Центральноукраїнський національний технічний університет
Центр заочної та дистанційної освіти
Кафедра кібербезпеки та програмного забезпечення

”Допущено до захисту”
Завідувач кафедри кібербезпеки
та програмного забезпечення
д.т.н., професор
_____ Олексій СМІРНОВ
“____” _____ 2023 р.

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі”

Виконав здобувач вищої освіти
ІІ курсу, групи КН-22МЗ
ОПП «Комп’ютерні науки»
спеціальності 122 «Комп’ютерні науки»
_____ Скирда О.В.
«____» _____ 2023 р.

Керівник проекту
кандидат технічних наук
_____ Буравченко К.О.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Центр *Заочної та дистанційної освіти*
Кафедра *Кібербезпеки та програмного забезпечення*
Рівень вищої освіти *магістр*
Галузь знань _12 _____*"Інформаційні технології"*_____
Спеціальність 122 *"Комп'ютерні науки"*
Освітньо-професійна (освітньо-наукова) програма _*"Комп'ютерні науки"*_

ЗАТВЕРДЖУЮ
Завідувач кафедри
д.т.н., проф.
_____ Олексій СМІРНОВ
« 6 » вересня 2023 року

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

Скирді Олександру Віталійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі*

2. Керівник роботи *Буравченко Костянтин Олегович, канд. техн. наук*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 37-13 від 04.08.2023 року

3. Строк подання студентом роботи до захисту *10.12.2023 р.*

4. Мета та завдання випускної кваліфікаційної роботи*: Метою розробки є дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі*

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.	05.10.2023	14.11.2023
Охорона праці	Оришака О.В.	06.10.2023	16.11.2023

7. Дата видачі завдання « 6 » вересня 2023 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	10.10.2023 р.	
2.	Постановка задачі, оформлення ТЗ	15.10.2023 р.	
3.	Розробка моделі компонента	20.10.2023 р.	
4.	Розробка структур даних	25.10.2023 р.	
5.	Розробка алгоритмів зв’язку та відображення	30.10.2023 р.	
6.	Програмування алгоритмів	10.11.2023 р.	
7.	Розрахунок економічної ефективності	13.11.2023 р.	
8.	Розрахунки з охорони праці та техніки безпеки	15.11.2023 р.	
9.	Оформлення ПЗ	17.11.2023 р.	
10.	Попередній захист роботи	10.12.2023 р.	

Дата видачі завдання
« 6 » вересня 2023 р.

Підпис керівника

_____ _____
(прізвище та ініціали)

Завдання прийнято до виконання
« 6 » вересня 2023 р.

Підпис здобувача

_____ _____
(прізвище та ініціали)

АНОТАЦІЯ

Скирда О.В. Дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі. 122 Комп'ютерні науки. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виділення й розпізнавання обличчя користувача у мережі.

Метою розробки є дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

Об'єктом дослідження є процес виділення й розпізнавання обличчя користувача у мережі.

Предметом дослідження є методи виділення й розпізнавання обличчя користувача у мережі.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

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.

Ключові слова: комп'ютерні науки, розпізнавання обличчя

ABSTRACT

Skyrda O.V. Research and software implementation of a system for identifying and recognizing a user's face in the network. 122 Computer Science. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software is developed, which is intended for the system of selection and recognition of the user's face in the network.

The purpose of the development is the research and software implementation of a system for identifying and recognizing the user's face in the network.

The object of research is the process of selecting and recognizing a user's face in the network.

The subject of research is the methods of identifying and recognizing the user's face in the network.

Research methods are based on artificial intelligence methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the system for identifying and recognizing the user's face in the network.

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 environment.

Keywords: computer science, facial recognition

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ВО	–	виявлення облич
ЗНМ	–	згорткова нейронна мережа
ККНК	–	комбінований каскад нейромережних класифікаторів
КСК	–	каскад слабких класифікаторів
РО	–	розпізнавання облич

ВСТУП

Актуальність теми. Програмне забезпечення для розпізнавання обличчя (FRS) визначається як біометричний інструмент, який використовується для зіставлення облич на зображеннях, як правило, з фотографій і відеозаписів, із існуючою базою даних ідентичності. Її можна розбити на три частини – виявлення (пошук обличчя на зображенні), аналіз (карта обличчя) і розпізнавання (підтвердження особи).

Прикладом технології розпізнавання обличчя є функція автоматичного позначення фотографій у Facebook або навіть Google Photos. Соціальні медіа та технічні гіганти, такі як ці, відображають обличчя користувача на фотографії, сортуючи існуючу базу даних завантажених зображень. Оскільки риси обличчя набагато складніші, ніж інші існуючі біометричні методи, такі як відбитки пальців і райдужна оболонка ока, інструменти FRS потребують складних алгоритмів зі штучним інтелектом.

Згідно зі звітом за 2023 рік Opens a new windowЗгідно з NIST, алгоритми розпізнавання облич тепер мають середній рівень помилок лише 0,08%, порівняно з 4,1% у 2014 році. Нейронні мережі та технології глибокого навчання відтоді значно вдосконалилися, дозволивши значний розвиток програмного забезпечення для розпізнавання 3D. Справа не лише в базових алгоритмах, ми тепер маємо потужніші мікроконтролери та процесори та вдосконалену технологію камер для об’єктивів і обробки на чіпі. Доступ до цього апаратного забезпечення у вигляді смартфонів став благом для галузі FRS.

На початку цього року Juniper Research повідомила Відкриває нове вікнощо апаратне забезпечення для розпізнавання обличчя, таке як FaceID від Apple, є найшвидше зростаючою формою біометричного обладнання для смартфонів. За оцінками, до 2024 року їх використовуватиме понад 800 мільйонів смартфонів. Беручи до уваги прогрес технологій і прискорене зростання ринку, це був би вдалий час для впровадження технології розпізнавання облич у ваш бізнес.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

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

– Огляд існуючих систем виділення й розпізнавання обличчя користувача у мережі.

– Дослідження системи виділення й розпізнавання обличчя користувача у мережі.

– Програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

Об'єктом дослідження є процес виділення й розпізнавання обличчя користувача у мережі.

Предметом дослідження є методи виділення й розпізнавання обличчя користувача у мережі.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод виділення й розпізнавання обличчя користувача у мережі.

– Розроблено вітчизняний продукт виділення й розпізнавання обличчя користувача у мережі, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі виділення й розпізнавання обличчя користувача у мережі.

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 LVII Науково-технічній конференції здобувачів вищої освіти «Наука – виробництву», 2023, основні положення випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти надруковані у статті збірника праць молодих науковців ЦНТУ, випуск №14.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі, 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

Призначенням системи є розробка програмного забезпечення виділення й розпізнавання обличчя користувача у мережі**.**

Технологія розпізнавання облич допомагає порівнювати людські обличчя на зображеннях і відео в масштабі.

Попередньо навчені моделі розпізнавання обличчя та демографічні дані Визначайте присутність облич на зображеннях і відео. Використовуйте риси обличчя, як-от відстань між очима, форму носа або контур скул, щоб визначити, чи належать кілька облич одній людині.

Технологія розпізнавання облич використовується фінансовими, державними, виробничими та роздрібними організаціями для забезпечення додаткового рівня безпеки для процесів і фізичних приміщень, прискорення входу в будівлі, цифрового входу людей, зменшення витрат, пов’язаних із крадіжками, запобігання крадіжці особистих даних і забезпечення безпеки простору, залишаючись непомітним.

Перевірка посвідчення особи

Значно прискорити безпеку та час обробки будь-якого посвідчення особи з фотографією, одночасно захищаючи конфіденційну інформацію клієнтів.

Знай свого клієнта

Використовуйте розпізнавання облич, щоб ідентифікувати VIP-персон і потенційних клієнтів, щоб забезпечити персоналізований досвід клієнтів.

Експлуатація ЗМІ

Відстежуйте та виявляйте бажані цілі у відеоканалах і зображеннях у 100 разів швидше, значно прискорюючи розслідування.

Управління доступом

Забезпечте безпеку доступу до шкіл, аеропортів та офісів і забезпечте безпеку людей. Надсилайте сповіщення, коли у фізичному просторі виникають проблеми.

Переваги**:**

– Зберігайте основну правду для особи, яку ви хочете впізнати, і порівнюйте нові обличчя з ними. Зіставляйте обличчя на фотографіях і відео, щоб пришвидшити перевірку особи.

– Позначайте, навчайте та розгортайте свої моделі на одній платформі, яка масштабується відповідно до ваших потреб. Розгорніть свою платформу в хмарі, на місці, на межі або на голому металевому обладнанні.

Реалізовано та експериментально досліджено наступні складові узагальненої структури системи комп'ютерного розпізнавання:

– виявлення облич;

– визначення ракурсу облич;

– ідентифікація облич;

– класифікація облич.

1.2 Область застосування

Основними областями застосування, в яких використовується розпізнавання облич є:

– Охоронні системи.

– Криміналістика.

– Комп'ютерна графіка.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі, 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

SecurOS Face

SecurOS Face – програмно-апаратний комплекс, що забезпечує автоматичне виділення з "живого" відеопотоку оптимального зображення особи для розпізнавання, збереження в базі даних і наступної ідентифікації в режимі реального часу.

Інноваційна технологія розпізнавання осіб забезпечує високу (не менш 80%) імовірність розпізнавання осіб, у тому числі, при зміні фізичних характеристик особи: старінні, появі бороди й вусів, зміні зачіски.

Відсутність фізичного контакту із системою, розпізнавання осіб всіх людей, що потрапили в поле зору відеокамери, робота із зовнішніми базами даних і інші переваги – аргумент для використання системи в місцях масового скупчення людей, на секретних і стратегічно важливих об'єктах.

Інтеграція системи "SecurOS Face" у єдиний комплекс безпеки для забезпечення необхідної реакції його компонент за результатами розпізнавання й ідентифікації – високоефективний спосіб забезпечення безпеки всіляких об'єктів.

Використання системи "SecurOS Face" – високотехнологічне рішення завдань реєстрації й ідентифікації людей. Функціональні можливості й характеристики дозволяють успішно використовувати систему на всіляких об'єктах, для забезпечення безпеки яких необхідна обов'язкова реєстрація й надання даних про людей, що перебувають на об'єкті, а також їхня ідентифікація:

– автоматична реєстрація осіб всіх людей, що потрапили в поле зору відеокамери;

– розпізнавання осіб у режимі реального часу.

Переваги:

– Відсутність фізичного контакту користувача із системою.

– Розпізнавання осіб людей у русі, без необхідності зупинки на рубежі контролю.

– Одночасне розпізнавання осіб всіх людей, що потрапили в поле зору відеокамери.

– Розпізнавання при зміні фізичних характеристик особи: старіння, поява й зникнення бороди й вусів, зміна кольору шкіри.

Адаптація до різних умов освітленості:

– візуалізація зображення на екрані монітора: відображення живого відео й останніх декількох десятків розпізнаних осіб;

– формування зручного для навігації відеоархіву й бази даних зі збереженням дати, часу, напрямку проходу.

Можливий обсяг відеоархіву: для реєстрації осіб обмежений обсягом жорсткого диска комп'ютера; для розпізнавання осіб – до 500'000 осіб:

– забезпечення пошуку інформації із заданих параметрів у базі даних і відеоархіві;

– ідентифікація – порівняння осіб людей із зображеннями із власної або зовнішньої бази даних.

Переваги:

– Висока точність розпізнавання й ідентифікації в результаті використання для порівняння двох і більше ракурсів особи людини.

– Висока швидкість ідентифікації: процес обробки зображення особи займає 5-10 секунд.

– Використання для ідентифікації зображень різних видів: фотографій, відеозаписів.

– Зіставлення осіб людей по "білому" і "чорному" списках.

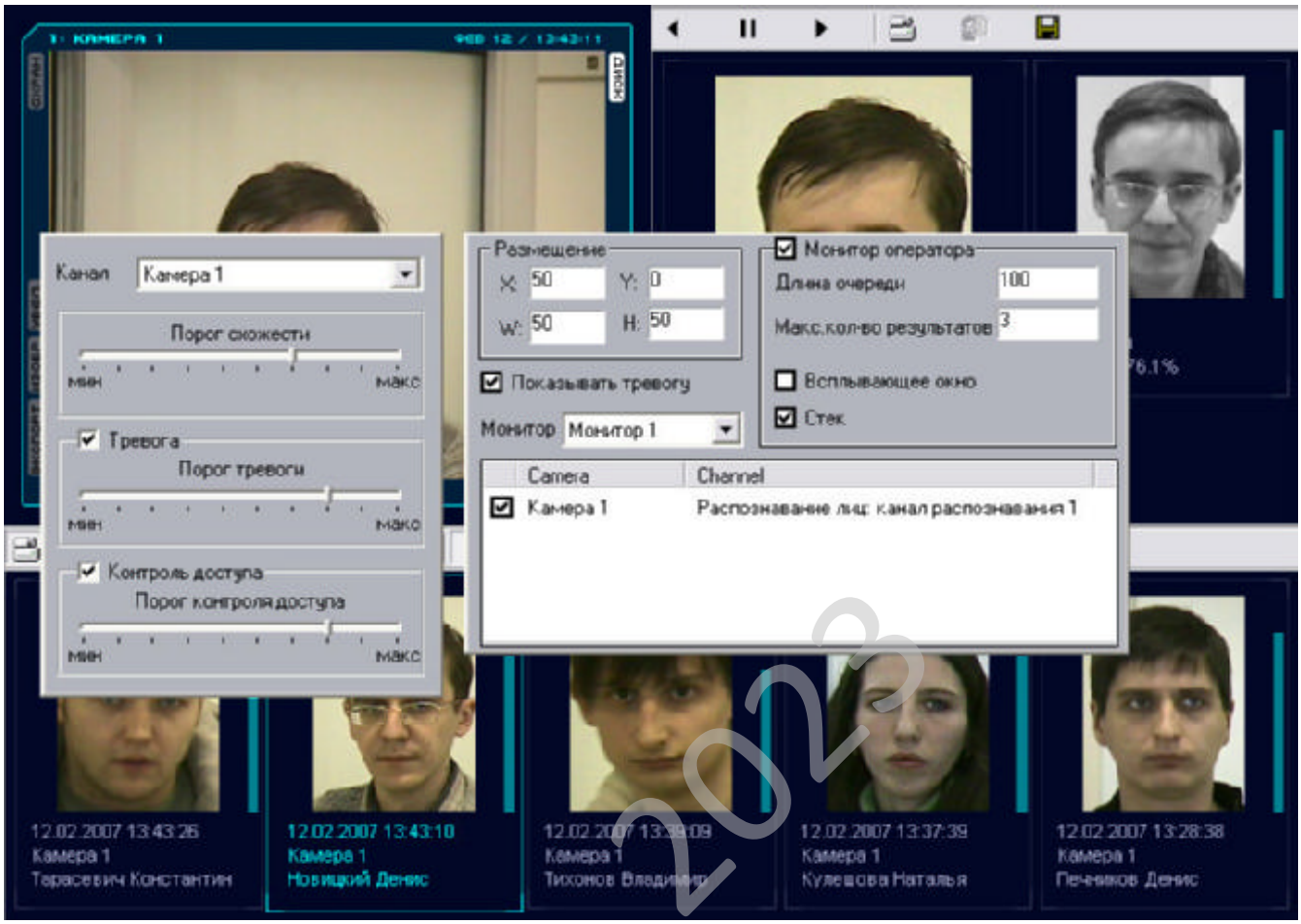

Рисунок 2.1 – Інтерфейс користувача SecurOS Face

Система "SecurOS Face" – ефективний компонент інтегрованої системи безпеки. Функціональні можливості системи дозволяють вирішувати завдання, пов'язані з організацією відеоспостереження, контролю й обмеження доступу, на об'єктах, до рівня безпеки яких пред'являються підвищені вимоги. Система "SecurOS Face" може стати ефективним компонентом інтегрованого комплексу безпеки, надаючи численні переваги при роботі:

– організація мережі необмеженої кількості відеоканалів, робочих місць і централізованого сховища даних;

– віддалений моніторинг і доступ до відеоархіву, дистанційне адміністрування системи (настроювання логіки роботи, експорт даних і ін.);

– інтеграція з устаткуванням системи контролю доступу, виконавчими пристроями (турнікетами), датчиками й ін.;

– забезпечення комплексної реакції системи безпеки, окремих її компонентів за результатами розпізнавання й ідентифікації;

– настроювання різних видів автоматичного оповіщення (звукове, телефон, e-mail, sms) за результатами розпізнавання й ідентифікації осіб;

– спільна робота із зовнішніми базами даних, у тому числі, для дозволу/заборони проходу людей по "білому"/"чорному" списках;

– можливість одночасної (на одному комп'ютері) обробки даних від 4 відеоканалів у режимі реального часу;

– зберігання довгострокового відеоархіву великого обсягу (оперативний архів – до 2 років);

– миттєвий пошук інформації у відеоархіві й базі даних;

– формування зручних для моніторингу обстановки в режимі реального часу й роботи відеоархівом і базою даних, інтерфейс системи не вимагає спеціальної підготовки для роботи.

Програмне забезпечення для розпізнавання обличчя зазвичай не доступне як окреме програмне забезпечення, а зазвичай є частиною послуг. Підприємства, які хочуть інтегрувати технологію розпізнавання облич у власні продукти та програми, можуть вибрати програмне забезпечення на основі веб-сервісів, доступне сьогодні. Ось 10 найкращих програм для розпізнавання обличчя, доступних сьогодні на ринку.

Застереження: ці списки базуються на загальнодоступній інформації та веб-сайтах постачальників. Читачам радимо провести власні детальні дослідження кожного програмного забезпечення для зустрічей. Компанії розташовані в алфавітному порядку.

Amazon Rekognition

Основні послуги: Amazon Rekognition – одне з найнадійніших імен у програмному забезпеченні розпізнавання облич.

