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
“____” _____ 2022 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-21М-1,4
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Ситнік Є.Ю.
«____» _____ 2022 р.

Керівник проекту
доктор технічних наук, професор
_____ Смірнов О.А.
«____» _____ 2022 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

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
« 6 » вересня 2022 року

ЗАВДАННЯ НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ ЗА ДРУГИМ (МАГІСТЕРСЬКИМ) РІВНЕМ ВИЩОЇ ОСВІТИ ЗДОБУВАЧА ВИЩОЇ ОСВІТИ

Ситніку Євгену Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи*

2. Керівник роботи *Смірнов Олексій Анатолійович, докт. техн. наук, професор*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 19-13 від 17.08.2022 року

3. Строк подання студентом роботи до захисту *10.12.2022 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи*

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

1. Призначення та область використання.	*6. Наукова новизна.*
2.Перегляд аналогічних існуючих систем.	*7. Економічна ефективність розробленої програми.*
3. Опис і обґрунтування проектних рішень.	*8. Заходи з охорони праці та техніки безпеки.*
4. Етапи програмування системи.	*9. Висновки.*
5. Впровадження системи в промислову експлуатацію	

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

Наукова новизна	*1 аркуш*
Структурна схема системи	*1 аркуш*
Функціональна схема системи	*1 аркуш*
Діаграма процесів	*1 аркуш*
Блок-схема алгоритму роботи додатку	*2 аркуша*
Показники економічної ефективності	*1 аркуш*

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Економічний	Савеленко Г.В.	05.10.2022	14.11.2022
Охорона праці	Оришака О.В.	06.10.2022	16.11.2022

7. Дата видачі завдання « 6 » вересня 2022 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Строк виконання етапів випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти	Примітка
1.	Аналіз існуючих систем	10.10.2022 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2022 р.	
3.	Розробка моделі компонента	20.10.2022 р.	
4.	Розробка структур даних	25.10.2022 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2022 р.	
6.	Програмування алгоритмів	10.11.2022 р.	
7.	Розрахунок економічної ефективності	13.11.2022 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2022 р.	
9.	Оформлення ПЗ	17.11.2022 р.	
10.	Попередній захист роботи	10.12.2022 р.	

Дата видачі завдання
« 6 » вересня 2022 р.

Підпис керівника
_____ Смірнов О.А.
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2022 р.

Підпис здобувача
_____ Ситнік Є.Ю.
(прізвище та ініціали)

АНОТАЦІЯ

Ситнік Є.Ю. Дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2022.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розпізнання образів у структурі технічного захисту інформації банківської установи.

Метою розробки є дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

Об'єктом дослідження є процес розпізнання образів у структурі технічного захисту інформації банківської установи.

Предметом дослідження є методи розпізнання образів у структурі технічного захисту інформації банківської установи.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

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 архітектури IBM PC з ОС Windows 10/11.

Програму розроблено в середовищі RAD Studio Delphi 10.4.

Ключові слова: комп'ютерна інженерія, розпізнання образів, технічний захист інформації

ABSTRACT

Sytnik E.Yu. Research and software implementation of a pattern recognition system in the structure of technical information protection of a banking institution. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2022.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the pattern recognition system in the structure of technical information protection of a banking institution.

The purpose of the development is research and software implementation of a pattern recognition system in the structure of technical information protection of a banking institution.

The object of the study is the process of pattern recognition in the structure of technical information protection of a banking institution.

The subject of the research is pattern recognition methods in the structure of technical information protection of a banking institution.

Research methods are based on information protection methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the pattern recognition system in the structure of the technical information protection of the banking institution.

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 PCs of IBM PC architecture with Windows 10/11 OS.

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

Keywords: computer engineering, pattern recognition, technical information protection

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

БД	–	база даних
ПЗ	–	програмне забезпечення

ВСТУП

Актуальність теми. Розвиток алгоритмів аналізу відеозображення й технологій розпізнавання об'єктів, підвищення «інтелектуальності» систем відеоспостереження значно розширює їхню сферу застосування. Наприклад, за допомогою відеоспостереження можна автоматично виявляти збої у виробничих процесах (наприклад, затор на конвеєрній лінії), вести облік готової продукції й матеріалів, аналізувати якість. Застосування «інтелектуальних» систем відеоспостереження на стадіонах і концертних залах, вокзалах і т.д. дозволяє виявляти людей з неадекватним поводженням. Аналіз відеозображення транспортних потоків у режимі реального часу використовується для визначення інтенсивності дорожнього руху й пробок.

Особливу необхідність у системах відеоспостереження вимагає банківський сектор, зокрема різні системи охорони приміщень і будинків, де розміщені банківські установи. Завдання які стоять перед такими системами зосереджуються у двох областях:

– Розпізнавання автомобільних номерів. Система використовується для автоматичної реєстрації й розпізнавання автомобільних номерів на контрольно-пропускних пунктах у банках. Інтеграція із системою контролю доступу дозволяє автоматизувати контрольно-пропускний режим.

– Захват і розпізнавання особи. 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 банків полегшує роботу з архівами при розслідуванні позаштатних ситуацій. Далі система розпізнавання особи ідентифікує особи і автоматизує пошук зображень у базах даних.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

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

– Огляд існуючих систем розпізнання образів у структурі технічного захисту інформації банківської установи.

– Дослідження системи розпізнання образів у структурі технічного захисту інформації банківської установи.

– Програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

Об'єктом дослідження є процес розпізнання образів у структурі технічного захисту інформації банківської установи.

Предметом дослідження є методи розпізнання образів у структурі технічного захисту інформації банківської установи.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод розпізнання образів у структурі технічного захисту інформації банківської установи.

– Розроблено вітчизняний продукт розпізнання образів у структурі технічного захисту інформації банківської установи, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі розпізнання образів у структурі технічного захисту інформації банківської установи.

Достовірність наукових результатів підтверджена теоретичними викладеннями, даними комп'ютерного моделювання, коректними дослідженнями параметрів на функціонуючій обчислювальній мережі, а також відповідністю отриманих результатів окремим результатам, наведеним у науковій літературі.

Робота апробована на LVI Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2022, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №13.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

1 ПРИЗНАЧЕННЯ ТА ОБЛАСТЬ ВИКОРИСТАННЯ

1.1 Призначення системи

Система призначення для розпізнання образів у структурі технічного захисту інформації банківської 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". 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 ввагу 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 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.

У чому переваги використання засобів охоронного телебачення як технічні засоби 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".

1.2 Область застосування

Областю застосування розробляємого програмного забезпечення є відеодетектори систем безпеки банківських 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

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи, є актуальною задачею, яка потребує вирішення у даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти.

2 ПЕРЕГЛЯД АНАЛОГІЧНИХ ІСНУЮЧИХ СИСТЕМ

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

Як правило, відеодетектори мають у своєму арсеналі набір різних налаштувань для мінімізації помилок. У їхньому числі регулювання чутливості й контрастності, мінімальні й максимальні розміри об'єктів, спеціалізовані налаштування для запобігання помилкових спрацьовувань. Системи поставляються з визначеними усередненими налаштуваннями "за замовчуванням" (заводськими). Досвід показує, що більша частина встановлених систем відеоспостереження працює саме з налаштуваннями відеодетектора "за замовчуванням". Далеко не кожний інсталятор або користувач буде витрачати час (і значний) на юстировку детектора й вибір оптимального режиму детектора. Представляється цілком логічним основну масу тестів провести саме з налаштуваннями "за замовчуванням". Фахівець, що бажає більш глибоко вивчити всі можливості відеодетекторів, може використовувати наведені дані як відправну точку у своїх власних дослідженнях. Проте в матеріалі розглянуті деякі спеціалізовані функції відеодетекторів руху.

У даному тестуванні ми поставили своєю метою вивчити характеристики відеодетекторів найбільш популярних неспеціалізованих цифрових систем відеоспостереження. Дослідження має емпіричний характер і не претендує на об'єктивність. Всі тести вироблялися шляхом подачі на відеовхід системи стандартних відеороликів з визначеним тестовим записом. У трьох роликах записано кілька стандартних типових ситуацій, що найчастіше зустрічаються на "реальних" об'єктах:

– рух людини по коридорі уздовж вектора спостереження відеокамери (швидкість руху близько 1 м/с, освітленість близько 500 лк);

– рух об'єктів по невеликій вуличній території на різних відстанях з перешкодами у вигляді колихання віток і листів дерев (освітленість близько 10 000 лк);

– великий вуличний простір з елементами проїзної частини, автостоянки, перехрестя й пішохідних переходів (освітленість близько 10 000 лк).

Для тестування всі системи приводилися до наступних умов: 1 відеоканал, 25 кадр/с на канал, розрішення 720x288.

Videonet

Система Videonet давно й визначенено займає більшу частину ринку цифрових систем відеоспостереження й постійно вдосконалюється. До основних її переваг (за матеріалами попередніх досліджень) можна віднести високу якість відеозображення (кодек стиску DV-pack), не менш високу швидкість відеообробки (плати відеовведення власної розробки), а також гнучкість і налаштовуваність програмного забезпечення.

Рисунок 2.1 – Інтерфейс системи

В основу детектора руху системи VideoNet покладений механізм передустановок, використовуваний для завантаження необхідних налаштувань детектора в тимчасовій або ситуативній залежності, наприклад день/ніч або

робоче/неробочий час. Маскування області детектування відбувається шляхом довільного розташування на відеозображенні номерних зон детектування загальним числом до 999. По кожній зоні доступні свої індивідуальні налаштування детектора. Реакція системи на відпрацьовування детектора також може налаштуватися окремо для кожної зони.

Доступні такі налаштування, як стандартні регулювання чутливості детектора, контрастності й мінімального розміру об'єкта, а також функції фільтрації: максимальне й мінімальне співвідношення сторін і максимальний розмір детектуємого об'єкта.

Робота детектора візуалізується шляхом виділення червоними прямокутниками області відеозображення, де виявлений рух.

1. Чутливість відеодетектора руху. При налаштуваннях "за замовчуванням" відеодетектор виявив об'єкт на відстані 11 м, коли той займав приблизно 2% площі відеокадру.

2. Завадостійкість відеодетекторів. На типовому відеозображенні вуличного дворика при налаштуваннях "за замовчуванням" детектор увизначенено виявляв людину, що рухається, на відстані 23-25 м, що займає 1,5-2% площі кадру, і не зауважував людей, що розвантажують вантажівку, на відстані близько 40 м, зображення яких займало порядку 0,5% загальної площі зображення. Помилкових спрацьовувань на вітки дерев, що займають приблизно 3% загальної площі зображення, не спостерігалося. Шляхом регулювання чутливості й контрастності вдалося домогтися детектування людей, але при цьому стали з'являтися помилкові спрацьовування від віток дерев.

3. Зведений тест відпрацьовування різних об'єктів, що рухаються. На сумбурній картинці відрізка проїзної частини й стоянки з пішоходами відеодетектор з налаштуваннями "за замовчуванням" виявляв джип на відстані приблизно 30 м і розміром 2% площі кадру. На перехресті (приблизно 80 м) об'єкти розміром 0,25-0,5% загальної площі зображення детектувалося зі змінним успіхом залежно від контрастності. Пішоходи, що переходять перехрестя, не

детектувалися. Збільшення чутливості дозволило виявити всі автомобілі й пішоходів, що перетинають перехрестя, але при цьому додалися помилкові спрацьовування від віток дерев і хмар вихлопних газів. До специфічних функцій системи VideoNet відноситься можливість фільтрації детектуємих об'єктів по геометрії їхніх границь. Дослідження показали, що налаштування мінімального співвідношення сторін 4x10 дозволила детектувати тільки проїжджаючі автомобілі й при цьому ігнорувати все інше. Слід ще зазначити, що всі тривоги детектора рухи записуються в журнал подій із вказівкою номера камери, зони й часу, з якого відразу ж можна перейти на визначений відеофрагмент в архіві.

Trassir

Цифрова система відеоспостереження Trassir однією з перших з'явилася на російському ринку ЦСВН із апаратним стиском середнього цінового діапазону. У лінійці Trassir є недорогі рішення на платах із програмним стиском. Високі споживчі властивості Trassir укупі з активною маркетинговою політикою дозволяють системі займати гідне місце на ринку засобів безпеки.

Рисунок 2.2 – Інтерфейс системи

До основних установок відеодетектора руху системи Trassir відносяться: регулювання чутливості, налаштування маски й зон детектування, фільтри по розмірах об'єкта. Маска являє собою необмежену кількість прямокутних зон довільного розміру; можна змінювати колір маски й відображати чи ні маску на відеозображенні. У налаштуваннях детектора можна вказати, який рух

детектувати: швидкий або повільний. Відображати об'єкти, що рухаються, можна кольоровою сіткою. Налаштування чутливості й фільтра по розмірі об'єкта єдині для всіх зон одного відеоканалу. Є можливість задати час скидання детектора, протягом якого детектор не реагує на рух у кадрі після спрацьовування.

1. Чутливість відеодетектора руху. Об'єкт, що наближається, був виявлений відеодетектором з налаштуванням "повільний рух" на відстані 8 м (близько 3% площі кадру), установка "швидкий рух" дала можливість розпізнавати рух об'єкта на відстані 16 м (1,5% кадру).

2. Завадостійкість відеодетекторів. При установці "повільний рух" детектор реагує практично на всі зміни в кадрі, включаючи людей на відстані до 40 м (0,5% площі кадру) і листя дерев (3% площі кадру). При налаштуванні "швидкий рух" діапазон реагування звузився до людини, що переміщається (20-25 м, 1,5-2% загальної площі зображення), помилкові спрацьовування на вітки дерев припинилися.

3. Зведений тест відпрацьовування різних об'єктів, що рухаються. При використанні налаштувань "за замовчуванням" детектор розпізнав дівчину, що перетинає проїзну частину, на відстані 25 м (1% площі зображення), і автомобіль (близько 40 м, 1,5% зображення). Установка "повільний рух" дозволила детектувати автомобілі й пішоходів на перехресті (80 м, 0,25-0,5% зображення), але з'явилися помилкові спрацьовування від хмар вихлопних газів (приблизно 0,5% площі зображення).

Відеодетектор Trassir має унікальну можливість розділяти реєструємі рухи на швидкі й повільні. Використовуючи цю функцію, можна настроїти детектор таким чином, щоб він розпізнавав об'єкти, що рухаються з визначеною швидкістю, і не реагував на більше повільні. Також у системі Trassir є функція "детектора залишених предметів", що дозволяє відслідковувати статичний кадр і детектувати об'єкти, що з'явилися або пропали. Регулювання чутливості й розміру об'єктів поширюються на цей детектор з налаштувань детектора руху. Детектор залишених предметів Trassir був провірений на типовому офісному

відеозображенні, тестування з додаванням в обстановку, наприклад, портфеля довело працездатність даної функції.

Avialle

Програмно-апаратний комплекс Avialle по праву вважається спадкоємцем цифрової системи відеоспостереження Sivineya. У лінійці Avialle є рішення класів "Стандарт" і "Економ", що поставляються з ПЗ Avialle і відрізняються тільки платами відеоуведення. Високі показники якості відеозображення й швидкості відеоуведення вкупі з досить демократичною вартістю дозволили системі Avialle зайняти більшу частку ринку цифрових відеореєстраторів.

Рисунок 2.3 – Інтерфейс системи

Маскування в системі Avialle відбувається шляхом виділення на відеозображенні восьми прямокутних зон довільного розміру. Для кожної камери доступна тільки одна маска детектора руху. Різні регулювання детектора єдині для всіх зон однієї камери, індивідуальне налаштування кожної зони неможливі. Є стандартні установки: регулювання максимального розміру детектуємого об'єкта й налаштування чутливості відеодетектора. Можлива візуалізація роботи детектора шляхом окреслення по контурі й/або сіткою об'єктів, що рухаються. Є лічильник тривог (на кнопці у верхньому правому куті вікна камери відображається кількість спрацювань детектора).

1. Чутливість відеодетектора руху. Відеодетектор системи Avialle з налаштуваннями "за замовчуванням" виявив на типовому відеоролику людину,

що наближається, на відстані 9 м, коли та займала приблизно 3% площі відеокадру.

2. Завадостійкість відеодетекторів. При налаштуваннях "за замовчуванням" помилкові спрацьовування не спостерігалися, зображення людини, що рухається, детектувалося приблизно з 20 м (приблизно 2% загальної площі зображення). Підвищення чутливості з метою виявлення більше віддалених об'єктів, що рухаються, приводило до появи великої кількості помилкових спрацьовувань через вітки дерев і листя на тротуарі.

3. Зведений тест відпрацьовування різних об'єктів, що рухаються. Автомобіль, що віддаляється, детектувався аж до 30 м (2% площі кадру), пішохід жіночої статі, який пересік дорогу на відстані приблизно 25 м (1% площі кадру), теж був виявлений детектором. Автомобілі й пішоходи на перехресті (приблизно 80 м) залишалися непоміченими.

Цифрова система відеоспостереження Avialle має кілька оригінальних функцій відеодетектора руху. "Регулювання глибини фільтра детектування" дозволяють фільтрувати помилкові спрацьовування шляхом відстеження часу змін у кадрі, тобто спрацьовування викликається лише тим рухом, що тривало довше зазначеного користувачем. Це дозволило, наприклад, отфільтровувати коливання віток дерев у процесі тестування. Ще одна цікава функція – "детектор залишених предметів" – дозволяє відслідковувати появу в статичному кадрі об'єктів, яких раніше там не було. Для налаштування цього детектора варто задати мінімальний розмір детектуємого об'єкта й затримку спрацьовування – період часу, по закінченні якого система буде перевіряти наявність змін у кадрі. При тестуванні ефективність детектора була перевірена в такий спосіб. На типовому офісному відеозображенні був зроблений "еталонний кадр", після чого в експозицію була додана сумка (що займала близько 1% площі кадру). Через заданий проміжок часу система виділила сумку контуром, повідомивши, що раніше тут цього предмета не було.

Intellect

Система Intellect за минулий рік придбала величезну популярність на ринку цифрового відеоспостереження, що було досягнуто за рахунок високої надійності апаратної й програмної платформи, модульної структури, розширених мережних можливостей, інтеграції з великою кількістю сторонніх продуктів охоронно-пожежної сигналізації й контролю доступу. Також зіграло свою роль наявність декількох додаткових функцій: робота з касовими терміналами, з IP-камерами (мережними камерами), розпізнавання осіб і т.д.

Рисунок 2.4 – Інтерфейс системи

Детектор руху підтримує роботу одночасно з декількома масками, які можуть бути поставлені під охорону або зняті з охорони оператором або автоматично. Також можна задавати зони, що постійно перебувають під охороною. Отмальовування зони довільне, налаштування контрастності й розміру детектуємого об'єкта індивідуальні для кожної зони. Візуалізація роботи детектора можлива шляхом виділення контуром об'єкта, що рухається. Є функція відстеження закриття/засвітки відеокамери з налаштуванням граничного значення засвітки у відсотках. Доступні функції "відкату", "дозапису" і "гарячого

запису", зміни швидкості обробки по камері за рахунок камер, що не реєструє рухи в кадрі.

1. Чутливість відеодетектора руху. Людину, що наближається, відеодетектор виявив на відстані 9 м, коли та займала приблизно 3% площі відеокадру (випробування налаштування детектора "за замовчуванням").

2. Завадостійкість відеодетекторів. При налаштуваннях "за замовчуванням" на типовій вуличній сцені детектор розпізнавав людину, що рухається, на відстані 25 м, коли та займала 1,5% площі кадру. Помилкові спрацьовування через вітки дерев не відбувалися. Об'єкти на відстані 40 м розпізнавалися при підвищенні чутливості, але при цьому з'являлися помилкові спрацьовування.

3. Зведений тест відпрацьовування різних об'єктів, що рухаються. На перехресті (близько 80 м) детектувалося автомобілі (близько 0,5% площі відеокадру) темного кольору, контрастні стосовно тла. Світлі автомобілі проїжджали непоміченими. У межах 30 м розпізнавався рух великого джипу (2% площі кадру) і пішохода (25 м, 1% площі кадру).

Особливостями відеодетектора системи Intellect є можливість одночасної роботи з декількома масками зон детектування, функція зняття/узяття зон на охорону оператором у будь-якій їхній сукупності, а також ведення лог-файлу тривог із вказівкою конкретної зони, де відбулося спрацьовування.

Ewclid

Система Ewclid у її сучасному виді з'явилася на ринку відносно недавно. Розроблювачі роблять великий упор на апаратну складову системи й роблять велику кількість різних модифікацій модулів відео– і аудіовведення, а також сигнальних плат. Програмне забезпечення побудоване по модульному принципі й має багаторівневу структуру.

Налаштування детектора руху в системі Ewclid обмежені регулюванням чутливості в діапазоні від 0 (детектор виключений) до 127. Також можна встановити мінімальні ширину й висоту об'єкта у відсотках від усього

зображення. Зони детектування мають прямокутну форму будь-яких розмірів, максимальна кількість зон 16, налаштування можуть бути індивідуальні для кожної зони. Візуалізація роботи детектора відбувається шляхом виділення на зображенні об'єктів, що рухаються, прямокутниками червоного кольору.

Рисунок 2.5 – Інтерфейс системи

1. Чутливість відеодетектора руху. Детектором з налаштуваннями "за замовчуванням" об’єкт, що наближається, був виявлений на відстані 6 м, коли він займав 4% площі кадру.

2. Завадостійкість відеодетекторів. На коливання віток дерев (приблизно 3% площі кадру) і переміщення листя по тротуарі детектор не реагував. Людина, яка рухається, розпізнана детектором на відстані 5 м (5% площі кадру).

