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

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

ВИПУСКНА КВАЛІФІКАЦІЙНА РОБОТА
за другим (магістерським) рівнем вищої освіти
на тему
“Дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку”

Виконав здобувач вищої освіти
ІІ курсу, групи КІ-22М-1
ОПП «Комп’ютерна інженерія»
спеціальності 123 «Комп’ютерна інженерія»
_____ Лісовий В.А.
«____» _____ 2023 р.

Керівник проекту
доктор філософії (PhD)
_____ Усік П.С.
«____» _____ 2023 р.
Рецензент _____
_____

м. Кропивницький

Центральноукраїнський національний технічний університет

Факультет *Механіко-технологічний*

Кафедра *Кібербезпеки та програмного забезпечення*

Рівень вищої освіти *магістр*

Галузь знань _12 _____*"Інформаційні технології"*_____

Спеціальність 123 *"Комп'ютерна інженерія"*

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

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

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

Лісовому В'ячеславу Андрійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи *Дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку*

2. Керівник роботи *Усік Павло Сергійович, доктор філософії (PhD)*
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу № 34-13 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 стабілізації та підвищення якості відеопотоку*

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

Лісовий В.А. Дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку. 123 Комп'ютерна інженерія. Центральноукраїнський національний технічний університет. Кропивницький. 2023.

В даній випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стабілізації та підвищення якості відеопотоку.

Метою розробки є дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку.

Об'єктом дослідження є процес стабілізації та підвищення якості відеопотоку.

Предметом дослідження є методи стабілізації та підвищення якості відеопотоку.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

Результат роботи – програмна реалізація системи стабілізації та підвищення якості відеопотоку.

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 Builder C++.

Ключові слова: комп'ютерна інженерія, відеопотік

ABSTRACT

Lisovyi V.A. Research and software implementation of the system of stabilization and improvement of the quality of the video stream. 123 Computer engineering. Central Ukrainian National Technical University. Kropyvnytskyi. 2023.

In this graduation thesis for the second (master's) level of higher education, software was developed, which is intended for the system of stabilization and improvement of the quality of the video stream.

The purpose of the development is the research and software implementation of the system of stabilization and improvement of the quality of the video stream.

The object of the study is the process of stabilizing and improving the quality of the video stream.

The subject of the study is the methods of stabilization and improvement of the quality of the video stream.

Research methods are based on computer graphics methods, mathematical statistics methods, and software development methods.

The result of the work is the software implementation of the stabilization system and improvement of the quality of the video stream.

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 Builder C++ environment.

Keywords: computer engineering, video stream

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

DS – DirectShow мультимедійне розширення Microsoft;

TrFl – Трансформ-фільтр;

RnFl – Рендер-фільтр;

SourceFilter – Фільтр джерела;

IC (Intelligent Connect) – інтелектуальне з'єднання;

ООП – об'єктно-орієнтоване програмування;

DSP – MS Direct Show Components and class to write Multimedia Applications – бібліотека компонентів для розробки мультимедійних програм;

VWINDOW – вертикальна синхронізація;

HWINDOW – горизонтальна синхронізація;

PLL – Цикл Блокування Стадії;

USB – Universal Serial Bus – універсальна послідовна шина;

URL – universal resource locator – локатор ресурсів Інтернет;

HTML – HyperText Murkup Language – мова розмітки гіпертекстових документів;

ОС – операційна система;

ПЕОМ – персональна електронно-обчислювальна машина;

КС – комп’ютерна система;

ПЗ – програмне забезпечення;

ПК – персональний комп’ютер;

ВСТУП

Актуальність теми. Підвищення ефективності сучасних систем відеоспостереження дозволяє знизити працезатрати персоналу. Але в більшості випадків застосування таких систем обмежується установкою відеокамер, і при цьому операторові необхідно постійно стежити за тим, що відбувається на декількох камерах, дані з яких можуть відображатися на безлічі моніторів. Тому поряд із впровадженням систем відеоспостереження на автостоянках, заправних станціях, перехрестях зі складним рухом потрібно здійснювати аналітичну підтримку таких систем. Виникла потреба у використанні детекторів руху, наявності зручного користувальницького інтерфейсу, що дозволяє інтуїтивно стежити за декількома камерами, у системі сигналів для оператора, що залучає увагу до певної ситуації на моніторі, в аналізі руху об'єктів і їхніх параметрів. Також варто врахувати, що при організації відеоспостереження на відкритій місцевості часто виникають проблеми, пов'язані з несприятливими погодними умовами.

Основною вимогою до систем відеоспостереження є ведення журналу подій у вигляді набору кадрів або відеопослідовностей, на яких відбиті порушники, з одночасним оповіщенням оператора. Функціонування блоку, що виконує ці дії, засновано на роботі детектора руху, однак у більшості існуючих систем використовується детектор руху, інтегрована в камери. У свою чергу, це приводить до необхідності дуже ретельного підбора параметрів визначення руху для зниження ймовірності помилкового спрацьовування. При програмній реалізації детектора руху можливе використання адаптивних алгоритмів і алгоритмів, що враховують попередню обробку у вигляді стабілізації, колірної і яркістної корекції.

Мета й завдання дослідження. Метою роботи є дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку.