– Аналіз облич і пошук по обличчю використовуються для верифікації користувачів, підрахунку людей і забезпечення громадської безпеки.

– Розпізнавання може ідентифікувати об’єкти та сцени, надаючи їм мітки.

– Amazon має перевагу в тому, що у своєму розпорядженні величезний набір даних. Це забезпечує точність.

– Розпізнавання дозволяє користувачам додавати власні мітки до об’єктів і сцен відповідно до потреб бізнесу. Це забезпечує вищі показники успіху матчів. Все, що потрібно зробити компанії, це надати зображення об’єктів і місць, які потрібно ідентифікувати.

– Amazon забезпечує модерацію вмісту. Він має перевірений набір даних, який використовує для позначення неприйнятного вмісту. Він також забезпечує виявлення тексту для розпізнавання імен.

– Він також забезпечує пошук обличчя та перевірку в приватному сховищі фотографій. Зверніть увагу, що в цьому випадку стягуються витрати на зберігання.

– Розпізнавання також забезпечує виявлення облич і аналіз у відео – живому або збереженому.

Клієнтська база: поточна клієнтська база Amazon Rekognition включає медіа-компанії, аналітичні компанії ринку, сайти електронної комерції та кредитні рішення.

Підтримка клієнтів: рішення надає документи, зразки та навчальні посібники.

Модель ціноутворення: модель ціноутворення базується на використаних послугах і кількості проаналізованих зображень. У рамках безкоштовного рівня AWS обробка зображень може використовуватися безкоштовно. Безкоштовний рівень триває 12 місяців і дозволяє аналізувати 5000 зображень на місяць і зберігати 1000 фрагментів метаданих обличчя на місяць. Після цього місячна вартість розраховується на основі кількості оброблених зображень (плюс плата за

зберігання, якщо є). Подібні багаторівневі пакети доступні для обробки відео та настроюваних міток.

Amazon Rekognition можуть використовувати організації, які не мають досвіду машинного навчання. Це дороге рішення, тому для підприємств із вільним бюджетом. Єдине, на що користувачі попереджають звернути увагу, – це додавання вартості зберігання.

Betaface

Основні послуги: Betaface в основному зосереджується на аналізі зображень і відео, а також на розпізнаванні облич і заперечень.

–

– Він пропонує три види послуг – SDK для розпізнавання обличчя, послуги з розробки програмного забезпечення для клієнтів і розміщені веб-сервіси.

– Послугами можуть бути просте розпізнавання обличчя або складне розпізнавання обличчя, ідентифікація та перевірка.

– Він використовує біометричні вимірювання для відстеження рис обличчя як на зображеннях, так і на відео. Він може розпізнавати емоції та етнічну приналежність.

– Він також може відстежувати шкіру, волосся, риси обличчя та форму зачіски.

– Він надає програмні рішення для відеоспостереження та безпеки.

Клієнтська база: Клієнтська база Betaface включає агентства веб-реклами та розваг, виробників медіа-контенту та розробників програмного забезпечення B2B.

Підтримка клієнтів: Betaface надає допомогу в інтеграції. Це також допомагає з повною підтримкою веб-сервісу та інфраструктури бази даних , якщо це вибрано.

Модель ціноутворення: Betaface дозволяє безкоштовно використовувати його API для 500 зображень на день і 0,035 євро за зображення після цього. Він

також пропонує три плани передплати на місяць: базовий план за 245 доларів США за обробку 40 000 зображень, преміальний план за 490 доларів США за 100 000 зображень і ультраплан за 1 595 доларів США за 300 000 зображень. За обробку кожного додаткового зображення стягується додаткова плата відповідно до плану передплати.

Betaface можуть використовувати організації, які трохи підковані в техніці та, можливо, мають власних розробників, які хочуть інтегрувати свою програму з FRS. Це рішення може добре задовольнити невеликі підприємства.

3. BioID

Основні послуги: BioID – це рішення, сумісне з GDPR, яке надає біометричні дані як послугу. Він надає хмарні служби FRS, до яких ваш продукт може отримати доступ за допомогою API. Програмне забезпечення пропонує три продукти:

– Веб-сервіс BioID: це пропозиція SaaS, яку можна розгорнути локально або в хмарі.

– Виявлення живості: це служба розпізнавання для визначення присутності користувача за допомогою розпізнавання обличчя, очей і голосу. Він використовується для запобігання онлайн-шахрайству та загрозам ідентифікації.

– PhotoVerify: це рішення поєднує технологію виявлення облич із службою Liveness Detection від BioID для перевірки фотографій, які використовуються як підтвердження особи.

Клієнтська база: це рішення призначене для компаній, яким потрібна електронна ідентифікація та електронний підпис. Він надає послуги фінансовим службам, KYC з біометрією, оренді автомобілів із самообслуговуванням, охороні здоров’я та організаторам онлайн-іспитів.

Підтримка клієнтів: BioID має надійну онлайн-документацію своїх API.

Модель ціноутворення: послуги BioID можна придбати на рівні оплати за користувача. Ви можете зв'язатися з BioID, щоб дізнатися ціни.

Організації з власними програмами або продуктами можуть розглянути BioID. Він ідеально підходить для підприємств, де перевірка особи є важливою, наприклад для оренди автомобілів. Компанії, які представлені на різних континентах, можуть розглянути це рішення, оскільки воно відповідає GDPR.

Cognitec

Основні послуги: Cognitec надає клієнтам масштабовану та настроювану FRS за допомогою своєї відкритої системної архітектури через «FaceVacs». Cognitec пропонує п'ять рішень:

– FaceVACS-VideoScan ES Live: це можна використовувати для розпізнавання облич у прямих відеопотоках. BioID розширює це рішення, надаючи можливість підраховувати людей, створювати демографічну статистику та відстежувати потоки людей.

– FaceVACS-VideoScan ES: це корпоративне рішення Cognitec на основі передплати для встановлення та керування скануванням відео. Cognitec вибирає комп’ютерне обладнання та обладнання для камери та керує ним локально або в хмарі , розгортаючи партнера Cognitec.

– FaceVACS-DBScan ID: це рішення для біометричної перевірки та ідентифікації.

– FaceVACS-DBScan LE: це рішення для біометричної перевірки для правоохоронних органів.

– FaceVACS-Entry: це рішення інтегрує програмне забезпечення FRS з апаратним забезпеченням найвищого класу для створення електронних воріт на контрольно-пропускних пунктах кордону.

Клієнтська база: Cognitec наразі має клієнтів у сфері управління документами, правоохоронними органами, фізичною безпекою та прикордонним контролем. Програмне забезпечення також працює з комерційними продуктами.

Підтримка клієнтів: клієнти Cognitec можуть порушувати проблеми на порталі клієнтів Cognitec. Він також надає підтримку на місці, навчання та консультаційні послуги. Також доступна команда віддаленої підтримки.

Модель ціноутворення: ціни починаються від 0,01 дол. США за раз за користувача та відрізняються залежно від використання.

Cognitec найкраще підходить для організацій, які шукають рішення безпеки та контролю натовпу на основі FRS. Він має значну присутність на ринку розпізнавання облич і може надавати ефективні масштабні рішення з мінімальними витратами на налаштування.

DeepVision AI

Основні послуги: DeepVision AI надає рішення FRS для маркетингу та планування, а також для компаній, які хочуть використовувати перевірку обличчя для безпеки.

– Він збирає дані про кількість відвідувачів у певному районі міста за віком, статтю та етнічною приналежністю. Ці дані використовуються, щоб допомогти рекламодавцям і брендам орієнтуватися на клієнтів за допомогою персоналізованих оголошень. Розпізнавання обличчя та перевірка для безпеки.

– Рішення DeepVision для перевірки обличчя ідентифікує людей користувачами, як правоохоронні органи для безпеки.

– Він надає інформаційну панель аналітики в реальному часі, яку можна налаштувати.

Клієнтська база: DeepVision наразі орієнтується на розумних міських планувальників. Його клієнтами в основному є роздрібні фірми та рекламні агентства.

Підтримка клієнтів: DeepVision AI надає службу запитань і відповідей електронною поштою для запитів щодо інтеграції веб-служб. Він також надає практичну підтримку експертів з комп'ютерного зору. Він також пропонує цілодобову підтримку протягом усього року.

Ціноутворююча модель: це програмне забезпечення встановлюється відповідно до розміру споживання. Ваш рахунок буде визначено кількістю ваших запитів. Програмне забезпечення стягує плату в розмірі 0,008 доларів США за 10

запитів на основі розпізнавання обличчя, віку та статі, розпізнавання автомобіля, візуального контексту, візуального пошуку та розпізнавання бренду.

DeepVision найкраще підходить для забудовників нерухомості , фірм роздрібної торгівлі та маркетингових агентств. Функцію на основі штучного інтелекту тут можна застосувати до безпеки та мобільності пішоходів, виявлення інцидентів і розпізнавання транспортних засобів, серед іншого, для забезпечення автоматизованого аналізу відео. Він ідеально підходить для середніх і великих підприємств, яким потрібні масштабовані рішення.

Face++

Основні послуги: Face++ надає чотири типи технологічних рішень:

– Розпізнавання облич для визначення облич, порівняння облич і пошуку облич.

– Розпізнавання тіла людини для виявлення тіла, виявлення скелета та контуру тіла.

– Прикрашання зображення для об’єднання облич на кількох фотографіях.

– Розпізнавання зображень – для позначення облич на фотографіях.

Face++ надає SDK та API для використання кожної з цих технологій, а також спеціальні хмарні служби. Пропозиції API включають виявлення обличчя, порівняння, пошук, інформацію про орієнтири обличчя, розпізнавання емоцій, оцінку погляду, аналіз стану шкіри та 3D-реконструкцію обличчя. Його пропозиції SDK включають порівняння облич і інформацію про орієнтири обличчя.

Клієнтська база: сьогодні Face++ використовується в автомобільній промисловості, освіті, онлайн-маркетингу та індустрії мобільних телефонів.

Підтримка клієнтів: Face++ має вичерпну онлайн- документацію для API та SDK. Він також має онлайн-консоль підтримки для своїх користувачів.

Модель ціноутворення: пропонує чотири моделі ціноутворення: безкоштовна, платна за ходом, щоденний/місячний план і оплата за ліцензування.

Безкоштовна модель дозволяє користувачам тестувати свої API та SDK протягом необмеженого часу, обмежуючи кількість QPS. Щоденні та місячні ставки варіюються від $1000 до $10 500 на місяць залежно від вибраних функцій.

Face++ чудово підходить для порівняння облич зі збереженими зображеннями. Деталізація функцій, які він пропонує, і його гнучкі моделі ціноутворення роблять його природним вибором для компаній, які все ще намагаються з'ясувати, як вони хочуть інтегруватися з FRS.

FaceFirst

Основні послуги: FaceFirst має на меті використовувати DigitalID для заміни карток і паролів. В основному він пропонує рішення на основі FRS у чотирьох ключових сферах:

– Рішення безпеки: вони включають автентифікацію, контроль доступу, перевірку ідентифікатора та перевірку віку.

– Залучення клієнтів: для програм лояльності та персоналізованої реклами.

– Безпека: для запобігання втратам і пом'якшення шахрайства за допомогою сповіщень у реальному часі про спроби підробки ідентифікаційної інформації.

– Бізнес-аналітика: для оцінки настроїв, аналітики трафіку, аудіоаналітики та аналітики шахрайства компаній.

Клієнтська база. Клієнтами Facefirst є роздрібні торговці, транспортні центри та інші організації, які потребують автентифікації, залучення клієнтів і рішень для лояльності.

Підтримка клієнтів: Facefirst надає підтримку за викликом – він має онлайн-центр підтримки для створення запитів на обслуговування.

Модель ціноутворення: зв'яжіться з FaceFirst, щоб дізнатися ціни.

FaceFirst найкраще підходить для компаній, які прагнуть інтегрувати FRS у свої продажі та маркетинг. Це вже довело свою ефективність у великій роздрібній торгівлі, особливо з роздрібними крадіжками.

Kairos

Основні послуги: Kairos надає підприємствам веб-служби на основі FRS і SDK для інтеграції своїх рішень. Він надає послуги виявлення обличчя, ідентифікації та перевірки. Програмне забезпечення також може відстежувати важливі демографічні дані. Його функція автоматичного позначення тегами дозволяє швидше шукати та індексувати зображення та відео.

Клієнтська база: Kairos наразі використовується в дослідженнях ринку, управлінні капіталом, автомобільній промисловості, банківській справі, охороні здоров'я та безпеці.

Підтримка клієнтів: Kairos надає онлайн- документацію та підтримку електронною поштою.

Модель ціноутворення: Kairos має 14-денну безкоштовну пробну програму. Він пропонує три плани: Student Cloud за 19 доларів США на місяць, Developer Cloud за 99 доларів США на місяць і Business Cloud за 249 доларів США на місяць.

Kairos найкраще підходить для організацій, які хочуть оснастити свої мобільні програми функціями FRS. Хоча користувачі повідомляють про крутий процес навчання під час інтеграції та початкового використання, вони також відзначають розширені функції Kairos як безперечний плюс.\

SenseTime

Основні послуги: SenseTime надає технологію аналізу обличчя та тіла, окрім своїх автономних служб FRS. Його рішення відрізняються високою точністю. Він надає такі послуги, як:

– Розпізнавання обличчя.

– Розташування рис обличчя: позиціонування рис позначається незалежно від широких кутів, зміни виразів або руху.

– Атрибути обличчя: рішення може точно розпізнавати більше десяти атрибутів обличчя.

– Виявлення активності: рішення перевірки користувача для запобігання атакам спуфінгу.

– Точка характеру тіла: використовується для аналізу тіла, і він використовує 14 точок характеру тіла для розпізнавання різних частин, зокрема під час руху.

Клієнтська база: Клієнтська база SenseTime включає розумні планувальники міст і компанії в автомобільному секторі, охороні здоров'я та рекламі.

Підтримка клієнтів: SenseTime надає документацію для користувачів.

Модель ціноутворення: зв'яжіться з SenseTime, щоб дізнатися ціни.

Цей продукт можна використовувати на підприємствах, які мають більше потреб у розпізнаванні облич на основі відео.

Sky Biometry

Основні послуги: Sky Biometry – це постачальник веб-послуг, який пропонує три Основні послуги:

– Розпізнавання обличчя

– Визначення ознаки

– Розпізнавання обличчя

Клієнтська база. Поточна клієнтська база Sky Biometry складається з компаній, які проводять рекламні кампанії, керування фотографіями, автентифікацію користувачів або модерацію спільноти, наприклад модерацію та перевірку профілів знайомств.

Підтримка клієнтів: це програмне забезпечення надає детальну документацію API в Інтернеті. Час відповіді служби підтримки залежить від вибраного плану підписки.

Модель ціноутворення: Sky Biometry має три тарифні плани – перший безкоштовний, другий – 50 євро на місяць і третій – 100 євро на місяць. Ви навіть можете зв'язатися з Sky Biometry, щоб отримати індивідуальний тарифний план. Плани підписки базуються на кількості викликів API на місяць/день/годину та

кількості навчених тегів. Sky Biometry ідеально підходить для організацій, які мають власні групи розробників, які створюють програми. Вона була виділена з Neurotechnologies, компанії, яка більше двох десятиліть займається біометричною ідентифікацією та ідентифікацією об’єктів. Це означає, що поряд із застарілою версією Sky Biometry має одну з найбільш гнучких і масштабованих пропозицій FRS SaaS.

Trueface.ai

Основні послуги: Trueface.ai надає рішення FRS у трьох режимах – із SDK, контейнером, що розгортається, і програмним рішенням plug-and-play (бета-версія). Програмне забезпечення пропонує чотири Основні послуги:

– Розпізнавання обличчя
– Виявлення зброї
– Космічна аналітика
– Перевірка в реальному часі для запобігання спробам спуфінгу

Клієнтська база: її клієнтська база складається з компаній із секторів фінансових технологій, освіти, безпеки, охорони здоров’я, роздрібної торгівлі та гостинності. Він також пропонує рішення для державних установ.

Підтримка клієнтів: Trueface.ai надає онлайн-документацію та має активний центр розробників.

Модель ціноутворення: ціна починається від 99 доларів США на місяць за послугу. Trueface ідеально підходить для компаній із досвідом програмування. Це може змінитися, коли його рішення plug-and-play перейде з бета-версії.

Висновок

Технологія розпізнавання обличчя більше не є справою майбутнього. Він уже тут, і багато організацій уже почали використовувати FRS для безпеки, маркетингу та бізнес-цілей. Інтелектуальна аналітика, отримана від FRS, може бути використана для прогнозного аналізу та подальшого оснащення продуктів. Для успішного розгортання FRS підприємства повинні протестувати кожне рішення FRS за допомогою повного та значущого набору даних.

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 виділення й розпізнавання обличчя користувача у мережі.

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

Будь-який хороший FRS має три ключові компоненти:

– Обладнання для захоплення зображень. Ці зображення також можна вводити в програмне забезпечення з незалежних пристроїв.

– Інтелект для порівняння захоплених облич з наявними даними.

– База даних, тобто існуюча колекція ідентичностей. Це може бути що завгодно: від баз даних співробітників до зображень, видалених із соціальних мереж.

Тепер давайте розберемося, як працює FRS.

1. Виявлення: Виявлення починається з виділення обличчя із зображення, яке подається в систему. Згодом на обличчі людини відзначаються різні риси. Певні риси обличчя не змінюються з віком або розміром. Це відстань між очима, глибина очниці і форма носа. Існує близько 80 таких об’єктів, які називаються «орієнтирами». Потім розміри цих орієнтирів об’єднуються, щоб створити код. Цей код називається «відбитком обличчя», і він унікальний для кожної людини.