3. Зведений тест відпрацьовування різних об'єктів, що рухаються. Детектор супроводжував автомобіль на дистанції до 10 м (приблизно 4% площі кадру), далі детекція припинилася. Пішохід на відстані 25 м (1% площі кадру) проскочив непоміченим. Посилення чутливості привело до виявлення пішохода, збільшення дистанції виявлення автомобіля до 40 м і появи помилкових спрацьовувань від хмар вихлопних газів. Об'єкти на перехресті (100 м) не детектувалося.

Специфічними особливостями відеодетектора Ewclid є можливість зняття/постановки зон на охорону й механізм підтвердження оператором прийняття тривоги детектора руху.

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 другим (магістерським) рівнем вищої освіти, реалізації підлягає програмне забезпечення, яке призначено для системи розпізнання образів у структурі технічного захисту інформації банківської установи.

В процесі розробки випускної кваліфікаційної роботи за другим (магістерським) 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

Опис алгоритму розпізнавання образів

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

Розпізнавання образів (об'єктів, сигналів, ситуацій, явищ або процесів) – завдання ідентифікації об'єкта або визначення яких-небудь його властивостей по його зображенню (оптичне розпізнавання) або аудіо запису (акустичне розпізнавання) і іншим характеристикам.

Одним з базових є, такий, що не має конкретного формулювання поняття множини. У комп'ютері множина представляється набором неповторюваних однотипних елементів. Слово "неповторюваних" означає, що якийсь елемент у множини або є, або його там немає. Універсальна множина включає всі можливі для розв'язуваного завдання елементи, порожня не містить жодного.

Образ – класифікаційне угруповання в системі класифікації, що поєднує (виділяє) певну групу об'єктів по деякій ознаці. Образи мають характерну властивість, що проявляється в тому, що ознайомлення з кінцевим числом явищ із тієї самої множини дає можливість дізнаватися як завгодно велике число її представників. Образи мають характерні об'єктивні властивості в тому розумінні, що різні люди, що навчаються на різному матеріалі спостережень, здебільшого однаково й незалежно друг від друга класифікують й ті самі об'єкти. У класичній постановці завдання розпізнавання універсальна множина розбивається на частини-образи. Кожне відображення якого-небудь об'єкта на сприймаючі органи системи, що розпізнає, незалежно від його положення щодо цих органів, прийнято називати зображенням об'єкта, а множини таких зображень, об'єднані якими-небудь загальними властивостями, являють собою образи.

Методика віднесення елемента до якого-небудь образа називається вирішальним правилом. Ще одне важливе поняття – метрика, спосіб визначення відстані між елементами універсальної множини. Чим менше ця відстань, тим більше схожими є об'єкти (символи, звуки й ін.) – те, що ми розпізнаємо. Звичайно елементи задаються у вигляді набору чисел, а метрика – у вигляді функції. Від вибору подання образів і реалізації метрики залежить ефективність програми, один алгоритм розпізнавання з різними метриками буде помилятися з різною частотою.

Навчанням звичайно називають процес виробітку в деякій системі тієї або іншої реакції на групи зовнішніх ідентичних сигналів шляхом багаторазового впливу на систему зовнішнього коректування. Таке зовнішнє коректування в навчанні прийнято називати "заохоченнями" і "покараннями". Механізм генерації цього коректування практично повністю визначає алгоритм навчання. Самонавчання відрізняється від навчання тим, що тут додаткова інформація про вірність реакції системі не повідомляється.

Адаптація – це процес зміни параметрів і структури системи, а можливо – і керуючих впливів, на основі поточної інформації з метою досягнення певного стану системи при початковій невизначеності й умовах, що змінюються, роботи.

Навчання – це процес, у результаті якого система поступово здобуває здатність відповідати потрібними реакціями на певні сукупності зовнішніх впливів, а адаптація – це підстроювання параметрів і структури системи з метою досягнення необхідної якості керування в умовах безперервних змін зовнішніх умов.

Приклади завдань розпізнавання образів:

– Розпізнавання букв.

– Розпізнавання штрих-кодів.

– Розпізнавання автомобільних номерів.

– Розпізнавання осіб і інших біометричних даних.

– Розпізнавання мови.

Методи розпізнавання образів

У цілому, можна виділити три методи розпізнавання образів:

Метод перебору. У цьому випадку виробляється порівняння з базою даних, де для кожного виду об'єктів представлені всілякі модифікації відображення. Наприклад, для оптичного розпізнавання образів можна застосувати метод перебору виду об'єкта під різними кутами, масштабами, зсувами, деформаціями й т.д. Для букв потрібно перебирати шрифт, властивості шрифту й т.д. У випадку розпізнавання звукових образів, відповідно, відбувається порівняння з деякими відомими шаблонами (наприклад, слово, вимовлене кількома людьми).

Другий підхід – виробляється більше глибокий аналіз характеристик образа. У випадку оптичного розпізнавання це може бути визначення різних геометричних характеристик. Звуковий зразок у цьому випадку піддається частотному, амплітудному аналізу й т.д.

Наступний метод – використання штучних нейронних мереж (ШНМ). Цей метод вимагає або великої кількості прикладів завдання розпізнавання при навчанні, або спеціальної структури нейронної мережі, що враховує специфіку даного завдання. Проте, його відрізняє більше висока ефективність і продуктивність.

Загальна характеристика завдань розпізнавання образів і їхні типи

Завдання розпізнавання мають наступні характерні риси.

Це інформаційні завдання, що складаються із двох етапів:

– перетворення вихідних даних до виду, зручного для розпізнавання;

– властиво розпізнавання (вказівка приналежності об'єкта певному класу).

У цих завданнях можна вводити поняття аналогії або подоби об'єктів і формулювати правила, на підставі яких об'єкт зараховується в той самий клас або в різні класи.

У цих завданнях можна оперувати набором прецедентів-прикладів, класифікація яких відома і які у вигляді формалізованих описів можуть бути

пред'явлені алгоритму розпізнавання для налаштування на завдання в процесі навчання.

Для цих завдань важко будувати формальні теорії й застосовувати класичні математичні методи (часто недоступна інформація для точної математичної моделі або виграш від використання моделі й математичних методів непорівнянний з витратами).

Виділяють наступні типи завдань розпізнавання:

– завдання розпізнавання – віднесення пред'явленого об'єкта по його опису до одному із заданих класів (навчання із учителем);

– завдання автоматичної класифікації – розбивка множини об'єктів, ситуацій, явищ по їхніх описах на систему непересічних класів (таксономія, кластерний аналіз, самонавчання);

– завдання вибору інформативного набору ознак при розпізнаванні;

– завдання приведення вихідних даних до виду, зручному для розпізнавання;

– динамічне розпізнавання й динамічна класифікація – завдання 1 і 2 для динамічних об'єктів;

– завдання прогнозування – суть попередній тип, у якому рішення повинне ставитися до деякого моменту в майбутньому.

3.2 Розробка структурної схеми

Структурна схема системи наведена на рисунку 3.1.

Відносно технічної сторони питання, як ми вже відзначали раніше, усе перебуває на досить високому рівні. Розв'язна здатність систем оптичного уведення вже впритул наблизилася до можливостей людського зору й навіть перевершує його. Прикладом може служити цифрова фотокамера H2 D-39, представлена шведською компанією Hasselblad, обладнана ПЗС-матрицею в 39

мегапікселей. А сучасним обчислювальним системам цілком під силу швидкісна обробка різних параметрів зображення.

Однак завдання розпізнавання, саме по собі, припускає інтелектуальну обробку отриманої інформації, що представляє певні складності. Безсумнівно, у завданні розпізнавання символів (розпізнавання тексту, автомобільних номерів) досягнуті величезні успіхи. Але, проте, яких-небудь універсальних методів обробки зображення, порівнянних по продуктивності і якості розпізнавання з людськими здатностями, немає. Наприклад, у завданнях, які відносяться перед експертними системами, потрібно більше глибокий інтелектуальний аналіз і високу швидкодія, цими ж властивостями повинні володіти роботизовані системи обслуговування. Тому обробка зображення в завданні розпізнавання є однією із центральних проблем.

Структурно система складається з наступних частин:

1. Відеокамера спостереження, з якої поступає інформація систему розпізнання образів.

2. Система розпізнання образів, яка складається з наступних блоків:

– Блок читання картинки з відеокамери. Він призначений для читання картинок з відеокамери й подання даних на блок аналізу та виділення ознак за допомогою багатопрохідної схеми розпізнавання образів на основі кластерного аналізу.

– Блок аналізу та виділення ознак за допомогою багатопрохідної схеми розпізнавання образів на основі кластерного аналізу. Він є основою системи, й за допомогою нижчеописаного алгоритму проводить розпізнання осіб, та машин, які перетнули межу території банківської установи.

– Блок класифікації та опису об’єкта. Він дозволяє, виходячи з даних, отриманих від блоку аналізу та виділення ознак за допомогою багатопрохідної схеми розпізнавання образів на основі кластерного аналізу, розподілити куди заносити отримані дані, у поля бази даних, які відповідають за осіб, або у поля бази даних, які відповідають за машини.

3. База даних журналювання розпізнаних образів. У цю баз даних заносяться усі дані, які відносяться до розпізнаних об’єктів, будь то людина, або автомобіль.

4. База даних образів облич. У цій базі даних зберігаються фотографії усіх працівників установи та відвідувачів, з виділенням характерних точок, для кожного обличчя, за якими можливо ідентифікувати або працівника банку, або відвідувача.

5. База даних образів цифр та букв на номерах автомобілів. У цій базі даних зберігаються образи усіх цифр та букв, з яких можуть складатися номера автомобілів, а також номера усіх автомобілів, які перетинали кордон приміщення банківської установи, який встаткований відеокамерами спостереження.

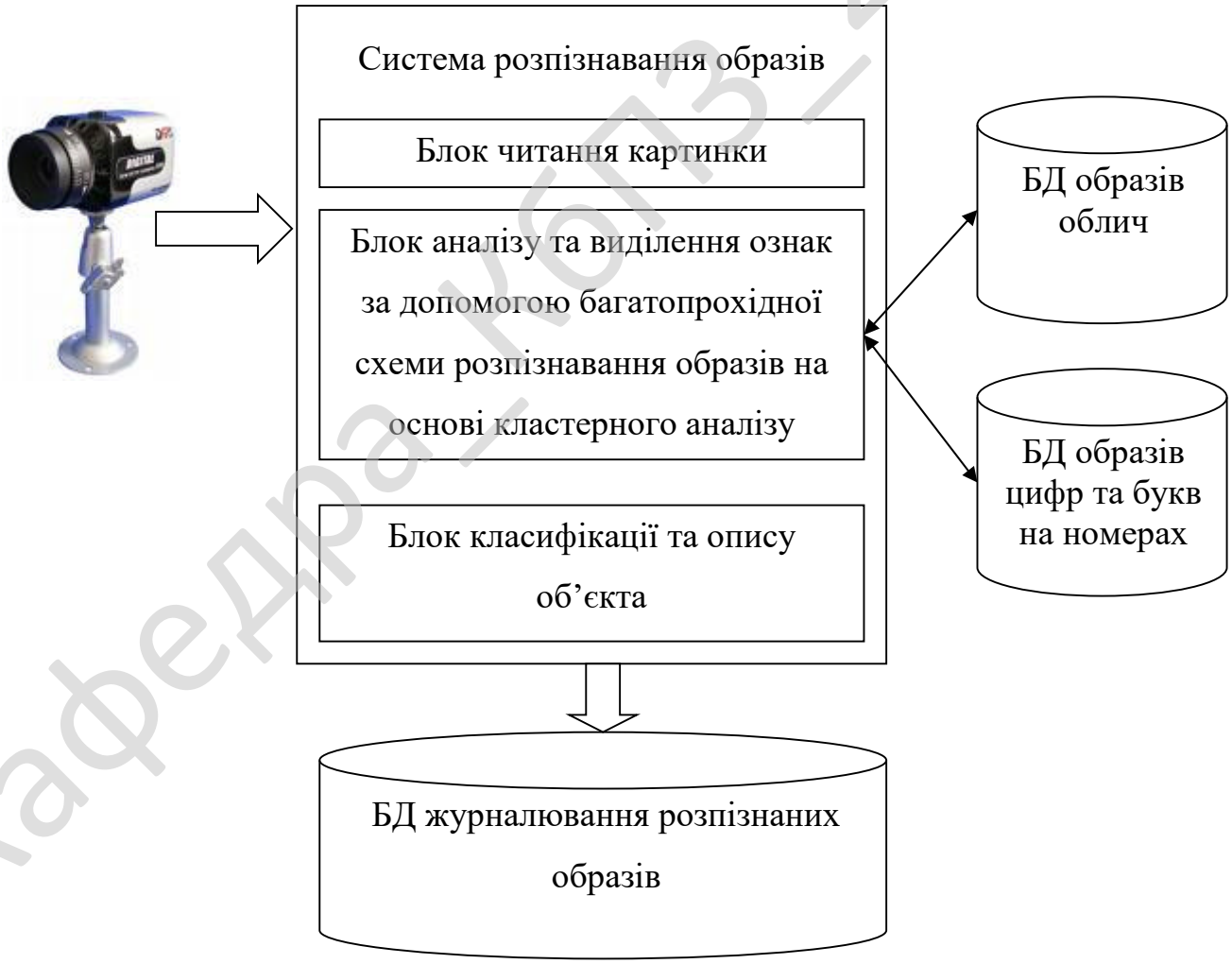

Рисунок 3.1 – Структурна схема системи

Так як розпізнавання образів відбувається за допомогою алгоритму багатопрохідної схеми розпізнавання образів на основі кластерного аналізу, то наведемо цей алгоритм.

Багатопрохідна схема розпізнавання образів на основі кластерного аналізу

Багатопрохідна схема складається в послідовній класифікації тих самих образів спочатку за допомогою образонезалежних алгоритмів розпізнавання, а потім – алгоритмів, що використовують особливості образів номерів автомобілів, і особливості розпізнавання осіб людини. Метою багатопрохідної схеми розпізнавання з навчанням є адаптація до особливостей образів. При цьому на кількість і характеристики використовуваних образів не накладається істотних обмежень.

Багатопрохідна схема розпізнавання містить у собі попереднє розпізнавання образів, формування бази даних результатів попереднього розпізнавання, додаткове самонавчання на підставі отриманих результатів розпізнавання, наступні перерозпізнавання образів з урахуванням самонавчання, формування остаточних результатів.

Навчальна вибірка готується першим проходом розпізнавання, що за допомогою образонезалежного алгоритму розпізнавання образів **ℜ1**. Алгоритм **ℜ1** надає колекції альтернатив (варіанти розпізнавання з оцінками) образів, що відповідають розпізнаним образам, і атрибути образів.

Крім цього перший прохід забезпечує валидацію образів, тобто позначку надійно розпізнаних образів, за допомогою наступних двох механізмів:

– облік оцінок альтернатив, сформованих алгоритмом з монотонними оцінками

– словникове або контекстне підтвердження досить довгого слова.

Комбінація цих механізмів валідирує частину результатів розпізнавання. Таким чином, множина розпізнаних образів розбито на підмножини, що задаються розмірами й атрибутами образа. Як контейнер навчання, призначеного

для зберігання розмічених образів, може виступати база даних, що формується на диску, або динамічна структура в оперативній пам'яті.

Метою навчання є побудова набору кластерів, частина яких надійно відділений друг від друга, а кластери, що залишилися, містять вказівки на їхню близькість до інших з погляду обраної Хаусдорфової метрики.

Термін кластерний аналіз у дійсності містить у собі набір різних методів і алгоритмів класифікації. Щораз, коли необхідно класифікувати більші масиви інформації із прийнятного для подальшої обробки групам, кластерний аналіз виявляється досить корисним і ефективним.

Багаторазові спроби класифікації самих методів кластерного аналізу приводять до десятків, а то й сотень різноманітних класів [2-4]. Таке різноманіття породжується великою кількістю можливих способів обчислення відстані між окремими об'єктами, не меншою кількістю функцій обчислення відстані між кластерами в процесі кластеризації й різноманітними оцінками оптимальності кінцевої кластерної структури.

Найбільше поширення одержали дві групи методів кластерного аналізу: ієрархічні агломеративні методи й ітеративні методи угруповання.

Серед ітераційних найбільш популярним є метод **k-середніх** Мак-Кіна. На відміну від ієрархічних алгоритмів сам користувач повинен задати шукане число кінцевих кластерів, що звичайно позначається як **k**. Як і в ієрархічних методах кластеризації, при цьому можна вибрати той або інший тип метрики. Різні алгоритми методу k-**середніх** відрізняються також способом вибору початкових центрів кластерів, що задаються. Оскільки ми заздалегідь не знаємо, скільки образів перебуває, використання методу k-**середніх** у нашім випадку представляється скрутним.

В агломеративно-ієрархічних алгоритмах (agglomerative hierarchical algorithms) спочатку всі об'єкти розглядаються як окремі, самостійні кластери, що складаються всього лише з одного елемента. Процедура кластеризації складається в поступовому об'єднанні об'єктів у досить більші кластери,

використовуючи деяку міру подібності або відстань між об'єктами. Типовим результатом такий кластеризації є ієрархічне дерево. На першому кроці, коли кожний об'єкт являє собою окремий кластер, відстані між цими об'єктами визначаються обраною мірою. Однак коли зв'язуються разом кілька об'єктів, виникає питання, як варто визначити відстані між кластерами?

Існує багато способів об'єднання кластерів, перелічимо деякі з них.

– Одиночний зв'язок (метод найближчого сусіда). У цьому випадку відстань між двома кластерами визначається відстанню між двома найбільш близькими об'єктами (найближчими сусідами) у різних кластерах.

– Повний зв'язок (метод найбільш віддалених сусідів). У цьому випадку відстані між кластерами визначаються найбільшою відстанню між будь-якими двома об'єктами в різних кластерах (тобто "найбільш віддаленими сусідами").

– Незважене попарне середнє. У цьому методі відстань між двома різними кластерами обчислюється як середня відстань між всіма парами об'єктів у них.

– Зважене попарне середнє. Метод ідентичний методу незваженого попарного середнього, за винятком того, що при обчисленнях розмір відповідних кластерів (тобто число об'єктів, що втримуються в них) використовується як ваговий коефіцієнт.

– Незважений центроідний метод. У цьому випадку відстань між двома кластерами визначається як відстань між їхніми центрами ваги.

– Зважений центроідний метод (медіана). Цей метод ідентичний попередньому, за винятком того, що при обчисленнях використовуються ваги для обліку різниці між розмірами кластерів (тобто числами об'єктів у них).

– Метод Варда. Цей алгоритм використовує методи дисперсійного аналізу для оцінки відстаней між кластерами. Метод мінімізує суму квадратів для будь-яких двох (гіпотетичних) кластерів, які можуть бути сформовані на кожному кроці.

Питання, що природно виникає в агломеративних алгоритмах, – коли припинити об'єднання кластерів. Для цього використовують число отриманих

кластерів, міжкластерну відстань, максимальний стрибок у зміні межкластерної відстані. Також використовуються основні статистичні характеристики кластерів, як кількість об'єктів у кластері, середні значення ознак у кожному кластері, дисперсії й т.д.

У нашій програмі для кластеризації ми використовували алгоритм ланцюгового розгорнення, алгоритм відноситься до групи методів одиночного зв'язку. Алгоритм кластеризації, що базується на відстані Хаусдорфа й наведений в [4] якісно відрізняється від широко відомих методів, насамперед, через структуру інформації, що підлягає класифікації під час навчання й цілей кластеризації. Мінімізація числа кластерів, швидкодія й інші питання класичної кластеризації мають для нас важливе, але не першорядне значення.

Результатом кластеризації служить множина об'єктів, кожний з яких містить суму стандартизованих ненормалізованих растрів, із близькими ознаками й загальними атрибутами, ланцюгова відстань між якими не перевищує певного заздалегідь межі. Кожний із кластерів крім успадкованих ознак має потужність (числом складових його растрів).

Ознаки кластера використовуються на етапі побудови еталонів накладення, що повинні сформувати базу надійних кластерів, а для кластерів, що залишилися, відповістити на запитання про можливість їхнього використання. Аналіз кластерів образів певного алфавіту містить кілька операцій:

– Перейменування кластера, що складається в розпізнаванні центра сумарного растра (або всієї суми растрів, розглянутого як напівтоновий образ) досить точним алгоритмом розпізнавання образів. Від перерозпізнавання очікується зняття систематичних помилок алгоритму першого проходу, що не зобов'язаний бути адаптивним до особливостей накреслень використовуваних образів.

– Об'єднання двох різноїменних прилеглих кластерів з метою перейменування одного з них. Вирішує завдання, аналогічні завданням перейменування кластера.

– Знищення кластера, тобто вивід ненадійно розпізнаного кластера з розгляду.

– Угруповання кластера з одним або декількома різноїменними з ним кластерами. Фіксує неможливість або ненадійність розрізнення результатів, отриманих накладенням кластерів з однієї групи.

Після первинного аналізу кластерів залишаються тільки надійні кластери, що володіють достатньої валідністю й потужністю. Серед надійних кластерів проводиться ітераційний процес пошуку образів, оскільки на картинці перебувають не просто якісь образи, а образи, що відбуваються з одного або декількох образів. Кластери розбиваються на кілька груп їх складових, і, можливо, по своїх атрибутах (якщо атрибути присутні). Якщо в процесі аналізу виявиться, наприклад, що на картинці присутня тільки один образ, але є кілька кластерів якої-небудь букви, то в остаточну вибірку кластерів треба взяти тільки один, кращий у деякому змісті, а інші можуть бути помилками розпізнавання, помилками сегментації, можуть виникнути через погану якість зображення.