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

– Огляд існуючих систем стабілізації та підвищення якості відеопотоку.

– Дослідження системи стабілізації та підвищення якості відеопотоку.

– Програмна реалізація системи стабілізації та підвищення якості відеопотоку.

Об'єктом дослідження є процес стабілізації та підвищення якості відеопотоку.

Предметом дослідження є методи стабілізації та підвищення якості відеопотоку.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод стабілізації та підвищення якості відеопотоку.

– Розроблено вітчизняний продукт стабілізації та підвищення якості відеопотоку, який має більш широкі можливості, на відміну від існуючих аналогів.

Практична цінність отриманих результатів полягає в тому, що розроблені алгоритми дозволяють успішно вирішувати задачі стабілізації та підвищення якості відеопотоку.

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 стабілізації та підвищення якості відеопотоку, 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

Пропонована програмна система призначена для поліпшення якості відеоматеріалу, отриманого в складних умовах освітлення, при організації відеоспостереження на АЗС, паркуваннях, у дворах і т.п. У системі реалізований триступінчастий детектор руху, що розділяє ситуацію в кадрі на три види:

– немає руху;

– слабкий;

– сильний рух.

У зв'язку із проблемами детекторів руху, а саме:

– великою чутливістю до погодних умов;

– тремтінню камери;

– незначних перешкод у кадрі;

пропонується алгоритм стабілізації відео, що дозволяє значно підвищити стійкість до таких ефектів. Застосування методу MSR для підвищення якості кольорових зображень дозволяє поліпшити якість сприйняття відеоданих у вечірній час доби, а також знизити вплив перешкод від самих камер.

1.2 Область застосування

Областю застосування розробляє мого програмного забезпечення є системи відеоспостереження.

Завдань, які вирішує відеоспостереження, безліч. Це може бути спостереження за входом у будинок, за співробітниками в офісі, спостереження за робочим процесом на підприємстві або в торговельному залі супермаркету. Яке завдання не стояло б, основним критерієм все-таки залишається якість

одержуваного зображення. Коли мова йде про високе розрішення, то багато фахівців в один голос затверджують, що треба встановлювати IP-рішення. Почасти вони звичайно праві. IP-відеоспостереження, як правило, дає більше можливостей по розпізнаванню деталей об'єкта зйомки. Але якщо бюджет обмежений, то доводиться шукати альтернативні рішення. Більше низькі за ціною, але порівнянні по якості. Один з варіантів рішення – це використання сучасних технологій, наприклад, відеокамери на матриці HDIS (High Definition Image Sensor). В основі нової HDIS-матриці лежить уже давно відома всім CMOS (Complementary Metal Oxide Semiconductor, комплементарна структура метал-оксид-напівпровідник, КМОП) технологія. HDIS сенсор дає можливість зчитувати відеоінформації з високою швидкістю й дозволом 700ТВЛ. Крім цього матриця такого типу має більше ощадливе енергоспоживання, а також, що дуже важливо, широкий робочий температурний діапазон від -40 до +50°С (витримає будь-яку українську зиму).

Дивлячись на всі переваги HDIS-матриць можна було б припустити, що ціна буде висока. Але й тут технологія HDIS нас дивує – ціна не вище, ніж аналогічні рішення на CCD-матрицях, а в більшості випадків навіть нижче.

Таким чином, виходячи з вищеперерахованого, дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку, 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

Ivideon

На перший погляд Ivideon – це один з багатьох сервісів по організації відеоспостереження. Якщо копнути більш детально, то він радикально відрізняється від аналогів. Наприклад, для особистого користування сервіс безкоштовний, що є безпрецедентним випадком. Звичайно учасники цього ринку не розділяють клієнтів на бізнес-користувачів і звичайних людей, яким може знадобітися система відеоспостереження. Хочеш віддалено спостерігати за чимсь або за кимсь – плати за сервіс (причому за кожний відеоканал окремо), за встаткування, за виділену IP-адресу.

У випадку з Ivideon не тільки не прийдеться платити, але можна використовувати вже наявне практично в кожного власника ПК або ноутбука встаткування. Тобто, у більшості випадків взагалі не буде потрібно ніяких капіталовкладень. Крім того, якщо користувач є щасливим власником iPhone/iPod touch або iPad, те він до того ж одержить мобільний термінал для спостереження за своїм об'єктом, перебуваючи в будь-якій точці землі, де є доступ до Інтернету. Властиво, про клієнта Ivideon для iOS і піде мова нижче, але колись варто згадати ще й про те, що для запуску особистої системи відеоспостереження за допомогою розглянутого рішення не потрібні якісь спеціальні знання. Все настільки просто, що мені особисто, поки не випробував Ivideon у справі, взагалі все це здавалося якимось хитрим розіграшем.

Зокрема, не потрібні спеціальні камери, не потрібний відеореєстратор або спеціальна плата розширення в ПК, не потрібний виділений IP-адреса у випадку

використання мережної камери. Досить простої веб-камери, причому підійде й та, що убудовано в ноутбук. Всю роботу із трансляції відео в будь-яке зручне для користувача місце бере на себе сервіс Ivideon і робіть він це безкоштовно. А захват зображення реалізується за допомогою спеціального ПЗ для Windows або Linux. Клієнта для OS X поки ні, але буде – про це ми поговоримо трохи пізніше. А поки давайте побудуємо власну систему відеоспостереження за п'ять хвилин.