2. Зіставлення: цей відбиток обличчя потім зіставляється з відбитками, збереженими в системі. На цьому етапі зображення проходить кілька технологічних рівнів для забезпечення точності. Оскільки більшість наших баз даних наразі є двовимірними фотографіями, зображення бази даних потрібно обробляти за допомогою рівня технології. Ця обробка зазвичай включає витягування орієнтирів обличчя, щоб вони були схожі на їхні тривимірні аналоги. Якщо зображення об’єкта має низьку роздільну здатність, його необхідно закодувати та декодувати, щоб створити деталі з бажаною роздільною здатністю. Алгоритми повинні враховувати різницю в освітленні, виразі обличчя та кутах.

3. Ідентифікація: мета цього кроку залежить від того, для чого

використовується програмне забезпечення для розпізнавання обличчя – для спостереження чи автентифікації. В ідеалі цей крок має забезпечити відповідність об’єкта 1:1. Це можна зробити кількома способами: швидким переходом, щоб звузити параметри, а потім увімкненням більш складних шарів. Деякі компанії аналізують текстуру шкіри разом із алгоритмами розпізнавання обличчя, щоб підвищити точність.

Кожен постачальник програмного забезпечення для розпізнавання обличчя зосереджується на різних аспектах технологічних рівнів, щоб забезпечити майже бездоганне обслуговування. Наприклад, одне програмне забезпечення може зосереджуватися на коригуванні умов освітлення, а інше – на аналізі текстури шкіри.

Хто користується програмним забезпеченням для розпізнавання обличчя?

Такі компанії, як Mastercard, уже використовують FRS як ідентифікатор під час платежів і для підвищення безпеки. FRS має потенційне застосування в роздрібній торгівлі, готельному секторі, банках, банкоматах та аеропортах. Компанії, орієнтовані на мобільну комерцію, отримають велику користь від FRS. Маркетингові фірми розглядають можливість використання FRS для персоналізованого обслуговування клієнтів.

Наприклад, деякі компанії електронної комерції, які продають окуляри, працюють над використанням FRS, щоб рекомендувати окуляри, які добре виглядають для структури вашого обличчя. Це позбавляє необхідності відвідувати магазин, щоб їх приміряти. Однак найвагоміші варіанти використання FRS сьогодні пов’язані з безпекою.

Ключові обов’язкові функції програмного забезпечення для розпізнавання облич

Програмне забезпечення для розпізнавання обличчя можна використовувати для автентифікації, спостереження або маркетингу. Залежно від вашого випадку використання, ось деякі ключові функції, на які варто звернути

увагу під час розгляду варіантів FRS:

Ключові обов'язкові функції програмного забезпечення для розпізнавання облич

1. Навчена та зростаюча база даних: Рівень точності будь-якої FRS залежить від бази даних, на якій навчався її штучний інтелект. Дані повинні постійно зростати, різноманітні за статтю та етнічним походженням. Навчальні дані також повинні відрізнятися в освітленні, ракурсах і виразах обличчя. Хороша база даних також містить різні роздільності зображень, з якими система може працювати. Програми машинного навчання ефективні настільки, наскільки хороша база даних, яку вони використовують для навчання, і FRS не є винятком.

2. Безпека та конфіденційність користувача: будь-яке біометричне програмне забезпечення тісно пов'язане з особистістю людини. Це означає, що дані (в даному випадку відбитки обличчя), накопичені FRS, є дуже конфіденційними. Дані користувача необхідно шифрувати та очищати через регулярні проміжки часу. Постачальники програмного забезпечення повинні мати надійний план на випадок витоку даних.

3. Точність алгоритму: ключовими показниками, на які слід звернути увагу під час розгляду FRS, є коефіцієнт помилкового прийняття (FAR) і коефіцієнт помилкового відхилення (FRR). FAR – це коли різні зображення хибно зіставляються як ідентичні. У цьому випадку, якщо ви використовуєте його для безпеки, доступ може бути дозволено не тій особі. У FRR точні зображення хибно не збігаються як різні. У цьому випадку потрібна особа може отримати відмову в доступі. У практичному сценарії безпеки FAR має бути низьким, а FRR високим.

4. Масштабованість: для великих підприємств, які хочуть використовувати FRS для автентифікації, масштабованість є важливою, оскільки програмне забезпечення потрібно розгортати в кількох місцях.

5. Адаптованість і підтримка: постачальники FRS повинні пропонувати резервні варіанти до уваги. У разі збою системи може знадобитися людська

підтримка та нагляд, поки система повернеться до нормального стану. Підтримка також потрібна для налаштування обладнання, зокрема камер, для максимальної точності.

6. Прозорість і етика: лише за останній рік FRS кілька разів критикували через відсутність прозорості. Переконайтеся, що програмне забезпечення, яке ви використовуєте, не вдається до неетичних методів, як-от очищення соціальних мереж для збору навчальних даних або порушує конфіденційність користувачів.

Контрольний список для вибору правильного програмного забезпечення

Будь-яка організація, яка збирається вибрати відповідне програмне забезпечення для розпізнавання обличчя, повинна розглянути такі питання:

1. Чи FRS відповідає потребам вашого бізнесу? Вам може знадобитися FRS для ідентифікації облич у закритому наборі даних (автентифікація співробітників), відкритому наборі даних (відстеження роздрібних клієнтів) або просто для перевірки (просто перевірте, чи два зображення однакові).

2. Чи безпечне рішення? Дані мають бути зашифровані та захищені від злому. Це також має захищати конфіденційність користувачів.

3. Чи перевірено програмне забезпечення? Щоб перевірити точність ваших власних даних, існують доступні відкриті джерела даних, як-от LFW і MegaFace. Ви також можете найняти для цього сторонніх постачальників даних. Просто переконайтеся, що набір даних, вибраний для тестування системи, відображає фактичних людей, які використовують систему у вашому випадку використання.

4. Чи перевірили ви показники FAR і FRR? Розглядаючи ці показники, також подумайте про поріг точності, який вас задовольняє. Наприклад, чи нормально для вашого бізнесу, якщо FRS показує збіги, які збігаються лише на 70%?

5. Чи є у нього сильна команда підтримки? Рішення FRS є складними, тому для безперебійної інтеграції у вашу існуючу систему та безперебійної роботи потрібна хороша команда підтримки. Команда підтримки також має

допомогти налаштувати програмне та апаратне забезпечення відповідно до ваших потреб.

6. Чи порушують якісь із умов програмного забезпечення закони штату? Розпізнавання обличчя зараз є актуальною темою, і правила щодо цього різняться залежно від штату США. Переконайтеся, що ви не перетинаєте жодних законних меж під час використання FRS.

3.2 Розробка структурної схеми

За результатами аналізу відомих методів ВО встановлено, що доцільно використовувати групу методів на основі моделювання зображення обличчя, які демонструють високу достовірність виявлення [2]. Обрані за базові методи ВО К. Гарсіа і М. Делакіс [4] та П. Віоли і М. Джонса [6] мають, відповідно, або високу достовірність, або високу швидкодію. Для подолання вказаних недоліків розроблено узагальнену інформаційну модель процесу ВО (рис. 3.5), яка використовує каскадний (багаторівневий) підхід до побудови класифікатора позитивних/негативних прикладів і комбінацію в одному каскаді декількох гетерогенних класифікаторів. За допомогою даної моделі визначено функції та сформовано критерії роботи складових компонентів комбінованого каскаду класифікаторів, що дозволило запропонувати шляхи підвищення достовірності та швидкодії методів виявлення облич на півтонових і кольорових зображеннях.

На основі запропонованої моделі розроблено метод виявлення облич на півтонових зображеннях, який базується на комбінованому каскаді нейромережних класифікаторів (ККНК) [10]. Як перший рівень даного комбінованого каскаду, що відповідає за виявлення облич-кандидатів, запропоновано використовувати каскад слабких класифікаторів (КСК) на основі Хаар-подібних ознак, який характеризується високою швидкодією. Для другого рівня ККНК, що призначений для верифікації облич-кандидатів, запропоновано застосувати згорткову нейронну мережу (ЗНМ), яка в задачах класифікації

володіє більшою стійкістю до деформацій вхідних образів, ніж інші відомі класифікатори, що дозволяє отримати високу достовірність виявлення.

Для спрощення реалізації каскаду слабких класифікаторів запропоновано використовувати нейромережну технологію, коли КСК являє собою багатошаровий персептрон, що складається із рівнів (нейронних шарів), кожний з яких містить один і більше слабких класифікаторів (нейронів).

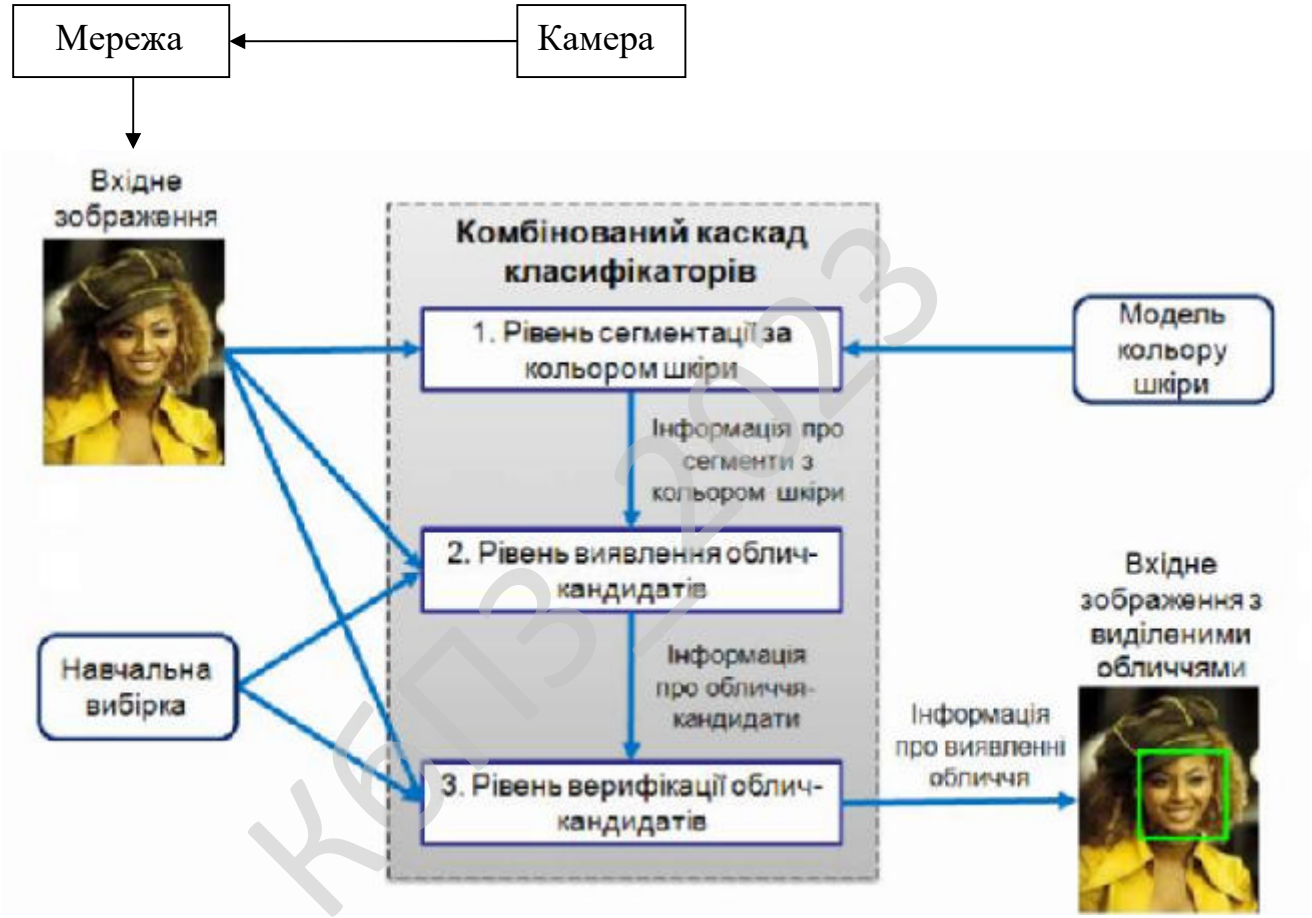

Рисунок 3.1 – Структурна схема системи

Входом для слабкого класифікатора є Хаар-подібна ознака прямокутної форми, яка складається зі «світлих» і «темних» прямокутників, а її значення *Feat* розраховується за формулою [6]:

$$Feat(x) = s_w \times SUM_w + s_b \times SUM_b,$$

де x – вхідне зображення, s_w та s_b – синаптичні ваги для всього прямокутника ознаки і для його темної частини відповідно, SUM_w та SUM_b – суми пікселів всього прямокутника ознаки і його темної частини відповідно. Вихідне значення слабкого класифікатора h знаходиться наступним чином [6]:

$$h(x) = \begin{cases} 1, \text{якщо } Feat(x) < \theta \\ -1, \text{якщо } Feat(x) > \theta, \end{cases}$$

де θ – порогове значення слабкого класифікатора. У свою чергу вихідне значення КСК H являє собою лінійну комбінацію слабких класифікаторів [6]:

$$H(x) = \sum_{t=1}^{T} \eta_t \times h_t(x),$$

де T – кількість слабких класифікаторів, η_i – вага t-слабкого класифікатора.

Для верифікації облич-кандидатів в рамках ККНК запропоновано використовувати згорткову нейронну мережу із площинами, які здійснюють і згортку, і підвибірку одночасно, що дозволило у 2,7 рази зменшити кількість елементарних операцій для обробки вхідного зображення у порівнянні із ЗНМ, використаною у методі ВО К. Гарсіа та М. Делакіс. Для генерації структури ЗНМ розроблено алгоритм, який дозволив автоматизувати проектування несиметричної розрідженої структури мережі [10].

Враховуючи, що в ЗНМ використовано біполярну сигмоїдну функцію активації, вихідне значення нейрона У з координатами (m, n) р-площини /-шару знаходиться за формулою:

$$y_{m,n}^{l,p}(x) = \frac{2}{1 + \exp(-WSUM_{m,n}^{l,p}(x))} - 1,$$

а зважена сума нейрона $WSUM$ у свою чергу обчислюється згідно з наступним виразом:

$$WSUM_{m,n}^{l,p}(x) = \left(\sum_{k=0}^{K-1}\sum_{r=0}^{R-1}\sum_{c=0}^{C-1} y_{2m+r,2n+c}^{l-1,k}(x) \times w_{r,c}^{l,p,k}\right) - b^{l,p},$$

де K – кількість вхідних площин, R та C – висота і ширина ядра згортки, w – синаптична вага з координатами (r, c) у ядрі згортки між k-площиною $(l-1)$-шару і p– площиною l -шару, b – порогове значення нейронівр-площини l -шару, $y(x)$ – вихідне значення $(l-1)$-шару k-площини з координатами $(2m + r, 2n + c)$.

3.3 Розробка функціональної схеми

На рисунку 3.8 зображена функціональна схема системи. Нижче розглянемо її більш докладно.

Функціональна схема системи включає в себе наступні функціональні блоки:

1. Блок виділення обличчя користувача у мережі.
2. Блок розпізнавання обличчя користувача у мережі.
3. Блок відеодетекції.
4. Блок динамічного спостереження за порушником.
5. Блок подання сигналу тривоги.
6. Блок документування подій на об’єкті.

Розглянемо ці блоки більш детально.

Блок виділення обличчя користувача у мережі

Виділимо етапи при рішенні завдання розпізнавання зображень:

– Сприйняття поля зору.

– Сегментація.

– Нормалізація виділених об'єктів.

– Розпізнавання.

Виходячи із цього, використовуються наступні основні принципи:

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Передобробка

Процедура попередньої обробки використовується практично завжди після одержання інформації з датчика, і являє собою застосування операцій усереднення й вирівнювання гістограм, різного типу фільтрів для виключення

перешкод, що виникають у результаті апаратної дискретизації й квантування, а також придушення зовнішніх шумів.

