша – забезпечення безпеки й здійснення контролю території. Друга – оперативний

облік вільних парковочних місць і наявності транспортних засобів (ТЗ), а також збір статистики.

Система відеоконтролю паркування автотранспорту призначена для використання на платних і безкоштовних паркуваннях. Поряд з охоронними функціями відеоспостереження, система дозволяє автоматично відслідковувати завантаженість паркування й показувати наявність вільних місць, а також точно визначати час стоянки ТЗ – ця функція корисна для платних паркувань.

Перелік функцій:

– визначення й підрахунок зайнятих і вільних місць;

– графічна індикація на карті паркування стану всіх парковочних місць;

– підрахунок часу стоянки ТЗ;

– охоронне відеоспостереження;

– ідентифікація неправильно припаркованих ТЗ.

Переваги

Головною перевагою системи відеоконтролю паркування автотранспорту в порівнянні із традиційними системами є простота монтажу й обслуговування системи. Система не вимагає установки спеціальних датчиків на парковочних місцях, замість цього використовуються встановлені над паркуванням відеокамери, наприклад, на щоглах висвітлення. Такий підхід полегшує розгортання системи на вже діючих паркуваннях і дозволяє без витрат міняти конфігурацію паркування шляхом простого інтерактивного настроювання карти зон на зображеннях з камер відеоспостереження.

Контроль в'їзду/виїзду

Система відеоконтролю паркування автотранспорту може працювати разом із системою розпізнавання номерів, установлюваної на в'їзді/виїзді паркування й забезпечуючи виконання наступних додаткових функцій:

– розпізнавання державних реєстраційних знаків ТЗ;

– визначення ТЗ по декількох ознаках, таким як колір, клас і т.д.

Для організації контрольно-пропускної системи розпізнавання номерів може використовувати базу даних ТЗ для автоматичного режиму проїзду на територію автостоянки. У цьому випадку система розпізнавання номерів може працювати в трьох режимах:

– автоматичне розпізнавання державних реєстраційних знаків ТЗ для звірення з базою даних контролю проїзду;

– контроль доступу за допомогою карт пропуску й перевірка відповідності реєстраційного знака й зовнішнього вигляду автомобіля з еталонними даними, що зберігаються в базі даних СКУД і пов'язаними з номером карти пропуску;

– контроль відповідності зовнішнього вигляду автомобіля й державного реєстраційного знака на в'їзді й на виїзді з паркування.

Состав системи:

– Камери відеоспостереження.

– Обчислювальний модуль системи відеоконтролю паркування автотранспорту.

– Обчислювальний модуль системи розпізнавання номерів.

– Архів відеоданих.

– АРМ моніторингу.

Для організації контролю за більшими по площі паркуваннями рекомендується використовувати цифрові камери високого дозволу. Високий дозвіл дозволяє одній камері замінити кілька аналогових камер (наприклад, модель NetCam 51C14L з дозволу зображення еквівалентна 12 аналоговим камерам). А це особливо актуально при охороні велких територій, де потрібна розміщення безлічі відеокамер, проведення великої кількості кабельний комунікацій і т.д. Завдяки високому дозволу, наявності прогресивного кадрового розгорнення й високій якості зображення підвищується ефективність роботи алгоритмів відеоаналізу.

3.3Розробка функціональної схеми

На рисунку 3.2 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема системи включає в себе наступні функціональні блоки:

1. Блок розпізнання образів.
2. Блок відеодетекції.
3. Блок розпізнавання номерів автомобілів.
4. Блок розпізнавання людей.
5. Блок динамічного спостереження за порушником.
6. Блок подання сигналу тривоги.
7. Блок документування подій на об’єкті.

Розглянемо ці блоки більш детально.

Блок відеодетекції

Виявлення переміщення в зоні спостереження (відеодетекція). Такі пристрої часто вбудовуються в стандартні мультиплексори. При цьому оператор може задавати зону на екрані монітора, рух у якій викликає сигнал тривоги.

Блок розпізнавання номерів автомобілів

Система використовується автоматичної реєстрації й розпізнавання автомобільних номерів на контрольно-пропускних пунктах на підприємствах, платних стоянках і гаражних комплексах, на постах ДПС. Захист із системою контролю доступу дозволяє автоматизувати контрольно-пропускний режим. Система дозволяє вести розшук автомобілів у викраденні.

Функціональні можливості:

– автоматична реєстрація автомобільних номерів і розпізнавання;

– збереження номера й відеозапису проїзду транспортного засобу в базі даних із вказівкою дати й часу;

– автоматичне зіставлення автомобільного номера з наявними базами даних (наприклад, автомобілів, що мають право допуску на територію підприємства, або автомобілів, що перебувають у викраденні) і видача відповідного повідомлення операторові;

– автоматизація контрольно-пропускного режиму при використанні із пристроями контролю доступу;

– пошук у базі даних по номері, даті, часу;

– формування звітів по номері, даті, часу.

Характеристики:

– Ширина контрольованої зони проїзної частини – від 1,5 до 3,5 метрів.

– Глибина зони контролю – 10 метрів (для КПП: 2–4 метри).

– Кількість оброблюваних кадрів (з обліком 100 % потокової завантаженості на всіх камерах):

– канал – 25 к/с, до 150 км/год;

– каналу й більше – 16 к/с (сумарне кіл-у кадрів), до 75 км/ч.

– Імовірність розпізнавання – не менш 90%.

– Припустимі кути установки відеокамери:

– По вертикалі – не більше 30º.

– По горизонталі – не більше 20º.

– Крен номерного знака ±15º.

– Освітленість – не менш 50 люкс.

– Розпізнавані номери – 7 типів (тло білий і жовтий).

– Вимога до бази даних – підтримка інтерфейсу ADO.

– Мінімальна конфігурація ПК – Celeron 1700 256MB Windows XP.

Блок розпізнавання людей

Система захвата осіб дозволяє виділяти тільки особи людей, вибирати найбільш виразне зображення з декількох варіантів і зберігати їх у базі даних. Створення бази осіб людей на прохідних підприємств, у місцях масового

скупчення людей (культурно-розважальні центри, вокзали, стадіони й т.д.) полегшує роботу з архівами при розслідуванні позаштатних ситуацій.

Далі система розпізнавання особи ідентифікує особистість і автоматизує пошук зображень у базах даних.

Блок динамічного спостереження за порушником

Системи динамічної цілевказівки аналізують зміни координат характерних точок об'єкта, наприклад центра ваги, кольору.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Розглянувши усі блоки функціональної схеми перейдемо до розгляду діаграми взаємодії процесів, які відбуваються у системі.

Блок подання сигналу тривоги

Відеокамера використовується разом з технічним засобом охорони для підтвердження факту спрацьовування останнього.

Блок документування подій на об'єкті

Матеріал відеоархівів може виявитися корисним як доказова база при розслідуванні несанкціонованих дій.

Блок розпізнання образів

Виділимо етапи при рішенні завдання розпізнавання зображень:

– Сприйняття поля зору.

– Сегментація.

– Нормалізація виділених об'єктів.

– Розпізнавання.

Виходячи із цього, використовуються наступні основні принципи:

– Принцип цілісності – розпізнаваний об'єкт розглядається як єдине ціле, що складається зі структурних частин, зв'язаних між собою просторовими відносинами.

– Принцип двонаправленості – створення моделі ведеться від зображення до моделі й від моделі до зображення.

– Принцип передбачення. Полягає у формуванні гіпотези про зміст зображення.

– Принцип цілеспрямованості, що включає сегментацію зображення й спільну інтерпретацію його частин.

– Нічого не робити без процедури розуміння (сприйняття поля зору).

– Принцип максимального використання моделі проблемного середовища, використання заздалегідь відомих, апріорних параметрів.