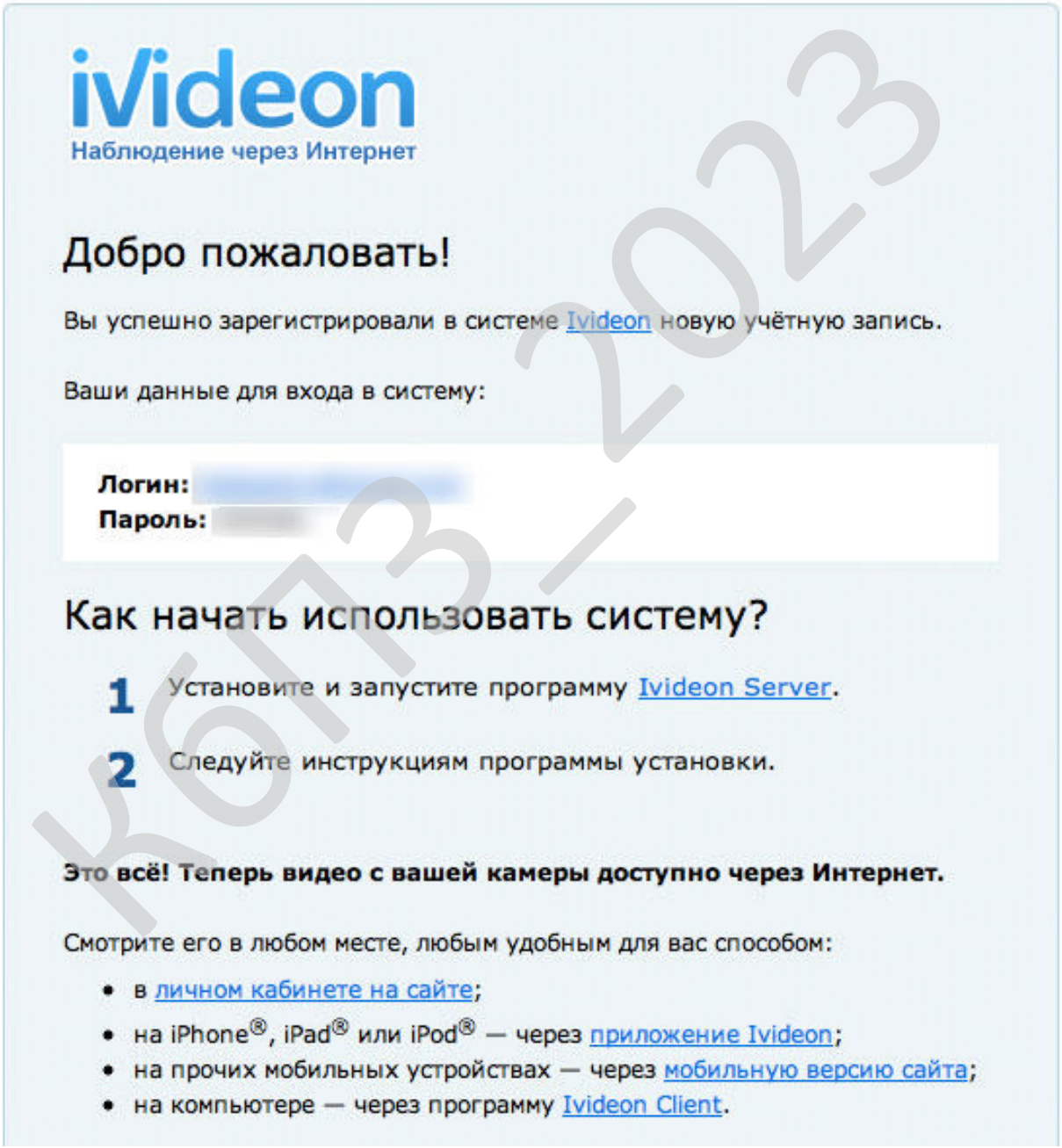

Рисунок 2.1 – Інтерфейс користувача Ivideon

Потім переходимо по посиланнях, завантажуємо серверне ПЗ для комп'ютера й клієнт для iOS.

Установка першого проста й невигадлива, хоча кілька дій зробіти прийдеться, а саме, підключити додаток до створеного раніше облікового запису Ivideon (або створити новий), вибрати веб-камеру й мікрофон (раптом захочете писати звук із зовнішнього мікрофона й використовувати окрему USB-камеру, а не з ноутбуку), указати шлях до архіву з відеозаписами й обсяг виділеного для них дискового простору, визначитися з варіантами запуску ПЗ (автоматом при запуску ОС і т.п.). Після цього запускається сам додаток і починає вести запис.

Отже, серверна частина запущена. Зверніть увагу, що коли працює веб-камера, на ній загоряється зелений світлодіод, тобто, непомітно за кимсь стежити не вдасться. Хіба що якщо використовувати зовнішню камеру й десь її заховати.

Ivideon уже веде захват, і при бажанні можна переглянути що ж таке відбувається «на іншому кінці проведення» у будь-якому веб-браузері, включаючи мобільний, для цього досить зайти в особистий кабінет на сайті.