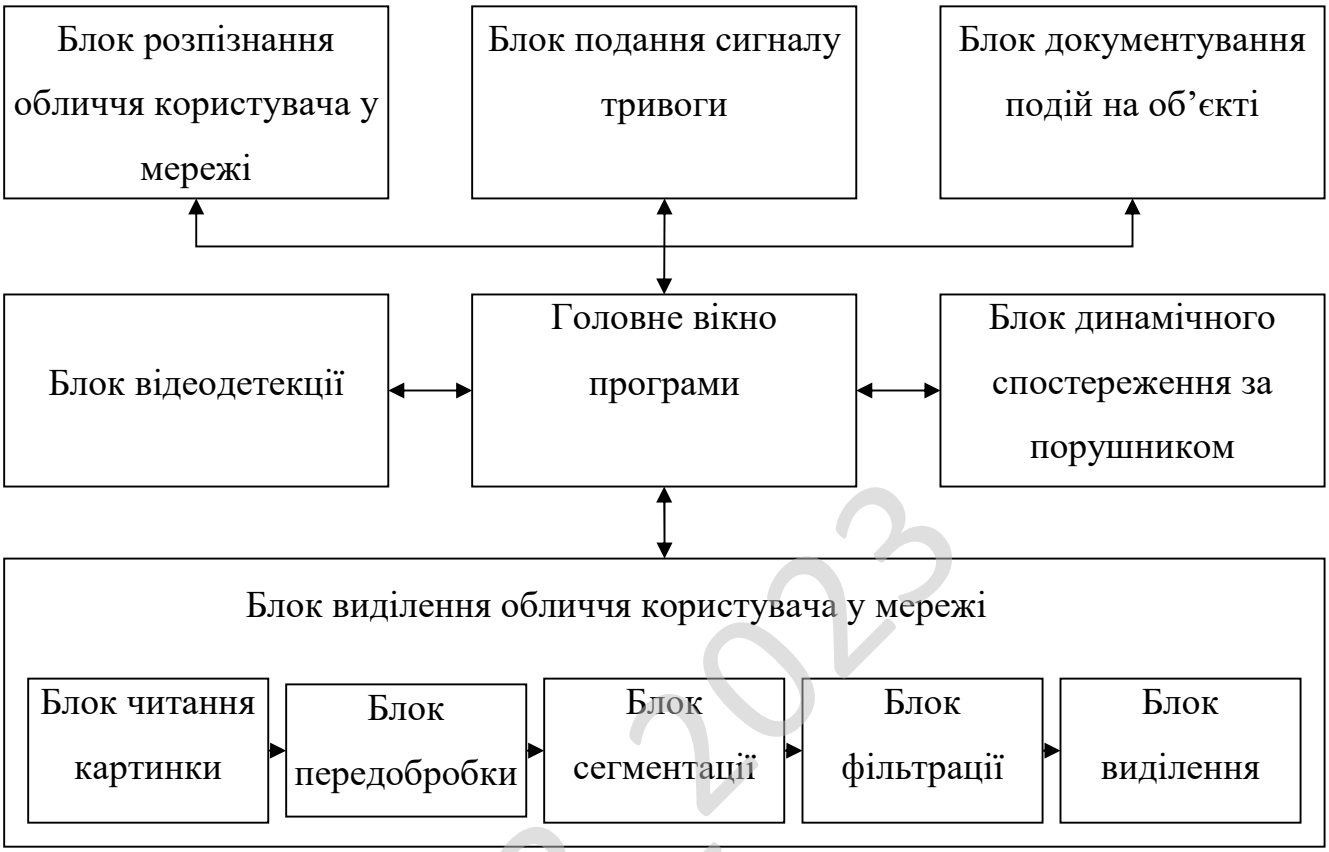

Рисунок 3.8 – Функціональна схема системи

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 – розпізнавання. Для цього етапу вхідними даними є зображення, отримані в результаті шумозаглушення й процесу сегментації. Як правило, вони відрізняються від

еталонних геометричними і яскравостними перекручуваннями, а також збереженими шумами.

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

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

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

3. Кореляційно-ознаковий метод має на увазі під собою обробку статистичними методами ознак, отриманих у такий спосіб. Споконвічно застосовується метод приватних кореляцій для різних фрагментів еталонного зображення, а потім у сигнальному просторі отримані кореляційні коефіцієнти розглядаються як ознаки.

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

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

2. Ознаки зображень різних класів повинні істотно розрізнятися.

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

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

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

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

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

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

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

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

– Аналіз множини довільно розташованих об'єктів, виділення конкретних предметів при відсутності або неможливості визначити деякі ознаки (наприклад, коли на 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 розпізнавання обличчя користувача у мережі

Система захвата осіб дозволяє виділяти тільки особи людей, вибирати найбільш виразне зображення з декількох варіантів і зберігати їх у базі даних. Створення бази осіб людей на прохідних підприємств, у місцях масового скупчення людей (культурно-розважальні центри, вокзали, стадіони й т.д.) полегшує роботу з архівами при розслідуванні позаштатних ситуацій.

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

Блок відеодетекції

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

Блок динамічного спостереження за порушником

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

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

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

Блок подання сигналу тривоги

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

Блок документування подій на об’єкті

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.9.

Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ, далі до блоку відеодетекції, виведення зображення з камери чи до блоку динамічного спостереження за порушником далі до блоку розпізнавання обличчя користувача у мережі та блоку подання сигналу тривоги.

Крім цього з головного вікна ПЗ можна потрапити до блоку читання картинки, блоку передобробки, блоку сегментації, фільтрації, виділення та в остаточному результаті до блоку документування подій на об’єкті.

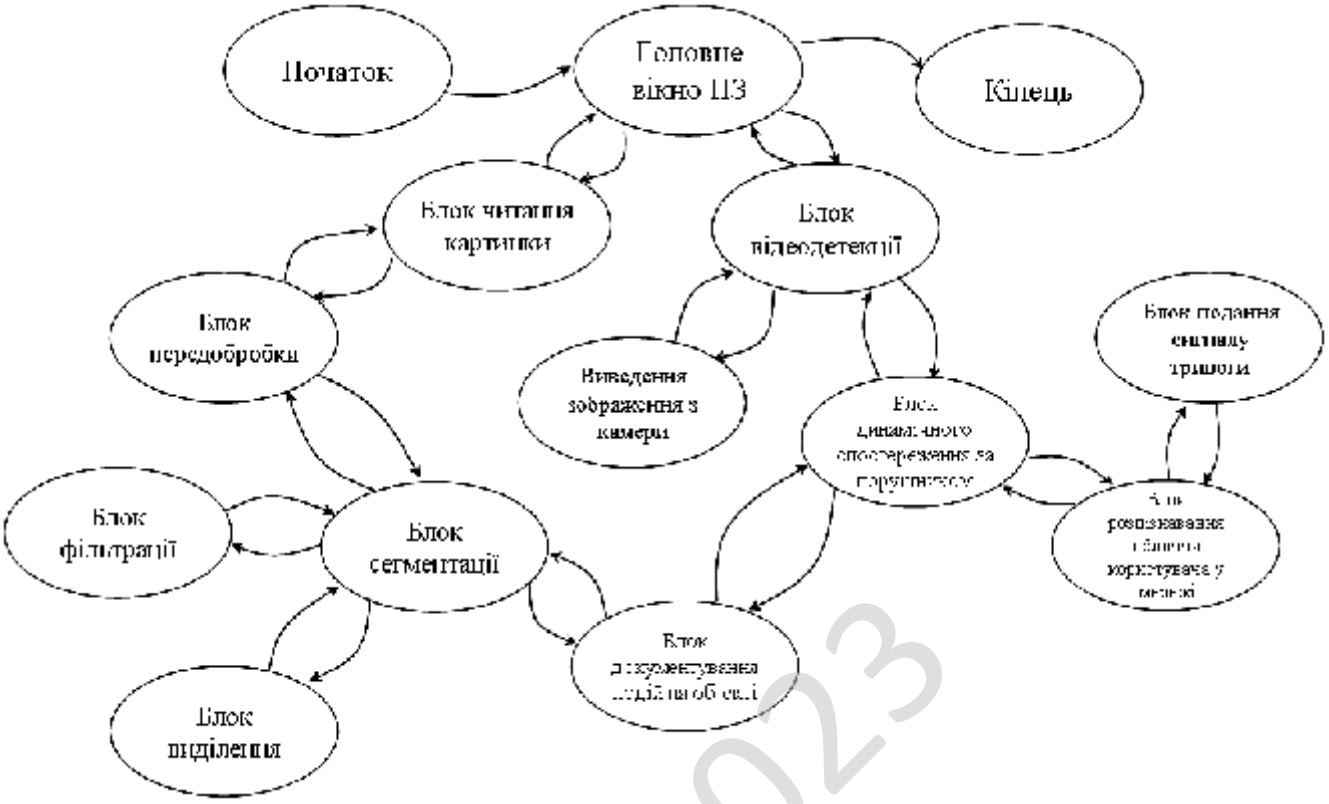

Рисунок 3.9 – Діаграма взаємодії процесів

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

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

4.1 Блок-схеми та опис алгоритмів функціонування системи

На рисунку 4.1 наведено блок-схему основної програми. Її робота складається з виконання наступних кроків. Спершу відбувається виведення основного вікна програми, після цього:

– Підключення камери до ПЗ.

– Знайдено декілька камер (запит).

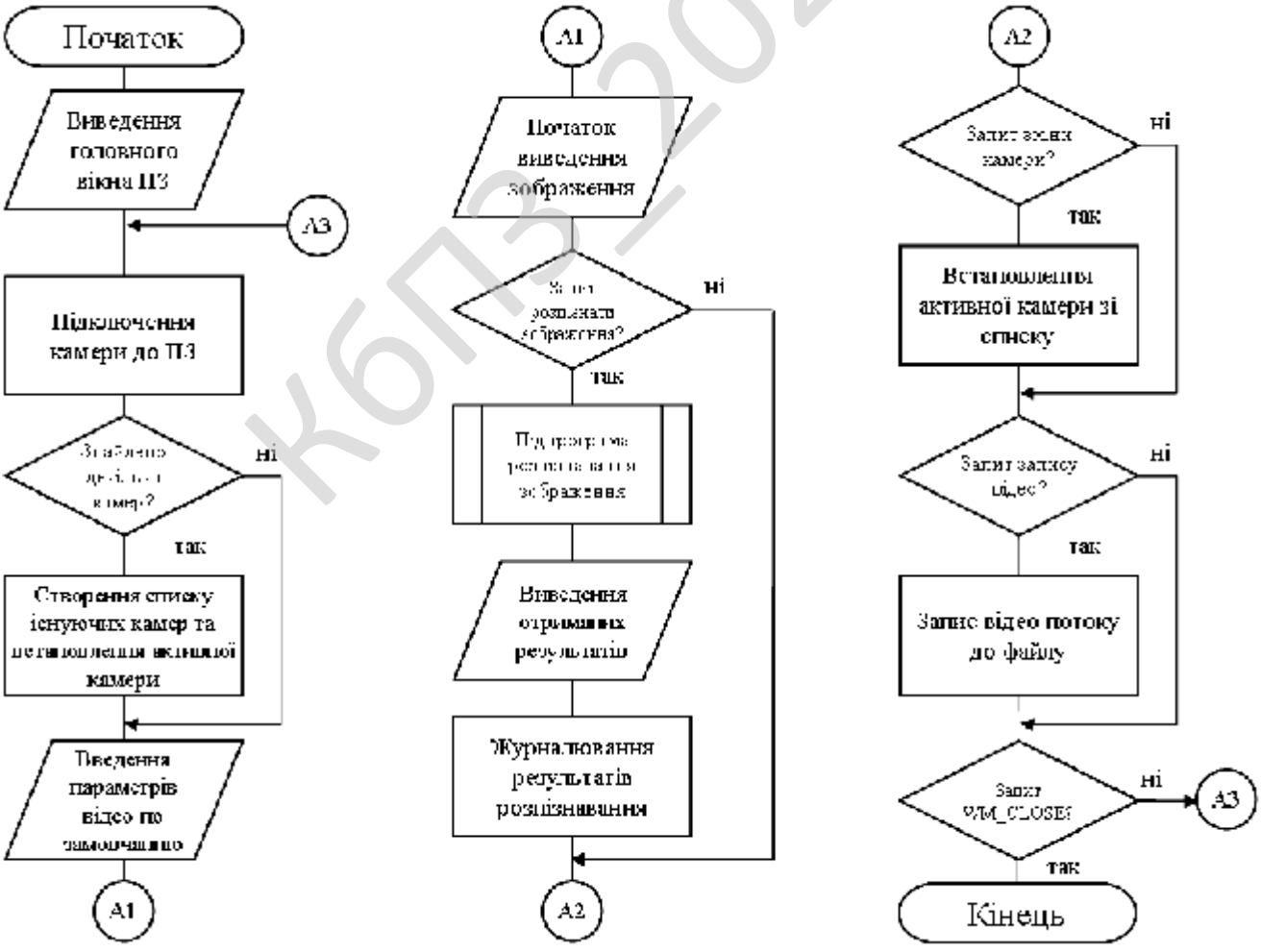

Рисунок 4.1 – Блок-схема основної програми

– Створення списку існуючих камер та встановлення активної камери.

– Введення параметрів відео по замовчанню.

– Початок виведення зображення.

– Запит розпізнати зображення?

– Підпрограма розпізнавання зображення.

– Виведення отриманих результатів.

– Журналювання результатів розпізнавання.

– Запит зміни камери?

– Встановлення активної камери зі списку.

– Запит запису відео?

– Запис відео потоку до файлу.

– Запит WM_CLOSE?

На рисунку 4.2 зображено роботу підпрограми розпізнавання зображення де проходять наступні дії:

– Система пройшла процес навчання? .

– Автоматичне налаштування системи.

– Введення та обробка значень яскравості та чіткості зображення.

– Введення налаштувань розпізнання.

– Обробка отриманого зображення з камери.

– Бінарізація зображення .

– Сегментація зображення .

– Пошук та виділення ознак для подальшого обчислення.

– Поворот, повторне вихоплювання ознак .

– Перетворення зображення у матричні значення.

– Аналіз матриці зображення.

– Формування звіту результату розпізнання кадру.

Опис алгоритмів функціонування системи

Всі об'єкти універсальної безлічі можна розмістити у вершинах одиничного квадрата, таким чином, безлічі фігур, зображених на двопіксельному

полі, може бути зіставлена безліч точок у двовимірному просторі. Ребру цього квадрата буде відповідати перехід від одного зображення до іншого.

Для переходу від (1,1) до (0,0) потрібно буде пройти два ребра, для переходу від (0,1) до (0,0) – одне. Відзначимо, що число ребер у нашому переході – це кількість незбіжних пікселів двох зображень.

Вивід цікавий: відстань від одного малюнка до іншого дорівнює числу незбіжних пікселів у них. Ця відстань називається відстанню за Хемінгом.

Тепер уявимо собі, що в нас малюнок складається із трьох пікселів. Коди зображень тоді будуть складатися із трьох значень, універсальна безліч – з восьми елементів, які ми розмістимо у вершинах одиничного куба.

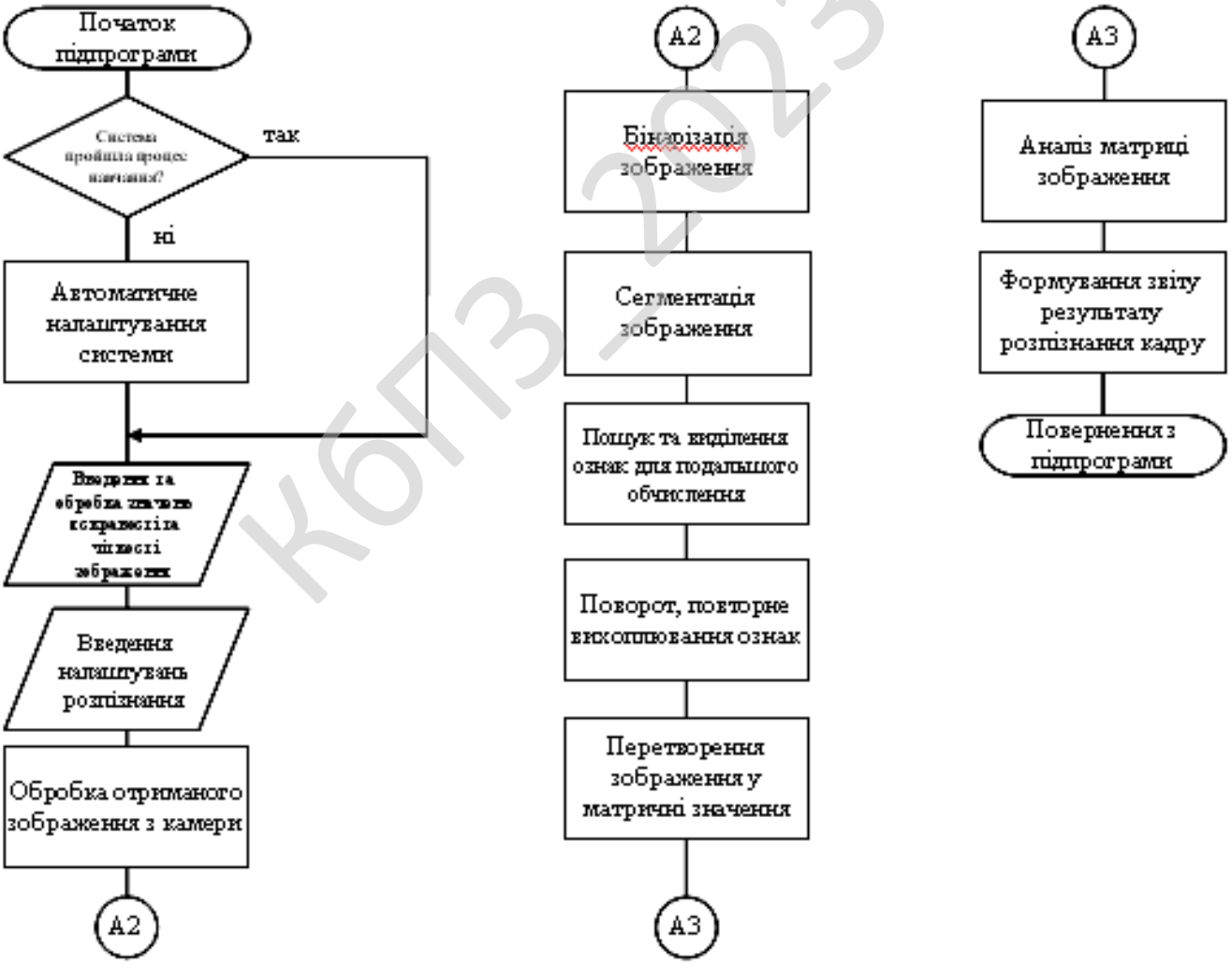

Рисунок 4.2 – Блок-схема підпрограми розпізнавання зображення

Але принципово нічого не зміниться, і відстань за Хемінгом обчислюється так само. У тестовій програмі використовується малюнок 50x70 = 3500 пікселів. Легко зміркувати, що в цьому випадку код будь-якого зображення складається з 3500 значень, універсальна безліч – з 23500 = 4,027 * 101053 елементів, які ми будемо розміщати у вершинах одиничного 3500-мірного куба.

Уявити собі такий 3500-мірний куб нелегко, але зміст від цього не міняється абсолютно. Основна ідея полягає в тому, що в цьому багатомірному кубі зображення, що відповідають якомусь певному образу, лежать недалеко друг від друга.

Ця ідея одержала назву "Гіпотеза про компактність образів". Тепер можна сформулювати завдання: потрібно універсальну безліч розбити на "шматки", компактні безлічі, кожні з яких відповідає образ.