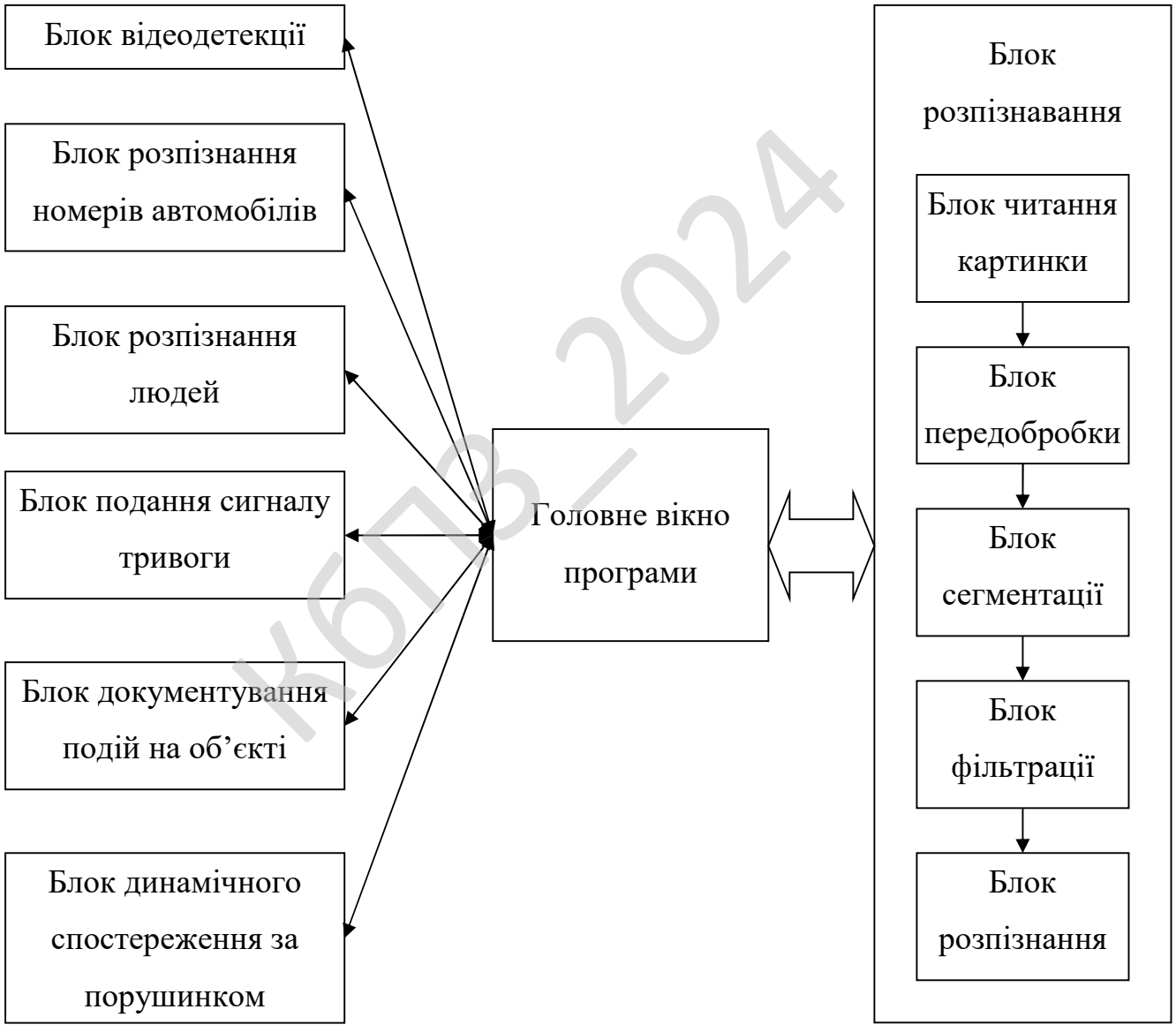

Рисунок 3.2 – Функціональна схема системи

Етап інтерпретації зображення не позначений чіткими границями й включається частково в процес сегментації й остаточно завершується на етапі розпізнавання.

Природно задатися питанням: а чи не можна брати зображення й послідовно порівнювати його з еталонами по ряду яких-небудь ознак? Але отут виникає ряд проблем і складностей:

– Тло. Як правило, зображення пред'являються на складному динамічному тлі.

– Орієнтація. Зображення еталона й вхідних зображень відрізняються положенням у поле зору.

– Перешкоди. Вхідні зображення не збігаються з еталонами за рахунок випадкових і локальних перешкод.

– Висвітлення. Відмінності вхідних і еталонних зображень виникає за рахунок зміни освітленості, підсвічування.

– Перетворення. Еталони й зображення можуть відрізняти складні геометричні перетворення.

Для рішення завдання в цілому й на окремих її етапах застосовуються різні методи сегментації, нормалізації й розпізнавання. На наведеній схемі зазначені основні процедури й методи обробки – від початкового етапу сприйняття поля зору за допомогою датчиків до кінцевого, котрим є розпізнавання. Коротко прокоментуємо наведену схему.

Передобробка

Процедура попередньої обробки використовується практично завжди після одержання інформації з датчика, і являє собою застосування операцій усереднення й вирівнювання гистограмм, різного типу фільтрів для виключення перешкод, що виникають у результаті апаратної дискретизації й квантування, а також придушення зовнішніх шумів.

Сегментація

Під сегментацією будемо розуміти процес пошуку однорідних областей на зображенні. Найбільше часто застосовуються методи, засновані на визначенні однорідних квітів або текстур, однак для довільного завдання цей етап не має чіткого алгоритму.

При існуванні стабільних розходжень у освітленості окремих областей поля зору застосовуються граничні методи. Приведемо приклад: для сегментації методом граничного розподілу необхідно одержати бінарне зображення з напівтонового. Для цього встановлюється деяке граничне значення. Після квантування функція зображення відображає елементи зображення з рівнем яскравості більше граничного в значення 1, менше граничного – 0.

При наявності стійкої зв’язності усередині окремих сегментів ефективні методи нарощування областей. Цей принцип полягає в тому, що відбувається угруповання сусідніх елементів з однаковими або близькими рівнями яскравості, а потім об'єднання їх в однорідні області. Один з типів – центроїдне зв'язування – припускає вибір стартових точок або за допомогою оператора, або автоматично. Ефективним представляється метод вододілів, заснований на пошуку локальних мінімумів з наступним угрупованням навколо них областей по зв’язності.

Метод виділення границь добре застосовувати, якщо границі досить чіткі й стабільні. Виділення контурних ліній найбільше часто використовується в системах технічного зору й засновано на обліку зміни яскравості й подальшому її порівнянні із граничної.

Перераховані методи служать для виділення сегментів за критерієм однорідних яскравостей. Всі перераховані принципи прийнятні з погляду обчислювальних витрат, проте, для кожного з них характерна не одиничність розмітки точок у реальних ситуаціях через необхідність застосування евристик.

Для опису й сегментації властивостей зображень, до яких відносяться однорідність, шорсткість, регулярність, застосовують текстурні методи, що підрозділяються умовно на дві категорії – статистичні й структурні.

Використання матриць збігів, формованих з вихідних зображень, з наступним підрахунком статистичних моментів і ентропії – є основа статистичного методу. При структурному підході будується множина багатокутників і виробляється дослідження на предмет загальних властивостей. Багатокутники із загальними властивостями поєднують в області.

Розпізнавання

Перейдемо до кінцевого етапу обробки зображення – розпізнаванню. Для цього етапу вхідними даними є зображення, отримані в результаті шумозаглушення й процесу сегментації. Як правило, вони відрізняються від еталонних геометричними і яскравостними перекручуваннями, а також збереженими шумами.

Для рішення завдань розпізнавання застосовуються, в основному, чотири підходи:

1. Кореляційний. Підхід, заснований на прийнятті рішень за критерієм близькості з еталонами. В основному застосовується при виявленні й розпізнаванні зображень у системах навігації, спостереження, промислової роботизації. Найбільш трудомісткий підхід з погляду споживання обчислювальних ресурсів. Має на увазі під собою багатокрокову кореляцію при повністю заданому еталоні, шляхом сканування вхідного поля зору. Інакше кажучи, відбувається перебір всіх вхідних сигналів і порівняння їх з еталонним.

2. Ознаковий. Такі методи засновані на переході в простір ознак, а відповідно, вимагають значно менших обчислювальних потужностей. Залежно від поставленого завдання, виконується кореляційна обробка ознак, отриманих від еталона й вхідного зображення. При цьому виникає завдання об'єднання й комплексної обробки ознак різної розмірності (метричних, статистичних, логічних, текстурних і т.д.), отриманих різними вимірювальними засобами з метою рішення завдання розпізнавання.

3. Кореляційно-ознаковий метод має на ввазі під собою обробку статистичними методами ознак, отриманих у такий спосіб. Споконвічно

застосовується метод приватних кореляцій для різних фрагментів еталонного зображення, а потім у сигнальному просторі отримані кореляційні коефіцієнти розглядаються як ознаки.