Stem IZON 2.0

Stem IZON 2.0 являє собою бездротову камеру з убудованим модулем Wi-Fi і набором датчиків, що дозволяють налаштувати умови реагування пристрою. У плані апаратної начинки – це продукт не самий складний, але адже всі ми знаємо, що «залізо» без софту є лише марною оболонкою. Stem же пропонує готове й, що найважливіше, просте у використанні рішення. Але перш ніж перейти до базового налаштування пристрою, давайте розберемося, що воно вміє й що пропонує за свої гроші.

Камера поставляється в прозорій акриловій коробці, у якій також є магнітна база для камери, пари дюбелів для кріплення системи на стіну, 2,7-метровий кабель miniUSB-USB для підзарядки й блок живлення. При необхідності пристрій можна живити не тільки від електромережі, але й від ПК. Дуже сподобалося рішення з магнітною базою, що дозволяє міняти кут камери без усяких шарнірів і кріплень. Просто вибираємо потрібне положення й примагнічуємо IZON одним рухом. Технічно камері можливостей досить, щоб

упоратися з поставленим завданням. Вона здатна записувати відео в дозволі 640x480 точок із частотою 30 кадрів у секунду й бітрейтом 1,5 Мбіт/с. Кут огляду становить 60°. При прямої трансляції в реальному часі якість відеопотоку знижується (320x240 точок, 10-кадрів у секунду, 300 Кбіт/с), але зате картинка без проблем передається навіть через мобільне EDGE-з'єднання, хоча 3G, звичайно, краще. Існує можливість підключатися до пристрою з будь-якої точки планети, де є доступ до Інтернету.

Крім відео записує й транслюється звук у діапазоні від 40 Гц до 8 кГц. IZON реагує на рівень звуку від 35 до 95 дб. На жаль, у камері немає убудованого інфрачервоного ліхтаря й можливості працювати в повній темряві. Виробник заявляє, мол, застосування таких засобів негативно впливає на якість запису відео у звичайних умовах освітлення. Не впевнений, що це так, скоріше, справа в економії. Зате пристрій досить непогано працює навіть при слабкому освітленні. Досить світла від 7,5-ватної лампочки.

Природно, для роботи IZON 2.0 потрібне підключення до Мережі й от отут вже без софта не обійтися. Пристрій контролюється тільки через iOS-додаток і через нього ж забезпечується доступ до трансляції в реальному часі або ж до архіву записів. Однойменний безкоштовний додаток доступно в App Store.

Перш ніж приступитися безпосередньо до моніторингу, прийдеться підключити камеру до iPhone або iPad, до домашньої мережі й налаштувати умови роботи. Процес нескладний, але недосвідченого користувача може небагато спантеличити, тому трішки його отут розпишу. Для початку додаємо камеру (можна використовувати одночасно кілька пристроїв). Щоб система зрозуміла, що до пристрою приєднується хазяїн, а не інша особа, використовується проста система світлового коду. Світлодіод на камері поморгає у певній послідовності жовтогарячим і зеленим кольором, після чого програма запропонує вибрати правильний варіант. Потім підключаємо камеру до мережі Wi-Fi, і після того, як на екрані з'явиться QR-код, його треба буде показати пристрою. Все, от тепер камера синхронізована з iPhone.

Важливою особливістю системи Stem IZON 2.0 є гнучка система налаштування. Адже користі від такого роду пристрою буде мало, якщо пропустиш саме цікаве. Таким чином, пристрій можна налаштувати, щоб він активувався при певній зміні зовнішніх умов і посилало при цьому повідомлення на iPhone. Передбачено реакцію на звук і на рух. В останньому випадку потрібно задати активну зону. Це може бути, наприклад, вхідні двері, ліжечко з дитиною, певна частина кімнати, де любіть пустувати домашній вихованець і т.п.

Рисунок 2.2 – Інтерфейс користувача Stem IZON 2.0

Система здатна самостійно записати відео при активації камери й зберегти його на сервері Stem. Але для використання цієї функції необхідно зареєструватися. Процедура традиційна (email, пароль) і не займе більше хвилини. На жаль, є обмеження. Так, довжина ролика не перевищує 30 секунд і таких роликів у день можна записувати до 25 штук. До того ж, перенести їх на ПК або зовнішній накопичувач не можна. Stem дозволяє лише переглядати контент через фірмовий додаток.

Що стосується якості відео, то воно залишає бажати кращого. Хоча, я вже писав про його параметри вище й про причини – економія трафіку. Проте, видаваної картинки цілком достатньо для розуміння того, що відбувається на підконтрольній території. А якщо ведеться автоматичний запис, те тоді й картинка буде краще.

Dropcam HD

Компанія Dropcam – це один з ветеранів на споживчому ринку систем віддаленого відеоспостереження й не перший рік вона радує людей такого роду пристроями. Dropcam HD компактна, стильна, а при необхідності її нескладно замаскувати. Компактний модуль із діаметром близько 80 мм розташовується на функціональній алюмінієвій підставці, з якої при необхідності він легко витягає. Сама ж підставка може виконувати як роль настільного упору, так і настінного кріплення.

Сподобався монолітний корпус камери, у якому є лише один отвір – рознімання micro-USB. Через нього пристрій живиться й підключається до ПК для базового налаштування, про яку мова йтиме трохи нижче, а поки давайте глянемо на технічні характеристики шпигунського пристрою.