Програмі в процесі навчання повідомляються зображення (точки багатомірного куба) і вказівки, до якого образа кожне зображення відноситься.

При розпізнаванні програма просто дивиться, у яку з відомих компактних областей потрапило вхідне зображення.

Швидше за все, всі зазначені машині зображення ляжуть більш-менш компактно, тому універсальну безліч буде можна розділити.

Властиво розділяти універсальну безліч ми не будемо, а будемо користуватися деякою характеристикою, що показує далекість одного малюнка (точки у вершині багатомірного куба) до групи таких же зображень. Як міра далекості малюнка від групи малюнків використовується потенціал.

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

$$P = a\frac{q}{R^2} \qquad 4.1$$

де a – деякий постійний коефіцієнт, q – величина заряду, R – відстань від даної точки до заряду.

Якщо електричне поле утворене двома або більше зарядами, то потенціал у даній точці дорівнює сумі потенціалів кожного заряду. Аналогія очевидна – кожний малюнок, на якому програма навчалася, створює в просторі універсальної безлічі потенціал.

Після навчання програмі дають розпізнати який-небудь малюнок (точку у вершині багатомірного куба), програма обчислює потенціал, створюваний у цій точці всіма об'єктами образа "а", образа "б"... на які програму вчили й розпізнаваний малюнок відноситься до образа, що створив найбільший потенціал.

Отже, при запуску програми в масив Data: array of array [0..9] of TBitmap; записуються цифри від 0 до 9, написані наступними шрифтами: Arial, Century Gothic, Courier New Cyr, Goudy Old Style і Times New Roman – усього п'ять комплектів (можна легко збільшити). Всі ці зображення були збережені мною й дбайливо викладені в папку \fonts.

```
Procedure Data;
var i1,j:integer;
path:string;
begin
SetLength(Data,5);
for i1 := 0 to 4 do
 begin
 path := ExtractFilePath(Application.ExeName)+'\fonts\';
 case i1 of
  0: path  := path + 'Arial\';
  1: path  := path + 'Century Gothic\';
  2: path  := path + 'Courier New Cyr\';
  3: path  := path + 'Goudy Old Style\';
  4: path  := path + 'Times New Roman\';
 end;
 for j := 0 to 9 do
  begin
  Data[i1,j] := TBitmap.Create;
  Data[i1,j].LoadFromFile(path + IntToStr(j) + '.bmp');
  end;
 end;
end;
```

Після завантаження еталонних зображень користувач малює на поле розміром 50x70 пікселів цифру, що програма буде розпізнавати.

При натисканні кнопки "розпізнати" вираховуються відстані від розпізнаваного малюнка до кожного з еталонних (відстань за Хемінгом).

```
function Compare(b1,b2:TBitmap):integer;
var i1,j,count:integer;
begin
count := 0;
for i1 := 0 to 49 do
 for j := 0 to 69 do
  if b1.Canvas.Pixels[i1,j] <> b2.Canvas.Pixels[i1,j] then
    inc(count);
Result := count;
end;
```

Знаючи цю відстань R, легко обчислити потенціал, створюваний кожним еталонним малюнком у точці, що відповідає намальованому користувачем зображенню.

Я небагато змінив формулу розрахунку потенціалу, щоб уникнути розподілу на 0 у випадку R=0 і для кращого сприйняття домножив на 1 000 000:

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

```
for i1 := 0 to 4 do
  for j := 0 to 9 do
   begin
   r := Compare(Image1.Picture.Bitmap,Data[i1,j]);
   p[j] := p[j] + 1000000/(1+r*r);
   end;
```

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

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

Нижче наведені основні моменти реалізації пропонованого алгоритму.

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

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

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

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

Дія 2. Формування масиву еталонних зразків символів.

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

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

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

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

Обчислюємо координати границь (описаного прямокутника) образа (еталонного або розпізнаваного) шляхом сканування рядків/стовпців. При цьому тут і при подальшому аналізі зображення передбачається, що символ

намальований чорним кольором (№0 у палітрі квітів) і відповідно всі значущі пікселі мають значення 0.

```
for j1 := 0 to Img.Height - 1 do
// Зверху
  begin
    for i1 := 0 to Img.Width - 1 do
      if Mas[j1][i1] = 0 then
      begin yTop := j1; break; end;
    if yTop = j1 then break;
  end;
  for j1 := Img.Height - 1 downto 0 do
 // Знизу
  begin
    for i1 := 0 to Img.Width - 1 do
      if Mas[j1][i1] = 0 then
      begin yBottom := j1 + 1; break; end;
    if yBottom = j1 + 1 then break;
  end;
  for i1 := 0 to Img.Width - 1 do
// Зліва
  begin
    for j1 := 0 to Img.Height - 1 do
      if Mas[j1][i1] = 0 then begin xLeft := i1; break;
  end;
    if xLeft = i1 then break;
  end;
  for i1 := Img.Width - 1 downto 0 do
// Праворуч
  begin
    for j1 := 0 to Img.Height - 1 do
      if Mas[j1][i1] = 0 then begin xRight := i1 + 1; break; end;
    if xRight = i1 + 1 then break;
  end;
```

Для подальшого аналізу буде потрібно якийсь критерій, по якому буде вироблятися згортка вихідного зображення символу в матрицю 16x16. Таким критерієм був обраний загальний відсоток заповнення – відношення кількості значимих пікселів (з яких складається символ) до загальної кількості пікселів в описаному навколо вихідного зображення прямокутнику.

Даний параметр може впливати на якість розпізнавання, причому якщо він більше 1 для розпізнаваного символу буде відповідати меншу кількість можливих альтернатив, при значенні меншому 1 – навпаки. У нашому випадку коефіцієнт виправлення прийнятий рівним 0,99.

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

Далі розбиваємо прямокутник із зображенням символу на 16x16 осередків шляхом розподілу сторін нового осередку на 2. Запам'ятовуємо відносні координати кожного осередку й приступаємо до заповнення матриці 16x16. Приймаємо як критерій загальний відсоток заповнення.

Якщо в аналізованому осередку відсоток заповнення більше, ніж загальний відсоток – відповідний елемент матриці 16x16 установлюється в 1, у противному випадку – в 0.

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

Дія 3. Розпізнавання мальованих (від руки) символів.

Розпізнавання здійснюємо шляхом порівняння матриці 16x16 розпізнаваного символу з матрицею еталона (шляхом перебору наявних у наявності). Порівняння робимо поелементно за допомогою оператора XOR. Результат – матриця 16x16, що містить одиниці в місцях розбіжностей тест-символу й еталона.

Шляхом підрахунку кількості розбіжностей формуємо вектор, що містить цю інформацію для кожного еталонного символу, і робимо сортування його елементів по зростанню кількості розбіжностей.

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

В багатьох областях техніки використовуються різні автомати й пристрої, що більш-менш вдало вирішують завдання розпізнавання (це й автомат для сортування поштових конвертів по індексу, і різні системи аналізу супутникових знімків, і голосовий виклик вашого мобільника, і багато чого іншого). В описаному вище алгоритмі , є один сильний недогляд: цю програму неможливо навчати, тому що у нього порівняння відбувається тільки з одним набором еталонів. Запропоную алгоритм, яким можна навчати. Особливу пікантність алгоритму надає той факт, що його математичне обґрунтування було запропоновано радянськими математиками на початку 60-х років.

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

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.

4.2 Захист розробленого програмного забезпечення

Дані у програмному забезпеченні я захищаю за допомогою MISTY1. MISTY1 – блоковий алгоритм шифрування, створений для компанії Mitsubishi Electric криптологом Міцуру Мацуї. Назва є абревіатурою Mitsubishi Improved Security Technology. Алгоритм був розроблений в 1995-1996 рр. Відомі також дві модифікації алгоритму MISTY1: MISTY2 і KASUMI

Шифр став переможцем на Європейському конкурсі NESSIE. У результаті аналізу алгоритму експерти зробили вивід, що ніяких серйозних уразливостей даний алгоритм не має (переважно, завдяки вкладеним мережам Фейстеля, що суттєво утрудняє криптоаналіз). У нього високий запас криптостійкості, алгоритм має високу швидкість шифрування й досить ефективний для апаратної реалізації.

Алгоритм був розроблений на основі теорії «підтвердженої безпеки» проти диференціального й лінійного криптоаналізу. Цей алгоритм був спроектований, щоб протистояти криптоатакам, відомим на момент створення.

З моменту публікації MISTY1 було проведено багато досліджень, щоб оцінити його рівень безпеки.

Диференціальний і неможливий диференціальний криптоаналіз високого порядку ефективно застосовується до блокових шифрів з малим ступенем. Найкращі результати для обох варіантів були отримані для 5-рівневого алгоритму MISTY1 без FL функцій.

Саме FL функції й широкобітні AND/OR операції в сильно утрудняють використання диференціального криптоаналізу, що не заважає проведенню в цьому напрямку всі нових досліджень і досягненню усе більш близьких до розв'язку результатів.

Параметри вихідних даних

MISTY1 – це шифр на основі вкладених мереж Фейстеля з варьюємим числом раундів. Рекомендоване використання 8-раундової версії, але може використовуватися будь-яка кількість раундів, кратне 4-м. Розмір блоку вихідного тексту – 64 біта, розмір ключа – 128 біт.

Для роботи алгоритму також попередньо виконується процедура розширення ключа, яка для 8-мі раундів обчислює 1216 бітів ключової інформації з 128-бітного ключа шифрування.

Структура алгоритму

Для задоволення вимогам конкурсу NESSIE, а також для задоволення завдання мультиплатформеності, в алгоритмі MISTY1 використовувалися наступні методи шифрування:

– Логічні операції.

– Арифметичні операції.

– Операції зрушення.

– Таблиці перестановок.

Як говорилося вище, алгоритм MISTY1 заснований на «вкладених» мережах Фейстеля. Спочатку блок вихідного тексту розбивається на два 32-бітних субблоки, після чого виконується r раундів наступних перетворень[1]:

– У кожному непарному раунді обоє субблоки обробляються операцією FL

– Над оброблюваним субблоком виконується операція FO.

– Результат цих операцій накладається логічною операцією «, що виключає або» (XOR) на неопрацьований субблок.

– Субблоки міняються місцями. Після заключного раунду обоє субблоки ще раз обробляються операцією FL.

Операція FL

Оброблюваний 32-бітний субблок розбивається на два 16-бітних фрагмента, до яких застосовуються операції, де:

– L і R – вхідні значення лівого й правого фрагментів відповідно;

– L' і R' – вихідні значення;

– і – фрагменти j-го підключа i-го раунду для функції FL (процедура розширення ключа докладно описана далі);

– і – побітіві логічні операції «і» і «або» відповідно.

Операція FO

Саме ця функція є вкладеною мережею Фейстеля. Тут, як і раніше, виконується розбивка вхідного значення на два 16-бітних фрагмента, що проходять 3 раунду наступних перетворень:

– На лівий фрагмент операцією XOR накладається фрагмент ключа, де k – номер раунду функції FO.

– Лівий фрагмент обробляється операцією FI.

– На лівий фрагмент накладається операцією XOR значення правого фрагмента.

– Фрагменти міняються місцями.

Після третього раунду операції FO на лівий фрагмент накладається операцією XOR додатковий фрагмент ключа.

Операція FI

Дана операція також представляє собою третій рівень вкладеності мережі Фейстеля. На відміну від двох верхніх рівнів, дана мережа є незбалансованою: оброблюваний 16-бітний фрагмент ділиться на дві частини: 9-бітну ліву й 7-бітну праву. Потім виконуються 3 раунду перетворень, що випливають:

– Ліва частина зазнає обробці S-box.9-бітна частина (в 1-м і 3-м раундах) обробляється таблицею S9, а 7-бітна (в 2-м раунді) – таблицею S7. Дані таблиці описані нижче.

– На ліву частину операцією XOR накладається поточне значення правої частини. При цьому, якщо праворуч 7-бітна частина, вона доповнюється нулями ліворуч, а в 9- бітної частини віддаляються ліворуч два біти.

– У другому раунді на ліву частину операцією XOR накладається фрагмент ключа раунду, а на праву – фрагмент. В інших раундах ці дії не виконуються.

– Ліва й права частини міняються місцями.

Для оптимального розв'язку завдання мультиплатформеності, таблиці S7 і S9 алгоритму MISTY1 можуть бути реалізовані як за допомогою обчислень, так і безпосередньо таблицями.

Розширення ключа

Для 8 раундів алгоритму результатом процедури розширення ключа буде наступний набір ключових значень:

– 20 фрагментів ключа (), кожний з яких має розмір по 16 бітів;

– 32 16-бітних фрагмента ();

– 24 7-бітних фрагмента (при k=4, тобто в 4-м раунді функції FO, операція FI не виконується);

– 24 9-бітних фрагмента.

Виконується дане обчислення в такий спосіб:

1. 128-бітний ключ ділиться на 8 фрагментів … по 16 бітів кожний.

2. Формуються значення: у якості використовується результат обробки значення функцією FI, яка в якості ключа (тобто сукупності необхідних 7- і 9-бітного фрагментів) використовує значення(якщо індекс n фрагмента ключа перевищує 8, то замість нього використовується індекс n-8).

Необхідні фрагменти розширеного ключа «набираються» у міру виконання перетворень із відповідних масивів і згідно з відповідними таблицями

16-бітний фрагмент ділиться на 7-бітний фрагмент і 9-бітний .

Розшифрування

Розшифрування проводиться виконанням тих же операцій, що й при зашифруванні, але з наступними змінами:

– фрагменти розширеного ключа використовуються у зворотній послідовності,

– замість операції FL використовується зворотна їй операція – FLI.

Схеми виконання функції FLI і процедури розшифрування наведено на малюнках 6 і 7 відповідно:

Методи аналізу

Як говорилося на початку розділу, диференціальний і неможливий диференціальний аналізи виявилися ефективні лише до версій шифру з меншою кількістю раундів і без операції FL [2][3]. Проте, на даний момент цей напрямок аналізу, особливе використання слабких ключів, найбільше перспективно, тому що наближене до реальних можливих допущень при використанні алгоритму.

Так само, ученим з Японії був проведений інтегральний аналіз повного алгоритму, використовуючи відкритих текстів зі складністю обчислення, рівної [4].

Лінійний аналіз дав результати тільки для 7-раундової версії шифру, і також без операції FL[5].

Так як MISTY1 створювався, у тому числі, з розрахунку на апаратну реалізацію, має сенс диференціальний аналіз, заснований на використанні атаки по помилках обчислень, що в цьому випадку наближене до реальності.

Висновок

Таким чином, була докладно описана структура алгоритму шифрування MISTY1 і розглянуті методи його аналізу, найбільш прагматичні напрямки дослідження. Далі має бути створення програмної реалізації для більш детального розгляду алгоритму й набір статистичних даних для повного дослідження й пошуку оптимального підходу до аналізу MISTY1.

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

На рисунку 5.1 зображено головне вікно програми. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню: Система; Налаштування; Довідка.

– Вікно розпізнавання: Джерела відео даних; Список знайдених камер; Встановити; Сканувати; Налаштувати.

– Функції програми: Розпізнавання; Параметри; Стан камер; Вибір відео режиму; Метод розпізнавання; Налаштування; Про програму.

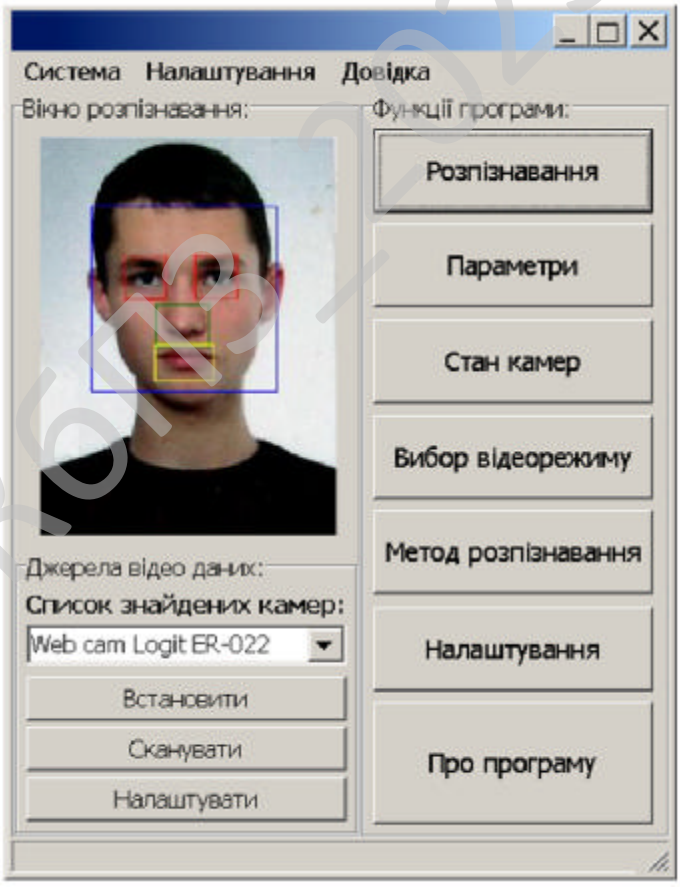

Рисунок 5.1 – Головне вікно програми

На рисунку 5.2 зображено форму авторського права, де відображені дані розробника. Було обрано Shareware умову розповсюдження.

Під умовно-безплатним програмним забезпеченням можна розуміти спосіб або метод розповсюдження комерційного ПЗ на ринку (тобто на шляху до кінцевого користувача), при якому випробувачеві пропонується обмежена за можливостями (неповнофункціональна або демонстраційна версія), терміном дії (тріал версія) або версія з вбудованим набридливим нагадуванням про необхідність оплати використання програми.