Основною проблемою в ознакових методах становить вибір ознак. При цьому виходять із природних правил:

1. Ознаки зображень одного класу можуть розрізнятися лише незначно (за рахунок впливу перешкод, шумів).

2. Ознаки зображень різних класів повинні істотно розрізнятися.

3. Набір ознак повинен бути мінімальним (від їхньої кількості залежить надійність, складність, швидкість обробки).

1. Синтаксичний метод заснований на одержанні структурно-граматичних ознак, коли в зображенні виділяються непохідні елементи – ознаки. Уводяться правила з'єднання цих елементів, однакові для еталона й вхідного зображення. Аналіз отриманої в такий спосіб граматики забезпечує прийняття рішень.

Кожний з підходів у розпізнаванні має право на існування. Більше того, у рамках кожного підходу є свої конкретні алгоритми, що мають певну область застосування, що залежить від характеру розходжень вхідних і еталонних зображень, від перешкодової обстановки у полі зору, вимог до обсягів обчислень і швидкості прийняття рішень. Ознакові й синтаксичні методи – найпоширеніші в теорії розпізнавання графічних образів.

2. Нормалізація. Завдання нормалізації зображення – це завдання визначення параметрів геометричних перетворень, яким піддалося зображення, з метою їхньої компенсації. Компенсація може проводитися за рахунок зміни просторового положення системи уведення зображення, або алгоритмічно шляхом застосування зворотного перетворення до вхідного зображення. Процедура перетворень виробляється за допомогою операторів нормалізації – нормалізаторів, а обчислення параметрів виконується функціоналами, що діють на множини зображень.

Методи нормалізації при розпізнаванні займають проміжне місце між кореляційними й ознаковими алгоритмами. На відміну від ознакових, при нормалізації зображення не виключається з розгляду, а тільки заміщається зображенням того ж класу еквівалентності. У той же час, на відміну від кореляційних методів, множина вхідних зображень заміняється безліччю нормалізованих зображень. Кожна нормалізована картинка, загалом кажучи, перебуває набагато ближче до свого еталона (з позиції групових перетворень), що значно скорочує кількість кореляцій на завершальному етапі розпізнавання.

Найбільший інтерес у цей час у теорії нормалізації представляють послідовні методи, засновані на поетапному обчисленні параметрів складних перетворень і застосуванні часткових нормалізаторів на кожному етапі.

Додаткові проблеми при рішенні завдання зорового сприйняття роботизованих систем у порівнянні із традиційними завданнями обробки й розпізнавання зображень:

– Опис середовища функціонування. Необхідно комплексний опис на основі обліку значного обсягу апріорної інформації, створення моделі проблемного середовища, на відміну від завдання виділення конкретних ознак або виділення окремих характеристик. Аналіз 3D-об'єктів і облік законів перспективи. Необхідно враховувати не тільки проекції реальних об'єктів, але й проводити аналіз у плані визначення об'ємних просторових відносин.

– Аналіз множини довільно розташованих об'єктів, виділення конкретних предметів при відсутності або неможливості визначити деякі ознаки (наприклад, коли на плоскопаралельної проекції видна тільки частина контуру необхідного об'єкта, але унікальна й достатня для його ідентифікації).

– Необхідність роботи в реальному динамічному середовищі. У загальному випадку, відсутність постійного завдання й необхідність оперативно реагувати на виникаючі завдання.

– Необхідність погодженості при взаємодії в реальному часі декількох підсистем робота.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання бакалаврського проектування, наведена на рисунку 3.3. Згідно цієї схеми процеси взаємодіють наступним чином.

Спершу запускається процес завантаження головного вікна програми.

Він взаємодіє з наступними процесами:

– Процесом перегляду розпізнаної та збереженої інформації.

– Процесом ввімкнення/вимкнення камери.

Процес перегляду розпізнаної та збереженої інформації взаємодіє, у свою чергу, з процесом документування подій.

Процес документування подій взаємодіє з процесом журналювання розпізнаних образів.

Процес журналювання розпізнаних образів взаємодіє з наступними процесами:

– Процес збереження номерів автомобілів.

– Процес збереження облич людей.

– Процес класифікації та опису об’єкту.

Процес класифікації та опису об’єкту взаємодіє з наступними процесами:

– Процес динамічного спостереження.

– Процес читання зображення з камери.

Процес читання зображення з камери взаємодіє з наступними процесами:

– Процес аналізу зображення.

– Процес ввімкнення/вимкнення камери.

Процес аналізу зображення взаємодіє з процесом розпізнавання образів на основі кластерного аналізу.

Процесом ввімкнення/вимкнення камери взаємодіє також з процесом виведення зображення з камери.

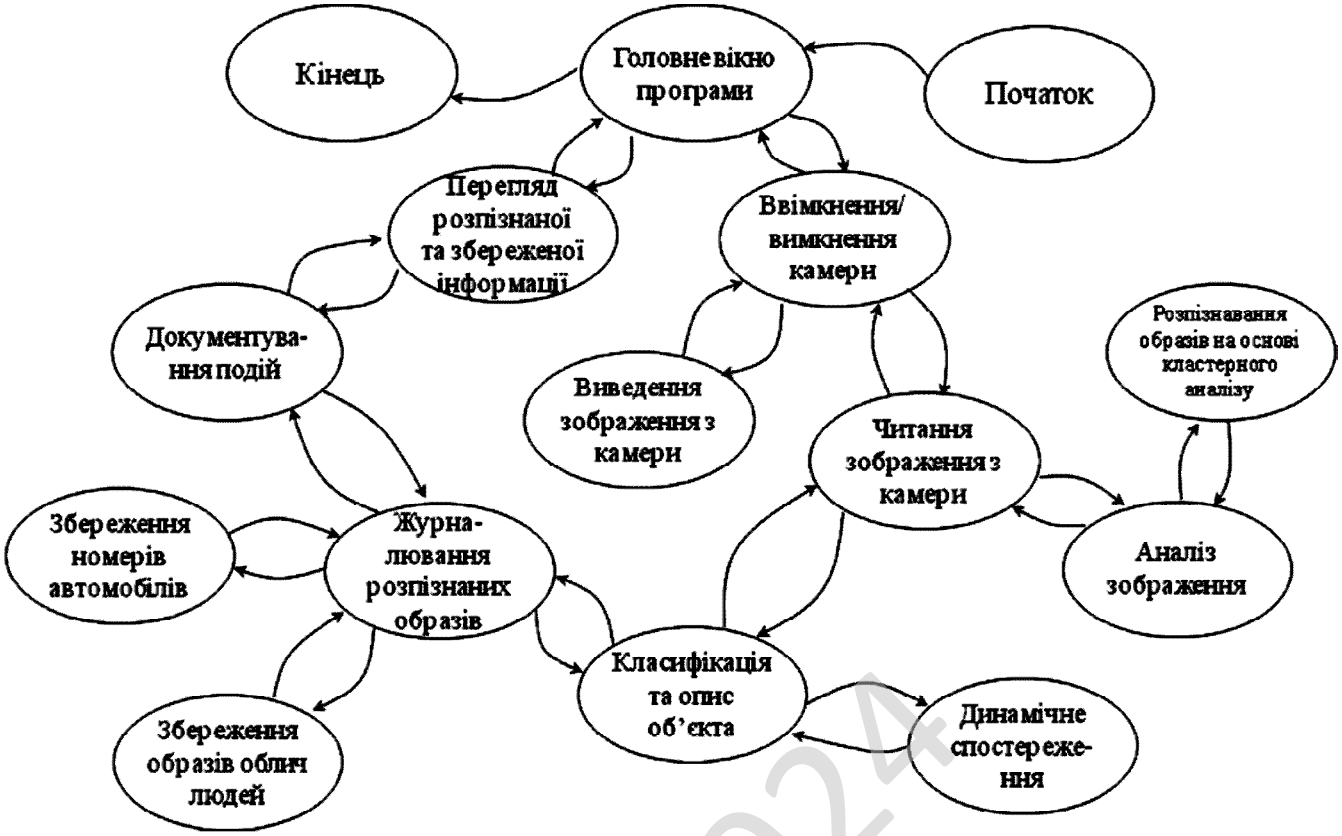

Рисунок 3.3 – Діаграма взаємодії процесів

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 ПРОЕКТУ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ПРАВИЛЬНІСТЬ ПРОЕКТНИХ РІШЕНЬ

4.1 Блок–схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків.

Спершу відбувається виведення основного вікна програми. Після цього відбувається підключення камер.