В аксесуарі використовується якісна матриця й широкоугольний світлочутливий об'єктив. Таке зв'язування дозволило вести запис в умовах досить слабкої освітленості й навіть у півмороку, причому підсумкова картинка виходить у дозволі 1280x720 точок і видаються із частотою 30 кадрів у секунду, що дуже круто. У стандартних побутових рішень звичайний розрішення запису

VGA (640x480), та й швидкість якщо й досягає 15 кадрів у секунду, те вже добре. Варто відзначити, що кодування відео відбувається з використанням кодеку H.264, тобто непогано ужимається, але через HD-розрішення потік однаково виходить немаленький і для його підтримки потрібно досить широкий канал. Рекомендується від 10 Мбіт, хоча в ідеалі – 50 Мбіт.

Але зі швидким Інтернетом зараз начебто б як проблем і немає. До того ж Dropcam HD залишається функціональної навіть практично в повній темряві. Цю проблему вирішує убудований набір з потужних інфрачервоних світлодіодів і наявність відповідного режиму роботи. Кут огляду становить 107°.

Ще одна важлива фішка камери – це наявність власного якісного й чутливого мікрофона. Частенько на цьому інші виробники заощаджують. Більше того, навіть є динамік і при бажанні можна віддалено гаркнути на порушника або нахабного домашнього коте, що творить шкоду. Убудований динамік і функція передачі власного голосу в принципі дають масу забавних можливостей.

Комплект поставки містить все необхідне для швидкої установки системи в ньому є, включаючи 3-метровий кабель USB.

Камера бездротова, з домашньою мережею зв'язується через Wi-Fi, щоправда, власного акумулятора в неї немає. Так що по факті одне проведення із пристрою буде стирчати постійно, а поблизу потрібне наявність розетки. Але перш ніж аксесуар установлювати на постійне місце перебування, його прийдеться попередньо підключити до комп'ютера й налаштувати.

Виробник завіряє, що на налаштування Dropcam HD піде не більше 60 секунд. Підключивши пристрій до USB-порту ПК одержимо доступ до фірмового ПЗ для Windows і OS X. Пари кличів, після чого пропонується завести обліковий запис в онлайновому сервісі Dropcam – без цього ніяк, адже всі дані шифруються й зберігаються в хмарі. Реєстрація швидка й легка. Потім досить вибрати точку доступу Wi-Fi і всі, можна користуватися.

Компанія пропонує кілька варіантів доступу до трансляції: через будь-який веб-браузер на ПК, за допомогою iOS-пристрою або Android-пристрою.

Крім того, для мобільної ОС Apple підтримуються Push-повідомлення, плюс є оповіщення по E-mail. Такі приходять, якщо камера визначає рух або звук.

Взагалі, запис відбувається постійно, більше того, весь він може зберігатися на серверах Dropcam, але тільки за гроші. Підтримувати 7-денний архів даних обійдеться в $10 на місяць, 30-денний – утроє дорожче. Для другої й всіх наступних камер діє 50-процентна знижка на сервіс. А от базова підтримка обійдеться безкоштовно, але вона включає лише живу трансляцію. При бажанні відеопотоком можна поділитися із друзями по E-mail (але їм теж прийдеться зареєструватися в сервісі) або ж опублікувати його у відкритому доступі.