В угоді про використання (ліцензії для кінцевого користувача, EULA) також може бути обумовлена заборона на комерційне або професійне (не тестове) її використання. Основний принцип умовно-безплатного ПЗ – «спробуй, перш ніж купити» (try before you buy). ПЗ що поширюється як умовно-безплатний, надається користувачам безоплатно.

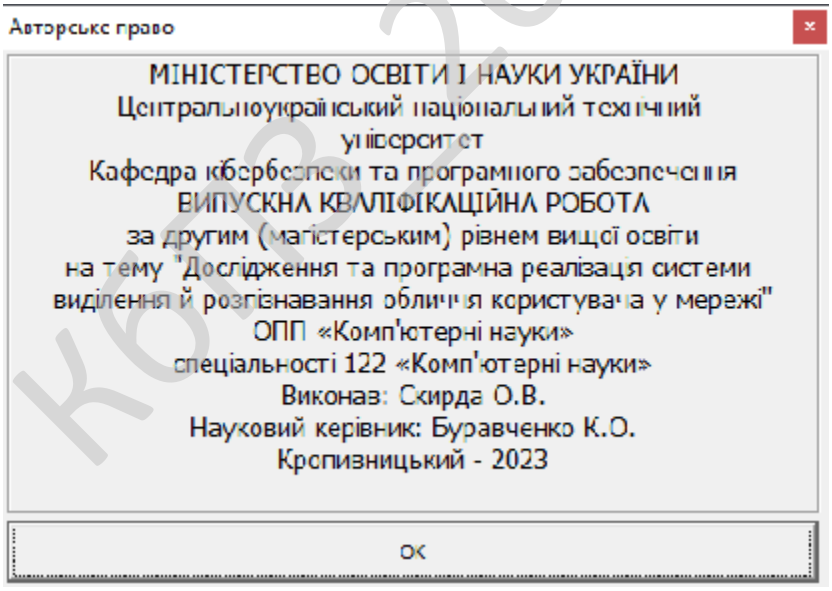

Рисунок 5.2 – Довідка автора

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи виділення й розпізнавання обличчя користувача у мережі.

Метою розробки є дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

Об'єктом дослідження є процес виділення й розпізнавання обличчя користувача у мережі.

Предметом дослідження є методи виділення й розпізнавання обличчя користувача у мережі.

Методи дослідження базуються на методах штучного інтелекту, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод виділення й розпізнавання обличчя користувача у мережі.

– Розроблено вітчизняний продукт виділення й розпізнавання обличчя користувача у мережі, 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 48 днів (два місяці).

В магістерській роботі проведене дослідження та виконана програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

Розроблене програмне забезпечення має достатню надійність і задовольняє усім поставленим умовам, а саме:

а) невеликий розмір та системні потреби;

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

в) 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	60
3. Запланований термін розробки, днів	Fpq	48 (2 місяці)
4. Група задачі підсистеми управління (1-6)	–	1
5. Ступінь новизни задачі (А, Б, В, Г)	–	Б
6. Складність алгоритму (1, 2, 3)	–	2
7. Кількість макетів вхідної інформації	–	3

Продовження таблиці 7.1

1	2	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 ПО для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн	–	60000
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 є найбільш тривалою і трудомісткою, що 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 А – коефіцієнт Боема, А=2,45;

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

B – показник ступеня, що визначається співвідношенням

$$B = 1{,}01 + 0{,}001\sum W_i \tag{7.2}$$

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.:

$$T_{уточн}=T_{ном}\, ПВ_j, \tag{7.3}$$

де $ПВ$j – добуток сімнадцяти додаткових коефіцієнтів, приведених в МВ додатку 3.

$$T_{уточн}= 6{,}78\cdot(0{,}88\cdot0{,}93\cdot0{,}88\cdot0{,}91\cdot0{,}95\cdot1\cdot1\cdot0{,}87\cdot1{,}22\cdot1{,}16\cdot1{,}1\cdot1{,}1\cdot$$
$$\cdot1{,}12\cdot1{,}1\cdot1\cdot1{,}1\cdot1{,}1)=9{,}37 \text{ люд-міс.}$$

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:

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

де C – визначений емпірично коефіцієнт, запропонований авторами методики, (МВ, додаток 4); S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПО згідно встановленим вимогам. Вибираємо в межах (25…350)%

$$T_{РП}=0{,}3\cdot2{,}66\cdot9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)}\cdot130=265 \text{ люд/день}$$

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	265	Ф 7.1-7.4
Впровадження	13	Д13
Всього	306	–

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,

$$Ч = \frac{306 \cdot 1}{48\text{-}5} = 7{,}1 \text{ ставки}$$

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	11	990	16,5
Монітор	60	11	660	11
Клавіатура	30	11	330	5,5
Маніпулятор «мишка»	30	11	330	5,5
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	2	240	4
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор– маршрутизатор	30	2	60	1
Кабельні господарства ЛВС на 1 м. п.	2,5	200	500	8,33
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	55,49

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{55 \cdot 2}{1,2} = 92 \text{ год}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$Ч_{ел}$=92/(48·8)=0,24 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 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	0,8	0,2
	Налаштування і конфігурування базової станції безпровідного зв’язку (CMTS)	0,2	
	Розробка та впровадження проектів з організації зв’язку між віддаленими об’єктами, ЛОМ	0,2	
	Забезпечення цілодобової роботи зв’язку клієнтів до мережі Інтернет	0,4	
Всього		1,6	

Продовження таблиці 7.4

Посада	Вид роботи	Час	Кількість штатних одиниць
Продакт-менеджер	Презентації нової продукції, пошук каналів збуту	2	0,5
	Підтримка постійних клієнтів	1	
	Оформлення договорів, ведення тендерів	0,5	
	Контроль взаєморозрахунків з постачальниками	0,5	
Всього		4	
Дизайнер WEB	Розробка концепції оформлення та інтерфейсу сайту, оптимізація дизайну існуючих, проектує їх структуру та навігацію	0,5	0,2
	Створення графічних і стилістичних елементів сайту	0,5	
	Оформлення банерів і промо-сторінок	0,3	
	Розміщення графіки і контенту на Інтернет сторінках	0,3	
Всього		1,6	
Інженер верстальник	Розробка та верстка макетів рекламної продукції та технічної документації	1	0,2
	Верстка друкованих видань	0,2	
	Додрукова підготовка макетів	0,2	
	Розміщення графіки і контенту на Інтернет сторінках	0,2	
Всього		1,6	

Складемо штатний розклад виконавців у таблицю 7.5.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньо-місячний оклад, грн.	Всього за період роз-робки, грн.
Керівник (IT-менеджер)	1	17000	34000
Продакт-менеджер	0,5	14965	14965
Інженер-програміст	7,1	16200	230040
Інженер-електронщик	0,24	10000	4800
Інженер-системотехнік	0,2	10000	4000
Адміністратор мережі	0,2	11000	4400
Системний програміст	0,2	10000	4000
Дизайнер WEB	0,2	11000	4400
Інженер-верстальник	0,2	11000	4400
Бухгалтер-економіст	0,2	10000	4000
Всього за період розробки	$R_{сп}$=10,04	-	$Ф_{роб}$=309005

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.

$$З_{сд} = \frac{309005}{10,4 \cdot 48} = 619 \; грн.$$

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 8 робочих місць.

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

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

Згідно даних інтернет ресурсу DOM.RIA (https://dom.ria.com) ціна одного квадратного метра площі, 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 Цм – 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 фірми Комп'ютерторг за 20.10.23 – джерело http://computorg.ua/ price.html

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		11771
Системний блок		7771
Процесор	AMD Ryzen 3 2200G PRO AM4, 4 ядра, потоки, 3.5 GHz, 3.7 GHz, TDP – 65 Вт 14 nm, Radeon Vega 8, Zen, Tray	-
Системна плата	ASUS PRIME A520M-A II/CSM, AM4 DDR4, 128 ГБ, 4800 MHz, LAN – 1 Гбіт/с D-Sub (VGA), DisplayPort, HDMI, 1 x M.2 2280, 4 x Sata 6.0 Gb/s, Micro-ATX	-
Жорсткий диск	240 GB SSD M.2 Samsung Evo	-
Оперативна пам’ять	DDR4 8GB 2400 MHz Patric (PSD48G240081) DDR4, 8 ГБ, В наборі – 1	-
Система охолодження	Vinga CL3011	-
Корпус	Vinga CS104B-500W	-
Відеоадаптер	вбудований	-
інше	Клавіатура, мишка	Подаруно
Монітор	Монітор BenQ GL2450HM Black	2600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USI cable	5500
Сканер	Epson Perfection V37	2800
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965
Пристрій безперебійног живлення	Powercom BNT-600AP USB	1400

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	11771	9416,8	103584,8
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	1	2800	280	3080
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	125216,3

Таблиця 7.8 – Вартість основних фондів та амортизаційні відрахування розробника

Групи та види основних фондів	Балансова вартість, грн.	Амортизація	
		Норма, %	Відрахування, грн.
1	2	3	4
Група 3			
1. Будівлі	1280000	-	-
2. Передавальні пристрої	128000	-	-
Всього по групі	1408000	5	70400

Продовження таблиці 7.8

1	2	3	4
Група 4			
3. Обчислювальна техніка	125216	-	-
Всього по групі	125216	50	62608
Група 5,6			
4. Вимірювальні пристрої	5190	-	-
5. Транспортні засоби	143000	-	
6. Господарський інвентар	28000	-	-
Всього по групі	176190	20	35238
7. Нематеріальні активи	60000	10	6000
Разом	$К_р$= 1769406		$А_р$= 174246

Примітка: вартість автомобіля взята по даним автобазару «Авто-РІА», джерело https://auto.ria.com/uk/auto_dacia_ logan_30226404.html, 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.

$$З_о = 619 \cdot 306/60 = 3158 \text{ грн}$$

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, %

$$З_д = 3158 \cdot 10 \cdot 0{,}01 = 316 \text{ грн}$$

Відрахування на соціальні потреби за нормативом Нс=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, %

$$С_{оц} = 0{,}01 \cdot 22(3158+316) = 764 \text{ грн}$$

Визначимо загальногосподарські витрати (електроенергію, ремонт і утримання приміщень і т.д) за нормативом *Нг*=15% від основної зарплати

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

де $Н_г$ – загальногосподарські витрати, %

$$Г_{осп} = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

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, картриджів, тонеру, грн., $N_е$ – кількість екземплярів програм, шт.

Згідно прийнятих норм на підприємстві $n_{вит}$ приймаємо 0,4 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=200 грн., визначаємо вартість паперу за період розробки:

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

$$З_{М1} = 200 \cdot 0{,}4 = 80 \text{ грн.}$$

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 10):

$$З_{М2} = \sum Ц_д \,, \tag{7.17}$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 23,6 грн./шт., DVD-R box – 49 грн./шт.

$$З_{М2} = 49 \cdot 10 = 490 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \tag{7.18}$$

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.

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

$$З_М = (80 + 490 + 1702)/60 = 38 \text{ грн.}$$

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

$$О_п = З_о \cdot Н_п \cdot 0{,}01, \tag{7.19}$$

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

$$О_п = 3158 \cdot 15 \cdot 0{,}01 = 474 \text{ грн}$$

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 60 прим.)

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \tag{7.20}$$

де $А_р$ – загальна річна сума амортизаційних відрахувань, грн.

$$А_м = 174246 \cdot 2/(60 \cdot 12) = 484 \text{ грн}$$

Повна собівартість ПЗ визначається як сума витрат за попередніми статтями калькуляції

$$С_п = З_о + З_д + С_{оц} + Г_{осп} + З_м + О_п + А_м \,. \tag{7.21}$$

$$С_п = 3158+316+764+474+38+474+484 = 5708 \text{ грн.}$$

Визначимо плановий прибуток за рівнем рентабельності (Рп) 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 $P_с$ – рівень рентабельності, %

$$П_р = 0{,}01 \cdot 50 \cdot 5708 = 2854 \text{ грн.}$$

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	$З_о$	3158
2. Додаткова зарплата виконавців	$З_д$	316
3. Відрахування на соціальні потреби	$С_{оц}$	764
4. Загальногосподарські витрати	$Г_{осп}$	474
5. Витрати на матеріали	$З_М$	38
6. Освоєння нових операційних систем, мов програмування	$О_п$	474
7. Амортизація основних фондів	$А_м$	484
8. Повна собівартість програмного забезпечення	$С_п$	5708
9. Плановий прибуток	$П_р$	2854
10.Ціна підприємства $Ц_п=С_п+П_р$	$Ц_п$	8562
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{ов} \cdot Ц_п$	$ПДВ$	1712,4
12.Відпускна ціна програмної продукції Ц = Цп + ПДВ	$Ц$	10274,4

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	–	10274
Всього капітальних витрат	–	10274

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	Позна-чення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	42944	16104
2. Витрати на електроенергію	$З_{ел}$	243	91
3. Витрати на амортизацію	$З_{ам}$	0	5137
Всього витрат за рік	$І$	43187	21332

Витрати на технічне обслуговування:

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

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 320 годин на рік до 120 годин на рік, тому витрати на технічне обслуговування зменшилися з

$$З_{р\ баз} = 320 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 42944\ грн.$$

до

$$З_{р\ нов} = 120 \cdot 100 \cdot 1{,}1 \cdot 1{,}22 = 16104\ грн.$$

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

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

Зел баз=0,2·320·3,8=243 грн

Зел нов=0,2·120·3,8=91 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	50	–	10274	–	5137
Всього відрахувань	-	–	10274	–	5137

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.

$$Ев = (8562 - 5708) \cdot 60 - (0,05 \cdot 1408000 + 0,5 \cdot 125216 +$$
$$+0,2 \cdot 176190 + 0,1 \cdot 60000) \cdot 2/12 = 142200 \text{ грн.}$$

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 без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{1769406}{(8562 - 5708) \cdot 60 \cdot 12 / 2} = 1,72 \text{ років}$$

Визначимо величину економічного ефекту у користувача програмної продукції за формулою:

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

де $I_б$, $I_н$ – величина експлуатаційних витрат за базовим и новим варіантом відповідно, $K_б$, $K_н$ – об'єм капітальних вкладень за варіантами, що порівнюються

$$E_{сп} = (43187\text{-}21332)\text{-}0,5 \cdot 10274 = 16718 \text{ грн.}$$

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

$$T_{сп} = \frac{K_н - K_б}{I_б - I_н} \quad (7.28)$$

$$T_{сп} = \frac{10274}{43187 - 21332} = 0,47 \ року$$

Показники економічної ефективності програмної продукції зводимо до таблиці 7.13.

Таблиця 7.13 – Показники економічної ефективності програмної продукції

Найменування показників	Одиниця виміру	Величина
1. Кількість екземплярів програми	Прим.	60
2. Повна собівартість розробленої програми	Грн.	5708
3. Ціна розробленої програми	Грн.	8562
4. Плановий прибуток від реалізації розробленої програми	Грн.	2854
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1769406
7. Загальний прибуток від реалізації програмної продукції	Грн.	171240
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	142200
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років	1,72
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	10274
11.Величина економічного ефекту у користувача програмної продукції	Грн.	16718
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,47

7.9 Висновки

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

8 ЗАХОДИ З ОХОРОНИ ПРАЦІ ТА ТЕХНІКИ БЕЗПЕКИ

8.1 Вступ

Законом України “Про охорону праці” [1] регламентуються загальні положення державної політики в галузі охорони праці, а конкретизуються ці положення нормативно-правовими актами про охорону праці, зокрема Наказом Міністерства соціальної політики України 14.02.2018 № 207 [4], який зареєстровано в Міністерстві юстиції України 25 квітня 2018 р. за №508/31960 «Про затвердження Вимог щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями», НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин», та ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2].

Програмісти у процесі роботи мають негативний вплив на органи зору, а також мають значну розумову напругою і нервово-емоційне навантаження. Руки (суглоби пальців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [2]. До шкідливих факторів, які впливають на робітників галузі інформаційних технологій (ІТ) спеціалісти відносять високочастотні електромагнітні коливання (випромінювання) роботи апаратної частини ЕОМ та виділення шкідливих газів.

Ці шкідливі фактори можуть привести до професійних захворювань.

При розгляді шкідливих чинників роботи програмістів та інших спеціалістів ІТ будемо керуватись наступними нормативно-правовими актами: «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» ДСанПіН 3.3.2-007-98 [2], та «Правила охорони праці під час експлуатації електронно-обчислювальних машин» НПАОП 0.00-1.28-10,

Умови праці програміста включають наступні фактори:

– параметри повітряного середовища в приміщенні;

– вентиляція приміщення;

– освітлення приміщення;

– параметри повітряного середовища в приміщенні, тощо.

Щоб запропонувати заходи щодо зменшення впливу комп'ютера на організм програміста визначемо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Це важливе питання, адже робота з комп'ютером може впливати на здров'я людини. За ДСанПіН 3.3.2.007-98 шкідливі і небезпечні фактори при роботі з комп'ютером можуть бути такі:

– Електромагнітне випромінювання – це випромінювання, яке створюється комп'ютером і його периферійними пристроями, такими як монітор, пртер, сканер тощо. Це випромінювання може викликати головний біль, запаморчення, розлади сну, зниження імунітету та інші негативні ефекти.

– Електростатичне поле – це поле, яке утворюється внаслідок накопичення електричних зарядів на поверхні комп'ютера і його частин. Це поле може спричиняти сухість шкіри, свербіж, подразнення очей, алергічні реакції та інші проблеми.

– Нервово-емоційне напруження – це напруження, яке виникає внаслідок трвлої концентрації уваги, високої вимогливості до результату роботи, нестабільності програмного забезпечення, конфліктних ситуацій тощо. Це напруження може призводити до стресу, депресії, роздратування, погіршення пам'яті та інших порушень.