Ітераційний процес аналізу первинних кластерів закінчується формуванням еталонів алгоритму накладення, що здатний відповісти на ряд питань, пов'язаних з можливістю розрізнення близьких образів. Алфавіт розпізнавання, містить ряд образів невідмінних друг від друга у всій множині накреслень цих образів, для яких, проте, необхідно на якімсь із етапів розпізнавання ухвалити рішення щодо виборі одного зі значень. Подібні по накресленню друковані образи, близькість яких визначається властивостями алгоритмів розпізнавання образів, також є джерелом помилок образонезалежних алгоритмів.

У той же самий час у межах одного образа, як правило, деякі з образів алфавіту й родинних образів помітні геометрично, наявність же декількох образів на картинці може як утруднити, так і спростити розпізнавання близьких образів. Після завершення етапу аналізу кластерів, що досліджує подібні можливі конфліктні ситуації, стають відомим, наскільки добре побудовані еталони можуть

дозволяти колізії родинних образів і образів з однаковим накресленням у різних образах. Інформація про це втримується в переліку груп різноїменних родинних образів і в списку відстаней між нерозрізненими образами.

З оброблених кластерів може бути витягнута множина еталонних кістякових і розширених образів, міра близькості до яких забезпечує достатні характеристики якості розпізнавання. Можливе подання еталонів, що складає із трьох об'єктів:

– кістякова підмножина SKEL, що містить значення растра кластерів у границях щирого кістякового образа;

– розширена підмножина COVER, що містить нулі в границях щирого розширеного образа, мабуть, COVER $\supseteq$ SKEL;

– опис штрафу, що залежить від відстані до найближчої точки кістяковий або розширений образи, що обчислюється за допомогою функцій Pen(i, j, M), аргументами якої є координати точки й множина M.

Кожне з підмножин є матрицею того ж розміру, що й стандартизовані растри, що підлягають кластеризації, і залежать від обсягу й інших характеристик кластера. Накладення, тобто обчислення міри близькості довільного образа й еталона E=||eij|| відбувається в кілька етапів, першим з яких є центрування образа. Для відцентрованого, тобто стандартизованого, образа R =||rij|| підраховується сума покомпонентних добутків значень растрів R і E:

$$\Sigma(R,E) = \Sigma\, \delta\,(rij=1 \wedge eij>0) \bullet eij -$$
$$\Sigma\, Pen(i,j,SKEL(S,\alpha))(\,(rij=0 \wedge eij>0) -$$
$$\Sigma\, Pen(i,j,COVER(S,\beta))(\,(rij=1 \wedge eij<0),$$

яка містить у собі як позитивні добутки точок растра, що потрапили в кістякову область, так і негативні компоненти точок, що не потрапили в розширену область, і штраф за недолік точок у кістяковій зоні. У цьому вираженні присутні як позитивні значення суми растрів, що склали кластер, так і негативні штрафні значення. У такий спосіб обчислена близькість відповідає на запитання про те, наскільки добре розпізнаваний образ відповідає розподілу даного кластера, тобто

поліпшує імовірнісні властивості оцінок накладення. Зрозуміло, не слід забувати не тільки про евристичні штрафи, що не володіють імовірнісною природою, але й про обсяг кластера, що породжує кістякову область, тому що утворення малих кластерів не є чимсь винятковим для більшості картинок. Внесок штрафів у загальну суму також поліпшує оцінки за умови оптимізації штрафів за видалення від границі розширеного образа. Результатом накладення є альтернатива:

$$(S(E), W \bullet \Sigma(R,E)),$$

де S(E) – код образа кластера з растром E.

W – масштабний коефіцієнт для одержання оцінок, при цьому успадковуються властивості кластера (кегль, атрибути образа).

Спосіб центрування стандартизованого розпізнаваного растра, що полягає в сполученні геометричних центрів вихідного растра, розширюваного симетрично до стандартних розмірів, не може дати задовільних результатів у загальному випадку накладення. Пошук центра доповнюється зрушеннями розпізнаваного растра в невеликій околиці геометричного центра еталона з вибором найкращого результату накладення. Кластерному накладенню властивий ряд проблем, пов'язаних із проблемою пошуку геометричного центра образа. Центрування стандартизованих растрів не дозволяє розпізнавати образи із сильно деформованою рамкою. Для складних випадків центрування необхідне залучення інших алгоритмів, наприклад, обчислення моментів або використання поліграфічних базових ліній.

Відзначимо, що обчислення досить трудомістких оцінок накладення може бути прискорено перериванням підрахунку суми в ситуації набору значного числа штрафів. Значна різниця в розмірах растрів R і E дозволяє прийняти рішення про відмову накладення даного еталона, це міркування в сукупності з фільтрацією по атрибутах еталонів також прискорюють алгоритм накладень.

Еталони містять ряд додаткових ознак, наприклад, для заборони перерозпізнавання образа по кластеру, породженого цим же образом без участі інших образів.

Побудована система еталонів дозволяє використовувати алгоритм накладення як алгоритм, що формує після перегляду всіх еталонів, що задовольняють розміру розпізнаваного образа, колекцію альтернатив, породжених найближчими еталонами. Також можливе використання еталонних накладень і як алгоритм-експерта, що перевіряє гіпотези про те, наскільки добре досліджуваний образ може бути розпізнаний з деяким заданим кодом образа. Можуть бути отримані наступними результатами перевірки близькості розпізнаваного образа одному з еталонів із заданим кодом:

– найменша відстань досягнута на еталоні, далекому від інших еталонів;

– найменша відстань досягнута на еталоні, що потрапив у групу близьких різноїменних еталонів;

– перевірка близькості не може бути зроблена через відсутність еталонів з даним кодом образа.

Отриманий результат може бути проінтерпретований на відміну від образонезалежного алгоритму-експерта, що відповідає тільки на питання про близькість досліджуваного образа до одному з еталонів, у такий спосіб. Перший результат залежно від того, чи є кластер досить представницьким (великий обсяг, валідність його растрів, що склали), може бути визнаний надійним або рекомендаційним як для обчислення автономних оцінок, так і для рішення конфліктів. Другий результат може бути визнаний надійним тільки для перевірки приналежності образа до всієї групи еталонів, у цьому випадку до колекції альтернатив, збагачуваної кластерною інформацією, можуть бути додані відсутні альтернативи родинних образів. Тобто другий результат фіксує конфлікт родинних або нерозрізнених альтернатив як нерозв'язний у рамках кластерної моделі, що може надалі вирішуватися за допомогою словникового пошуку, словникової корекції. Третій результат є відмовою кластерного накладення. У цьому випадку неможливе порівняння двох образів, код одного одного з яких є присутнім в еталонах, а іншого відсутній.

Очікувана відсутність ряду еталонів припускає комбінування результатів образонезалежного алгоритму й алгоритму кластерного накладення, адаптивного до образів у розпізнаваній картинці. Схема комбінування будується в припущенні, що значна частина (80-90%) картинок уже розпізнана без помилок, внаслідок чого для частки, що залишилася, дорозпізнаваємих образів можливе застосування достатнє трудомістких алгоритмів розпізнавання образів. Це означає, що комбінувати кластерне накладення доцільно з алгоритмом, точність якого перевищує точність алгоритму $\Re 1$ розпізнавання образів на першому проході. Комбінування з більше точним алгоритмом (наприклад, з нейронною мережею [4]) забезпечує як збереження кластерних оцінок у випадку успішно розпізнаних обома алгоритмами образів і дозволених конфліктів, так і збереження точності алгоритму $\Re 1$ з одночасним зниженням його оцінок у випадку не підтвердження його результатів надійними кластерами. Це поліпшує й точність, і монотонність оцінок. Використання образонезалежного алгоритму дозволяє розпізнавати й виставляти оцінки образам, для яких не зібралися підтверджувальні еталони. Недоліком описаного комбінування є одержання оцінок різної природи: як образонезалежних, так і кластерних, зіставлення яких у загальному випадку важко. Початкові параметри схеми комбінування, у першу чергу відносяться порогів оцінок, по яких приймається рішення про надійність розпізнавання, можуть змінюватися після завершення кластеризації, за рахунок чого відбувається додаткова адаптація до результатів першого проходу розпізнавання картинок.

Властиво дорозпізнавання, тобто другий прохід розпізнавання, містить у собі розпізнавання комбінованим алгоритмом як окремо стоячих, так і склеєних і розсипаних образів, які деяким чином були сегментовані на першому проході. Дорозпізнавання рядка образів починається з експертної оцінки як незмінених, так і сегментованих розпізнаних образів. Кожний розпізнаний образ піддається експертизі на предмет підтвердження оцінки його провідної альтернативи алгоритмом, використовуваним як експерт. Образи, що не одержали

підтвердження, перерозпізнаються; деякі групи образів піддаються повторної сегментації. Сегментація, що навіть опирається на той самий перелік відрізків розрізування, при використанні іншого алгоритму розпізнавання образів може дати інші результати визначення границь образів. У той же час відзначимо, що часто успішно працює алгоритм розрізування (як, втім, і склейки), заснований винятково на кластерному розпізнаванні, без складання переліку відрізків можливого розрізування, хоча чисто кластерне розпізнавання працює не завжди у зв'язку із уже згадуваною проблемою можливої неповноти кластерів. Перерозпізнані ланцюжки образів конкурують зі своїми прототипами, утвореними на першому проході. Порівнянню підлягають слова, для яких можливо не тільки обчислення функцій над оцінками образів, що склали слово, але й підтвердження словниково-лінгвістичними методами.

Таким чином, побудований комбінований алгоритм (позначимо його $\Re 2$) дозволяє поліпшувати точність і монотонність оцінок розпізнавання як за допомогою образонезалежного алгоритму, так і за рахунок надійних еталонів кластерного накладення. Крім цих поліпшень не можна не відзначити ще одного результату другого проходу, що складається в тому, що в перерозпізнаних рядках частина образів одержала додаткову валідацію від надійних еталонів. По суті справи частина образів, що не одержала такий валідації, може бути змінена наступними етапами розпізнавання, а валидированні образи з високою ймовірністю не можуть піддаватися змінам. Окремо слід зазначити валідацію системою еталонів, що містить один єдиний образ, така гіпотеза перевіряється під час кластеризації.

3.3 Розробка 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 подання сигналу тривоги.
3. Блок документування подій на об’єкті.
4. Блок розпізнавання номерів автомобілів.
5. Блок розпізнавання людей.
6. Блок відеодетектування.
7. Блок динамічного спостереження за порушником.

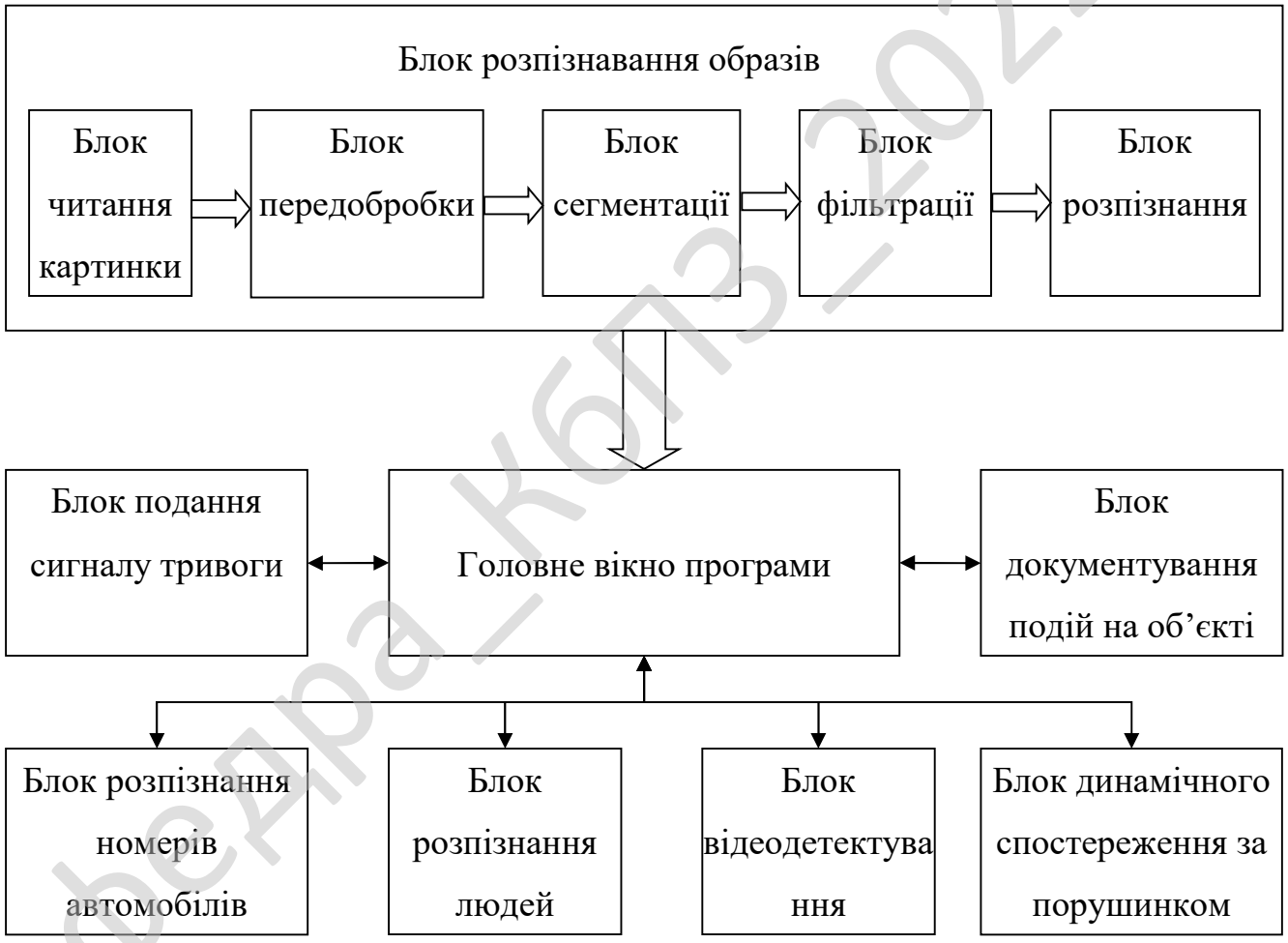

Рисунок 3.2 – Функціональна схема системи

Розглянемо ці блоки більш детально.

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 двунаправленості – 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.

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.

– Засвітлення. 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 – центроідне 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 ввазі 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).

4. 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 образів.

5. 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 плоскопараллельної 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.

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 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 30^{o}.

– По горизонталі – не більше 20^{o}.

– Крен номерного знака $\pm 15^{o}$.

– 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-розрішенняьні 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 відеодетектування

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 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.

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.3.

У системі робота процесів починається з запуску процесу початку/кінця роботи.

Цей процес взаємодіє з двома процесами:

– Процес читання зображення з телекамери.

– Процес документування подій на об’єкті.

Розглянемо з якими процесами взаємодіють вище перераховані процеси.

Процес читання зображення з телекамери взаємодіє з процесом аналізу зображення.

Процес аналіза зображення взаємодіє з наступними процесами:

– Процес розпізнавання образів на основі кластерного аналізу.

– Процес класифікацій та опису об’єкту.

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

– Процес динамічного спостереження.

– Процес журналювання розпізнаних об’єктів.

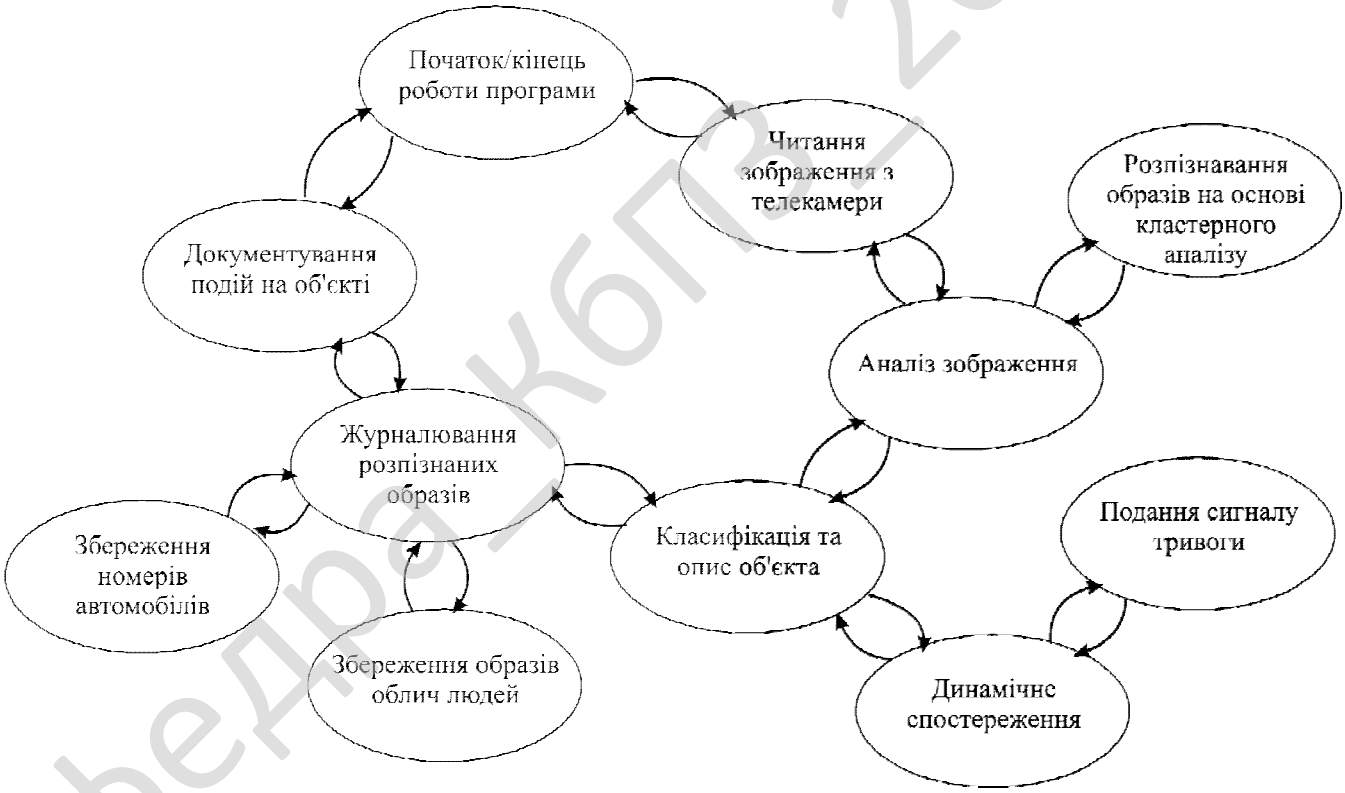

Рисунок 3.3 – Діаграма процесів системи

Процес динамічного спостереження у свою чергу взаємодіє з процесом подання сигналу тривоги.

Процес журналювання розпізнаних об’єктів взаємодіє з наступними

процесами:

– Процес збереження образів облич людей.

– Процес збереження номерів автомобілів.

– Процес документування подій на об’єкті.

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

4 РЕАЛІЗАЦІЯ РОБОТИ. РОЗРАХУНКИ І ЕКСПЕРИМЕНТАЛЬНІ ДАНІ, ЩО ПІДТВЕРДЖУЮТЬ ВІРНІСТЬ ПРОЕКТНИХ ТА ПРОГРАМНИХ РІШЕНЬ

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 зображено блок-схему основної програми.

Вона починається з виведення основного вікна програми.

Наступним етапом є підключення камери.

Відбувається перевірка на кількість камер.

Якщо камер декілька, то обирається активна камера.

У іншому випадку, вводяться параметри відео.

Після проведення описаних вище дій, відбувається виведення зображення з камери на екран.

Якщо зображення не потрібно розпізнавати, то програма переходить у блок запису відео з камери.

У іншому випадку, тобто, якщо потрібно розпізнавати зображення, то викликається підпрограма розпізнання образів нейронною мережею, блок-схема якої зображена на рисунку 4.2.

Після відпрацювання підпрограми розпізнання образів нейронною мережею, відбувається вивід результаті.

Логічним кроком є збереження виведених результатів.

Якщо користувач обирає меню запису відео з камери без розпізнання, то відбувається запис відео у файли по заданим шляхам.

Після цього користувач обирає працювати йому далі з програмою, або вийти з неї.