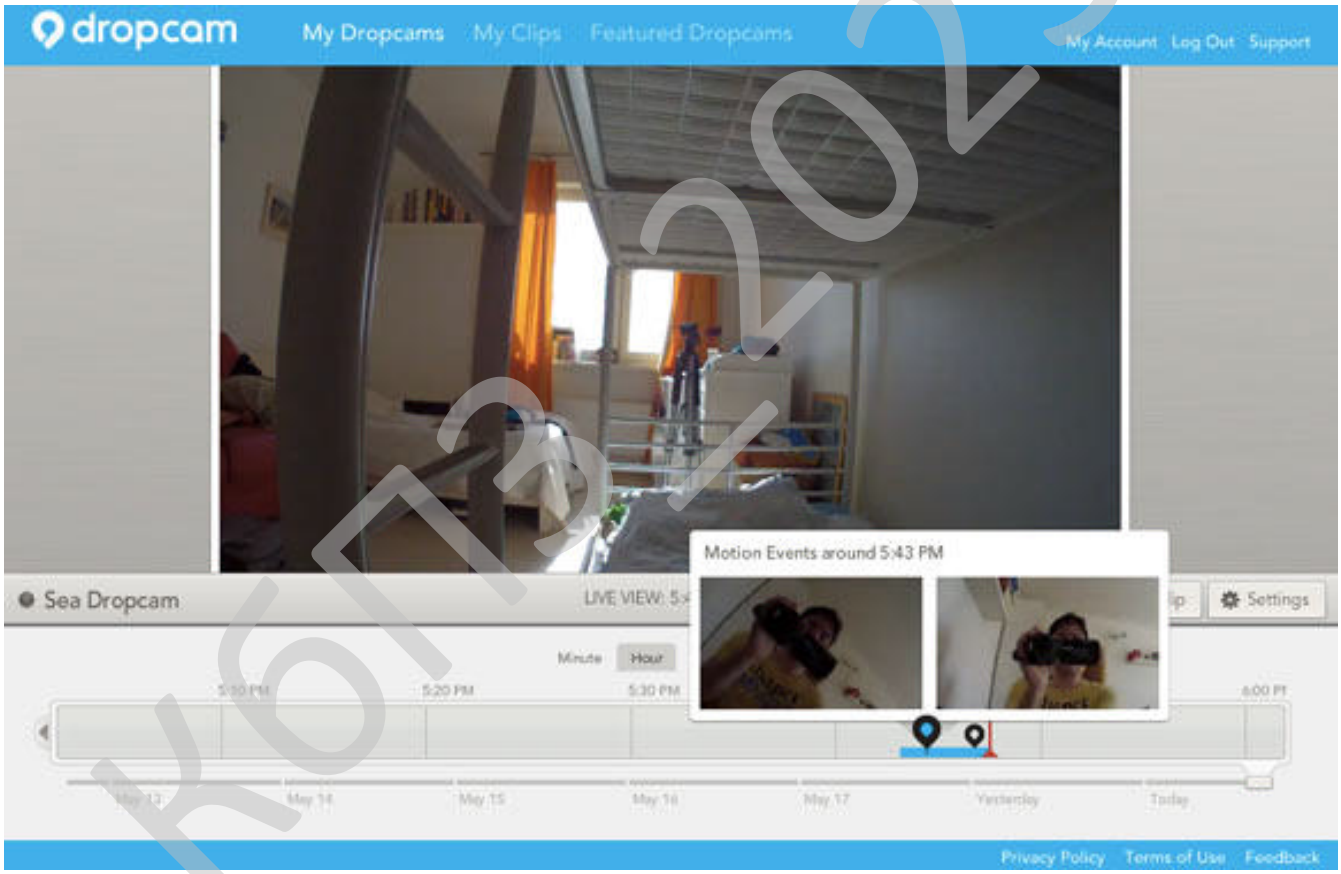

Рисунок 2.3– Інтерфейс користувача Dropcam HD

З підключенням камери в захищених мережах проблем бути не повинно, тому що підтримуються протоколи WEP (40-bit, 128-bit), WPA (TKIP, AES) і WPA2 (TKIP, AES). Сам бездротовий модуль працює в режимі 802.11b/g/n. Потік же шифрується із застосуванням 256-бітного алгоритму AES і SSL-з'єднання.

Якість відео досить на рівні, але не чекайте кристально чіткої картинки. Але за рахунок високої частоти підсумковий результат виходить краще, ніж у більшості конкуруючих рішень, а в умовах гарної освітленості зображення просто шикарне. Оптичного зуму або можливості рухати камеру віддалено немає. Доступний тільки цифровий зум – активна область ділиться на п'ять частин і кожну з них можна збільшити. Але картинка вийде мутною – краще систему використовувати без цифрового збільшення, що є лише іграшкою, не більше.

Отже, серед плюсів Dropcam HD виділимо просту установку й налаштування, двосторонню аудіозв'язок, режим нічного бачення, безкоштовний стримінг у режимі 24/7, можливість поділитися потоком із друзями й доступ з мобільних пристроїв через спеціалізовані клієнти.

Мінуси є теж, як же без них. Так, функція запису й зберігання відео коштує відчутних грошей, немає можливості локального зберігання знятих копій, немає убудованого акумулятора (камера прив'язана до електромережі) і немає вогне-влагозахищеності (можна використовувати тільки усередині приміщень).

Серед конкурентів варто згадати систему відеоспостереження Logitech Alert 750e, захищену від будь-якої непогоди, здатну живитися навіть від Ethernet-мережі й зберігати відео локально. От тільки коштує вона аж $350 і це в США. У Україні як мінімум буде те ж саме, але в євро, тобто сильно дорожче. Ще можна згадати про дійсно бездротову систему Avaak VueZone з убудованим акумулятором, але відео вона записує в SD-якості, а коштує – дорожче ($200 у США). У підсумку Dropcam HD хоч і не позбавлена мінусів, але видає одну із кращих по якості картинок, при цьому дозволяє зберігати до місяця записів у високому дозволі, нехай і за гроші. Крім того, у плані налаштування й установки – це найпростіша система, у якій розбереться навіть домогосподарка».

2.2 Обґрунтування вибору засобів для побудови системи та мови програмування

Оскільки потрібно розробити просту та легку у користуванні програму, яка б виконувалась під операційною системою Windows, то для її реалізації я обрав Builder C++. Існує велике число бібліотек написаних під Builder C++, тому це одна з важливих причин вибору мови програмування. Середовище Builder C++ досить просте в користуванні, його вихідний код значно менше по об'єму в порівнянні з Delphi чи деякими іншими програмами такого типу. Досить легко організувати взаємодію між модулями програм, об'єктно-орієнтований підхід дає можливість значно скоротити код програми, а отже і час його виконання.

На заміну старого розробленого набору елементів управління у Builder C++ інтегрована бібліотека візуальних компонентів VCL, представлених на палітрі компонентів. Після переносу на форму методом перетягування (drag-and-drop) компоненти відразу становляться діючими об'єктами вашої програми. Окрім типізованих інтерфейсних елементів Windows (кнопки, смуги прокручування, редагуємі текстові області, прості та комбіновані списки, та інше) у бібліотеку включені елементи підтримки діалогових вікон, обслуговування баз даних та багато іншого. Можливо не тільки модифікувати поведінку існуючих компонентів, але і будувати нові.