– Навантаження на органи зору – це навантаження, яке виникає внаслідок тривалого перебування перед монітором, низької якості зображення, недостнго

освітлення приміщення, поганої ергономіки робочого місця тощо. Це навантаження може спричиняти зниження гостроти зору, появу блимавок і кругів перед очима, синдром сухого ока та інші захворювання.

– Дрібні стереостатичні рухи кінцівок – це рухи, які пов'язані з керуванням клавіатурою і мишею. Ці рухи можуть призводити до перевантаження і запалення суглобів і сухожиль рук і пальців, а також до тендовагінітів і синдрому зап'ясткового каналу.

Таким чином, робота з комп'ютером не така безпечна, як може здатися на перший погляд. Тому дуже важливо дотримуватися правил охорони праці і гігієни при роботі з комп'ютером.

8.3 Аналіз санітарно-гігієнічних умов праці на робочому місці програміста

Розглянемо умови праці у приміщенні, в якому працюють програмісти. Геометричні розміри приміщення наведено у таблиці 8.1.

Таблиця 8.1 – Розміри приміщення

Найменування	Значення, м
Ширина	5,4
Довжина	6
Висота	3

Таблиця 8.2 – Площа та обсяг приміщення, на одного працюючого*

Геометрична характеристика	Одиниця виміру	Нормативне значення*	Фактичне значення
Площа, S	$м^2$	не менше 6.0	6,5
Обсяг, V	$м^3$	не менше 20.0	19,4

* Згідно ДСанПіН 3.3.2.007-98 (Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин) [2].

У зазначеному приміщенні працює 5 осіб. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа приміщення у розрахунку на одно робоче місце програміста відповідають нормативним вимогам (Наказу Міністерства соціальної політики України № 207, від 14.02.2018 «Про затвердження Вимог щодо безпеки та захисту здоров'я працівників під час роботи з екранними пристроями», ДСанПіН 3.3.2-007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» [2], а об'єм – НПАОП 0.00-1.28-10 «Правила охорони праці під час експлуатації електронно-обчислювальних машин»).

Температура повітря в приміщенні визначається впливом температури зовнішнього повітря і тепловою енергією, яка виділяється всередині приміщення. Джерелами виділення теплоти в даному приміщенні є електроустаткування, освітлювальні прилади, а також люди. У світлий час доби джерелом надлишкового тепла є сонячна радіація. Згідно Постанови Головного державного санітарного лікаря України [5], робота, яка виконується в даному приміщенні, відноситься до категорії Іа. В цьому випадку людина витрачає енергії до 120 ккал у годину. Вологість повітря у приміщенні визначається впливом багатьох факторів, серед яких: вологість атмосферного повітря, виділення вологи людьми (при диханні та випарами з поверхні шкіри).

Мікроклімат повітряного середовища в приміщенні характеризується запиленістю та загазованістю повітря. Мікроклімат приміщення визначається діючим на організм людини поєднанням, вологості, температури, швидкості руху повітря та інтенсивності теплового випромінювання. Аналіз мікроклімату складається з визначення зазначених вище факторів і порівняння результатів із встановленими нормами.

У таблиці 8.3 наведено оптимальні та фактичні значення параметрів мікроклімату як для категорії ваги робіт Іа, так і розглянутого приміщення. У

приміщеннях, де встановлено ЕОМ, рекомендується застосування тільки оптимальних значень показників мікроклімату.

Таблиця 8.3 – Оптимальні і фактичні значення параметрів мікроклімату

Пора року	Оптимальні для Іа			Фактичні		
	Температура °С	Воло-гість,%	Швидкість повітря, м/	Температура °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23,5	40-60	0,1
Тепла	23-25	50-70	0,1	23-25	52-70	0,12

Проведений аналіз показує, що показники мікроклімату в приміщенні відповідають установленим нормам. Штучне опалення застосовується у холодний період року.

В літню пору застосовується кондиціонер.

Для боротьби з пилом робляться регулярні провітрювання та вологі прибирання приміщенні.

У приміщенні знаходяться наступні джерела шуму: принтер HP Laser 107a, електродвигуни вентиляторів ЕОМ.

Одним з найважливіших факторів, які впливають на ефективність трудової діяльності людини, та попереджають травматизм і професійні захворювання програмістів є освітлення на робочому місці.

Працю працівника, який постійно працює за комп’ютером, згідно ДБН В.2.5 – 28 – 2006 р можна віднести до роботи з малою точністю (найменший розмір об'єкта розрізнення від 1 до 5 мм) V-го розряду зорової роботи, з великою контрастністю об'єкта розрізнення (символів на екрані дисплея), з темним тлом (під розряд зорової роботи В). Приміщення можна віднести до 1-ої групи приміщень, у яких проводиться розрізнення об'єктів зорової роботи при фіксованому напрямку лінії зору того, що працює на робочу поверхню. Для такого типу приміщень і розряду зорової роботи нормоване значення коефіцієнта природної освітленості (КПО) робочої поверхні (при поєднаному, спільному

освітленні), повинен становити не більше 1,5%, освітленість при штучному висвітленні повинна становити 300 лк. Крім того все поле зору повинне бути освітлено достатньо рівномірно – ця основна гігієнічна вимога. Так як яскраве світло на ділянці периферійного зору значно збільшує напруженість очей і, як наслідок, призводить до їх швидкої стомлюваності, ступінь освітлення приміщення і яскравість екрану комп'ютера повинні бути приблизно однаковими.

8.4 Розробка заходів з умов поліпшення охорони праці

Згідно аналізу умов праці в розглянутому приміщенні, ми одержали наступні результати:

– розмірі приміщення, у розрахунку на одному працюючого, відповідають нормативам;

– мікроклімат відповідає нормативному значенню;

– акустичні умови роботи не перевищують нормативних значеннь;

Таким чином можна припустити, що основною причиною можнивого зниження працездатності програміста є психофізіологічний фактор, тому основна пропозиція буде така: дотримання позитивної психологічної атмосфери в колективі та регламентованого режиму праці та відпочинку, організація робочого місця з урахуванням ергономічних вимог.

Рекомендовані заходи: регулярні періодичні наочні огляди персоналом шляхів для евакуації людей із приміщення, відповідно до плану евакуації (який повинен розташовуватись на видному місці у приміщенні), включення до колективного договору мінімально можливого вмісту аптечок з обов’язковою наявністю масок-клапанів, або іншого спорядження для штучного дихання. Регулярна періодична перевірка параметрів заземлення та занулення (вимірювання опору).

8.5 Розрахункова частина

Завдання: розрахувати *штучне освітлення робочого приміщення.*

Початкові дані: ширина *робочого* приміщення: 5,4 м.; довжина –6 м.; висота – 3 м.

Розрахунок штучного освітлення проводемо за методом коефіцієнта використання світлового потоку.

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 (у нашому випадку *S*=5,4 х 6 = 32,4 $м^2$);

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

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);*

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% *[6]*.

Обчислимо індекс приміщення за формулою:

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

де: S – площа приміщення, $S = 32,4$ м^2;

h – розрахункова висота підвісу, $h = 3$ *м;*

A – ширина приміщення, А = 5,4 м;

B – довжина приміщення, В = *6* м.

Підставимо всі значення у формулу та визначемо індекса приміщення: *i=0,57.*

Знаючи індекс приміщення, за знаходимо *n = 0.29 (з табличних даних коефіцієнтів використання світлового потоку (n) світильників відповідного типу).* Підставимо всі значення у формулу, визначемо світловий потік: *F=34865 Лм.*

Для штучного освітлення приміщення використовуються *LED панель MAXUS ASSISTANCE PRO 80W 5000K WHITE (M1052480531)*, світловий потік яких $F_л = 8000$ Лм.

Число світильників визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік одного *світильника*.

$$N= 34865/ 8000=4,36 \text{ шт.}$$

Приймаємо необхідну кількість світильників 5 шт.

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 приміщення, призначеного для праці програмістів, проведено розгляд небезпечних та шкідливих факторів, що негативно впливають на програмістів під час роботи. Виконано розрахунок захисного штучного освітлення. Розроблено заходи з охорони праці.

Список використаних джерел інформації

1. 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

2. 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

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

4. 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

5. 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

6. 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: 16.06.2023).

7. Методичні рекомендації до виконання розділу «Заходи з охорони праці та техніки безпеки» у магістерській дисертації / Л.Д. Третякова; М-во освіти і науки України, Національний технічний університет україни «Київський політехнічний інститут» – Київ, КПІ, 2014. – 26 с. – Режим доступу до ресурсу: http://surl.li/dhulo (дата звернення: 16.06.2023).

9 ОСНОВНІ ВИСНОВКИ

Програмне забезпечення, створене в результаті виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти, призначено для системи виділення й розпізнавання обличчя користувача у мережі.

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 виділення й розпізнавання обличчя користувача у мережі.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем виділення й розпізнавання обличчя користувача у мережі.

– Досліджена система виділення й розпізнавання обличчя користувача у мережі.

– На основі отриманих результатів досліджень створена програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання виділення й розпізнавання обличчя користувача у мережі.

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. 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 MISTY1.

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 16718 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,47 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Скирда О.В. Дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.
2. Aho A.V., Hopcroft J.E., Ullman J.D. Data Structures and Algorithms. – Pearson, 2001. – 620 с.
3. Brink H., Richards J., Fetherolf M. Real-World Machine Learning. – Manning, 2016. – 474 p.
4. Cormen T.H., Leiserson C.E., Rivest R.L., Stein C. Introduction to Algorithms, 3rd Edition (The MIT Press) 3rd Edition – The MIT Press, 2019. – 1292 p.
5. Fenner M. Machine Learning with Python for Everyone (Addison-Wesley Data & Analytics Series) 1st Edition, Kindle Edition. – Addison-Wesley Professional, 2019. – 586 p.
6. Foreman J.W. Data Smart: Using Data Science to Transform Information into Insight 1st Edition. – Wiley, 2013. – 432 p.
7. Hurbans R. Grokking Artificial Intelligence Algorithms. – Manning, 2020. – 631 p.
8. Gusfield D. Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology 1st Edition. – Cambridge University Press, 2008. – 556 p.
9. Kotu V., Deshpande B. Data Science: Concepts and Practice. – Elsevier Science, 2018. – 953 p.
10. Knowledge Base A Complete Guide – 2021 Edition // The Art of Service – Knowledge Base Publishing, 2020. – 306 p.
11. Knuth D. The Art of Computer Programming, Vol. 1: Fundamental Algorithms, 3rd Edition 3rd Edition. – Addison-Wesley Professional, 2019. – 672 p.

12. Mattmann C. Machine Learning with TensorFlow, Second Edition. – Manning, 2020. – 1124 p.

13. Mueller J.P., Massaron L. Machine Learning For Dummies. – Wiley, 2016. – 714 p.

14. Teofili T. Deep Learning for Search. – Manning, 2019. – 695 p.

15. Rungta K. TensorFlow in 1 Day: Make your own Neural Network. – Publishdrive, 2019. – 587 p.

16. Weidman S. Deep Learning from Scratch: Building with Python from First Principles. – O'Reilly. 2019. – 252 p.

17. Rajasekaran S., Vijayalakshmi Pai G.A. Neural networks, fuzzy logic, and genetic algorithms: synthesis and applications (with cd-rom) Kindle Edition. – PHI, 2013. – 628 p.

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

19. Smirnov, O., Neskorodieva, T., Fedorov, E., Rudakov, K., Neskorodieva, A. «Method Detection Audit Data Anomalies on Basis Restricted Cauchy Machine» CEUR Workshop Proceedings, Volume 3187, 2022, pp. 1-12.

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

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

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

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

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

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

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

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

28. Smirnov O., Kuznetsov A., Pushkar'ov A., Serhiienko R., Babenko V., Kuznetsova T., «Representation of Cascade Codes in the Frequency Domain». In: Radivilova T., Ageyev D., Kryvinska N. (eds) Data-Centric Business and Applications. Lecture Notes on Data Engineering and Communications Technologies, vol 48. Springer, Cham. 2021. pp 557-587.

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

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

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

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

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

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

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

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

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

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

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

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

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

42. Аль-Мудхафар Акіл Абдулхуссейн М., Смірнова Т.В., Буравченко К.О., Смірнов О.А. «Метод оцінки та підвищення користувальницького досвіду абонентів в програмно-конфігурованих мережах на основі використання машинного навчання». Сучасні інформаційні системи, 2023, том 7, № 2, С. 49-56.

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

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

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

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

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

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

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

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

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи виділення й розпізнавання обличчя користувача у мережі.

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. № 37-13 від 04.08.2023 року).

3 Мета та призначення розробки

Метою випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти є дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі.

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 виділення й розпізнавання обличчя користувача у мережі;

– 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.

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 2023 року.

8 Вимоги щодо охорони праці

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

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 – 101 аркуш.

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 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 21.12.2023 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

_____ Буравченко К.О.

Дослідження та програмна реалізація системи виділення й розпізнавання обличчя користувача у мережі

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 59

Літера: РП

Кропивницький – 2023 року

Файл UAddSample.pas – обробка зображення для виділення та ідентифікації обличчя користувача у мережі

```
unit UAddSample;
// один з модулів виділення й розпізнавання обличчя користувача у мережі,
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 хвилин в однієї й тої ж величини
// значить пійманий локальний мінімум

interface

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

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

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

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

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

  implementation

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


// щось для роботи з картинками без камери
```

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



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

const MinBPixelsPercent =10;

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



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

        Canvas.MoveTo(Round(left),Round(top));
        Canvas.LineTo(Round(right),Round(top));
        Canvas.LineTo(Round(right),Round(buttom));
        Canvas.LineTo(Round(left),Round(buttom));
        Canvas.LineTo(Round(left),Round(top));
```

```
//        Canvas.MoveTo(Round(x - 10),Round(y - 10));

        end;
    end;


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


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

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

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

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


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

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

```
      //tmp:=(2*Uxy)/( Ux-Uy+C);
      end;



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

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


    end;

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

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


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

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

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


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

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

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



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

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

    end;

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

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

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

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

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


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


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

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


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

    if debug then
```

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

      end;


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

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

      // Алгоритм зменшення зображення  - всі параметри плаваючі

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

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


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


    for iy := 0 to IconSize - 1 do
      begin

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

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


        end;
      end;

    // ну от і все - іконка готова - тепер або розпізнаємо її або
```

```
    // додаємо в БД.
    if debug then
    begin
    for y := 0 to IconSize - 1 do
    begin
      for x := 0 to IconSize - 1 do
         write(F,BitmapArr3[y,x]);
      writeln(F,' ');
    end;
        writeln(F,'--------------');
    end;

end;

// додавання нового семпла - перетворення його в іконку, додавання в масив
procedure TAddSampleForm.AddSampleButtonClick(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  Icon: TByteArr;
  y: Cardinal;
  IconType: Cardinal;

begin
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);

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

    IconType:=IconTypeSet.ItemIndex;
{  if RadioTSample.Checked then
    IconType := Icon
  else
    IconType := IconNot;
 }
  //IconTypes:=IconTypes+1;
  ConvertBitmapToMonoChrome(QP,BitmapArr);
  imFeatures(BitmapArr,1);
  AddSample(BitmapArr,IconType); // реально використовується BitmapArr3

  //Close;
end;

//подія - натискання "Зберегти"
procedure TAddSampleForm.SaveSampleButtonClick(Sender: TObject);
begin
  if SampleSaveDialog.Execute then
  begin
    SampleImage.Picture.Bitmap.SaveToFile(SampleSaveDialog.FileName);
  end;
end;
// читання іконки з файлу
function ReadIcon:TByteArr;
var
  x,y: Integer;
  Arr1:TByteArr;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
  buff:string;
  buff2:char;
  cnt:integer;

begin
  SamePixels := 0;
  Width := IconSize;
  Height := IconSize;
```

```
  SetLength(Arr1,IconSize);
  for y := 0 to IconSize - 1 do
    SetLength(Arr1[y],IconSize);

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

  cnt:=0;
  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        read(sample_db,buff2);
        Arr1[x,y]:=strtoint(buff2);
            if Arr1[x,y] = 1 then
            begin

            xVector[cnt] := 2;
            xInputVector[cnt] := 0;
            end
            else
            begin
            xVector[cnt] := -1;
            xInputVector[cnt] := 1;
            end;
          cnt:=cnt+1;
        //write(F,Arr1[x,y]);
      end;
      //WRITELN(f);
      readln(sample_db,buff);
    end;

  //NeuralNetHopf1.AddPattern(xVector);
  result:=Arr1;

end;


// подія відкриття форми, завантаження семплів з файла, ініціалізація й т.п.
procedure TAddSampleForm.FormShow(Sender: TObject);

  var i1,i,j,jj:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff:string;
    buff_file:textfile;
    max_icon_type:integer;

begin
  AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);


  matrix_size.text:=inttostr(IconSize);
  matrix_size.text:=inttostr(IconSize);



  if FileExists(sample_db_file_name) then
    begin
      assignfile(sample_db,sample_db_file_name);
      assignfile(n_debug,neuro_debug_file_name);
      rewrite(n_debug);
      reset(sample_db);

      //IconTypes:=2;
```

```
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconSize:=strtoint(buff);matrix_size.Text:=buff;
      readln(sample_db,buff); IconTypes:=strtoint(buff);//eIconTypes.Text:=buff;
      readln(sample_db,buff);
SetLength(IconArr,strtoint(buff)+1);NumIcons.Text:=buff;
      // вхідний вектор Нейрона мережа Хопфілда - нейронів стільки ж , скільки й
точок в іконці
      SetLength(xVector, IconSize * IconSize);
      // вихідний вектор багатошарової - дорівнює числу образів
      //IconTypes:=3;
      SetLength(xOutputVector, IconTypes);

      // вхідний вектор багатошарової - дорівнює числу нейронів
      SetLength(xInputVector, IconSize * IconSize);


      //  SetLength(xInputVector, 5);
      //  SetLength(xOutputVector, 1);