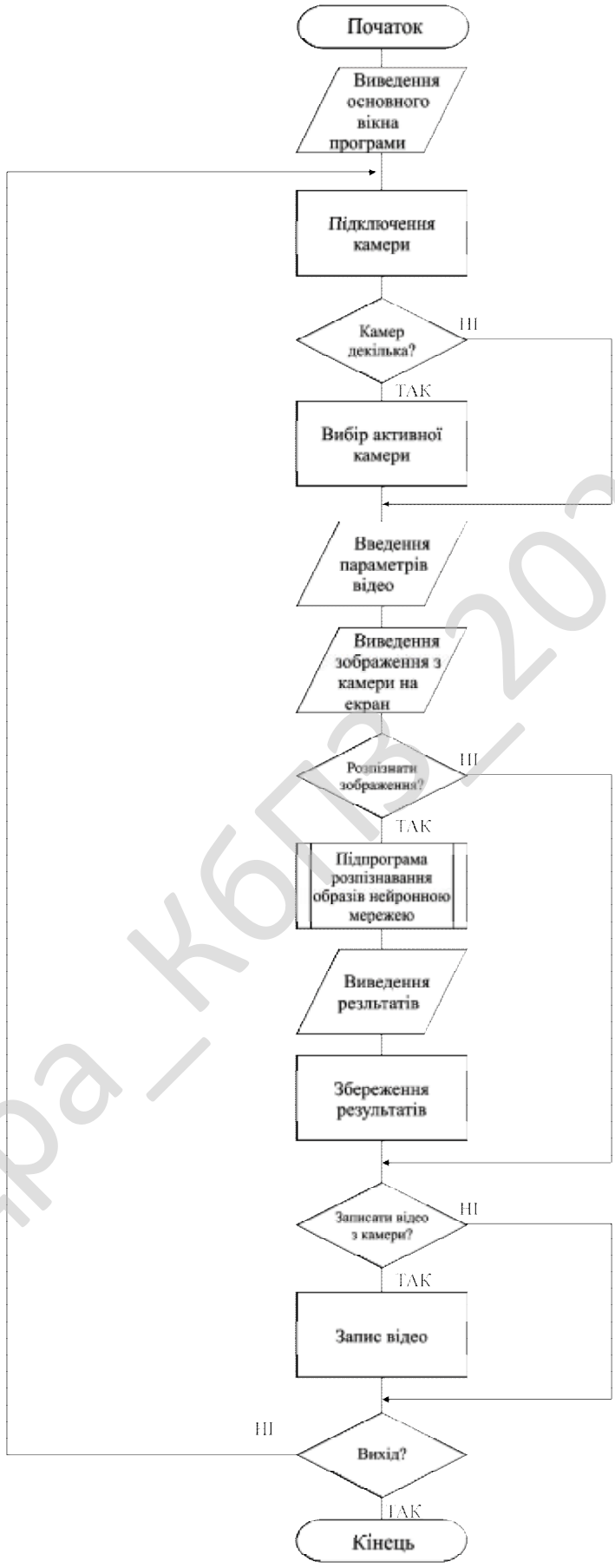

Рисунок 4.1 – Блок-схема роботи основної програми

Описавши блок-схему й алгоритм роботи основної програми, перейдемо до опису блок-схеми та алгоритму роботи підпрограми розпізнавання образів нейронною мережею.

Ця блок-схема зображена на рисунку 4.2.

Підпрограма працює наступним чином.

Спершу визначається, чи пройшла нейромережа процес навчання.

Якщо вона його не пройшла, то відбувається навчання нейромережі.

Якщо ж вона пройшла навчання, то тоді відбувається введення користувачем значення якості розпізнання.

Після цього наступним параметром користувач вводить метод, за допомогою якого буде проводитися розпізнання об’єктів.

Далі підпрограма отримує зображення від камери й починає його обробляти наступним чином:

– Відбувається бінарізація.

– Відбувається сегментація.

– Відбувається вихоплювання, обчислення ознак.

– Відбувається поворот, якщо треба, повторне вихоплювання.

– Відбувається стиснення в матрицю.

– Відбувається розпізнавання матриці.

Після виконання усіх вищеперерахованих дій, проводиться формування результату розпізнавання.

Розглянувши блок-схеми розробленого програмного забезпечення, наведемо більш детальний опис його реалізації.

Алгоритми розпізнавання образів

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

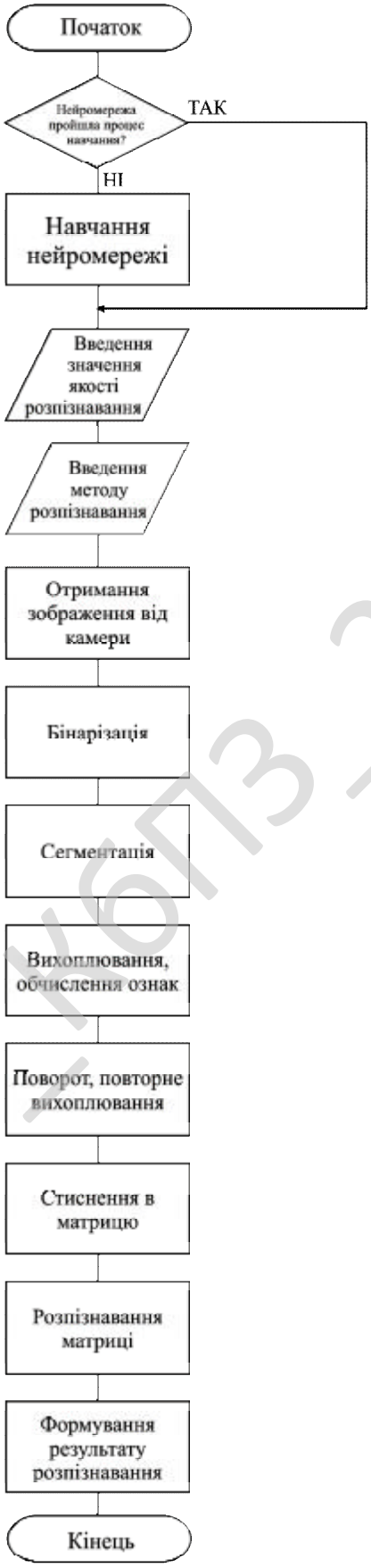

Рисунок 4.2 – Блок-схема роботи підпрограми розпізнавання образів нейронною мережею

Алгоритм скелетизації

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

Нейромережні структури

Солідне число нейронів вимагає солідні обчислювальні потужності, які звичайно відсутні на мобільних платформах. Однак треба мати, що нейромережі іноді дають досить цікаві результати, за рахунок своєї нелінійної структури, більше того деякі нейромережі здатні розпізнавати образи інваріантні щодо повороту без якої або зовнішньої передобробки. Так наприклад мережі на основі неокогнейтронів здатні виділяти деякі характерні риси образів, і розпізнавати їх як би образи не були повернені.

Інваріантні числа.

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

Поточечне процентне порівняння з еталоном.

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

Практична частина розпізнавання образів

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

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

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

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

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

Алгоритм представляє із себе послідовність наступних кроків:

1. Одержання зображення від камери.
2. Бінарізація.
3. Сегментація
4. Вихоплювання, обчислення ознак.
5. Поворот, повторне вихоплювання.
6. Стиск у матрицю.
7. Розпізнавання матриці нейромережею.

Далі буде описаний кожний пункт алгоритму.

Одержання зображення з камери в ОС Windows

Отже для початку нам потрібна картинка, з якої потім можна працювати. В ОС Windows всі джерела відео мають один шаблон. Сам я навчився працювати з відео за допомогою безкоштовних исходників DScap. Є пристрій, що може давати відеопотік (послідовність Bitmap) далі його потрібно настроїти, для цього запросити драйвер про всі можливі діалоги, які він (драйвер пристрою) може

дати. Далі руками в додатку налускати потрібні налаштування (яскравість, розрішення дають клацати всі пристрої, у деяких є число каналів, усякі фільтри й т.п). Потім установити число кадрів у секунду – як правило теж задається. І властиво по події захоплення кадру (драйвером якщо завгодно) буде викликатися функція користувача, на вхід іде той, хто її викликав, і ще параметр, властиво захоплений Bitmap у зазначеній палітрі й із зазначеною яскравістю й т.п.

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

Бінарізація

У даній програмі розпізнаються тільки бінарні образи, тому другим етапом після одержання картинки, вона бінарізуется. При роботі з кольоровою камерою перетворення з кольору в ЧБ іде по стандартній формулі: Y:=0.3*R+0.59*G+0.11*B.

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

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

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

Сегментація

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

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

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

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

Отже, деякі пояснення.

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

На стадії **бінарізації** ми повинні перетворити об'єкт зображення в бінарну матрицю даних.

Отже, із цього моменту для прискорення процесу починається робота вже не з об'єктом картинки, а з бінарною матрицею.

Наступним пунктом алгоритму повинна бути **сегментація**. Вона здійснюється Шляхом проходу по матриці зображення ліворуч праворуч, зверху долілиць. При проході виконуються наступні правила:

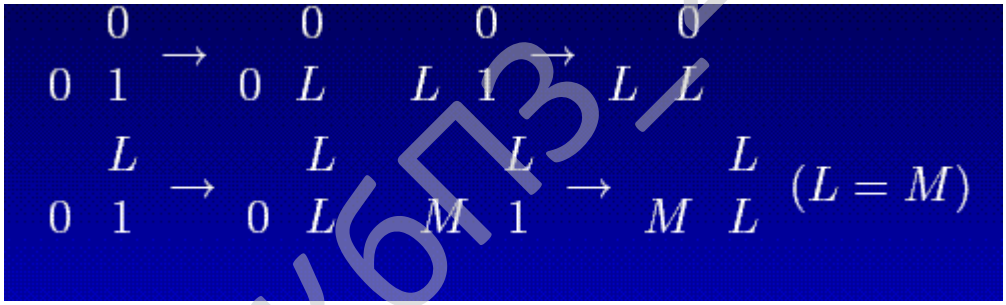

Рисунок 4.3 – Правило сегментації

Отже деякі коментарі за правилами: L і M це мітки, які після проходження алгоритму по масиві(до речі однократного) повинні бути привласнені ВСІМ пікселям об'єкта (елементам матриці, які до сегментації були одиничними).

Далі якщо вже мічені об'єкти зливаються, то повинне відбуватися й злиття міток.

На папірці це все виглядає досить просто, але якщо подумати те стає ясно, що для коректної роботи алгоритму потрібно перепризначувати мітки у ВСІЙ МАТРИЦІ зображення щораз, коли виконується останнє правило. Як я вже говорив раніше, одна із задач алгоритму це постаратися укластися в реально час,

тому проходити по масиві елементів 640x480 щораз, коли хочеться перепризначити мітку – це недозволенна розкіш.

Перше що приходить, на розум, щоб раціоналізувати цю частину алгоритму, це завести окремо масив міток, у якому треба прописувати яка мітка на яку посилається, і всі операції при проході по матриці зображення проводити тільки з масивом посилань міток. При цьому наприкінці проходу повинен бути масив міток, що посилаються один на одного, і тільки кілька міток повинні бути унікальними – які властиво й утворю об'єкти. Що б одержати унікальні об'єкти, які вже можна направити в модуль розпізнавання потрібно всього лише нормалізувати цей масив посилань і перепризначити всі елементи матриці на унікальні мітки виходячи з даних нормалізованого масиву посилань.

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

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

Термін відростки явно не буде зрозумілий читачеві, що знайомиться з матеріалом перший раз, тому пояснюю: на картинці вище (там де буква К и 1ка) стінки в 1ки ідеально рівні, на практиці ж, розрішення 640x480 у камери, спрямованої на підлогу, дає досить пристойну деталізацію, що може вихоплювати реальні або гадана камера "заусеніци" або перешкоди на рівних місцях образа (пускай навіть роздрукованого на принтері). У результаті алгоритм працює не з ідеальними картинками, а досить складними, але це відбувається тільки на стадії сегментації, **потім ці зайві деталі на більших об'єктах підуть**, як буде зрозуміло пізніше.

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

Приклад цей – це реальний вихід програми в текстовий файл, оскільки деякі мітки при проведенні сегментації стали двозначними, то картинка небагато поповзла, що утрудняє її сприйняття, однак при деякій фантазії й бажанні можна зрозуміти що відбувається.

Вихоплювання образа, обчислення деяких інваріантних чисел

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

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

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

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

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

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

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

При обчисленні ряду морфометричних ознак використовуються поняття механіки твердого тіла. Зокрема, це ставиться до довжин осей інерції об'єкта. Напрямку в тілі, що збігаються з півосями еліпсоїда інерції, називають головними осями інерції. Для знаходження головних осей інерції, що лежать у площині об'єкта, у функції **imfeature** використовуються наступні співвідношення [1, 2, 3].

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

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

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

Обчислимо кілька допоміжних величин:

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

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

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

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

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

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

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

Ексцентриситет визначається за допомогою співвідношення:

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

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

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

у противному випадку Про обчислюється як

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

Поворот образа, повторне вихоплювання

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

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

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

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

Стиск образа в матрицю заданого розміру

Як уже помітно з картинок вище, в оболонці програми з'явилася якась бінарна матриця, у яку стискається кожний окремо сегментований образ. Розмір матриці задається по потребам – тобто якщо потрібна більша деталізація, то краще використовувати матриці більшого розміру, ніж у моїй поточній версії

програми. Це дасть додаткові обчислення на стадії розпізнавання, але по ідеї й підвищить якість процесу. Однак **важливо розуміти,** що малому розрішенні розпізнаваної матриці дозволяє виправити деякі можливі помилки при повороті зображення(обчислення орієнтації образа), так як незначне відхилення в 5 градусів, не буде помітно після стиску образа.

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

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

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

Розпізнавання образа нейромережею

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

Структура обраної мною мережі досить стандартна:

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

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

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

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

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

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

4.2 Захист розробленого програмного забезпечення

Розроблене програмне забезпечення захистимо за допомогою алгоритму захисту інформації RSA. Спочатку необхідно обчислити пару ключів (секретний ключ і відкритий ключ). Для цього відправник електронних документів обчислює два більших простих числа P и Q, потім знаходить їхній добуток $N = P*Q$ і значення функції $\varphi(N) = (P-1)(Q-1)$. Далі відправник обчислює число E з умов $E < \varphi(N)$, НЗД $(E, \varphi(N)) = 1$ і число D з умов $D < N$, $E*D \equiv 1 (\text{mod } \varphi(N))$.

Пари чисел (E, N) є відкритим ключем. Цю пару чисел автор передає партнерам по переписці для перевірки його цифрових підписів. Число D зберігається автором як секретний ключ для підписування.

Допустимо, що відправник хоче підписати повідомлення M перед його відправленням. Спочатку повідомлення M (блок інформації, файл, таблиця) стискають за допомогою геш-функції h(-) у ціле число m: $m = h(M)$.

Потім обчислюють цифровий підпис S під електронним документом M, використовуючи геш-значення m і секретний ключ D: $S = m (\text{mod } N)$.

Пари (M, S) передається партнерові-одержувачеві як електронний документ M, підписаний цифровим підписом S, причому підпис S сформований власником секретного ключа D.

Після прийому пари (M, S) одержувач обчислює геш-значення повідомлення M двома різними способами. Насамперед, він відновлює геш-значення m', застосовуючи криптографічне перетворення підпису S з використанням відкритого ключа E: $m' = S^E (\text{mod } N)$.

Крім того, він знаходить результат гешування прийнятого повідомлення M с допомогою такої ж геш-функції h(-): $m = h(M)$.

Якщо дотримується рівність обчислених значень, тобто $S^E(\text{mod}) = h(M)$, то одержувач визнає пару (M, S) справжньою. Доведено, що тільки власник секретного ключа D може сформувати цифровий підпис S по документі M, а визначити секретне число D по відкритому числу E не легше, ніж розкласти

модуль *N* на множники. Крім того, можна строго математично довести, що результат перевірки цифрового підпису *S* буде позитивним тільки в тому випадку, якщо при обчисленні *S* був використаний секретний ключ *D*, що відповідає відкритому ключу *E*. Тому відкритий ключ E іноді називають "ідентифікатором" того, хто підписав.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розроблене програмне забезпечення реалізує систему розпізнання образів у структурі технічного захисту інформації банківської установи.

Програмно-апаратні вимоги:

– Загальний обсяг ОЗП: 512 Мбайт.

– Вільний простір на жорсткому диску: 25 Мбайт.

– Операційна система Microsoft Windows 10/11.

– USB камера, що може давати розрішення 640x480.

Програма складається з наступних модулів:

unit UMain – основна программа, в ній виконується ініціалізація камер;

unit UPreview – модуль роботи з камерами;

unit UCamTimers – модуль встановлення модуля камер;

unit UGraphConfig – модуль налагодження камер;

unit UCharAnalyz – модуль розпізнавання образів, наведемо основні функції модуля:

procedure TRobotRecognition.execute() – нитка паралельного процесу розпізнавання

procedure NeuroCompute – обчислення результатів нейромережею

procedure segment – сегментація бінарного зображення

function get_parent – рекурсивний пошук батьків мітки, для сегментації

function normalize – нормалізація масиву міток

procedure reader – сортування розпізнаного й читання вголос

procedure ConvertBitmapToMonoChrome – бінарізація

procedure BitmapToIcon – стиск у матрицю 16x16

function CompareIcons – процентне порівняння двох матриць

procedure AddSample – додавання приклада для навчання

function TSampleRule – порівняння образа відсотками або нейромережею

unit UAddSample – Обробка образа:

procedure imFeatures – Обробка образа, обчислення центру мас, орієнтація, поворот.

На рисунку 5.1 зображене основне вікно програми.

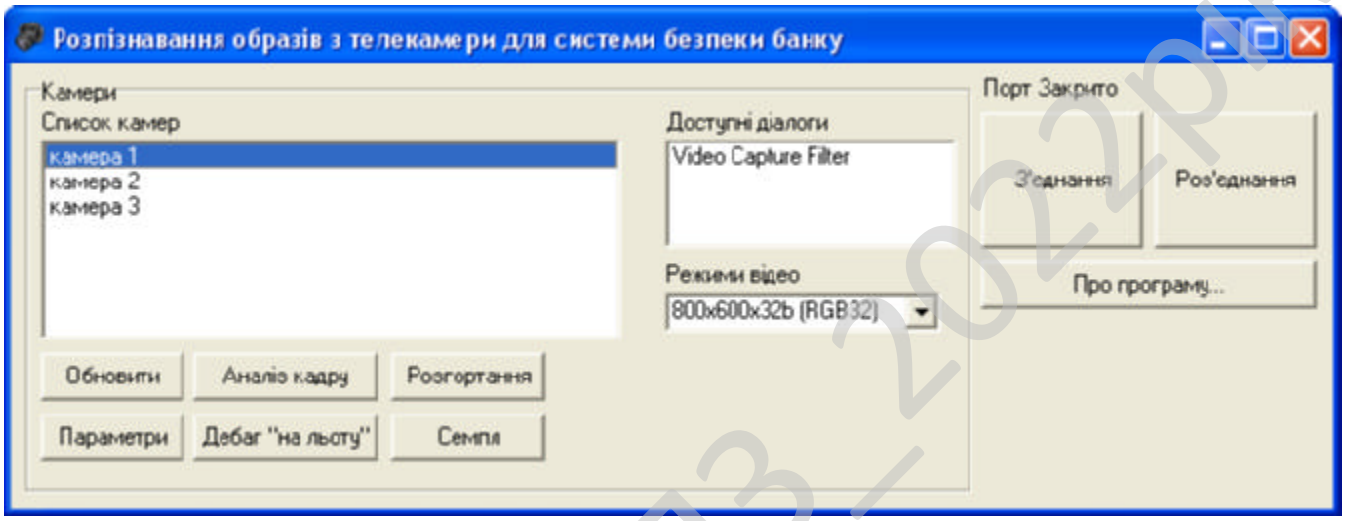

Рисунок 5.1 – Основне вікно програми

В основному вікні програми для початку треба вибрати активну камеру зі **«Списку камер»**, потім натиснути кнопку **«Параметри»** й у діалоговому вікні вказати параметри камери, поставити галки де потрібно (рисунок 5.2). В опції **«Функція»** потрібно вибрати **«полігон-вперед»**.

Потім у списку **Доступні діалоги** (у різних камер може бути різне число діалогів) треба вибрати параметри – число кадрів у секунду, яскравість, контрастність і т.п. І нарешті, для виведення головного діалогу програми треба натиснути кнопку **«Семпл»**.