Якщо камер декілька, то відбувається вибір активної камери, з якої буде відбуватися розпізнання образів, тобто автомобільних номерів, або облич.

Після цього відбувається введення параметрів відео, які дозволять записувати картинку більше, або менше якісно.

Наступним кроком є виведення зображення з камери на екран.

Якщо необхідно розпізнати зображення, то програма виконує наступні дії:

– Запускається підпрограмма розпізнання образів нейронною мережею.

– Виводяться результати розпізнавання.

– Зберігаються результати розпізнавання.

Якщо цього робити немає необхідності, тоді користувача обирає, чи потрібно йому записати відео з камери.

Якщо потрібно, то відбувається запис відео.

Якщо ні, то користувач обирає, чи потрібно йому змінити активну камеру.

Якщо потрібно, то відбувається зміна активної камери.

Після цього користувач закінчує роботу з програмою.

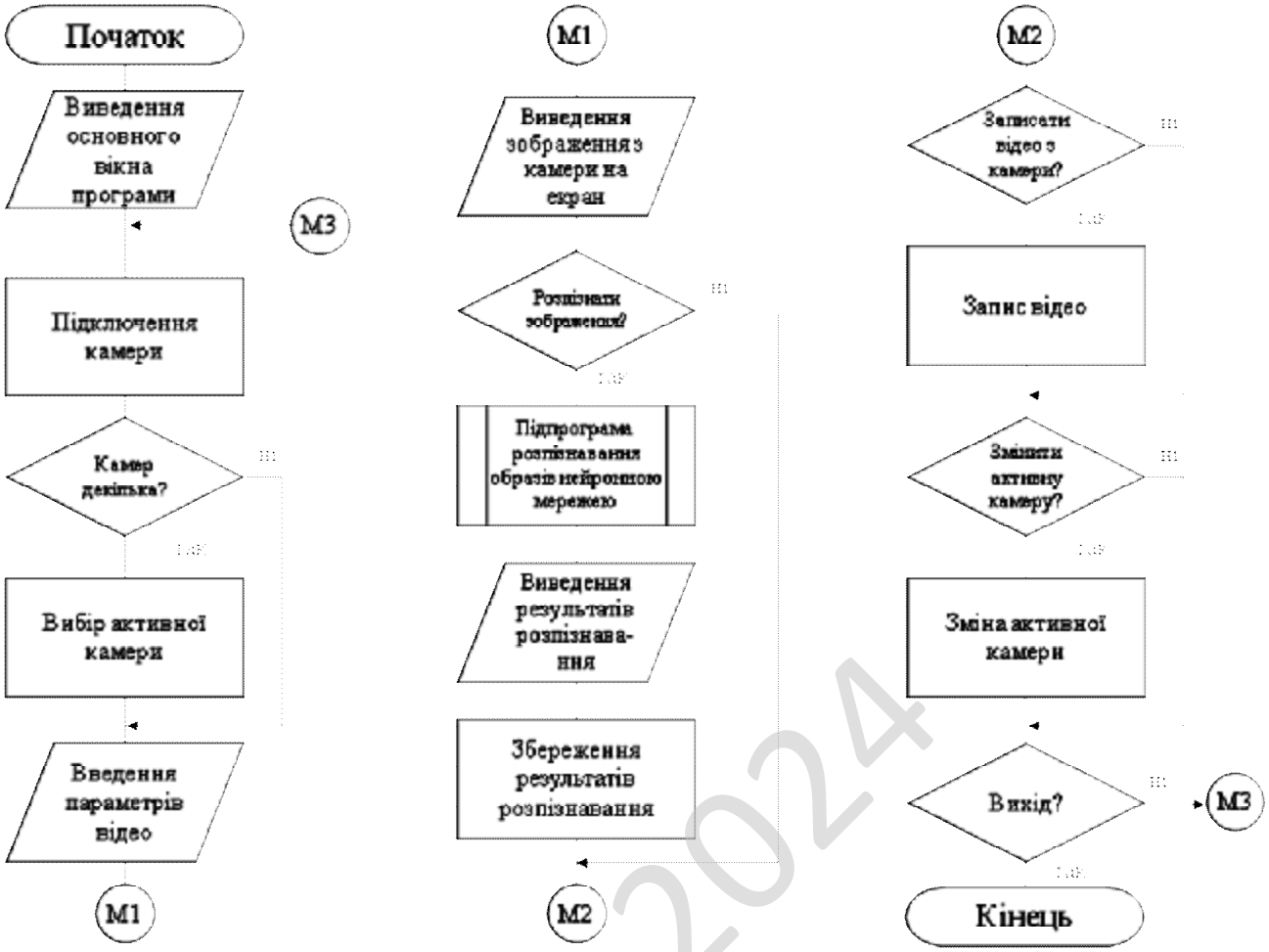

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 зображена блок-схема підпрограми розпізнавання образів нейронною мережею.

Вона працює наступним чином.

Спершу визначається, чи пройшла нейромережа процес навчання.

Якщо ні, тоді відбувається навчання нейромережі.

Якщо ж пройшла, тоді відбувається послідовне виконання наступних ітерацій:

– Вводиться значення якості розпізнавання.

– Вводиться метод розпізнавання.

– Отримується зображення від камери.

– Відбувається бінарізація.

– Відбувається сегментація.

– Відбувається вихоплювання та обчислення ознак.

– Відбувається поворот та повторне вихоплювання.

– Відбувається стиснення матриці.

– Відбувається розпізнання матриці.

– Відбувається формування результату розпізнання.

Після цього підпрограма закінчує свою роботу.

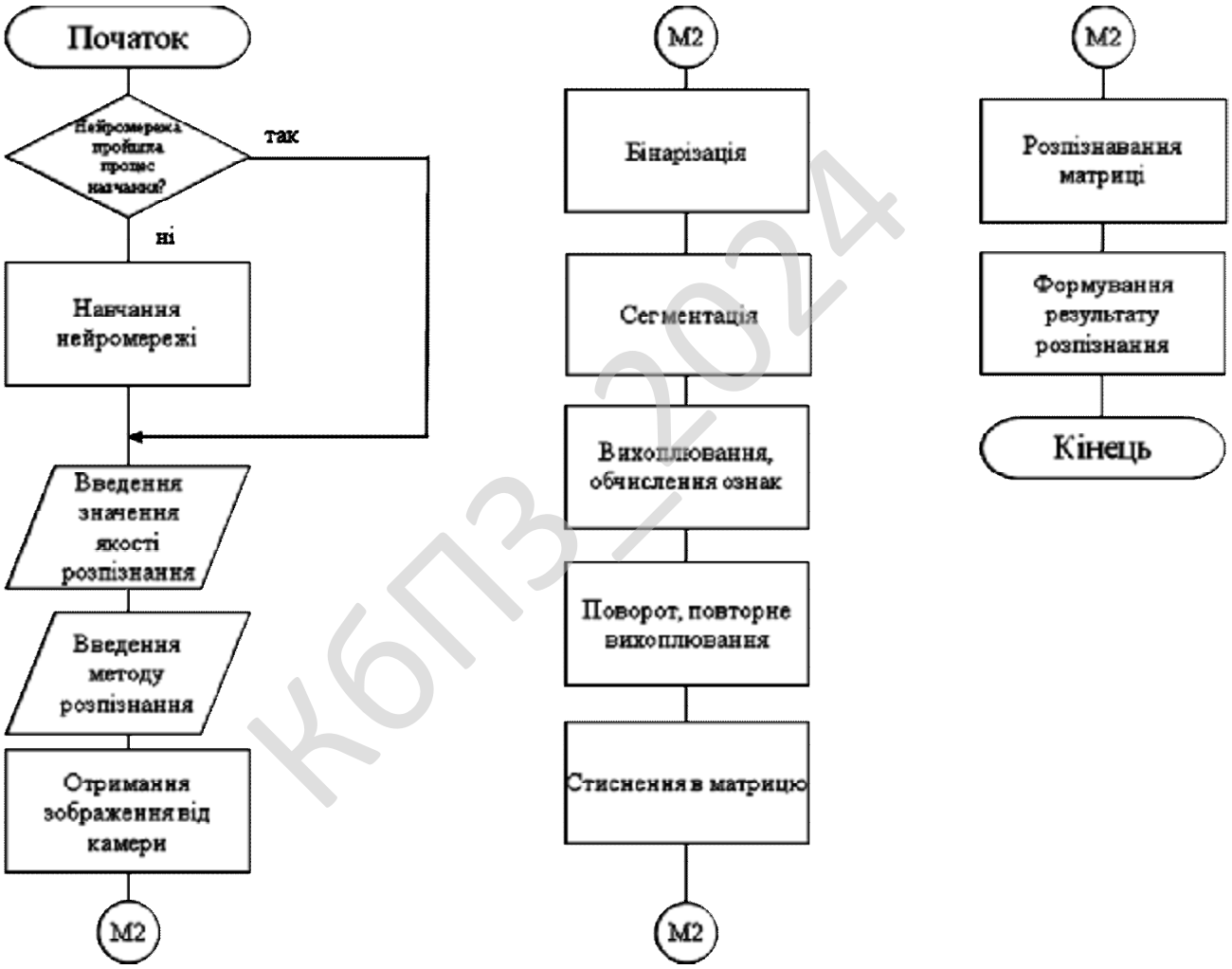

Рисунок 4.2 – Блок-схема підпрограми розпізнавання образів нейронною мережею

Опишемо алгоритм роботи програми на прикладі розпізнавання букв та цифр в номерах автомобілів, або рукописних прописних українських букв і цифр на основі методу порівняння з еталонними зображеннями відповідних символів.