Builder C++ підтримує останні розширення стандарту мови C++ та забезпечує швидку компіляцію та складання 32-розрядних програм для Windows. Результуючі програми оптимізовані з точки зору швидкості виконання програм та затрат пам'яті. Зручний відладгоджувальник (з асемблерним вікном, можливістю крокового виконання, завдання точок зупинки, трасування та інше) повністю інтегрований у систему проектування. Дизайнер форм, редактор коду, інспектор об'єктів та інші інструменти зостаються доступними під час виконання програми, саме через це вносити зміни до коду можна прямо у процесі відлагодження.

Дизайнер форм, Інспектор об'єктів і інші засоби залишаються доступними під час роботи програми, тому вносити зміни можна в процесі відлагодження.

Builder C++ поставляється в трьох варіантах: Standard (стандартний), Professional (для професіоналів розробників, орієнтованих на мережеву архітектуру) і Client/Server Suite (для розробки систем в архітектурі клієнт/сервер). Останні два варіанти доповнюють стандартний початковими текстами візуальних компонентів, різномасштабним словником даних, новими функціями мови запитів SQL для бази даних, пакетом підтримки систем Internet, службою моніторингу програм, а також рядом інших засобів.

Builder C++ підтримує зв'язок з різними базами даних 3-х видів: dBASE і Paradox: Sybase, Oracle, InterBase і Informix; Excel, Access, FoxPro і Btrieve. Механізм BDE (Borland Database Engine) додає обслуговуванню зв'язків з базами даних дивовижну простоту і прозорість. Провідник Database Explorer дозволяє зображати зв'язки і об'єкти баз даних графічно. Використовуючи компоненти баз даних, я побудував електронний записник згідно таблиці dBASE за півгодини роботи на комп'ютері. Спадкоємство готових форм і їх "підгонка" під специфічні вимоги помітно скорочують тимчасові витрати на вирішення подібних завдань.

Довідкова служба Builder C++ надавала мені допомогу в цій і багатьох інших подібних ситуаціях. Є повний опис кожного управляємого компонента, включаючи списки властивостей і методів, а також численні приклади. Виклад матеріалу в книзі був значно покращуваний і систематизований завдяки відомостям, почерпнутим мною з довідкової служби.

Завдяки засобам управління проектами, двосторонній інтеграції застосунку і синхронізації між засобами візуального і текстового редагування, а також вбудованому відладнику (з асемблерним вікном прокрутки, покрокового виконання, точок останову, трасуванням і тому подібне) – Builder C++ корпорації Borland надає собою вражаюче середовище розробки, яка, мабуть, витримає конкурентну боротьбу з такими модними продуктами як Developer Studio фірми Microsoft.

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 стабілізації та підвищення якості відеопотоку.

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

Оцінка руху

Найбільш значимим у системах відеоспостереження є модуль детектора руху, оскільки при виявленні руху встає необхідність залучення уваги оператора. Існує велика кількість методів виявлення руху, однак багато які є досить складними з обчислювальної точки зору. До самих швидкодіючих методів визначення руху відносяться методи на основі вирахування тла й порівняння сусідніх кадрів. У їхню основу заставляється порівняння поточного кадру з попереднім або з деяким опорним кадром. При цьому для обчислення різниці між кадрами існують три основних підходи по обчисленню значень на основі компонентів колірної моделі RGB, яскравості пікселів, хроматичних компонентів колірної моделі YUV.

Для розрахунку величини руху був розроблений метод, що припускає використання факторів колірного посилення із застосуванням обчислення значення на основі колірної моделі RGB. Розрахунок абсолютної різниці між значеннями колірних компонентів R, G, B (Red, Green, Blue) пікселів аналізованих кадрів I_{op} і I_{cur} проводиться в нормованому діапазоні [0..1] відповідно до формули:

$$S_{op}(s,x,y) = \begin{cases} 1, \text{ если } S_{op}(s,x,y) \geq TS \times F_{boost}, \\ S_{op}(s,x,y) \times F_{boost}, \text{ если } S_{op}(s,x,y) \geq \\ \qquad \geq TS \text{ и } S_{op}(s,x,y) < \\ \qquad < TS \times F_{boost}, \\ S_{op}(s,x,y), \text{ если } S_{op}(s,x,y) < TS, \end{cases} \tag{3.1}$$

де s – покажчик компонента колірної моделі (R, G, B); x і y – просторові координати; F_{boost} – величина, що визначає ступінь активації режиму посилення;

K_{boost} – коефіцієнт підсилення; TS – граничне значення, що визначає, чи відбулися зміни в кольорі пікселя; $S_{ad}(s, x, y)$ – функція обчислення абсолютної різниці по зазначеному колірному компоненті між заданими кадрами, що обчислюється за формулою:

$$S_{ad}(s,x,y) = \left| Sv_{Iop}(s,x,y) - Sv_{Icur}(s,x,y) \right|, \quad (3.2)$$

де Sv_{Iop}, Sv_{Icur} – значення компонента колірної моделі (R, G, B) опорного й поточного кадрів відповідно.