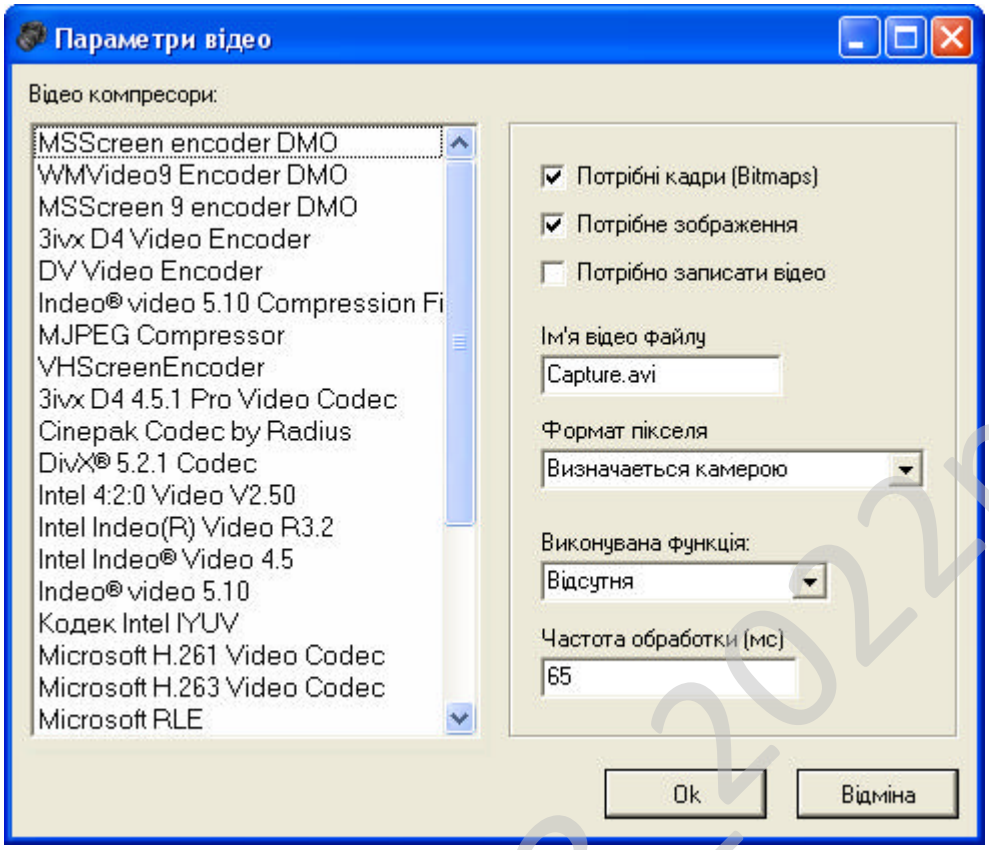

Рисунок 5.2 – Вікно параметрів відео

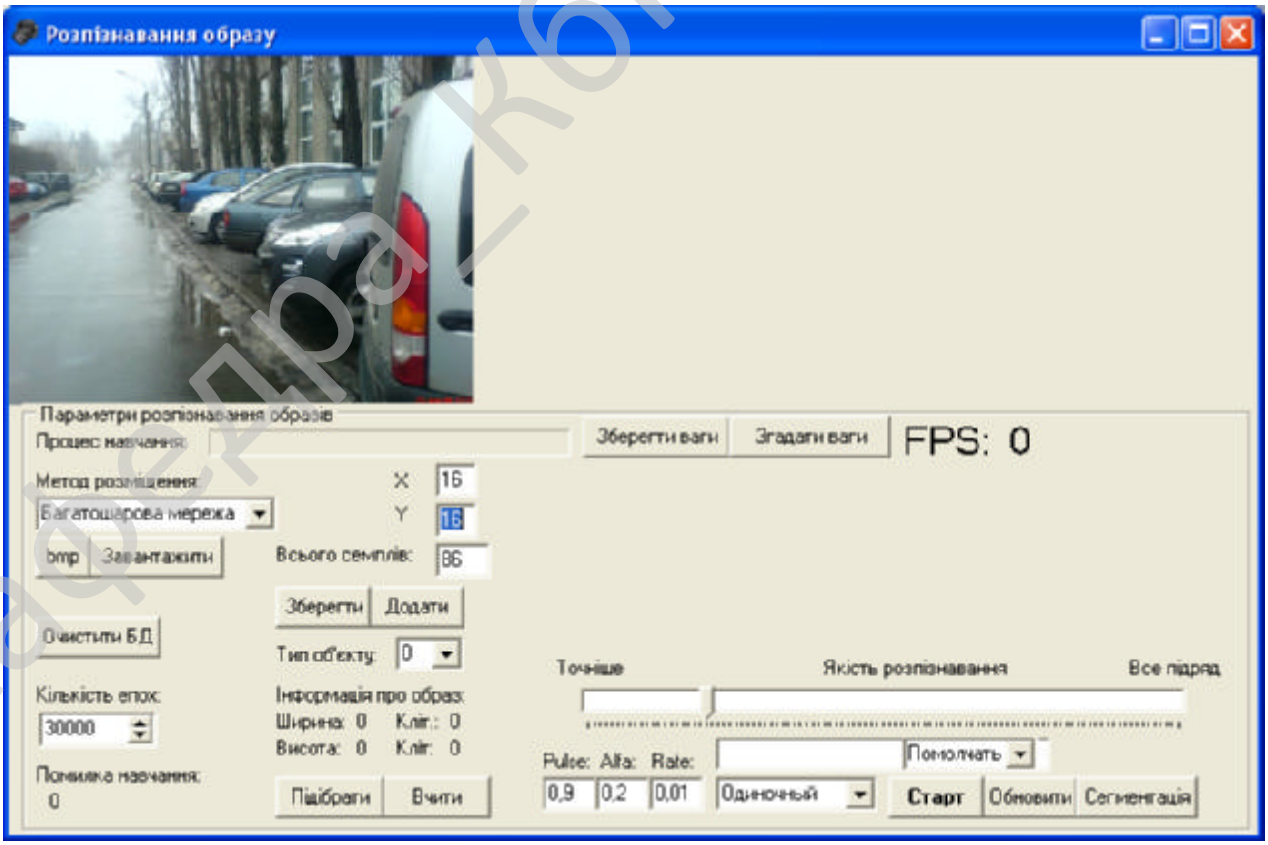

Рисунок 5.3 – Вікно розпізнавання образів

Для початку потрібно заготовити образи й навчити програму. Образи потрібно пред'являти білі на чорному(або поміняти в проге) і окремо.

Порядок дій при навчанні повинен бути наступний:

1. З папки DATA стерти файл config – у якому втримується інформація про попередні розучені образи.

2. Далі запустити програму, настроїти камеру, і відкрити вікно розпізнавання образів.

3. Нажати кнопку "обновити". (повинна з'явитися чиста ЧБ картинка.

4. Указати тип образа, що повинен бути доданий.

5. Нажати кнопку "додати". Повторити операцію(4 потім 5) відновлення й додавання стільки разів, скільки образів використовується.

6. Після того, як додані всі образи нажати кнопку "Save" поруч із "додати".

7. Закрити програму, відкрити в блокноті фаил config з папки DATA, переконатися, що дані для навчання вийшли коректными.

8. Запустити програму заного, відкрити вікно розпізнавання, увести параметри навчання й роботи нейромережі.

9. Нажати кнопку "учити".

10. Спостерігати за навчанням нейромережі (2-20 хвилин, залежно від машини й складності завдання).

11. Після успішного навчання (помилка менше 0, а краще 0.1) нажати кнопку "зберегти ваги".

12. Закрити програму.

Якщо все пройшло успішно, то на цьому етапі програма навчена на розпізнавання зазначених образів і готова приступитися до роботи.

Що б змусити програму працювати треба:

1. Запустити програму, вибрати й настроїти камеру, дуже важливо, що б налаштування були ідентичними як і при навчанні.

2. Нажати кнопку семпл, відкрити основне діалогове вікно розпізнавання

3. По потребі нажати кнопку обновити, після того як буде підготовлений об'єкт перед камерою.

4. Нажати кнопку "сегментація". Після чого повинна бути виведена вся інформація дебага на екран, і у фаили

5. Якщо дебаг пройшов нормально, всі образи виділяються й розпізнаються, то можна нажати кнопку "Пішов" – це запустить паралельный процес розпізнавання, але вже без візуальної інформації, що б не витрачати на неї час.

Для перегляду короткої довідки про програму слід натиснути на основному вікні кнопку **«Про програму…»**, після чого на екрані з’явиться вікно показане на рисунку 5.4.

Рисунок 5.4 – Довідка

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи розпізнання образів у структурі технічного захисту інформації банківської установи.

Метою розробки є дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

Об'єктом дослідження є процес розпізнання образів у структурі технічного захисту інформації банківської установи.

Предметом дослідження є методи розпізнання образів у структурі технічного захисту інформації банківської установи.

Методи дослідження базуються на методах захисту інформації, методах математичної статистики, методах розробки програмного забезпечення.

Наукова новизна отриманих результатів. У процесі рішення завдань, обумовлених цілями дослідження, отримані наступні результати:

– Удосконалено метод розпізнання образів у структурі технічного захисту інформації банківської установи.

– Розроблено вітчизняний продукт розпізнання образів у структурі технічного захисту інформації банківської установи, який має більш широкі можливості, на відміну від існуючих аналогів.

7 ДАНІ ПРО ЕКОНОМІЧНУ ЕФЕКТИВНІСТЬ РОЗРОБЛЕНОЇ ПРОГРАМИ

7.1 Техніко-економічне обґрунтування теми випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти

Після ознайомлення з підприємством та засобами розробки програмної продукції був розроблений план розробки програми. Був підрахований необхідний час для розробки та впровадження програми. Цей час склав 26 днів (один місяць).

В магістерській роботі були проведені дослідження та виконана програмна реалізація системи розпізнавання образів у структурі технічного захисту інформації банківської установи. Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір;

б) невеликі системні потреби;

в) незалежність від встановлених на комп'ютері баз даних;

г) зручність у користуванні та надійність.

Таблиця 7.1 – Початкові дані

Показники	Позначення	Характеристика або величина
1	2	3
1. Кількість розроблених програм період, шт.	N	1
2. Кількість екземплярів програм, шт.	Ne	20
3. Запланований термін розробки, днів	Fpq	26 (1 місяць)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2

Продовження таблиці 7.1

1	2	3
7. Кількість макетів вхідної інформації	–	3
8. Кількість форм вихідної інформації.	–	4
9. Мова програмування (1-6)	–	1
10. Попередній досвід (1-6)	–	3
11. Гнучкість проекту ПП (1-6)	–	3
12. Детальність проекту ПП (1-6)	–	2
13. Рівень спрацьованості колективу (1-6)	–	2
14. Ступінь вимірності процесів (1-6)	–	3
15. Необхідна надійність програмного забезпечення (1-6)	–	2
16. Розмір бази даних (порівняно з розміром програми) (1-6)	–	2
17. Складність кінцевого програмного продукту (1-6)	–	2
18. Необхідний рівень забезпечення повторного використання (1-6)	–	2
19. Документованість відповідно до планованого життєвого циклу (1-6)	–	2
20. Вимоги до швидкодії ПП (1-6)	–	2
21. Обмеження на розміри основного сховища даних (1-6)	–	2
22. Різноманітність використовуваних обчислювальних платформ (1-6)	–	2
23. Професійний рівень аналітиків (1-6)	–	2
24. Професійний рівень програмістів (1-6)	–	2
25. Постійність складу команди розробників (1-6)	–	2
26. Досвід розробки додатків (1-6)	–	2
27. Досвід роботи з обчислювальною платформою (1-6)	–	2

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	20000
33. Норматив додаткової зарплати, % :	$Н_д$	10
34. Норматив відрахувань у соціальні фонди, %	Нс	22
35. Норматив загальногосподарських витрат, %	Нг	15
36. Норматив витрат на освоєння нових мов програмування, %	Нп	15
37. Рівень рентабельності програмної продукції, %	Ре	50
38. Ставка податку на додану вартість, %	Ндв	20

7.2 Розрахунок трудомісткості розробки програмної продукції

Значення трудомісткості розробки програмного забезпечення для стадій ТЗ, ЕК, ТП та ВП визначаємо по типовим нормам часу приведеним в додатках МВ. Стадія РП як одна з найбільш тривалих і трудомістких робить значний вплив на інші стадії проекту.

Визначимо трудомісткість розробки ПЗ для стадії РП.

Обчислюємо номінальні трудовитрати, люд-міс.:

$$T_{ном} = A\ \text{Size}^B, \qquad (7.1)$$

де: A – коефіцієнт Боема, A = 2,45;

Size – загальний об'єм відлагодженого програмного коду, тис. рядків;

B – показник ступеня, що визначається співвідношенням:

$$B = 1{,}01 + 0{,}001\sum W_i, \qquad (7.2)$$

де: *Wi* – сумарне значення п'яти показників (МВ, додаток 2), що відображають особливості розробки проекту програмного продукту (ПП) і колективу розробників.

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}026.$$

$$T_{ном} = 2{,}45 \cdot 2{,}7^{1{,}026} = 6{,}78 \text{ люд-міс.}$$

Визначаємо уточнені (з урахуванням приведених в МВ додатку 3 сімнадцяти додаткових коефіцієнтів) трудовитрати, люд-міс.:

$$T_{уточн} = T_{ном}\, ПВ_j, \qquad (7.3)$$

де: $ПВ_j$ – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн} = 6{,}78 \cdot (0{,}88 \cdot 0{,}93 \cdot 0{,}88 \cdot 0{,}91 \cdot 0{,}95 \cdot 1 \cdot 1 \cdot 0{,}87 \cdot 1{,}22 \cdot 1{,}16 \cdot 1{,}1 \cdot 1{,}1 \cdot$$
$$\cdot 1{,}12 \cdot 1{,}1 \cdot 1 \cdot 1{,}1 \cdot 1{,}1) = 9{,}37 \text{ люд-міс.}$$

Ці коефіцієнти дозволяють диференційовано оцінювати результати роботи програмістів, беручи до уваги швидкодію програми, використання різноманітних обчислювальних платформ і інструментів розробки, взаємодію декількох серверів, вимоги до об'ємів баз даних і ін.

Визначаємо підсумкові трудовитрати по стадії робочий проект, люд-дні:

$$T_{РП} = 0{,}3 C T_{уточн}^{0{,}33+0{,}2(B-1{,}01)} S, \qquad (7.4)$$

де: *C* – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4);

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%.

$$T_{РП} = 0{,}3 \cdot 2{,}84 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 100 = 168 \text{ люд/день.}$$

Для зручності визначення загальної трудомісткості на розробку програмного забезпечення результати розрахунків по стадіям зводимо до таблиці 7.2.

Таблиця 7.2 – Визначення трудомісткості розробки програмного забезпечення

Стадії розробки	Трудомісткість за типовими нормами та розрахунками	
	Величина, люд/дні	Підстава
Технічне завдання	9	Д5
Ескізний проект	10	Д6
Технічний проект	9	Д7
Робочий проект	168	Ф 7.1-7.4
Впровадження	13	Д13
Всього	209	–

7.3 Визначення чисельності виконавців і планового фонду зарплати

Чисельність ставок інженерів-програмістів для розробки програмного забезпечення визначається за формулою:

$$Ч = \frac{T_{пз} N}{F_{рq} - H_{ев}}, \quad (7.5)$$

де: $F_{рq}$ – плановий фонд робочого часу одного спеціаліста, днів;

$T_{пз}$ – трудомісткість розробки програмного забезпечення люд-дні.

$$Ч = \frac{209 \cdot 1}{26\text{-}5} = 10 \text{ ставок.}$$

Чисельність інженерів-електронщиків для проведення технічного обслуговування та ремонту комп'ютерних мереж визначається в залежності від наявності технічних засобів і норм витрат часу на виконання профілактичних робіт на протязі року.

Визначаємо затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за період розробки. Результати розрахунку зводимо до таблиці 7.3.

Таблиця 7.3 – Затрати часу на виконання профілактичних робіт по обслуговуванню обладнання за розрахунковий період

Найменування обладнання	Профілактичне обслуговування			
	Кількість хв. на один. обл.	Кількість облад-нання	Затрати часу в хв.	Затрати часу в год.
Системний блок ПК	90	14	1260	21
Монітор	60	14	840	14
Клавіатура	30	14	420	7
Маніпулятор «мишка»	30	14	420	7
Принтер матричний	60	1	60	1
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	2	40	0,67
Концентратор-маршрутизатор	30	3	90	1,5
Кабельні господарства ЛОМ на 1 м. п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	67,83

Час на профілактику обладнання в загальному балансі робочого часу інженерів-електронщиків неповинен складати більше 10%.

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

$$Ф_{др}^{с} = \frac{З_ч \cdot n_{міс}}{1,2}, \quad (7.6)$$

$$Ф_{др}^{с} = \frac{67,83 \cdot 1}{1,2} = 56 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

$$Ч_{ел} = \frac{Ф_{др}^{с}}{F_{др} \cdot T_{зм}}, \quad (7.7)$$

$$Ч_{ел} = 56/(26 \cdot 8) = 0{,}25 \text{ ставки.}$$

Для забезпечення нормального технічного обслуговування засобів ТО та мереж, необхідно прийняти найбільше ціле значення розрахункової чисельності інженерів-електронщиків.

Чисельність інженерів-системотехніків, адміністраторів мережі, дизайнерів WEB вузлів, системних програмістів (аналітиків), бухгалтерів-економістів визначається за потребою в залежності від функціональних обов'язків. Після визначення чисельності персоналу складається штатний розклад.

Таблиця 7.4 – Розрахунок чисельності штатного персоналу сектору системного та адміністративного обслуговування засобів ОТ та комп'ютерних мереж

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2019, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,1	0,3
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,1	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,1	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	2,1	
Всього		2,4	

Продовження таблиці 7.4

Посада	Вид роботи	Час	К-ть штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	1	0,25
	Підтримка постійних клієнтів	0,5	
	Оформлення договорів, ведення тендерів	0,25	
	Контроль взаєморозрахунків з постачальниками	0,25	
Всього		2	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	1	0,25
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,25	
	Розміщення графіки і контенту на Інтернет сторінках	0,25	
Всього		2	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	0,2	0,1
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		0,8	

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	18000	18000
Продакт-менеджер	0,25	12000	3000
Інженер-програміст	10	8360	83600
Інженер-електронщик	0,25	9000	2250
Інженер-системотехнік	0,25	9000	2250
Адміністратор мережі	0,3	9000	2700
Системний програміст	0,2	15000	3000
Дизайнер WEB	0,2	16000	3200
Інженер-верстальник	0,1	11000	1100
Бухгалтер-економіст	0,1	15000	1500
Всього за період розробки	$R_{сп}$ = 12,65	-	$Ф_{роб}$ = 120600

Розрахуємо середньоденну зарплату одного виконавця:

$$З_{сд} = \frac{Ф_{роб}}{R_{сп} F_{рд}}, \quad (7.8)$$

де: $Ф_{роб}$ – загальна сума зарплати за плановий період, грн.

$$З_{сд} = \frac{120600}{12,65 \cdot 26} = 366 \; грн.$$

7.4 Розрахунок капітальних вкладень та амортизаційних відрахувань у розробника

Балансова вартість будівель визначається з урахуванням кількості робочих місць виконавців, питомої площі на одне робоче місце, та вартості одного квадратного метра виробничої площі:

$$Б_{уд} = R^1_{сп} S_у Ц_{пл}, \quad (7.9)$$

де: $R^1_{сп}$ – кількість робочих місць виконавців, шт. Приймаємо 13 робочих місць;

$S_у$ – питома площа на одне робоче місце, $м^2$;

$Ц_{пл}$ – вартість одного квадратного метра площі, грн.

Згідно даних ТОВ науково-дослідницького консалтингового підприємства «Пекторaль» (м. Кіровоград) ціна одного квадратного метра площі новобудови, вік якої не перевищує 30 років, по місту складає 500…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

$$Б_{уд} = 8 \cdot 8 \cdot 20000 = 1280000 \text{ грн.}$$

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн. Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

$$І_{нв} = R^1_{сп} \cdot Ц_м, \quad (7.10)$$

де: $Ц_м$ – ціна меблів для одного робочого місця, грн.

$$І_{нв} = 8 \cdot 3500 = 28000 \text{ грн.}$$

Балансова вартість обчислювальної техніки визначається по оптовим цінам постачальника з врахуванням витрат на транспортування.

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу Інтернет-магазину Компбест за 19.10.22 – джерело https://compbest.com.ua.

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп'ютер		10947

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Системний блок	LogicPower SFF	7347
Процесор	Intel Core i5-7400 (4 ядра по 3.0 - 3.5 GHz) Cache Memory 6 MB	-
Системна плата	Gigabyte GA-B250M-D3H, 4x USB 3.1, 4 USB 2.0, 8x Audio Ports, 1x LAN (RJ-45), 1 PS/2, 1x HDMI, 1x DP, 1x DVI, 1x VGA, 1 M.2, 1x SATA Express, 6x SATA	-
Відеокарта	Integrated Intel HD Graphics 630, частот графічної системи 350 MHz - 1.00 GHz	-
Жорсткий диск	240 GB SSD Goodram CX200	-
Оперативна пам’ять	16 GB DDR3	-
DVD-привод	DVD -RW/+RW , LG SATA SuperMulti Bul 22x, SecurDisc, black	-
Корпус	ATX LogicPower SFF, 3GTLA-489, PSU 350W(FSP Brand: ATX-350PNR, 12cm) black, (front bezel – black+light silver; bod material – 0.6mm), 80mm fan (rear) 2xUSB2.0/AUDIO/MIC, Air Duct, Tool-les chassis design,Thermally Advantaged Chassis	-
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
інше	Клавіатура, мишка, система охолодженн процесора	Подару-нок
Монітор	Fujitsu B24T-7 з матрицею 24" (1920x1080 TN LED	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USB cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Витрати на транспорт, монтаж та випробування можуть бути прийняті в межах до 10% від оптової ціни.

Для визначення необхідної кількості капітальних вкладень складемо таблицю 7.8.

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,64
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,14

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57443
Група 5, 6			
4. Вимірювальні пристрої	9031	25	2257,75
5. Транспортні засоби	143000	20	28600
6. Господарський інвентар	28000	25	7000
Всього по групі 5,6	180031	-	37857,75
Нематеріальні активи			
7. Нематеріальні активи	20000	10	2000
Разом	$К_р$ = 1722916		$А_р$ = 167701

Примітка: вартість автомобіля Sens (Standard+) взята по даним з автосалону «Кіровоград-Авто», джерело http://kirovograd-avto.ukravto.ua /catalog/tm-9/model-80/description, складає 143000 грн

7.5 Визначення собівартості розробки та ціни програмної продукції

Визначимо основну зарплату виконавців:

$$З_о = \frac{З_{сд} \cdot Т_{пз}}{N_е}, \quad (7.11)$$

де: $N_е$ – кількість екземплярів програм, шт.

$$З_о = 366 \cdot 209/20 = 3825 \text{ грн.}$$

Визначимо додаткову зарплату (оплата відпусток, виконання державних та суспільних обов'язків) на рівні 10%:

$$З_д = З_о \cdot Н_q \cdot 0{,}01, \quad (7.12)$$

де: $Н_q$ – норматив додаткової зарплати, %.

$$З_д = 3825 \cdot 10 \cdot 0{,}01 = 383 \text{ грн.}$$

Відрахування на соціальні потреби за нормативом $Н_с = 22\%$ від суми основної та додаткової зарплати:

$$С_{оц} = 0{,}01 \cdot Н_с (З_о + З_д), \quad (7.13)$$

де: $Н_с$ – відрахування на соціальні потреби, %.

$$С_{оц} = 0{,}01 \cdot 22(3825+383) = 926 \text{ грн.}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом $Н_г = 15\%$ від основної зарплати:

$$Г_{осп} = З_о \cdot Н_г \cdot 0{,}01, \quad (7.14)$$

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 3825 \cdot 15 \cdot 0{,}01 = 574 \text{ грн.}$$

Визначимо витрати на матеріали для розробки програмної продукції за нормами споживання та діючими цінами за одиницю виміру:

$$З_М = (З_{М1} + З_{М2} + З_{М3})/N_е, \quad (7.15)$$

де: $З_{М1}$ – вартість паперу, грн.; $З_{М2}$ – вартість запам'ятовуючих пристроїв, грн.; $З_{М3}$ – вартість фарби, картріджей, тонеру, грн.; $N_е$ – кількість екземплярів програм, шт.