Нижче наведені основні моменти реалізації пропонованого алгоритму.

Крок 1. Створення канви для малювання й формування її образа в пам'яті.

Як канву використовуємо клас TBitmap (для простоти роботи з бітмапом використовуємо режим 1 байт на піксель, тобто TBitmap.PixelFormat := pf8bit), візуалізуємо його на TPaintBox, відображаємо в пам'яті за допомогою структури:

```
type
  Mas = PByteArray;
var
  Mas : array of Mas // масив пікселів,
  {
  де Mas[ y-коорд][ x-коорд] = номер кольору в палітрі квітів (при 8 біт/піксель).
  Відображення здійснюємо з використанням TBitmap.ScanLine (швидко й просто):
  SetLength(Mas, TBitmap.Height);
  for j := 0 to TBitmap.Height - 1 do
    Mas[j] := TBitmap.ScanLine[j];
}
```

Тепер з картинкою у вигляді матриці XxY можна робити все що завгодно.

Крок 2. Формування масиву еталонних зразків символів.

Еталонні зразки будемо формувати на основі матриці розміром 16x16. Для цього розробимо процедуру генерації такої матриці по довільному зображенню еталона.

Процедура function Create_16x16(Img : TBitmap) : TMas16x16 одержує як параметр посилання на картинку, на якій намальований еталон символу (у нашім випадку – програмно), повертає наведену матрицю розміром 16x16.

Коротко пояснимо роботу процедури (більш повно див. коментарі в програмі).

Одержуємо посилання на бітмап і здійснюємо його відображення в пам'яті.

Обчислюємо координати границь (описаного прямокутника) образа (еталонного або розпізнаваного) шляхом сканування рядків/стовпців. При цьому тут і при подальшому аналізі зображення передбачається, що символ намальований чорним кольором (№0 у палітрі квітів) і відповідно всі значущі пікселі мають значення 0.

```
for j := 0 to Img.Height - 1 do                    // Зверху
  begin
    for i := 0 to Img.Width - 1 do
      if Mas[j][i] = 0 then
      begin yTop := j; break; end;
    if yTop = j then break;
  end;
  for j := Img.Height - 1 downto 0 do               // Знизу
  begin
    for i := 0 to Img.Width - 1 do
      if Mas[j][i] = 0 then
      begin yBottom := j + 1; break; end;
    if yBottom = j + 1 then break;
  end;
  for i := 0 to Img.Width - 1 do                    // Зліва
  begin
    for j := 0 to Img.Height - 1 do
      if Mas[j][i] = 0 then begin xLeft := i; break; end;
    if xLeft = i then break;
  end;
  for i := Img.Width - 1 downto 0 do                // Праворуч
  begin
    for j := 0 to Img.Height - 1 do
      if Mas[j][i] = 0 then begin xRight := i + 1; break; end;
    if xRight = i + 1 then break;
  end;
```

Для подальшого аналізу буде потрібно якийсь критерій, по якому буде вироблятися згортка вихідного зображення символу в матрицю 16x16. Таким критерієм був обраний загальний відсоток заповнення – відношення кількості значимих пікселів (з яких складається символ) до загальної кількості пікселів в описаному навколо вихідного зображення прямокутнику. Даний параметр може впливати на якість розпізнавання, причому якщо він більше 1 для розпізнаваного

символу буде відповідати меншу кількість можливих альтернатив, при значенні меншому 1 – навпаки. У нашому випадку коефіцієнт виправлення прийнятий рівним 0,99.

```
nSymbol := 0;
for j := yTop to yBottom do
for i := xLeft to xRight do
  if Mas[j][i] = 0 then inc(nSymbol);
Percent := nSymbol / ((yBottom - yTop)*(xRight - xLeft));
Percent := 0.99*Percent;
```

Далі розбиваємо прямокутник із зображенням символу на 16x16 осередків шляхом розподілу сторін нового осередку на 2. Запам'ятовуємо відносні координати кожного осередку й приступаємо до заповнення матриці 16x16. Приймаємо як критерій загальний відсоток заповнення. Якщо в аналізованому осередку відсоток заповнення більше, ніж загальний відсоток – відповідний елемент матриці 16x16 установлюється в 1, у противному випадку – в 0.

Інша частина алгоритму стосується питань малювання на TBitmap букв або цифр (у циклі), запам'ятовування в масиві матриць 16x16, що відповідають кожному еталонному символу.

Крок 3. Розпізнавання мальованих (від руки) символів.

Розпізнавання здійснюємо шляхом порівняння матриці 16x16 розпізнаваного символу з матрицею еталона (шляхом перебору наявних у наявності). Порівняння робимо поелементно за допомогою оператора XOR. Результат – матриця 16x16, що містить одиниці в місцях розбіжностей тест-символу й еталона. Шляхом підрахунку кількості розбіжностей формуємо вектор, що містить цю інформацію для кожного еталонного символу, і робимо сортування його елементів по зростанню кількості розбіжностей.

Параметр (1 – Result[i]/256)*100%, де Result[i] – кількість розбіжностей для i – го символу, показує "імовірність" відповідності образа конкретному символу.

В багатьох областях техніки використовуються різні автомати й пристрої, що більш-менш вдало вирішують завдання розпізнавання (це й автомат для сортування поштових конвертів по індексу, і різні системи аналізу супутникових

знімків, і голосовий виклик вашого мобільника, і багато чого іншого). В описаному вище алгоритмі , є один сильний недогляд: цю програму неможливо навчати, тому що у нього порівняння відбувається тільки з одним набором еталонів. Запропоную алгоритм, яким можна навчати. Особливу пікантність алгоритму надає той факт, що його математичне обґрунтування було запропоновано радянськими математиками на початку 60-х років.

Зрозуміло, для простоти ми будемо розглядати тільки чорно-білі зображення. Нехай у нас малюнок складається всього із двох пікселів. Тоді безліч всіх об'єктів, яку можна буде зобразити (універсальна безліч), складається із чотирьох об'єктів: (0,0), (0,1), (1,0), (1,1), де 1 – чорний піксель, 0 – білий.

Всі об'єкти універсальної безлічі можна розмістити у вершинах одиничного квадрата, таким чином, безлічі фігур, зображених на двопіксельному полі, може бути зіставлена безліч точок у двовимірному просторі. Ребру цього квадрата буде відповідати перехід від одного зображення до іншого. Для переходу від (1,1) до (0,0) потрібно буде пройти два ребра, для переходу від (0,1) до (0,0) – одне. Відзначимо, що число ребер у нашому переході – це кількість незбіжних пікселів двох зображень. Вивід цікавий: відстань від одного малюнка до іншого дорівнює числу незбіжних пікселів у них. Ця відстань називається відстанню за Хемінгом.

Тепер уявимо собі, що в нас малюнок складається із трьох пікселів. Коди зображень тоді будуть складатися із трьох значень, універсальна безліч – з восьми елементів, які ми розмістимо у вершинах одиничного куба. Але принципово нічого не зміниться, і відстань за Хемінгом обчислюється так само. У прикладеній тестовій програмі використовується малюнок 50x70 = 3500 пікселів. Легко зміркувати, що в цьому випадку код будь-якого зображення складається з 3500 значень, універсальна безліч – з 23500 = 4,027 * 101053 елементів, які ми будемо розміщати у вершинах одиничного 3500-мірного куба. Уявити собі такий 3500-мірний куб нелегко, але зміст від цього не міняється абсолютно. Основна ідея полягає в тому, що в цьому багатомірному кубі зображення, що відповідають

якомусь певному образу, лежать недалеко друг від друга. Ця ідея одержала назву "Гіпотеза про компактність образів".