      // настроювання нейроної мережі Хопфілда

      AddSampleForm.NeuralNetHopf1.LayerCount:=1;
      AddSampleForm.NeuralNetHopf1.InputNeuronCount:=IconSize * IconSize;

      //NeuralNetExtended1.InputFieldCount:=IconSize * IconSize;
      {
      NeuralNetExtended1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNetExtended1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNetExtended1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      }



      // настроювання багатошарової
      //NeuralNetBP1.LayerCount:=2;
      //NeuralNetBP1.LayersBP[0].NeuronCount:=IconSize * IconSize;
      //NeuralNetBP1.LayersBP[1].NeuronCount:=IconSize * IconSize;
      //NeuralNetBP1.LayersBP[1].NeuronCount:=IconTypes;
      //NeuralNetBP1.LayersBP[2].NeuronCount:=IconTypes;

      NeuralNetBP1.ResetPatterns;




      for i:=0 to High(IconArr) do
        begin
        readln(sample_db,buff); IconArr[i].IconType:=strtoint(buff)+1;


        for j:=1 to IconTypes do
          begin
          if j<>IconArr[i].IconType then xOutputVector[ j-1]:=0 else
xOutputVector[ j-1]:=1;
          end;



        IconArr[i].Icon:=ReadIcon;
        // додаємо вектор навчання нейроної мережі Хопфілда
        AddSampleForm.NeuralNetHopf1.AddPattern(xVector);

        // Додаємо вектор багатошарової мережі
        //SetLength(xInputVector, 100);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        for i1:=0 to length(xInputVector) do
        write(n_debug,inttostr(round(xInputVector[i1])));
        writeln(n_debug,'');
```

```
        // NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        //SetLength(xInputVector, 256);
        readln(sample_db,buff);
        end;
        writeln(n_debug,'-----');

        {
        readln(sample_db,buff); IconArr[0].IconType:=strtoint(buff);
        xOutputVector[0]:=0;  xOutputVector[1]:=1;
        IconArr[0].Icon:=ReadIcon; // SetLength(xInputVector, 256);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);

        //SetLength(xInputVector, 256);
        readln(sample_db,buff); IconArr[1].IconType:=strtoint(buff);
        xOutputVector[0]:=1;  xOutputVector[1]:=0;
        IconArr[1].Icon:=ReadIcon; //SetLength(xInputVector, 256);
        //NeuralNetExtended1.AddPattern(xInputVector, xOutputVector);
        NeuralNetBP1.AddPattern(xInputVector, xOutputVector);
        readln(sample_db,buff);
        }
        // Ініціалізувати ваги Нейроної мережі Хопфілда
        //NeuralNetHopf1.InitWeights;



      closefile(n_debug);
      closefile(sample_db);
    end;
  //NeuralNetBP1.ResetPatterns;

end;


// запис іконки у файл
procedure WriteIcon(Arr1: TByteArr);
var
  x,y: Integer;
  SamePixels: Cardinal;
  Width,Height: Cardinal;
begin
  SamePixels := 0;
  Width := Length(Arr1[Low(Arr1)]);
  Height := Length(Arr1);

  for y := 0 to Height - 1 do
    begin
      for x := 0 to Width - 1 do
      begin
        write(sample_db,Arr1[x,y]);
      end;
    writeln(sample_db,'');
    end;
end;

// закриття форми
procedure TAddSampleForm.FormClose(Sender: TObject;
  var Action: TCloseAction);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
//  recognition_run:=not(recognition_run);
  if recognition_run then
    begin
```

```
    RobotRecognition.Terminate;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    recognition_run:=not(recognition_run);
    end;

  //IconTypes:=strtoint(eIconTypes.text);

  {
  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;

  }
end;

// відновлення картинки для збереження семпла
procedure TAddSampleForm.Button2Click(Sender: TObject);
var canvas:tcanvas;
begin
if debug then
begin
      canvas:=AddSampleForm.cellImage.canvas;
      Canvas.Brush.Color := ClWhite;Canvas.FillRect(Canvas.ClipRect);
      Canvas:=AddSampleForm.ObjectImage.Canvas;
      AddSampleForm.ObjectImage.Height :=0;
      AddSampleForm.ObjectImage.Width :=0;

end;
upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
end;

procedure TAddSampleForm.NeuralNetBP1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNetBP1.TeachError);
  Application.ProcessMessages;

end;


// навчання нейромережі із зазначеними параметрами
procedure TAddSampleForm.Button3Click(Sender: TObject);
begin


      NeuralNetBP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
      NeuralNetBP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
      NeuralNetBP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
      NeuralNetBP1.Init;
 // Учимо багатошарову
 NeuralNetBP1.EpochCount := speEpochCount.Value;
 prbEpoch.Max := NeuralNetBP1.EpochCount;
 prbEpoch.Position := 0;

 // Запуск процесу навчання (offline)
 //NeuralNetExtended1.TeachOffLine;
```

```
 NeuralNetBP1.TeachOffLine;
end;


// перебір параметрів навчання, навчання, записати помилки в лог
// для дослідження  - яка нейромережа краще, і як не потрапити в локальний
// мінімум при навчанні.
procedure TAddSampleForm.Button4Click(Sender: TObject);
  var log:textfile;
  var i,j:integer;
  var cnt:integer;
begin


  NeuralNetBP1.TeachRate:=StrToFloat(AddSampleForm.neroRate.text);
  NeuralNetBP1.Momentum:=StrToFloat(AddSampleForm.neroIMp.text);
  NeuralNetBP1.Alpha:=StrToFloat(AddSampleForm.neroAlfa.text);
  NeuralNetBP1.EpochCount := 5000;


  prbEpoch.Max := NeuralNetBP1.EpochCount;
  prbEpoch.Position := 0;
      assignfile(log,neuro_teach_log_file_name);
      append(log);
      writeln(log,'---------------------------');
      closefile(log);
  cnt:=400;
  for i:=1 to 20 do
    begin
    NeuralNetBP1.Alpha:=NeuralNetBP1.Alpha-0.01;
    for j:=1 to 20 do
      begin
      prbEpoch.Position := 0;
      NeuralNetBP1.TeachRate:=NeuralNetBP1.TeachRate-0.005;
      NeuralNetBP1.TeachOffLine;
      assignfile(log,neuro_teach_log_file_name);
      append(log);

      writeln(log,'помилка='+floattostr(NeuralNetBP1.TeachError)+'
альфа='+floattostr(NeuralNetBP1.Alpha)+'
шаг навчання =',floattostr(NeuralNetBP1.TeachRate));
      closefile(log);
      cnt:= cnt-1;
      sttError.Caption := inttostr(cnt);
      //prbEpoch.Position := prbEpoch.Position + 1;
      Application.ProcessMessages;
      end;
    end;

end;

// подія кінця епохи навчання , зрушення процес бара, відображення помилки
procedure TAddSampleForm.NeuralNetExtended1EpochPassed(Sender: TObject);
begin
  prbEpoch.Position := prbEpoch.Position + 1;
  sttError.Caption := FloatToStr(NeuralNetExtended1.TeachError);
  Application.ProcessMessages;

end;


// дозвіл розпізнавання
procedure TAddSampleForm.Button5Click(Sender: TObject);
begin
  debug:=false;
  recognition_run:=not(recognition_run);


  if recognition_run then
```

```
    begin
    RobotRecognition := TRobotRecognition.Create(false);
    RobotRecognition.RecType:=AddSampleForm.recType.ItemIndex;
    AddSampleForm.Button5.Caption:='СТОП'
    end
    else
    begin
    //RobotRecognition.
    RobotRecognition.Terminate;
    //RobotRecognition.Destroy;
    AddSampleForm.Button5.Caption:='ПІШОВ!';
    end;

end;



// збереження образів у файл
procedure TAddSampleForm.Button6Click(Sender: TObject);
  var i,j:integer;
    x,y,bx,by,ix,iy: Integer; //Лічильники
    buff_name:string;
    buff_file:textfile;
    max_icon_type:integer;
begin
  //IconTypes:=strtoint(eIconTypes.text);


  assignfile(sample_db,sample_db_file_name);
  rewrite(sample_db);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,matrix_size.text);
  writeln(sample_db,inttostr(IconTypes));
  writeln(sample_db,inttostr(High(IconArr)));
  for i:=0 to High(IconArr) do
    begin
    writeln(sample_db,inttostr(IconArr[i].IconType));
    writeIcon(IconArr[i].Icon);
    writeln(sample_db,'');
    end;
  closefile(sample_db);

  //recognition_run:=true;


end;


// збереження ваг нейромережі у файл
procedure TAddSampleForm.Button7Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
begin
  AssignFile(neuro,neuro_weight_file_name);
  rewrite(neuro);
  writeln(neuro,floattostr(NeuralNetBP1.TeachRate));
  writeln(neuro,floattostr(NeuralNetBP1.Momentum));
  writeln(neuro,floattostr(NeuralNetBP1.Alpha));

  for i:=0 to NeuralNetBP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNetBP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to  length(NeuralNetBP1.Layers[i].Neurons[j].FWeights)-1 do
          begin

writeln(neuro,floattostr(NeuralNetBP1.Layers[i].Neurons[j].Weights[w]));
```

```
          end;
      end;
    end;
  closefile(neuro);
end;



// завантаження ваг нейромережі з файла
procedure TAddSampleForm.Button8Click(Sender: TObject);
var

  neuro:textfile;
  i,j,w:integer;
  buff:string;
begin
  AssignFile(neuro,neuro_weight_file_name);
  reset(neuro);


  readln(neuro,buff);NeuralNetBP1.TeachRate:=StrToFloat(buff);
  readln(neuro,buff);NeuralNetBP1.Momentum:=StrToFloat(buff);
  readln(neuro,buff);NeuralNetBP1.Alpha:=StrToFloat(buff);
  NeuralNetBP1.Init;
  for i:=0 to NeuralNetBP1.LayerCount-1 do
    begin
    for j:=0 to NeuralNetBP1.Layers[i].NeuronCount-1 do
      begin
        for w:=0 to  length(NeuralNetBP1.Layers[i].Neurons[j].FWeights)-1 do
          begin
            readln(neuro,buff);
            NeuralNetBP1.Layers[i].Neurons[j].Weights[w]:=strtofloat(buff);
          end;
      end;
    end;
  closefile(neuro);

end;

procedure TAddSampleForm.Button9Click(Sender: TObject);
var
  QP: TQuickPixels;
  BitmapArr: TByteArr;
  y: Cardinal;

begin
  debug:=true;
 // upreview.BitmapGrabEventArr[mainform.CamsListBox.ItemIndex].NeedAddSample :=
true;
  QP := TQuickPixels.Create;
  QP.Attach(SampleImage.Picture.Bitmap);


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

  ConvertBitmapToMonoChrome(QP,BitmapArr);

  //ConvertBitmapToMonoChrome(QP2,BitmapArr);
  if length(BitmapArr)>0 then
  segment(BitmapArr);
  debug:=false;
end;

procedure TAddSampleForm.TrackBar1Change(Sender: TObject);
begin
AddSampleForm.param.Caption:=inttostr( 100-AddSampleForm.TrackBar1.Position);
end;
```

```
procedure TAddSampleForm.Timer1Timer(Sender: TObject);
begin
AddSampleForm.lFPS.Caption:=floattostr(round(tm_tick/5*10)/10);
tm_tick:=0;
end;

end.
```

Файл UMain.pas - основна програма

```
unit UMain;

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

interface

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

const port = 'COM2';

type
  TMainForm = class(TForm)
    Button3: TButton;
    MoveForLine: TButton;
    CamsListBox: TListBox;
    Label7: TLabel;
    RefreshCamsListButton: TButton;
    CamConfigButton: TButton;
    CamsGroupBox: TGroupBox;
    DialogListBox: TListBox;
    Label1: TLabel;
    VideoModeComboBox: TComboBox;
    Label2: TLabel;
    RisovalkaButton: TButton;
    CommunicationButton: TButton;
    AnalyzFrameButton: TButton;
    RazvertkaButton: TButton;
    OnFlyDebugButton: TButton;
    Button1: TButton;
    Button2: TButton;
    TLabel1: TLabel;
    AddSampleButton: TButton;
    RestaranButton: TButton;
    Button4: TButton;
    Button5: TButton;
    Button6: TButton;
    Label3: TLabel;
    Timer1: TTimer;
    Edit1: TEdit;
```

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

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



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

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


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

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




var
  MainForm: TMainForm; //головна форма

  //масив настроювань камер
  GraphConfigExArr: TGraphConfigExArr;

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

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

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

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

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

  {Інтерфейс для перебору движків}
  aTTSEnum: ITTSEnum;

  {Покажчик на параметри движка}
  fpModeInfo: PTTSModeInfo;

  NumFound : DWord;
  ModeInfo : TTSModeInfo;

  QPixels: TQuickPixels;
  timercnt: Cardinal = 0;


   i: integer;
   dest1: mass;



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

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

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

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

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

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

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

function GetVideoDevicesListEm(): TStringList;

implementation

uses Math;

{$R *.dfm}

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


{Допоміжні процедури, що не входять у клас форми}

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

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

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

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

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

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

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

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

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

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

begin
  Result := true;

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

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

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

    //присвоєння файлу
    AssignFile(GraphConfigFile,GraphConfigFileName);

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

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

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

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

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

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

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

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

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

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

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

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

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

```
    VCapSource := CamName;

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

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

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

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

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

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

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

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

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

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

{Оброблювачі подій}
```

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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


  //зупинка каналу
  Communication.Destroy;
```

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



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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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

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

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

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

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

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

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



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


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

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


   mbReset(1);
   sleep(10);

   mbExecuteProgramFile(1,'image.raw');

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

   Communication.PortEn:=true;
  end;



   MainForm.Timer1.Enabled:=true;

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

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

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




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

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

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

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

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

    addsampleform.NeuralNetBP1.Compute(xInputVector);


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

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

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

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

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

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

end.
```

Файл UPreview.pas – модуль роботи з камерами

```
unit UPreview;

interface

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

type
  //захоплений кадр
  TCapturedBitmap = Vcap.TCapturedBitmap;

  TBitmap = Graphics.TBitmap;

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

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


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

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

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

  implementation

uses
  UMain,UCamTimers,Urestaran;

var
  F: TextFile;
//  BitmapArr: TByteArr;

{$R *.dfm}

//конструктор
```

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

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

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

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

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

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

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

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

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

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


    if recognition_run then
    begin
      QP.Attach(Bitmap);

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

      //TSampleRule(BitmapArr);

        RobotRecognition.BitmapReady:=true;
    end;
  end;

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

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

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

```
    RazvertkaTlumitsya.Destroy;
  end;

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

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

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

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

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

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

finalization
  QP.Destroy;
  CloseFile(F);

end.
```

Файл UCamTimers.pas – модуль встановлення таймерів камер

```
unit UCamTimers;

interface

uses
  Windows,URazvertka, UPreview, UTypeConst, SysUtils;

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

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

  //масив таймерів камер
  TCamTimerArr = array of TCamTimer;

var
  CamTimerArr: TCamTimerArr;//масив таймерів камер

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

implementation

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

end.
```

Файл UGraphConfig.pas – модуль налаштування камер

```
unit UGraphConfig;

{Модуль настроювання камери}

interface

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


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


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

var
  GraphConfigForm: TGraphConfigForm; //форма

  //індекс елемента масиву настроювань
  GraphConfigExIndex: Cardinal;

implementation

uses
  UMain;

{$R *.dfm}

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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


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

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

end;

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

end.
```

Файл UPicture_Analyz.pas - модуль виділення й розпізнавання обличчя користувача у мережі з камери за допомогою нейроної мережі

```
unit UPicture_Analyz;
// один з модулів виділення й розпізнавання обличчя користувача у мережі,
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 виділення й розпізнавання обличчя користувача у мережі:
// розпізнає весь екран цілком, як один образ
// або б'є по сегментах, і розпізнає кожний окремо

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

      recognition_run:=false;
```

```
      //debug:=false;
      BitmapReady:=false;
      if rectype =1 then
      begin
        segment(BitmapArr);

      end;
      if rectype =0 then
      begin

        TSampleRule(BitmapArr);

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

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

begin

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

  Canvas:=AddSampleForm.SampleImage.Canvas;


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

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

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

      BitmapArr[y,x] := PixelRes;

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

```
      end;

      // що- те на подобі простенького фільтра

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

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

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

      OldPix:=PixelRes;


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

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

  end;


end;


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

begin

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

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


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

    end;

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

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

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

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

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



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

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

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

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

end;



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

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


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

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

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

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

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

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

begin


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


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

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

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

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

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

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

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

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

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

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


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

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

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

                  end;

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

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

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

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

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

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

      end;


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

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

```
            end;
          end;

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

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

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

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

                  end;

            end;
      end;

 }


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

      end;

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

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

```
            if lables[BitmapArr[i,j]]<>0 then
             begin

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

             end;
           end;

      end;
    end;

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

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

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

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

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

      if debug then
         begin

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

```
          end;


        end;


    // ну от, переходимо до суті - властиво до виділення й розпізнавання обличчя користувача у мережі:
    // 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 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('МАГІСТЕРСЬКИЙ ПРОЕКТ');
Memo1.Lines.Add('');
Memo1.Lines.Add('на  тему:');
Memo1.Lines.Add('');
Memo1.Lines.Add(' Дослідження та програмна реалізація системи виділення й
розпізнавання обличчя користувача у мережі');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('Керівник: Буравченко К.О.');
Memo1.Lines.Add('');
Memo1.Lines.Add('Розробив: студент Скирда Олександр Віталійович');
Memo1.Lines.Add('                    гр. КН-22МЗ');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
Memo1.Lines.Add('м. Кропивницький 2023');
Memo1.Lines.Add('');
Memo1.Lines.Add('');
end;

procedure TAboutForm.Button1Click(Sender: TObject);
begin
AboutForm.Close;
end;

end.
```