Згідно норм приймаємо одну півпачки паперу на місяць розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$ = 210 грн., визначаємо вартість паперу за період розробки $N_м$ = 1 міс:

$$З_{М1} = Ц_п \cdot N_м \cdot n. \quad (7.16)$$

$$З_{М1} = 210 \cdot 1 \cdot 0{,}5 = 105 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо одну):

$$З_{М2} = \sum Ц_д , \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR TDK 700Mb, 80Min, 52x Cake box – 23 грн./шт., DVD-R LG 4,7Gb, 16x speed Cake box – 43 грн./шт.

$$З_{М2} = 43 \text{ грн.}$$

Згідно виданих викладачем норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

де: $Ц_з$ – вартість розхідних матеріалів друкуючих пристроїв: відновлення та заправка картриджу для Canon i-SENSYS LBP6030W – 574 грн.; картридж для Epson Stylus Photo P50 – 558 грн.; відновлення картриджу для MF217W – 570 грн.

$$З_{М3} = 574+558+570 = 1702 \text{ грн.}$$

$$З_М = (105 + 43 + 1702)/20 = 93 \text{ грн.}$$

Визначимо витрати на освоєння нових мов програмування або операційних систем за нормативом ($Н_п$ = 15%) від основної зарплати виконавців:

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \quad (7.19)$$

де: $Н_п$ – норматив витрат на освоєння нових мов програмування, %.

$$О_п = 3825 \cdot 15 \cdot 0{,}01 = 574 \text{ грн.}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 20 прим.):

$$A_м = \frac{A_р \cdot N_{міс}}{N_е \cdot 12}, \quad (7.20)$$

де: $A_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$A_м = 167701 \cdot 1/(20 \cdot 12) = 699 \text{ грн.}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції:

$$C_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м. \quad (7.21)$$

$$C_п = 3825+383+926+574+93+574+699 = 7074 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності ($P_п$) програмної продукції, яка залежить від складності програми та ступеня новизни задачі.

Для даного програмного забезпечення рівень рентабельності складає 50%.

$$П_р = 0{,}01 \cdot P_п \cdot C_п, \quad (7.22)$$

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 7074 = 3537 \text{ грн.}$$

Величини ціна підприємства, податок на додану вартість, відпускна ціна програмної продукції визначаються за формулами, приведеними в таблиці 7.9

Таблиця 7.9 – Нормативна калькуляція собівартості розробки програмного забезпечення задачі

Найменування статей витрат	Позначення	Величина, грн
1	2	3
1. Основна зарплата виконавців	$З_о$	3825
2. Додаткова зарплата виконавців	$З_д$	383
3. Відрахування на соціальні потреби	$С_{оц}$	926
4. Загальногосподарські витрати	$Г_{осп}$	574
5. Витрати на матеріали	$З_М$	93
6. Освоєння нових операційних систем, мов програмування	$О_п$	574

Продовження таблиці 7.9

1	2	3
7. Амортизація основних фондів	$А_м$	699
8. Повна собівартість програмного забезпечення	$С_п$	7074
9. Плановий прибуток	$П_р$	3537
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	10611
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	2122,2
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	12733,2

7.6 Визначення об’єму капітальних вкладень у споживача програмної продукції

Об’єм капітальних вкладень у споживача програмної продукції визначаємо на основі балансової вартості основних фондів, яка враховує ціну, транспортно-заготівельні витрати, вартість будівель, монтажних та пусконалагоджувальних робіт, а також витрати на випробування у виробничих умовах. Результати розрахунків зводимо у таблицю 7.10.

Таблиця 7.10 – Розрахунок об’єму капітальних вкладень у споживача програмної продукції

Найменування капітальних вкладень	Сума за варіантами, грн.	
	Базовий	Новий
Вартість програмної продукції	–	12733
Всього капітальних витрат	–	12733

7.7 Визначення експлуатаційних витрат

Експлуатаційні витрати у споживача програмної продукції визначаємо при умові роботи підсистеми на протязі року. Результати зводимо до таблиці 7.11.

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	20130	6710
2. Витрати на електроенергію	$З_{ел}$	3456	3072
3. Витрати на амортизацію	$З_{ам}$	0	3183
Всього витрат за рік	$І$	23586	12965

Витрати на обслуговування системи:

$$З_р = Т_р \cdot З_г \cdot (1 + 0{,}01 \cdot Н_q) \cdot (1 + 0{,}01 \cdot Н_c), \qquad (7.23)$$

де: $Т_р$ – кількість годин обслуговування системи за рік, *год.*;

$З_г$ – заробітна плата обслуговуючого персоналу, *грн / год*.

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 150 годин на рік до 50 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 150 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 20130 \text{ грн.}$$

до:

$$З_{р\,нов} = 50 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 6710 \text{ грн.}$$

Витрати на електроенергію визначаються з урахуванням споживаємої потужності ($П_{ел}$) в кіловатах, часу експлуатації технічних засобів ($Т_р$) в годинах та ціни однієї кіловат-години ($Ц_{ел}$):

$$З_{ел} = П_{ел} \cdot Т_р \cdot Ц_{ел}. \qquad (7.24)$$

$$З_{ел\ баз} = 0{,}45 \cdot 2400 \cdot 3{,}2 = 3456 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}40 \cdot 2400 \cdot 3{,}2 = 3072 \text{ грн.}$$

Витрати по амортизації визначаються на основі норм амортизаційних відрахувань, вартості програмної продукції і основних фондів. Для розрахунку складаємо таблицю 7.12.

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	12733	–	3183,25
Всього відрахувань	-	–	12733	–	3183,25

7.8 Визначення економічної ефективності програмної продукції

Економічна ефективність програмного забезпечення визначається для виготовлювача і споживача за такими показниками.

Величина економічного ефекту при виготовленні програмної продукції, розраховуємо за формулою:

$$E_в = (Ц_п - C_п) \cdot N_е - \sum_{i=1}^{m} E_{p_m} \cdot K_{p_m}, \quad (7.25)$$

де: $K_р$ – балансова вартість основних фондів розробника, грн.; $E_р$ – розрахунковий коефіцієнт капіталовкладень.

$$E_в = (10611 - 7074) \cdot 20 - \cdot(0{,}05 \cdot 1408000 + 0{,}5 \cdot 114885 +$$
$$+0{,}2 \cdot 143000 + 0{,}25 \cdot 37031 + 0{,}1 \cdot 20000)\ 1/12 = 58909 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$T_в = \frac{K_р}{(Ц_п - C_п) \cdot N_е}, \quad (7.26)$$

де: $K_р$ – балансова вартість основних фондів розробника.

$$T_б = \frac{1722916}{(10611 - 7074) \cdot 20 \cdot 12 / 1} = 2 \text{ роки}.$$

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	20
2. Повна собівартість розробленої програми	Грн.	7074
3. Ціна розробленої програми	Грн.	10611
4. Плановий прибуток від реалізації розробленої програми	Грн.	3537
5. Рентабельність програмної продукції	%	50
6. Об'єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1722816
7. Загальний прибуток від реалізації програмної продукції	Грн.	70740
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	58909
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Роки	2
10.Об'єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	12733
11.Величина економічного ефекту у користувача програмної продукції	Грн.	7438
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Роки	1,2

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

$$E_{сп} = (I_б - I_н) - E_н(K_н - K_б), \qquad (7.27)$$

де: $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно; $К_б$, $К_н$ – об'єм капітальних вкладень за варіантами, що порівнюються.

$$E_{сп} = (23586\text{-}12965)\text{-}0{,}25 \cdot 12733 = 7438 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у споживача програмної продукції за рахунок зниження експлуатаційних витрат:

$$T_{сп} = \frac{К_н - К_б}{I_б - I_н}, \qquad (7.28)$$

$$T_{сп} = \frac{12733}{23586 - 12965} = 1{,}2 \text{ роки}.$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

7.9 Висновки

Розроблена програма економічно вигідна. За рахунок впровадження програмного забезпечення досягається скорочення часу обробки інформації, підвищується культура праці, підвищення якості управлінських рішень.

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Охорона здоров'я працівників, забезпечення безпеки умов праці, ліквідація професійних захворювань і виробничого травматизму повинна складати одну з головних завдань роботодавця.

Основою охорони праці є науковий аналіз умов праці, технологічних процесів, виробничого обладнання, робочих місць, трудових операцій, організації виробництва з метою виявлення шкідливих і небезпечних виробничих факторів, їх властивостей, особливостей виливу на організм людини. На підставі такого аналізу розробляються заходи та засоби, спрямовані на мінімізацію несприятливого впливу виробничих факторів, створення безпечних та нешкідливих умов праці [3].

Для того, щоб об’єктивно проаналізувати відповідність умов праці діючим нормативно-правовим актам, необхідно здійснити санітарно-гігієнічну характеристику умов праці відділу, в якому працює програміст, над розробкою даного програмного продукту.

В зв’язку з цим необхідно сконцентрувати увагу на небезпечних і шкідливих чинниках пов’язаних з постійною роботою за комп’ютером.

Електробезпека є одним із критичних питань для співробітників, що працюють із технікою, яка одержує живлення з електричної мережі. При невиконанні норм електробезпеки можлива поразка електричним струмом.

8.2 Аналіз умов праці на робочому місці ІТ-фахівця

На робочому місці ІТ-фахівця (або програміста) виникають небезпечні та шкідливі для безпечної життєдіяльності фактори:

– підвищений рівень шуму;

– несприятливі мікрокліматичні умови;

– недостатній рівень освітленості;

– шкідливі речовини;

– підвищений рівень електромагнітних випромінювань радіочастот;

– висока напруга електричної мережі;

– статична електрика та інші.

Робота програміста супроводжується також підвищеним ступенем напруженості трудового процесу. При систематичному впливі виробничих факторів, які не відповідають нормативним показникам, зростає рівень професійно зумовленої захворюваності працюючих та можуть виникнути професійні захворювання органів зору, руху, нервової системи. Таким чином, вивчення умов праці на робочому місці програміста є необхідною умовою запобігання негативних наслідків впливу небезпечних та шкідливих факторів. Робоче місце, добре пристосоване до трудової діяльності інженера, правильно і доцільно організоване, щодо простору, форми, розміру забезпечує йому зручне положення при роботі і високу продуктивність праці при найменшому фізичному і психічному напруженні [2].

Нормування параметрів проводиться в залежності від періоду року та категорії важкості виконуваних робіт. Для постійних робочих місць, якими є робочі місця ІТ-фахівців, встановлені оптимальні параметри мікроклімату, а за неможливості їх дотримання використовують допустимі параметри. Робота ІТ-фахівця за важкістю відноситься до Іа (роботи, що виконуються сидячи і не потребують фізичного напруження) та Іб (роботи, що виконуються сидячи, стоячи або пов'язані з ходінням та супроводжуються деяким фізичним напруженням) категорій. В таблиці 8.1. наведені оптимальні параметри мікроклімату в приміщеннях.

Таблиця 8.1 – Параметри мікроклімату для приміщень з ПК

Період року	Параметр мікроклімату	Величина
Холодний	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	22...24°С; 40... 60%; до 0,1 м/с
Теплий	Температура повітря в приміщенні; відносна вологість; швидкість руху повітря	23...25 °С 40...60% 0,1...0,2 м/с

Виміряні за допомогою приладів температура та вологість у приміщеннях праці ІТ-фахівців повинні відповідати зазначеним у таблиці для теплого періоду року. Слід зазначити, що для нормалізації параметрів мікроклімату слід використовувати у приміщеннях кондиціювання повітря, або забезпечити подачу свіжого повітря системами вентиляції. Норми подачі свіжого повітря наведені у таблиці 8.2.

Таблиця 8.2 – Норми подачі свіжого повітря в приміщення

Характеристика приміщення	Об'ємна витрата свіжого повітря, що подається в приміщення, $м^3$ на одну людину в годину
Об'єм до 20 $м^3$ на людину 20... 40 $м^3$ на людину Більше 40 $м^3$на людину	Не менше 30 Не менше 20 Може біти використана природна вентиляція

Створення сприятливих умов праці і правильне естетичне оформлення робочих місць на виробництві має велике значення як для полегшення праці, так і для підвищення його привабливості, позитивно впливає на продуктивність праці.

Забарвлення приміщень і меблів повинні сприяти створенню сприятливих умов для зорового сприйняття, гарного настрою. У службових приміщеннях, у яких виконується одноманітна розумова робота, що вимагає значної нервової напруги і великого зосередження, забарвлення повинно бути спокійних тонів – малонасичені відтінки холодного зеленого або блакитного кольорів [1].

При розробці оптимальних умов праці програміста необхідно враховувати освітленість. Раціональне освітлення робочого місця є одним з найважливіших факторів, що впливають на ефективність трудової діяльності людини, що попереджають травматизм і професійні захворювання. Правильно організоване освітлення створює сприятливі умови праці, підвищує працездатність і продуктивність праці. Освітлення на робочому місці програміста повинно бути таким, щоб працівник міг без напруги зору виконувати свою роботу. Стомлюваність органів зору залежить від ряду причин: недостатність освітленості; надмірна освітленість; неправильний напрям світла. Недостатність освітлення приводить до напруги зору, ослабляє увагу, приводить до настання передчасної стомленості. Надмірно яскраве освітлення викликає засліплення, роздратування і різь в очах. Неправильний напрямок світла на робочому місці може створювати різкі тіні, відблиски, дезорієнтувати працюючого. Всі ці причини можуть призвести до нещасного випадку або профзахворювань. [2]

8.3 Пропозиції щодо підвищення працездатності ІТ-фахівців

Поява та впровадження нових інформаційно-комунікаційних технологій зумовлює необхідність подальшого вдосконалення охорони праці фахівців it-індустрії. Все це потребує розробки нових нормативно-правових актів з регламентації праці та відпочинку фахівців it-індустрії і стандартів підприємств, центрів комп'ютерної техніки, центрів інформаційних технологій, сучасних комп'ютерних класів. Для підвищення розумової працездатності то зорової роботи повинна здійснюватися ергономічна оптимізація в рамках системи

«оператор-термінал», яка сприятиме результативній фізичній та інтелектуальний працездатності і відновленню психосоматичного здоров'я фахівців it-індустрії. Всі наведені заходи щодо вдосконалення охорони праці фахівців it-індустрії повинні контролюватися службою охорони праці та комісією з охорони праці підприємства. Особливе значення у соціальному захисті цієї категорії працівників належить прийняття комплексного договору, який може забезпечити фахівців додатковими пільгами та компенсаціями.

Для більшого розуміння, пропозиції щодо підвищення працездатності it-фахівців, розіб'ємо на декілька категорій:

Середовище і розпорядок праці. Для мінімізації негативних ефектів, що пов'язані з перевтомленням it-фахівців, потрібно чітко прописати і реалізувати графік періодів праці-відпочинку, щоб фахівець міг можливість переключити увагу, дати можливість відпочити очам, мозку, елементарно, встати розім'яти ноги. Також потрібно зробити максимально комфортними умови мікроклімату у офісному приміщенні, де працюють it-фахівці. Мається на увазі встановлення і експлуатація, коли виникає необхідність, кондиціонерів, опалення, та системи вентиляції, задля попередження перегрівання, переохолодження it-фахівців, і подальшої неможливості ними виконувати свої функції. Також, за можливості, нами пропонується введення практики віддаленої праці it-фахівцями, якщо роботодавець не може забезпечити оптимальні і безпечні умови в офісному приміщенні, або якщо фахівця вони не влаштовують із певних причин.

Фізичні і психоемоційні чинники. Першим і найважливішим чинником, що впливає на працездатність it-фахівців є робоче місце, і саме тому, роботодавець має забезпечити максимальний його комфорт і безпеку. Гарантією цих факторів може слугувати сертифікація меблів, що використовуються на підприємстві it-галузі. Тому нами пропонується закупівля тільки меблів , які пошли сертифікацію на відповідність. Під психоемоційними чинниками ми розуміємо гарне самопочуття фахівців, позитивний настій, гарний психологічний клімат у колективі, тощо. Задля того, щоб психоемоційні чинники мали

максимально позитивний ефект, керівництву слід поводити заходи, які сприятимуть укріпленню і покращенню міжособистісних стосунків у колективі, таких як психологічні тренінги, тімбілдінг, спортивні змагання і естафети. Також, сюди можна віднести розробку і впровадження системи мотивації працівників, як фінансової, так моральної і адміністративної.

8.4. Пожежна безпека

Вимоги до пожежної безпеки на підприємстві неухильно повинен дотримуватися кожен співробітник, а організаційна складова при цьому покладається на посадових осіб за відповідним рішенням керівництва і прописується в посадових інструкціях і положеннях по структурним підрозділам.

Зокрема, вказуються конкретні території, ділянки, зони, об'єкти, цілі будівлі і їх частини, поверхи, на яких відповідального співробітника повинне проводити такі організаційні роботи.

Відповідальні особи зобов'язуються розробити, впровадити та підтримувати в певному інструкцією і положенням на ввірених їм об'єктах протипожежний режим і інструкції відповідно до вимог, викладених в нормативних актах.

Передбачено також створення підрозділу добровільної пожежної охорони та пожежно-рятувальної команди в його складі.

Встановлений режим включає порядки з описом місць спеціального призначення та правила їх користування та утримання, наприклад:

– евакуаційних шляхів;

– так званих «курилок»;

– місць складування продукції та сировини;

– стоянки транспорту.

Також встановлюється порядок роботи та технічного обслуговування:

– вентиляційного устаткування;

– засобів пожежогасіння і захисту від загорянь;

– нагрівальних приладів;

– електрообладнання.

Розробляються і впроваджуються правила роботи з відкритим вогнем і горючими матеріалами. Створюються графіки проходження інструктажів з пожежної безпеки співробітників, а також порядок і терміни перевірок знань пожежно-технічного мінімуму, в тому числі, тих працівників, які відповідальні за цю ділянку роботи на підприємстві. При цьому можуть передбачатися внутрішні лекції, семінари, тренінги та практичні заняття на підприємстві, а також зовнішні – на базі спеціалізованих навчальних центрів з професійними викладачами.

Важливою складовою протипожежного режиму на будь-якому об’єкті є розробка і впровадження порядку дій при виникненні пожежі. Неодмінно має бути план евакуації, описано, як повинні відключатися електроустановки, що і в якій послідовності необхідно робити співробітникам.

Відповідно, для кожного об’єкта, кожного приміщення (крім коридорів, санвузлів, басейнів і подібних приміщень), окремих видів робіт складаються інструкції, за якими повинен працювати персонал, залучений на певних ділянках і в виконанні окремих видів робіт. За інструкціями проводиться навчання (інструктаж) персоналу з подальшим контролем знань.

Детально про те, як розробити протипожежний режим, прописати порядки та інструкції, пояснюють на тематичних курсах і семінарах. [4]

8.5. Розрахункова частина

Система освітлення робочого місця користувача ПК має відповідати наступним вимогам (рис. 8.1).

Рисунок 8.1 – Вимоги до системи освітлення робочого місця користувача ПК

Проведемо розрахунок штучного освітлення за методом коефіцієнта використання світлового потоку для приміщення ширина якого складає 3 м, довжина – 4,2 м, висота – 3,2 м.

У зазначеному приміщенні працює 2 людей.

Для того, щоб визначити потрібну кількість світильників, які повинні забезпечити нормований рівень освітленості, визначимо світловий потік, що падає на робочу поверхню за формулою:

$$F=ESKZ/n,$$

де:

F – світловий потік, що розраховується, Лм;

E – нормована мінімальна освітленість, Лк; *E = 300 Лк;*

S – площа освітлюваного приміщення.

K – коефіцієнт запасу, що враховує зменшення світлового потоку лампи в результаті забруднення світильників в процесі експлуатації (його значення залежить від типу приміщення і характеру робіт, що проводяться в ньому, в нашому випадку *K = 1,5);*

Z – відношення середньої освітленості до мінімальної (зазвичай приймається рівним 1.1... 1.2, в нашому випадку *Z = 1,1*);

n – коефіцієнт використання світлового потоку, (відношення світлового потоку, що падає на розрахункову поверхню, до сумарного потоку всіх ламп і обчислюється в долях одиниці; залежить від характеристик світильника, розмірів приміщення, забарвлення стін і стелі, що характеризуються коефіцієнтами відбиття від стін **($\rho_{стін}$.)** і стелі **($\rho_{стелі}$),** значення коефіцієнтів дорівнюють **$\rho_{стін}$ = 50% і $\rho_{стелі}$ = 50%.**

Обчислимо індекс приміщення за формулою:

$$i=S/(h(A+B)),$$

де:

S – площа приміщення, *S = 12,6 м²;*

h – розрахункова висота підвісу, *h* = 3,2 *м. (співпадає з в*исотою стелі, т.я. лампи освітлення закріплюються на стелі)*;*

A – ширина приміщення, *A = 3 м;*

B – довжина приміщення, *B* = 4,2 *м.*

Підставимо всі значення у формулу та визначемо індекс приміщення:

$$i=0,58.$$

Знаючи індекс приміщення, за знаходимо *n = 0,46 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників з відповідним типом лампам) [11].* Підставимо всі значення у формулу, визначемо світловий потік: *F=13558 Лм.*

Для розрахунку дудемо використовувати стельовий с*вітильник LED функціональний круглий зоряне небо 100W 2800-6200K 220V VIDEX VL-CLSRS-*

100, енергоефективність якого дорівнює 70 Lm/W, а світловий потік $F_л$ = *7000 Лм.*

Число ламп визначається по формулі:

$$N=F/F_л$$

де:

F – світловий потік,

$F_л$ – світловий потік однієї лампи.