Тепер можна сформулювати завдання: потрібно універсальну безліч розбити на "шматки", компактні безлічі, кожні з яких відповідає образ.

Програмі в процесі навчання повідомляються зображення (точки багатомірного куба) і вказівки, до якого образа кожне зображення відноситься. При розпізнаванні програма просто дивиться, у яку з відомих компактних областей потрапило вхідне зображення. Швидше за все, всі зазначені машині зображення ляжуть більш-менш компактно, тому універсальну безліч буде можна розділити. Властиво розділяти універсальну безліч ми не будемо, а будемо користуватися деякою характеристикою, що показує далекість одного малюнка (точки у вершині багатомірного куба) до групи таких же зображень. Як міра далекості малюнка від групи малюнків використовується потенціал.

Відомо, що електричний заряд створює навколо себе поле, однієї з характеристик якого є потенціал. У будь-якій точці він може бути обчислений за формулою:

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

де a – деякий постійний коефіцієнт, q – величина заряду, R – відстань від даної точки до заряду. Якщо електричне поле утворене двома або більше зарядами, то потенціал у даній точці дорівнює сумі потенціалів кожного заряду. Аналогія очевидна – кожний малюнок, на якому програма навчалася, створює в просторі універсальної безлічі потенціал. Після навчання програмі дають розпізнати який-небудь малюнок (точку у вершині багатомірного куба), програма обчислює потенціал, створюваний у цій точці всіма об'єктами образа "а", образа "б"... на які програму вчили й розпізнаваний малюнок відноситься до образа, що створив найбільший потенціал.

Отже, при запуску програми в масив Data: array of array [0..9] of TBitmap; записуються цифри від 0 до 9, написані наступними шрифтами: Arial, Century

Gothic, Courier New Cyr, Goudy Old Style і Times New Roman – усього п'ять комплектів (можна легко збільшити). Всі ці зображення були збережені мною й дбайливо викладені в папку \fonts.

```
Procedure LoadData;
var i,j:integer;
path:string;
begin
SetLength(Data,5);
for i := 0 to 4 do
 begin
 path := ExtractFilePath(Application.ExeName)+'\fonts\';
 case i of
  0: path  := path + 'Arial\';
  1: path  := path + 'Century Gothic\';
  2: path  := path + 'Courier New Cyr\';
  3: path  := path + 'Goudy Old Style\';
  4: path  := path + 'Times New Roman\';
 end;
 for j := 0 to 9 do
  begin
  Data[i,j] := TBitmap.Create;
  Data[i,j].LoadFromFile(path + IntToStr(j) + '.bmp');
  end;
 end;
end;
```

Після завантаження еталонних зображень користувач малює на поле розміром 50x70 пікселів цифру, що програма буде розпізнавати. При натисканні кнопки "розпізнати" вираховуються відстані від розпізнаваного малюнка до кожного з еталонних (відстань за Хемінгом).

```
function Compare( b1,b2:TBitmap):integer;
var i,j,count:integer;
begin
count := 0;
for i := 0 to 49 do
 for j := 0 to 69 do
  if b1.Canvas.Pixels[i,j] <> b2.Canvas.Pixels[i,j] then
    inc(count);
Result := count;
end;
```

Знаючи цю відстань R, легко обчислити потенціал, створюваний кожним еталонним малюнком у точці, що відповідає намальованому користувачем зображенню. Я небагато змінив формулу розрахунку потенціалу, щоб уникнути розподілу на 0 у випадку R=0 і для кращого сприйняття домножив на 1 000 000:

Потенціали, створювані нулями всіх накреслень, підсумуються в p[0], одиницями – у p[1] і так далі.

```
for i := 0 to 4 do
  for j := 0 to 9 do
   begin
   r := Compare(Image1.Picture.Bitmap,Data[i,j]);
   p[j] := p[j] + 1000000/(1+r*r);
   end;
```

Після всього цього залишається знайти, якому образу відповідає найбільший потенціал.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації. Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена

версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

Як і інші учасники конкурсу AES, Crypton призначений для шифрування 128-бітових блоків даних[2]. При шифруванні використовуються ключі шифрування для декількох фіксованих розмірів – від 0 до 256 біт з кратністю 8 бітів. Структура алгоритму Crypton – структура «Квадрата» – багато в чому схожа на структуру алгоритму Square, створеного в 1997 році. Криптографічні перетворення для алгоритмів з даною структурою можуть бути виконані як для цілих рядків і стовпців масиву, так і над окремими його байтами. (Варто зазначити, що алгоритм Square був розроблений авторами майбутнього переможця конкурсу AES – авторами алгоритму Rijndael – Вінсентом Ріджменом і Джоан Дейменом.)

Шифрування

Алгоритм Crypton являє 128-бітовий блок шифруємих даних у вигляді байтового масиву 4×4, над якими в процесі шифрування проводиться кілька раундів перетворень. У кожному раунді передбачається послідовне виконання наступних операцій:

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

Алгоритм Crypton використовує 4 таблиці замін. Кожна з яких заміщає 8-бітне вхідне значення на вихідне такого ж розміру.

Лінійне перетворення π

Тут використовується 4 спеціальні константи. Ці константи об'єднані в маскуючі послідовності

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

Дана операція є побітовим складанням всього масиву даних з ключем раунду. Зауважимо, саме 12 раундів шифрування рекомендується автором алгоритму Чьо Хун Лімом, проте сувора кількість раундів не встановлена.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми.

З нього ми бачимо, що існують такі основні елементи меню користувача:

– Файл.

– Відео.

– Розпізнавання.

– Архів.

– Параметри.

– Довідка.

Крім блока меню, існують такі наступні блоки:

– Вікно стану камер.

– Кнопки швидкого доступу до елементів програми.

– Вікно доступних діалогів.

– Вікно вибору відеорежимів.

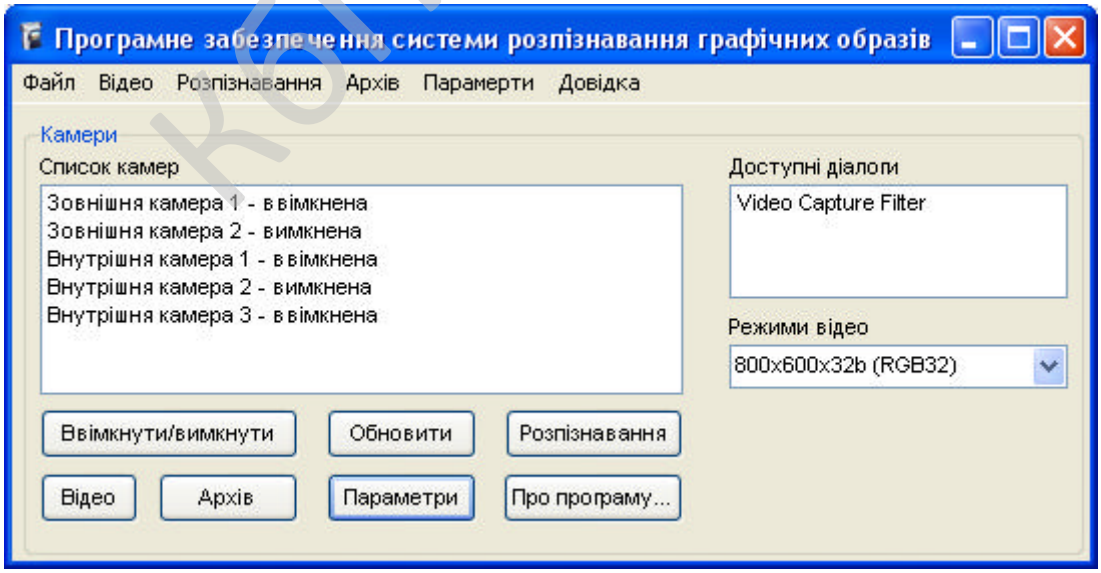

Рисунок 5.1 – Головне вікно програми

Кнопки швидкого доступу до елементів програми дозволяють виконати наступні операції:

– Ввімкнути/вимкнути.

– Обновити.

– Розпізнавання.

– Відео.

– Архів.

– Параметри.

– Про програму.

На рисунку 5.2 зображено вікно розпізнавання графічних образів.

У цьому вікні можливо встановити наступні параметри розпізнання:

– Процес навчання.

– Метод розпізнання.

– Кількість епох для навчання.

– Метод розпізнавання.

– Координати.

– Момент та альфа.

– Крок навчання.

– Тип об’єкта.

– Інформація про образ.

– Точність розпізнання.

Також сюди винесені кнопки швидкого доступу:

– Розпізнати.

– Призупинити розпізнавання.

– Навчати.

– Зберегти ваги.

– Згадати ваги.

– Додати в БД.

– Очистити БД.

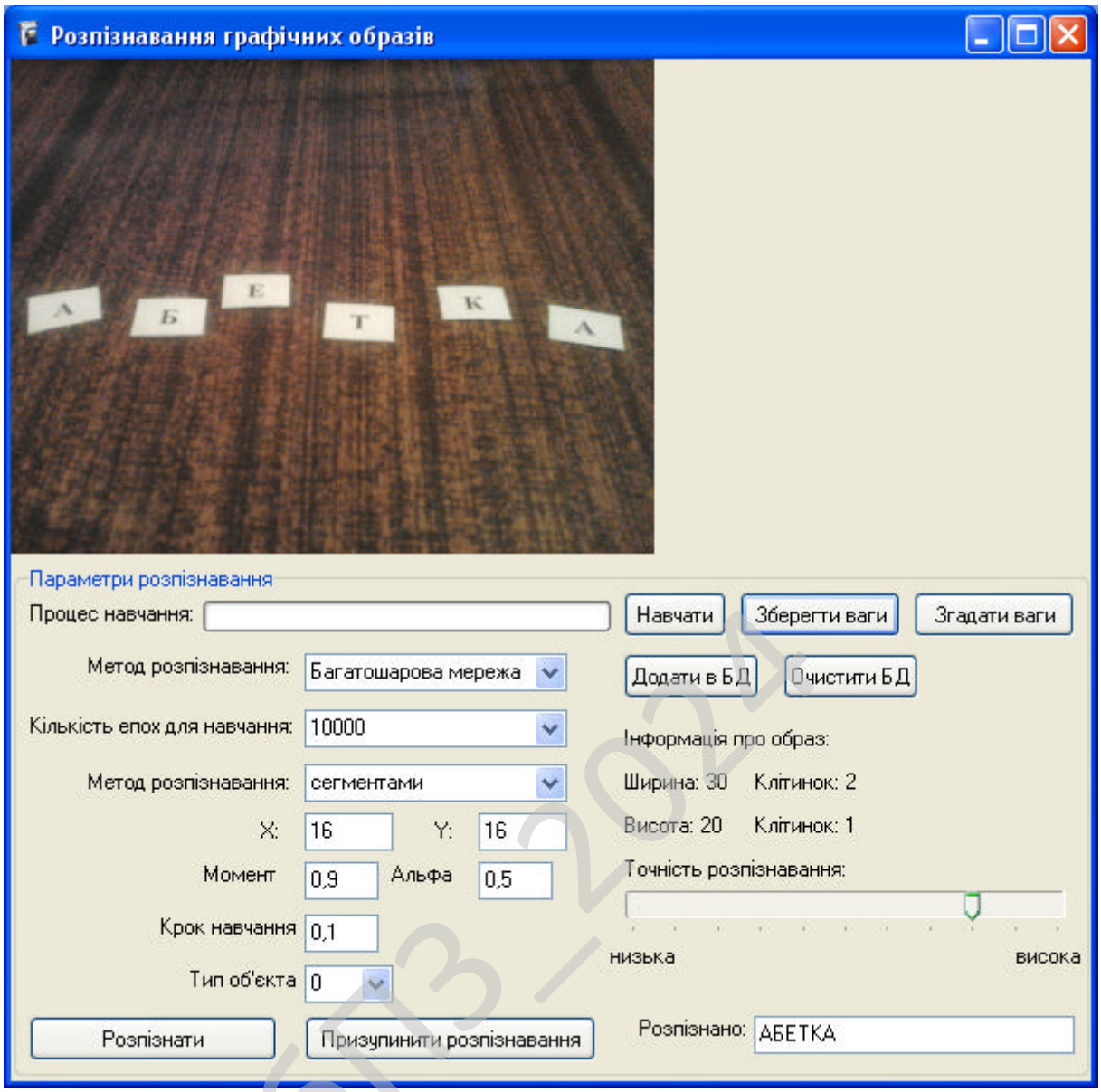

Рисунок 5.2 – Вікно розпізнавання графічних образів

На рисунку 5.3 зображена довідка, у якій наведена наступна інформація:

– Тема.

– Розроблювач програмного продукту.

– Керівник.

– Місце виконання проекту.

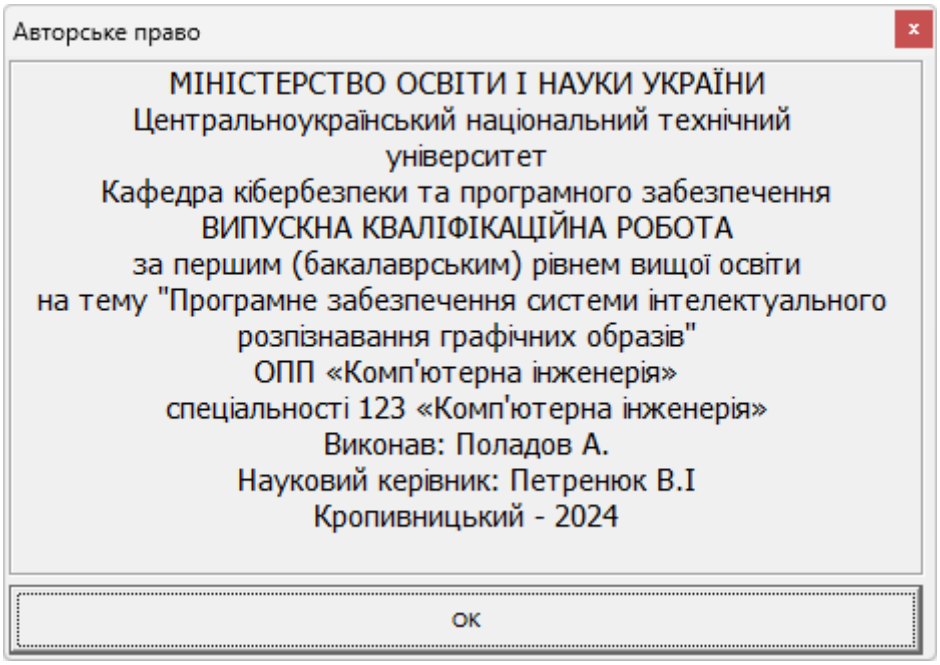

Рисунок 5.3 – Довідка

6 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти, призначено для системи інтелектуального розпізнавання графічних образів.

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

Рішення завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем інтелектуального розпізнавання графічних образів.

– Досліджена система інтелектуального розпізнавання графічних образів.

– На основі отриманих результатів досліджень створена програмна реалізація системи інтелектуального розпізнавання графічних образів.

Розроблені під час виконання випускної кваліфікаційної роботи за першим (бакалаврським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання інтелектуального розпізнавання графічних образів.

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 Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані призначені для системи інтелектуального розпізнавання графічних образів. 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 CRYPTON.

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. Malyukov V., Bebeshko B., Lakhno V., Smirnov O., Malyukova I., Mohylnyi H. «Managing the Purchase-Sale Process of Digital Currencies Under Fuzzy Conditions». *Lecture Notes in Networks and Systems,* 2023, 729 LNNS, pp. 104–112.

2. 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.

3. 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.

4. Smirnov, O., Karapetyan, A., Fedorov, E., «Creating Neural Network and Single Solution Human-Based Metaheuristic Methods of Solving the Traveling Salesman Problem». CEUR Workshop Proceedings, Volume 3312, 2022, pp. 47-58.

5. 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, pp. 1-12.

6. 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.

7. Smirnov, O., Lakhno, V., Akhmetov, B., Chubaievskyi, V., Khorolska, K., Bebeshko, B. «Selection of a Rational Composition of İnformation Protection Means Using a Genetic Algorithm». In: Rajakumar, G., Du, KL., Vuppalapati, C., Beligiannis, G.N. (eds) Intelligent Communication Technologies and Virtual Mobile Networks. Lecture Notes on Data Engineering and Communications Technologies, vol 131. 2023. Springer, Singapore. pp. 21-34.