Підставимо значення у формулу та отримаємо:

$$N= 13558 / 7000=1,9 \text{ шт.}$$

Приймаємо необхідну кількість *світлодіодних світильників* 2 шт.

8.6 Висновки до розділу

Дотримання всіх необхідних умов праці не лише сприяє збереженню здоров'я працівників, а також підвищує ефективність виробництва вцілому.

З цих міркувань було здійснено аналіз умов праці, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок штучного освітлення, як одного з ключових факторів впливу на працездатність та здоров'я програміста. Розроблено заходи з охорони праці.

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи розпізнання образів у структурі технічного захисту інформації банківської установи.

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

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти наведені теоретичне узагальнення й рішення наукового завдання дослідження методів розпізнання образів у структурі технічного захисту інформації банківської установи.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем розпізнання образів у структурі технічного захисту інформації банківської установи.

– Досліджена система розпізнання образів у структурі технічного захисту інформації банківської установи.

– На основі отриманих результатів досліджень створена програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання розпізнання образів у структурі технічного захисту інформації банківської установи.

Проведено аналіз предметної галузі в ході якого були виявлені об'єкти, взаємодія яких носить істотний характер для функціональної діяльності предметної галузі, і їхні основні характеристики; побудована алгоритм і вибраний середовище розробки.

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

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

Програма реалізована на мові високого рівня RAD Studio Delphi 10.4. Дана мова програмування дозволяє найбільш ефективно обробляти дані. 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 RSA.

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.

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 7438 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 1,2 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Ситнік Є.Ю. Дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи // Збірник праць молодих науковців ЦНТУ. – Вип. 13. – Кропивницький: ЦНТУ, 2022.

2. Донченко В.С. Теорія ймовірностей та математична статистика: навчальний посібник / В.С. Донченко, М.В.-С. Сидоров, М.М. Шарапов – К.: ВЦ «Академія», 2009. – 288 с.

3. Дреев А.Н. Использование неравномерного распределения единичных битов для дополнительного сжатия SPIHT кода / А.Н. Дреев, А.А. Смирнов // Информационные системы в управлении, образовании, промышленности: монография. Под редакцией профессора В.С. Пономаренко. – Х.: Вид-во ТОВ «Щедра садиба плюс», 2014. – С. 498.

4. Дрєєв О.М. Дослідження впливу шляху розгортки на ступінь ентропійного стиснення цифрового зображення / О.М. Дрєєв, О.В. Слюсар // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. Випуск 21. – Кіровоград: КНТУ. – 2008 – С. 115-118.

5. Дрєєв О.М. Метод розвантаження телекомунікаційного сервера за рахунок кешування зображень / О.М. Дрєєв // Збірник наукових праць Кіровоградського національного технічного університету. Техніка в сільськогосподарському виробництві, галузеве машинобудування, автоматизація. Випуск 25. Ч. І. – Кіровоград: КНТУ. – 2012 – С. 419-424.

6. Дрєєв О.М. Метод прогнозування завантаженості серверу телекомунікаційної мережі / О.М. Дрєєв, О.А. Смірнов, Є.В. Мелешко, О.В. Коваленко // Системи обробки інформації. Випуск 3(101) Том 2. – Х.: ХУПС. – 2012. – С. 181-188.

7. Дрєєв О.М. Оцінка якості стиснення зображень на основі дискретного перетворення Хартлі / О.В. Коваленко, О.П. Доренський, О.М. Дрєєв // Системи озброєння і військова техніка. Науковий журнал 2(34) – Х.: ХУПС – 2013. С. 99-102.

8. Дрєєв О.М. Дослідження впливу ступеня стиснення зображень на оперативність їх доставки у телекомунікаційній системі / О.А. Смірнов, О.М. Дрєєв, О.П. Доренський // Збірник наукових праць "Системи обробки інформації". – Випуск 8(115). – Х.: ХУПС – 2013. – С. 234-239.

9. Дреев А.Н. Сравнение битовых плотностей при использовании различных методов кодирования информации / А.Н. Дреев, А.А. Смирнов // Системи обробки інформації, 2014, випуск 2 (118), том 2 – Харків: ХУПС – 2014. С 64-66.

10. Дрєєв О.М. Моделювання впливу інтенсивності трафіку на оперативність доставляння інформації / О.М. Дрєєв // Науково-виробничій журнал "Зв'язок". – Київ: ДУТ, 2014. – № 2 (108) С. 24-29.

11. Дреев А.Н. Повышение вероятности доставки сообщений в телекоммуникационных системах и сетях для обеспечения информационной безопасности / А.Н. Дреев, А.А. Смирнов //«Безпека інформації» Том 21, №1 2015 р. – Київ: НАУ – 2015. – С. 22-28.

12. Дрєєв О.М. Узагальнення вейвлету Хаара / О.М. Дрєєв, Г.М. Дрєєва // Збірник тез доповідей Комбінаторні конфігурації та їх застосування, 15-16 жовтня 2010 р. – Кіровоград – С. 58

13. Дрєєв О.М. Узагальнення вейвлету Хаара / О.М. Дрєєв // Матеріали науково-практичної конференції, присвяченої 80-річчю фізико-математичного факультету КДПУ ім. В. Винниченка 26 листопада 2010 р. – Кіровоград – С. 12

14. Дрєєв О.М. Метод прогнозування завантаженості серверу телекомунікаційної мережі / О.М. Дрєєв, О.В. Коваленко // Тези доповідей Новітні технології – для захисту повітряного простору. Дев'ята наукова конференція. 18-19 квітня 2011 р. – Х.: ХУПС. – 2012. – С. 206

15. Дрєєв О.М. Метод довгострокового прогнозування навантаження серверу телекомунікаційної мережі / О.М. Дрєєв, Г.М. Дрєєва // Комбінаторні конфігурації та їх застосування. Кіровоград. 13-14 квітня 2012 р. – Кіровоград: "Екслюзів-систем". – 2012. – С. 50

16. Дрєєв О.М. Вдосконалення стиснення зображень SPIHT методу шляхом додаткового кодування та відкладеної передачі уточнення вейвлет коефіцієнтів / О.М. Дрєєв // Дискретна математика та її застосування у економіко-математичному моделюванні та інформаційних технологіях. 11-13 жовтня 2012 р. – Запоріжжя: ЗНУ – 2012. – С. 22-23.

17. Дрєєв О.М. Методи підвищення якості обслуговування у телекомунікаційних системах та мережах / О.М. Дрєєв, Г.М Дрєєва, О.А. Смірнов // Збірник тез доповідей. Академія внутрішніх військ МВС України "Застосування інформаційних технологій у підготовці та діяльності сил охорони правопорядку" 20-21 березня 2013р. – Харків: АВВ. – 2013. С. – 18-19

18. Дреев А.Н. SPIHT кодирование с отложенной передачей значимых битов / А.Н. Дреев // Тези доповідей. Новітні технології – для захисту повітряного простору. Дев'ята наукова конференція 17 квітня 2013 р. – Х.: ХУПС. – 2013. – С. 206

19. Дреев А.Н. Повышение оперативности доставки данных повышенной востребованности в телекоммуникационных системах и сетях / А.Н. Дреев, А.А. Смирнов, Е.В. Мелешко // Проблеми і перспективи розвитку ІТ-індустрії 25-26 квітня 2013 р. Системи обробки інформації. – Випуск 3 (110). Том 2. – Харків: ХУПС. – 2013. С. – 199.

20. Дрєєв О.М. Середньостатистичний та найімовірніший час доставки багатопакетного повідомлення в телекомунікаційній системі або мережі / О.М. Дрєєв, О.А. Смірнов // V Всеукраїнська науково-практична конференція "Інформатика та системні науки" ІСН – 2014, 13-15 березня 2014 року, м. Полтава – С. 92

21. Дрєєв О.М. Визначення оптимального розміру блоку при бітовому арифметичному кодуванні / О.М. Дрєєв, Г.М. Дрєєва // Збірник тез доповідей Комбінаторні конфігурації та їх застосування, 11-12 квітня 2014 р. – Кіровоград – С. 44

22. Дреев А.Н. Экстраполяция квазипериодических процессов с аддитивными помехами / А.Н. Дреев, А.А. Смирнов // П’ята Міжнародна науково-практична конференція "Інформаційні технології та моделювання в економіці" 15-16 травня 2014 р. – Черкаси – С. 59

23. Дреев А.Н. Статистическая модель передачи многопакетного сообщения в телекоммуникационной системе или сети / А.Н. Дреев, А.А. Смирнов // «Компьютерное моделирование в наукоемких технологиях (КМНТ-2014)» Харьков, 28-31 мая 2014 года – С. 137-140

24. ДСТУ 2481 – 94 Системи оброблення інформації інтелектуальні інформаційні технології. Терміни та визначення. – Х.: ДЕРЖСТАНДАРТ УКРАЇНИ, 1994. – 33 с.

25. ДСТУ В 3265 – 95. Зв'язок військовий. Терміни та визначення. – К.:УкрНДІССІ, 1995. – 23 с.

26. Дымарский Я.С. Управление сетями связи: принципы, протоколы, прикладные задачи / Я.С. Дымарский., Н.П. Крутякова, Г.Г. Яновский – М.: ЭкоТрендз, 2003. – 384 с.

27. Ершов В.А. Мультисервисные телекоммуникационные сети / В.А. Ершов, Н.А. Кузнецов – М.: Изд. МГТУ им. Н.Э. Баумана, 2003. – 432 с.

28. Закон України «Про Концепцію Національної програми інформатизації» (Відомості Верховної Ради (ВВР), 1998, N 27-28, ст.181) (Із змінами, внесеними згідно із Законом N 5463-VI (5463-17) від 16.10.2012, ВВР, 2014, N 4, ст.61). – http://zakon1.rada.gov.ua/laws/show/74/98-%D0%B2%D1%80.

29. Зайченко Ю.П. Компьютерные сети / Ю.П. Зайченко. – К.: Слово, 2003. – 256 с.

30. Зубко Р.А. Алгоритми стиснення зображень в системах цифрової обробки даних / Р.А. Зубко // Восточно-Европейский журнал передовых технологий. – 2013. – №1/2(61). – С. 40-44.

31. Иванов В.Г. Прогрессивные информационные технологии сжатия изображений / Иванов В. Г., Ломоносов Ю. В., Любарский М. Г. // Проблеми інформатики та комп'ютерної техніки (ПІКТ – 2014) : пр. ІІІ-ї Міжнар. наук.-практ. конф., 27–30 трав. 2014 р. – Чернівці : Родовід, 2014. – С. 172–174.

32. Ивахненко А.Г. Долгосрочное прогнозирование и управление сложными системами / А.Г. Ивахненко – Киев: «Техніка». – 1975. – 312 с.

33. Игнатенко Е.Г., Бессараб В.И. Алгоритм адаптивного мониторинга загрузки кластерных web-серверов / Е.Г. Игнатенко, В.И. Бессараб // Збірник тез. Нові технології в телекомунікаціях. VI міжнародний науково-технічний симпозіум. – Карпати, Вишків: ДУІКТ. – 2011. – С.34.

34. Карпенко С.В. Метод компактного представлення зображень в телекомунікаційних системах на основі тривимірного поліадичного кодування: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.12.02 – телекомунікаційні системи та мережі / С.В. Карпенко. – Харків: Харківський національний університет радіоелектроніки, 2009. – 22 с.

35. Касимов Р.Р. Вдосконалення алгоритмів QoS маршрутизації в мережах з технологією IP/MPLS на основі прогнозу трафіка: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.12.02 –телекомунікаційні системи та мережі / Р.Р. Касимов. – Київ: Київський державний університет інформаційно-комунікаційних технологій, 2011. – 24 с.

36. Кириченко Л.О. Влияние методов маршрутизации на QOS в мультисервисных сетях при самоподобной нагрузке / Л.О. Кириченко, Т.А. Радивилова, Э. Кайали // Восточно-Европейский журнал передовых технологий. Информационные технологии. 1/2 (49) 2011. – С. 15-18

37. Квєтний Р.Н. Методи та засоби передавання інформації у проблемно-орієнтованих розподілених комп'ютерних системах: монографія / Р.Н. Квєтний, А.Я. Кулик – Вінниця: ВНТУ, 2010. – 362 с.

38. Коваленко А.А. Методи та засоби підвищення оперативності передачі даних у мультисервісних мережах: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.13.05 «компьютерні системи та компоненти» / А.А. Коваленко. – Харків: Харківський національний університет радіоелектроніки, 2008. – 20 с.

39. Комарова Л.О. Алгоритми управління потоками даних у телекомунікаційній мережі реального часу / Л.О. Комарова // Телекомунікаційні та інформаційні технології. – 2014. – №2 – С. 13-18.

40. Кормен Т. Алгоритмы: построение и анализ / Томас Кормен, Чарльз Лейзерсон, Рональд Ривест, Клиффорд Штайн. – М.:"Вильямс", 2005. – 1296 с.

41. Конахович Г.Ф. Сети передачи пакетных данных / Г.Ф. Конахович, В.М.Чуприн. – К.:МК-Пресс, 2006. – 272 с.

42. Кононенко В.О. Колебательные системы с ограниченным возбуждением / В.О. Кононенко – М.: Наука, 1964. – 232 с.

43. Королев А.В. Адаптивная маршрутизация в корпоративных сетях / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2003. – 224 с.

44. Королев А.В. Управление сетевыми ресурсами / А.В. Королев, Г.А. Кучук, А.А. Пашнев. – Х.: ХВУ, 2004. – 272 с.

45. Красильников Н.Н. Теория передачи и восприятия зображений / Н.Н. Красильников – М.: Радио и Связь, 1986. – 248 с.

46. Кузнецов О.О. Методи обробки сигналів даних та зображень. Навчальний посібник / О.О. Кузнецов, Г.А. Кучук, С.Г. Семенов – Х.: НТУ «ХПІ», 2011. – 301 с.

47. Кучерявый Е.А. Управление трафиком и качество обслуживания в сети Интернет / Е.А. Кучерявый – М.: Наука и Техника, 2004. – 336 с.

48. Кучук Г.А. Метод дослідження фрактального мережевого трафіка / Г.А. Кучук // Системи обробки інформації. – Х.: ХВУ, 2005. – Вип. 5 (45). – С. 74-84.

49. Кучук Г.А. Метод агрегування фрактального трафіка / Г.А. Кучук, А.А. Можаев, О.В. Воробьов // Радіоелектронні та комп'ютерні системи. – 2006. – №6(18). – С. 181-188.

50. Лемешко А.В. Потоковые модели многоадресной и широковещательной маршрутизации в телекоммуникационных сетях [Электронный ресурс] / А.В. Лемешко, К.М. Арус // Проблеми телекомунікацій. – 2013. – No 1 (10), с. 38 – 45. – Режим доступа: http://pt.journal.kh.ua/2013/1/1/131_lemeshko_ multicast.pdf.

51. Лемешко А.В. Особенности математического описания процессов многоадресной маршрутизации потоковыми моделями / А.В. Лемешко, Арус Кинан // Труды Северо-кавказского филиала Московского технического университета связи и информатики, часть 1. – Ростов-на-Дону: СКФ МТУСИ, 2014. – С. 94-98.

52. Лемешко О.В. Результати порівняльного аналізу потокових моделей маршрутизації в телекомунікаційних мережах / О.В. Лемешко, О.А. Дробот, Д.В. Симоненко // Збірник наукових праць Харківського університету Повітряних Сил. Вип. 1(13), 2007. – С. 66-69.

53. Лемешко А.В. Тензорная модель многопутевой маршрутизации с гарантиями качества обслуживания одновременно по множеству разнородных показателей [Электронный ресурс] / А.В. Лемешко, О.Ю. Евсеева // Проблеми телекомунікацій. – 2012. – No 4 (9). – С. 16 – 31. – Режим доступа: http://pt.journal.kh.ua/2012/4/1/124_lemeshko_tensor.pdf.

54. Лемешко А.В. Модель отказоустойчивой маршрутизации многоадресных и широковещательных потоков в MPLS-сети / Лемешко А.В., Арус К.М. // Системи обробки інформації. – 2013. – №9 (116). – С. 160 – 163.

55. Малла С. Вейвлеты в обработке сигналов: пер. с англ./ С. Малла – М.: Мир, 2005. — 671 с.

56. Марчук Г.И. Методы вычислительной математики / Г.И. Марчук – М.: Наука, 1989. – 608 с.

57. Муранов О.С. Експериментальні дослідження механізмів прогнозування пульсацій пакетного трафіку / Муранов О.С. // Защита информации : Сб. научн. трудов Национального авиационного университета. – К. : НАУ, 2008. – Специальный выпуск. – С.137-142.

58. Муранов О.С. Підвищення якості технології адаптивного управління трафіком пакетних телекомунікаційних мереж: автореф. дис. на здобуття наук. ступеня кандидата техн. наук: спец. 05.13.06 – інформаційні технології / О.С. Муранов. – Київ: Національний авіаційний університет, 2010. – 29 с.

59. Державні будівельні норми України: ДБН В.2.5-28:2018. – Режим доступу до ресурсу: https://goo.su/9AkQ

60. Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин: ДСанПІН 3.3.2-007-98. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/v0007282-98

61. Закон України «Про охорону праці» від 14.10.1992 р. № 2694-XII. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/2694-12

62. Зеркалов Д. В. Охорона праці в Галузі: Загальні вимоги: навч. посіб. Київ: Основа. 2011. 551 с.

63. Наказ Міністерства соціальної політики України 14.02.2018 № 207 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями». – Режим доступу до ресурсу: https://zakon.rada.gov.ua/laws/show/z0508

64. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград:

КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358

65. Постанова № 42 від 01.12.1999 Головного державного санітарного лікаря України «Санітарні норми мікроклімату виробничих приміщень ДСН 3.3.6.042-99. – Режим доступу до ресурсу: https://zakon.rada.gov.ua/rada/show/va042282-99

66. Сакулин В.П., Шептовицкий В.М. Безопасность труда при монтаже и эксплуатации электроустановок / В.П.Сакулин, В.М.Шептовицкий. – Л. : “Колос”, 1973. – 238 с.

67. Центр післядипломної освіти та підвищення кваліфікації. – Режим доступу до ресурсу: https://cpo.stu.cn.ua

68. Оришака, О. В. Основи охорони праці: навч. посіб. / О. В. Оришака, Г. П. Горбачова, К. М. Марченко; М-во освіти і науки України, Центральноукраїн. нац. техн. ун-т. – Кропивницький : ЦНТУ, 2022. – 175 с. – Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/12161 (дата звернення 19.09.22).

Додаток А
(обов'язковий)

Технічне завдання

Зміст

					ВКРМ-123.22.0020.00.00.ТЗ		
Вим.	Арк.	№ документа	Підпис	Дата			
Розробив		Ситнік Є.Ю.			Дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи	Літ.	Аркуш 1 / Аркушів 6
Перевірив		Смірнов О.А.					
Н. Контр.		Гермак В.С.				ЦНТУ КІ-21М-1,4	
Затв.		Смірнов О.А.					

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи розпізнання образів у структурі технічного захисту інформації банківської установи.

2 Підстава для розробки

Підставою для розробки служить завдання на випускну кваліфікаційну роботу за другим (магістерським) рівнем вищої освіти, видане на кафедрі кібербезпеки та програмного забезпечення (нак. № 19-13 від 17.08.2022 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи.

4 Джерела розробки

Джерелом цієї випускної кваліфікаційної роботи за другим (магістерським) 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;

– техніко-економічне обґрунтування доцільності прийнятого до розробки програмного забезпечення;

– аналіз умов праці;

– розробка програми, що реалізує спроектовані алгоритми роботи системи.

5.2 Показники призначення

Система повинна забезпечувати:

– програмну реалізацію системи розпізнання образів у структурі технічного захисту інформації банківської установи;

– 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 архітектури IBM PC, працювати в ОС Windows 10/11 і з сумісними з цією платформою пристроями і прикладним програмним забезпеченням.

5.8 Вимоги до інформаційної і програмної сумісності

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

5.8.1 Обладнання

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

5.8.2 Мова програмування

Середовище RAD Studio Delphi 10.4.

5.8.3 Вхідні дані

Опис алгоритму роботи запропонованої системи.

5.8.4 Вихідні дані

Робоча програма.

6 Вимоги до програмної документації

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

7 Економічні вимоги

7.1 Для ПЗ необхідно виробити функціонально-вартісний аналіз варіантів розробки.

7.2 Виконати розрахунок витрат показників економічного ефекту з урахуванням цін на 3 вересня 2022 року.

8 Вимоги щодо охорони праці

В частині охорони праці випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти повинен бути розглянутий аналіз умов праці на робочому місці ІТ-фахівця.

9 Перелік документів, що розробляються

– Наукова новизна – 1 аркуш.
– Структурна схема системи – 1 аркуш.
– Функціональна схема системи – 1 аркуш.
– Діаграма процесів – 1 аркуш.
– Блок-схема алгоритму роботи програми – 2 аркуша.
– Показники економічної ефективності – 1 аркуш.
– Пояснювальна записка – 119 аркушів.

10 Етапи розробки

10.1 Збір і обробка інформації по темі випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти. Постановка задачі на виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти (складання ТЗ).

10.2 Проведення досліджень або експериментальних робіт для уточнення основних положень випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти.

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2022 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 22.12.2022 р.

Додаток Б

(обов’язковий)

Міністерство освіти і науки України

Центральноукраїнський національний технічний університет

ЗАТВЕРДЖУЮ

Керівник випускної кваліфікаційної роботи за

другим (магістерським) рівнем вищої освіти

_____ Смірнов О.А.

Дослідження та програмна реалізація системи розпізнання образів у структурі технічного захисту інформації банківської установи

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2022 року

Файл UMain.pas - основна програма

```
unit UMain;

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

interface

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

const port = 'COM2';

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

```
    Edit1: TEdit;
    Edit2: TEdit;
    Label4: TLabel;
    Button7: TButton;
    NeuroCount: TEdit;
    Button8: TButton;
    procedure FormCreate(Sender: TObject);
    procedure FormShow(Sender: TObject);
    procedure FormClose(Sender: TObject; var Action: TCloseAction);
    procedure MovingFormButtonClick(Sender: TObject);
    procedure Button3Click(Sender: TObject);
    procedure MoveForLineClick(Sender: TObject);
    procedure RefreshCamsListButtonClick(Sender: TObject);
    procedure CamConfigButtonClick(Sender: TObject);
    procedure CamsListBoxClick(Sender: TObject);
    procedure DialogListBoxDblClick(Sender: TObject);
    procedure VideoModeComboBoxChange(Sender: TObject);
    procedure CommunicationButtonClick(Sender: TObject);
    procedure RisovalkaButtonClick(Sender: TObject);
    procedure AnalyzFrameButtonClick(Sender: TObject);
    procedure RazvertkaButtonClick(Sender: TObject);
    procedure OnFlyDebugButtonClick(Sender: TObject);
    procedure Button1Click(Sender: TObject);
    procedure Button2Click(Sender: TObject);
    procedure AddSampleButtonClick(Sender: TObject);
    procedure RestaranButtonClick(Sender: TObject);
    procedure Button4Click(Sender: TObject);
    procedure Button6Click(Sender: TObject);
    procedure Button5Click(Sender: TObject);
    procedure Timer1Timer(Sender: TObject);
    procedure Button7Click(Sender: TObject);
    procedure Button8Click(Sender: TObject);

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



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

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


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

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

```
var
  MainForm: TMainForm; //головна форма

  //масив настроювань камер
  GraphConfigExArr: TGraphConfigExArr;

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

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

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

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

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

  {Інтерфейс для перебору движків}
  aTTSEnum: ITTSEnum;

  {Покажчик на параметри движка}
  fpModeInfo: PTTSModeInfo;

  NumFound : DWord;
  ModeInfo : TTSModeInfo;

  QPixels: TQuickPixels;
  timercnt: Cardinal = 0;


   i: integer;
   dest1: mass;
```

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

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

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

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

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

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


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

function GetVideoDevicesListEm(): TStringList;

implementation

uses Math;

{$R *.dfm}

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


{Допоміжні процедури, що не входять у клас форми}

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

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

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

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

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

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

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

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

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

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

begin
  Result := true;

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

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

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

    //присвоєння файлу
    AssignFile(GraphConfigFile,GraphConfigFileName);

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

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

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

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

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

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

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

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

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

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

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

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

 with GraphConfigEx.GraphConfig do
  begin
```

```
    //ім'я камери
    VCapSource := CamName;

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

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

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

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

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

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

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

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

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

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

{Оброблювачі подій}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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


  //зупинка каналу
  Communication.Destroy;
```

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

```
      //запускаємо або перезапускаємо або зупиняємо таймери якщо потрібно
      if GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.WantBitmaps then
      begin
        if CamTimerArr[CamsListBox.ItemIndex].Active then
          CamTimerArr[CamsListBox.ItemIndex].StopTimer;

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

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

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

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

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

  //ім'я камери
```

```
  PreviewWindow.Caption :=
GraphConfigExArr[CamsListBox.ItemIndex].GraphConfig.VCapSource;

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

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

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

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

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

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

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

//Подія - натискання на "Рисовалка"
procedure TMainForm.RisovalkaButtonClick(Sender: TObject);
```

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

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

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

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

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


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

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

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

```
  begin
    BitmapGrabEventArr[CamsListBox.ItemIndex].NeedAddSample := true;
  end
  else
    UAddSample.AddSampleForm.ShowModal;
end;

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

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

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



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


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

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


   mbReset(1);
   sleep(10);

   mbExecuteProgramFile(1,'image.raw');

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

   Communication.PortEn:=true;
  end;



   MainForm.Timer1.Enabled:=true;

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

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

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

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




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

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

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

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

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

    addsampleform.NeuralNetBP1.Compute(xInputVector);


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

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

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

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

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

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

end.
```

Файл UPreview.pas – модуль роботи з камерами

```
unit UPreview;

interface

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

type
  //захоплений кадр
  TCapturedBitmap = Vcap.TCapturedBitmap;

  TBitmap = Graphics.TBitmap;

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

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


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

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

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

  implementation

uses
  UMain,UCamTimers,Urestaran;

var
  F: TextFile;
//  BitmapArr: TByteArr;

{$R *.dfm}

//конструктор
```

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

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

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

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

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

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

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

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

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

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


    if recognition_run then
    begin
      QP.Attach(Bitmap);

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

      //TSampleRule(BitmapArr);

        RobotRecognition.BitmapReady:=true;
    end;
  end;

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

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

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

```
    RazvertkaTlumitsya.Destroy;
  end;

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

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

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

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

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

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

finalization
  QP.Destroy;
  CloseFile(F);

end.
```

Файл UCamTimers.pas – модуль встановлення таймерів камер

```
unit UCamTimers;

interface

uses
  Windows,URazvertka, UPreview, UTypeConst, SysUtils;

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

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

  //масив таймерів камер
  TCamTimerArr = array of TCamTimer;

var
  CamTimerArr: TCamTimerArr;//масив таймерів камер

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

implementation

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

end.
```

Файл UGraphConfig.pas – модуль налаштування камер

```
unit UGraphConfig;

{Модуль настроювання камери}

interface

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


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


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

var
  GraphConfigForm: TGraphConfigForm; //форма

  //індекс елемента масиву настроювань
  GraphConfigExIndex: Cardinal;

implementation

uses
  UMain;

{$R *.dfm}

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

{при натисканні кнопки 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 UCharAnalyz.pas - модуль розпізнавання образів з телекамери за допомогою нейроної мережі

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


interface

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

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

type

  TByteArr = array of array of integer;

  TRobotRecognition = class(TThread) //Потік розпізнання
    private


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


  TBitmap = Graphics.TBitmap;


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


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

  end;


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

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

procedure segment(BitmapArr: TByteArr);

procedure reader(letters_count:integer);

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

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

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

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

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

implementation

uses
  UMain,UAddSample,upreview;



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

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

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

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

      end;
      if rectype =0 then
      begin

        TSampleRule(BitmapArr);

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

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

begin

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

  Canvas:=AddSampleForm.SampleImage.Canvas;


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

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

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

      BitmapArr[y,x] := PixelRes;

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

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

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

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

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

      OldPix:=PixelRes;


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

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

  end;


end;


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

begin

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

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


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

    end;

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

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

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

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

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



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

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

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

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

end;



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

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


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

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

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

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

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

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

begin


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


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

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

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

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

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

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

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

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

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

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


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

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

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

                    end;

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

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

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

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

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

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

      end;


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

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

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

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

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

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

                  end;

            end;
      end;

 }


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

      end;

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

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

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

            end;
          end;

      end;
    end;

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

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

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

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

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

      if debug then
         begin

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

```
     end;


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

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


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

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

      for y := 0 to height-2 do
        begin

        for x := 0 to Width - 2 do
          begin

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

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

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


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

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

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

```
     end;



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

end;


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

 }

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

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


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

end;


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

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

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


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

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


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


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

  //Порівнюємо сіткою
  NeuroCompute(Icon);


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

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

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

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

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

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

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

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

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


    end;

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

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


//  cnt:=0;

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

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

      cnt:=cnt+1;

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


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

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


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


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

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

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

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

```
    end;

end;

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

finalization
  CloseFile(F);


end.
```

Файл UAddSample.pas – обробка розпізнаного образу

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


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

interface

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

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

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

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

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

  implementation

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


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

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



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

const MinBPixelsPercent =10;

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



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

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

```
        end;
    end;


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


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

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

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

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


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

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

```
      end;


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

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


    end;

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

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


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

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

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


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

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

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



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

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

    end;

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

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

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

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

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


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


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

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


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

    if debug then
      begin
```

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

      end;


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

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

      // Алгоритм зменшення зображення  - всі параметри плаваючі

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

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


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


    for iy := 0 to IconSize - 1 do
      begin

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

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


        end;
      end;

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

```
    if debug then
    begin
    for y := 0 to IconSize - 1 do
    begin
      for x := 0 to IconSize - 1 do
         write(F,BitmapArr3[y,x]);
      writeln(F,' ');
    end;
        writeln(F,'--------------');
    end;

end;

// додавання нового семпла - перетворення його в іконку, додавання в масив
procedure TAddSampleForm.AddSampleButtonClick(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  Icon: TByteArr;
  y: Cardinal;
  IconType: Cardinal;

begin
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);

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

    IconType:=IconTypeSet.ItemIndex;
{  if RadioTSample.Checked then
    IconType := Icon
  else
    IconType := IconNot;
 }
  //IconTypes:=IconTypes+1;
  ConvertBitmapToMonoChrome(QP,BitmapArr);
  imFeatures(BitmapArr,1);
  AddSample(BitmapArr,IconType); // реально використовується BitmapArr3

  //Close;
end;

//подія - натискання "Зберегти"
procedure TAddSampleForm.SaveSampleButtonClick(Sender: TObject);
begin
  if SampleSaveDialog.Execute then
  begin
    SampleImage.Picture.Bitmap.SaveToFile(SampleSaveDialog.FileName);
  end;
end;
// читання іконки з файлу
function ReadIcon:TByteArr;
var
  x,y: Integer;
  Arr1:TByteArr;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  buff:string;
  buff2:char;
  cnt:integer;

begin
  SamePixels := 0;
  Width := IconSize;
  Height := IconSize;
```

```
  SetLength(Arr1,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Arr1[y],IconSize);

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

  cnt:=0;
  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        read(sample_db,buff2);
        Arr1[x,y]:=strtoint(buff2);
            if Arr1[x,y] = 1 then
            begin

            xVector[cnt] := 2;
            xInputVector[cnt] := 0;
            end
            else
            begin
            xVector[cnt] := -1;
            xInputVector[cnt] := 1;
            end;
          cnt:=cnt+1;
        //write(F,Arr1[x,y]);
      end;
      //WRITELN(f);
      readln(sample_db,buff);
    end;

  //NeuralNetHopf1.AddPattern(xVector);
  result:=Arr1;

end;


// подія відкриття форми, завантаження семплів з файла, ініціалізація й т.п.
procedure TAddSampleForm.FormShow(Sender: TObject);

  var i1,i,j,jj:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff:string;
    buff_file:textfile;
    max_icon_type:integer;

begin
  AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);


  matrix_size.text:=inttostr(IconSize);
  matrix_size.text:=inttostr(IconSize);



  if FileExists(sample_db_file_name) then
    begin
      assignfile(sample_db,sample_db_file_name);
      assignfile(n_debug,neuro_debug_file_name);
      rewrite(n_debug);
      reset(sample_db);

      //IconTypes:=2;
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
```

```
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconTypes:=strtoint(buff);//eIconTypes.Text:=buff;
      readln(sample_db,buff);
SetLength(IconArr,strtoint(buff)+1);NumIcons.Text:=buff;
      // вхідний вектор Нейрона мережа Хопфілда - нейронів стільки ж , скільки й
точок в іконці
      SetLength(xVector, IconSize * IconSize);
      // вихідний вектор багатошарової - дорівнює числу образів
      //IconTypes:=3;
      SetLength(xOutputVector, IconTypes);

      // вхідний вектор багатошарової - дорівнює числу нейронів
      SetLength(xInputVector, IconSize * IconSize);


      //  SetLength(xInputVector, 5);
      //  SetLength(xOutputVector, 1);

      // настроювання нейроної мережі Хопфілда

      AddSampleForm.NeuralNetHopf1.LayerCount:=1;
      AddSampleForm.NeuralNetHopf1.InputNeuronCount:=IconSize * IconSize;

      //NeuralNetExtended1.InputFieldCount:=IconSize * IconSize;
      {
      NeuralNetExtended1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNetExtended1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNetExtended1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      }



      // настроювання багатошарової
      //NeuralNetBP1.LayerCount:=2;
      //NeuralNetBP1.LayersBP[0].NeuronCount:=IconSize * IconSize;
      //NeuralNetBP1.LayersBP[1].NeuronCount:=IconSize * IconSize;
      //NeuralNetBP1.LayersBP[1].NeuronCount:=IconTypes;
      //NeuralNetBP1.LayersBP[2].NeuronCount:=IconTypes;

      NeuralNetBP1.ResetPatterns;




      for i:=0 to High(IconArr) do
        begin
        readln(sample_db,buff); IconArr[i].IconType:=strtoint(buff)+1;


        for j:=1 to IconTypes do
          begin
          if j<>IconArr[i].IconType then xOutputVector[ j-1]:=0 else
xOutputVector[ j-1]:=1;
          end;



        IconArr[i].Icon:=ReadIcon;
        // додаємо вектор навчання нейроної мережі Хопфілда
        AddSampleForm.NeuralNetHopf1.AddPattern(xVector);

        // Додаємо вектор багатошарової мережі
        //SetLength(xInputVector, 100);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        for i1:=0 to length(xInputVector) do
        write(n_debug,inttostr(round(xInputVector[i1])));
        writeln(n_debug,'');
        // NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
```

```
        //SetLength(xInputVector, 256);
        readln(sample_db,buff);
        end;
        writeln(n_debug,'-----');

        {
        readln(sample_db,buff); IconArr[0].IconType:=strtoint(buff);
        xOutputVector[0]:=0;  xOutputVector[1]:=1;
        IconArr[0].Icon:=ReadIcon; // SetLength(xInputVector, 256);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);

        //SetLength(xInputVector, 256);
        readln(sample_db,buff); IconArr[1].IconType:=strtoint(buff);
        xOutputVector[0]:=1;  xOutputVector[1]:=0;
        IconArr[1].Icon:=ReadIcon; //SetLength(xInputVector, 256);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);
        }
        // Ініціалізувати ваги Нейроної мережі Хопфілда
        //NeuralNetHopf1.InitWeights;



      closefile(n_debug);
      closefile(sample_db);
    end;
  //NeuralNetBP1.ResetPatterns;

end;


// запис іконки у файл
procedure WriteIcon(Arr1: TByteArr);
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);

  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        write(sample_db,Arr1[x,y]);
      end;
    writeln(sample_db,'');
    end;
end;

// закриття форми
procedure TAddSampleForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
//  recognition_run:=not(recognition_run);
  if recognition_run then
    begin
    RobotRecognition.Terminate;
```

```
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    recognition_run:=not(recognition_run);
    end;

  //IconTypes:=strtoint(eIconTypes.text);

  {
  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;

  }
end;

// відновлення картинки для збереження семпла
procedure TAddSampleForm.Button2Click(Sender: TObject);
var canvas:tcanvas;
begin
if debug then
begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;Canvas.FillRect(Canvas.ClipRect);
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=0;
      AddSampleForm.ObjectImage.Width :=0;

end;
upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
end;

procedure TAddSampleForm.NeuralNetBP1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNetBP1.TeachError);
  Application.ProcessMessages;

end;


// навчання нейромережі із зазначеними параметрами
procedure TAddSampleForm.Button3Click(Sender: TObject);
begin


      NeuralNetBP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNetBP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNetBP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      NeuralNetBP1.Init;
 // Учимо багатошарову
 NeuralNetBP1.EpochCount := speEpochCount.Value;
 prbEpoch.Max := NeuralNetBP1.EpochCount;
 prbEpoch.Position := 0;

 // Запуск процесу навчання (offline)
 //NeuralNetExtended1.TeachOffLine;
 NeuralNetBP1.TeachOffLine;
```

```
end;


// перебір параметрів навчання, навчання, записати помилки в лог
// для дослідження  - яка нейромережа краще, і як не потрапити в локальний
// мінімум при навчанні.
procedure TAddSampleForm.Button4Click(Sender: TObject);
  var log:textfile;
  var i,j:integer;
  var cnt:integer;
begin


  NeuralNetBP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
  NeuralNetBP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
  NeuralNetBP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
  NeuralNetBP1.EpochCount := 5000;


  prbEpoch.Max := NeuralNetBP1.EpochCount;
  prbEpoch.Position := 0;
      assignfile(log,neuro_teach_log_file_name);
      append(log);
      writeln(log,'---------------------------');
      closefile(log);
  cnt:=400;
  for i:=1 to 20 do
    begin
    NeuralNetBP1.Alpha:=NeuralNetBP1.Alpha-0.01;
    for j:=1 to 20 do
      begin
      prbEpoch.Position := 0;
      NeuralNetBP1.TeachRate:=NeuralNetBP1.TeachRate-0.005;
      NeuralNetBP1.TeachOffLine;
      assignfile(log,neuro_teach_log_file_name);
      append(log);

      writeln(log,'помилка='+floattostr(NeuralNetBP1.TeachError)+'
альфа='+floattostr(NeuralNetBP1.Alpha)+'
шаг навчання =',floattostr(NeuralNetBP1.TeachRate));
      closefile(log);
      cnt:= cnt-1;
      sttError.Caption := inttostr(cnt);
      //prbEpoch.Position := prbEpoch.Position + 1;
      Application.ProcessMessages;
      end;
    end;

end;

// подія кінця епохи навчання , зрушення процес бара, відображення помилки
procedure TAddSampleForm.NeuralNetExtended1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNetExtended1.TeachError);
  Application.ProcessMessages;

end;


// дозвіл розпізнавання
procedure TAddSampleForm.Button5Click(Sender: TObject);
begin
  debug:=false;
  recognition_run:=not(recognition_run);


  if recognition_run then
    begin
```

```
    RobotRecognition := TRobotRecognition.Create(false);
    RobotRecognition.RecType:=AddSampleForm.recType.ItemIndex;
    AddSampleForm.Button5.Caption:='СТОП'
    end
    else
    begin
    //RobotRecognition.
    RobotRecognition.Terminate;
    //RobotRecognition.Destroy;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    end;

end;



// збереження образів у файл
procedure TAddSampleForm.Button6Click(Sender: TObject);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
  //IconTypes:=strtoint(eIconTypes.text);


  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;


end;


// збереження ваг нейромережі у файл
procedure TAddSampleForm.Button7Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
begin
  AssignFile(neuro,neuro_weight_file_name);
  rewrite(neuro);
  writeln(neuro,floattostr(NeuralNetBP1.TeachRate));
  writeln(neuro,floattostr(NeuralNetBP1.Momentum));
  writeln(neuro,floattostr(NeuralNetBP1.Alpha));

  for i:=0 to NeuralNetBP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNetBP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to  length(NeuralNetBP1.Layers[i].Neurons[j].FWeights)-1 do
          begin

writeln(neuro,floattostr(NeuralNetBP1.Layers[i].Neurons[j].Weights[w]));
          end;
```

```
      end;
    end;
  closefile(neuro);
end;



// завантаження ваг нейромережі з файла
procedure TAddSampleForm.Button8Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
  buff:string;
begin
  AssignFile(neuro,neuro_weight_file_name);
  reset(neuro);


  readln(neuro,buff);NeuralNetBP1.TeachRate:=StrToFloat(buff);
  readln(neuro,buff);NeuralNetBP1.Momentum:=StrToFloat(buff);
  readln(neuro,buff);NeuralNetBP1.Alpha:=StrToFloat(buff);
  NeuralNetBP1.Init;
  for i:=0 to NeuralNetBP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNetBP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to  length(NeuralNetBP1.Layers[i].Neurons[j].FWeights)-1 do
          begin
            readln(neuro,buff);
            NeuralNetBP1.Layers[i].Neurons[j].Weights[w]:=strtofloat(buff);
          end;
      end;
    end;
  closefile(neuro);

end;

procedure TAddSampleForm.Button9Click(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  y: Cardinal;

begin
  debug:=true;
 // upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);


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

  ConvertBitmapToMonoChrome(QP,BitmapArr);

  //ConvertBitmapToMonoChrome(QP2,BitmapArr);
  if length(BitmapArr)>0 then
  segment(BitmapArr);
  debug:=false;
end;

procedure TAddSampleForm.TrackBar1Change(Sender: TObject);
begin
AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);
end;
```

```
procedure TAddSampleForm.Timer1Timer(Sender: TObject);
begin
AddSampleForm.lFPS.Caption:=floattostr(round(tm_tick/5*10)/10);
tm_tick:=0;
end;

end.
```

Файл About.pas - довідка

```
unit about;

interface

uses
  Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls, Forms,
  Dialogs, StdCtrls, jpeg, ExtCtrls;

type
  TAboutForm = class(TForm)
    Memo1: TMemo;
    Button1: TButton;
    Image1: TImage;
    procedure FormCreate(Sender: TObject);
    procedure Button1Click(Sender: TObject);
  private
    { Private declarations }
  public
    { Public declarations }
  end;

var
  AboutForm: TAboutForm;

implementation

{$R *.dfm}

procedure TAboutForm.FormCreate(Sender: TObject);
begin
Memo1.Clear;
Memo1.Lines.Add('МАГІСТЕРСЬКА РОБОТА');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add('Дослідження та програмна реалізація системи розпізнання образів
у структурі технічного захисту інформації банківської установи');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Смірнов О.А.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Ситнік Євген Юрійович ');
Memo1.Lines.Add('                    гр. КІ-21М-1,4');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2022');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;

procedure TAboutForm.Button1Click(Sender: TObject);
begin
AboutForm.Close;
end;

end.
```