8. Kuznetsov, A., Oleshko, I., Chernov, K., Bagmut, M., Smirnova, T. «Biometric authentication using convolutional neural networks». Lecture Notes in Networks and Systems. Volume 152, 2021, Pages 85-98.

9. 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.

10. Smirnov O., Neskorodieva T., Fedorov E., Rymar P. «Neural Network Modeling Method of Transformations Data of Audit Production with Returnable Waste». CEUR Workshop Proceedings Volume 3101, 2021, Pages 192-207.

11. 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.

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

13. 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.

14. 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.

15. 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.

16. 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.

17. 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.

18. 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.

19. 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.

20. Smirnov, O., Ulichev, O., Meleshko, Y., Khokh, V., Goncharenko, I. «Method of Choosing Objects for Informational Influence in Social Networks during Information Campaign Based on the Analytic Hierarchy Process». CEUR Workshop Proceedings, Vol 2588, P. 215-227, 2019.

21. 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.

22. 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.

23. Smirnov, O., Kuznetsov, A., Kiian, A., Zamula, A., Rudenko, S., Hryhorenko, V., «Variance Analysis of Networks Traffic for Intrusion Detection in Smart Grids», 2019 IEEE 6th International Conference On Energy Smart Systems (2019 IEEE ESS), Kyiv, Ukraine April 17-19, 2019 P. 353-358.

24. 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.

25. 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.

26. Smirnov, S., Bulekbaeva, G., Kikvidze, O.G., Lakhno, V., Brzhanov, R., Tabylov, A. «Computer simulation in the MathCAD package of plastic deformation of the deposited layer on the flat surface of the part». Journal of Theoretical and Applied Information Technology Volume 97, Issue 20, 2019, Pages 2467-2484. (Scopus).

27. 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.

28. 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.

29. Смірнов О.А., Смірнова Т.В., Буравченко К.О., Кравченко С.С., Горбов В.О., «Хмарна система підтримки прийняття рішень технологічного процесу відновлення поверхонь конструкцій і деталей машин». Сучасні інформаційні системи. 2021. Т. 5, № 4. С. 79-95

30. Смірнов, О.А., Усік П.С., Полігенько О.О., Одарченко Р.С., Терещенко Л.Ю. «Інформаційна технологія та програмне забезпечення для підвищення ефективності планування підсистеми базових станцій стільникового зв'язку». Проблеми телекомунікацій. № 1(26). С. 83-96. 2020.

31. Смірнов О.А., Усік П.С., Миронець І.В., Буравченко К.О., Якименко Н.М. «Метод підвищення ефективності розподіленої обробки даних у комп'ютерних системах операторів стільникового зв'язку» Вісник Черкаського державного технологічного університету. Технічні науки. №4. С. 103-110. 2020.

32. 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.

33. Смірнов, С.А., Смірнова, Т.В., Минайленко, Р.М., Доренський, О.П., Сисоєнко С.В. «Хмарна автоматизована система інтелектуальної підтримки прийняття рішень для технологічних процесів». Вісник Черкаського державного технологічного університету. Технічні науки. №4, 2020, С. 84-92.

34. 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.

35. Smirnov, O., Kuznetsov, A., Kuznetsova., K. Synthesis of Discrete Signals with Improved Correlation Properties. Монографія: In.: ISCI'2019: Information Security in Critical Infrastructures. Collective monograph. Edited by Ivan D. Gorbenko and Alexandr A. Kuznetsov, ASC Academic Publishing, USA, 2019, pp. 281-299. – ISBN: 978-0-9989826-8-7 (Hardback), ISBN: 978-0-9989826-9-4 (Ebook).

36. О.А. Смірнов, Т.В. Смірнова, О.М. Дрєєв, Є.К. Солових, «Методи оптимізації технологічних процесів відновлення сталевих покриттів», Shipbuilding & marine infrastructure / Суднобудування і морська інфраструктура № 1 (11). с. 48-57, 2019.

37. Смірнова Т.В., Солових Є.К., Смірнов О.А., Дрєєв О.М. Побудова хмарних інформаційних технологій оптимізації технологічного процесу відновлення та зміцнення поверхонь деталей. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 184-194, 2019.

38. Смірнова Т.В., Дрєєв О.М., Смірнов О.А. Експертна система оптимізації процесу відновлення та зміцнення поверхонь деталей типу «вал» електродуговим напиленням. Системи управління, навігації та зв'язку, № 2 (54). С. 149-154, 2019.

39. Смірнов О.А., Котелянець В.В. Стійкі до колізій стохастичні моделі функціонування безпроводових сенсорних мереж. Вісник інженерної академії України, №3, с. 145-152, 2018

40. Смірнов О.А., Лисенко І.А. Інформаційна технологія проектування тестових наборів з урахуванням вимог до програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 4 (44). - Полтава: ПолтНТУ. - 2017. - С. 112-115.

41. Смірнов О.А., Коваленко О.В. Використання псевдобулевих методів бівалентного програмування для управління ризиками розробки програмного забезпечення. Системи управління, навігації та зв'язку. – Випуск 1 (37). - Полтава: ПолтНТУ. - 2016. - С. 98-103

42. Смірнов О.А., Лисенко І.А. Формалізація процесу проектування тестових наборів. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 3 (48). - Харків: ХУПС. - 2016. - С.96-100.

43. Смірнов О.А., Лисенко І.А. Удосконалення методу перевірки коректності таблиць рішень для подання тестових наборів. Збірник наукових праць "Системи обробки інформації". - Випуск 8 (145). - Х.: ХУПС - 2016. - С. 77-80.

44. Смірнов О.А., Лисенко І.А. Розробка впорядкованих каскадних таблиць рішень із використанням матриць слідування. Збірник наукових праць "Системи обробки інформації". - Випуск 6 (143). - Х.: ХУПС - 2016. - С. 216-220.

45. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод кількісної оцінки ризиків розроблення програмного забезпечення. Збірник наукових праць Харківського університету Повітряних Сил. Випуск 2 (47). - Харків: ХУПС. - 2016. - С. 128-133.

46. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Метод якісного аналізу ризиків розроблення програмного забезпечення. Наука і техніка Збройних Сил України. – Випуск 2(23). - Харків: ХУПС. - 2016. - С. 150-158.

47. Смірнов О.А., Смірнов С.А., Дідик А.К., Дрєєв А.М. Алгоритми формування безлічі маршрутів передачі метаданих у антивірусні хмарні системи. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). - Х.: ХУПС - 2016. - С. 148-152.

48. Смірнов О.А., Коваленко О.В., Якименко Н.М., Доренський О.П. Проблеми аналізу та оцінки ризиків інформаційної діяльності. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 40-42.

49. Смірнов О.А., Смірнов С.А. Дідик А.К., Дрєєв О.М. Моделі системи нейромережевих експертів безпечної маршрутизації у хмарних антивірусних системах. Збірник наукових праць "Системи обробки інформації". - Випуск 3 (140). - Х.: ХУПС - 2016. - С. 36-39.

50. Смірнов О.А., Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). – Х.: ХУПС – 2016. – С. 153-157.

51. Смірнов О.А., Коваленко О.В. Методи якісного аналізу та кількісної оцінки ризиків розробки програмного забезпечення. Збірник наукових праць "Системи обробки інформації". - Випуск 5 (142). – Х.: ХУПС – 2016. – С. 153-157.

Додаток А
(обов’язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на розробку системи інтелектуального розпізнавання графічних образів.

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 інтелектуального розпізнавання графічних образів.

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 інтелектуального розпізнавання графічних образів;

– 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 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 – 71 аркуш.

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 10.06.2024 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

_____ Петренюк В.І.

Програмне забезпечення системи інтелектуального розпізнавання графічних образів

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2024 року

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

```
unit UPicture_Analyz;
// один з модулів розпізнавання графічних 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 графічних 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 графічних 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 графічних 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,UPicture_Analyz,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 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
  UPicture_Analyz, //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;
    TLabel1: 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,UPicture_Analyz,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 OK 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 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('БАКАЛАВРСЬКИЙ ПРОЕКТ');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Програмне забезпечення системи інтелектуального розпізнавання
графічних образів');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Петренюк В.І.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Поладов Арслан ');
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.
```