Як опорний кадр I_{op} можна використовувати як попередній, так і фоновий кадр.

Для розрахунку величини зміни пікселя в нормованій формі колірної моделі RGB використовується формула:

$$P_{diff}(x,y) = \frac{(S_{ad}(R,x,y) + S_{ad}(G,x,y) + S_{ad}(B,x,y))}{3}. \quad (3.3)$$

Найчастіше шум у кадрах може бути зарахований як зміна в пікселі, і в підсумку у великій кількості цей шум буде впливати на величину руху. Для зменшення впливу шуму на величину руху необхідна додаткова корекція. Вона відбувається залежно від установленого граничного значення [TS] і коефіцієнта придушення малих відхилень [Lc] відповідно до формули:

$$PI_{diff}(x,y) = \begin{cases} 0, \text{ если } P_{diff}(x,y) < TS \times Lc, \\ P_{diff}(x,y), \text{ если } P_{diff}(x,y) \geq TS \times Lc. \end{cases} \quad (3.4)$$

Параметр коефіцієнта придушення малих відхилень обчислюється на основі експериментальних досліджень і може бути скоректований під час налаштування параметрів системи. Загальна величина, що характеризує зміни пікселя [PDiff(x, y)], розраховується за формулою:

$$P_{diff}(x,y) = \begin{cases} PI_{diff}(x,y), \text{ если } PI_{diff}(x,y) < TS, \\ PI_{diff}(x,y) + 1, \text{ если } PI_{diff}(x,y) \geq TS \\ \quad \text{и } PI_{diff}(x,y) < TS \times K_{boost}, \\ PI_{diff}(x,y) + 2, \text{ если } PI_{diff}(x,y) \geq \\ \quad \geq TS \times K_{boost}. \end{cases} \quad (3.5)$$

Значення величини, що характеризує рух між двома кадрами (зміни в кадрі), обчислюється на підставі загальної величини, що характеризує зміна значень пікселів аналізованої області кадру відповідно до формули:

$$F_{motion} = \frac{\sum_{i=0}^{W}\sum_{j=0}^{H} PDiff(x,y)}{H \times W} \times 100, \quad (3.6)$$

де W, H – розміри аналізованої області кадру по горизонталі й вертикалі (ширина й висота) відповідно.

Для класифікації руху використовується значення величини, що характеризує рух між двома кадрами, при цьому слід зазначити, що залежно від сцени діапазони класифікації рухи можуть варіюватися.

Для можливості зберігання інформації про рух був створений власний формат зберігання відеоданих. Спроектована структура файлу складається із двох основних блоків – заголовної частини й блоку даних. У заголовній частині описуються параметри камери, характеристики відеозапису, а також індексне поле, що містить інформацію про рух, тимчасові мітки й т.п. Запропонована структура файлу дозволяє здійснювати швидку навігацію й перевантажувати параметри оцінки руху, що зберігаються в заголовній частині. Наявність в індексному полі посилального зв'язку дає можливість організації проріджування відеопослідовності шляхом видалення кадрів, у яких або відсутній, або зафіксований дуже малий рух.

Стабілізація відеоданих

Модуль стабілізації відеопослідовності націлений на придушення небажаної вібрації руху внаслідок механічного тремтіння камери й погодних умов, а також на синтез нових зображень послідовності з урахуванням стабілізації траєкторії руху камери. При використанні стаціонарних IP-камер на відкритому просторі тремтіння кадру може виникати при русі камери або під впливом вітру. Для поліпшення якості відеопослідовності розроблений алгоритм стабілізації, що відрізняється від відомих знаходженням мінімальної відстані між

блоками й заснований на виводі функції вартості, шляхом відновлення границь кадру й за допомогою тимчасових фільтрів для поліпшення якості відеопотоку.

Після одержання кадру перебувають локальні вектори руху із застосуванням методу відповідності блоків. Для кожного блоку пікселів на попередньому кадрі шукається відповідний блок на поточному кадрі [1]. Функція порівняння переданих блоків, використовуючи метрику PSNR, розраховує різницю між кадрами. Розрахунок глобального вектора руху виробляється на основі знайдених локальних векторів за формулою:

$$GVx = \frac{1}{MN}\sum_{i=1}^{n-1}\sum_{j=1}^{m-1} LVx(i,j),$$
$$GVy = \frac{1}{MN}\sum_{i=1}^{n-1}\sum_{j=1}^{m-1} LVy(i,j). \quad (3.7)$$

Після знаходження глобального вектора руху кадру виконується компенсація руху. Даний етап розкладається на два завдання:

– облік глобального руху кадру;

– відновлення границь, що випали за рамки кадру [2].

Коректування знайденого тремтіння кадру виконується шляхом зрушення наступного кадру в напрямку, зворотному знайденому вектору глобального руху. Таким чином, відбувається зсув зображення на трохи пікселів, при цьому необхідно враховувати зміну границі кадру. Щоб уникнути втрати області видимості, частина інформації для граничних пікселів береться з попередніх кадрів відеопослідовності.

Останнім етапом пропонованого алгоритму стабілізації відео є постобробка кадру. Для неї використовуються автоматична корекція, а також обробка модифікованим для декількох кадрів фільтром 2d_cleaner. У цьому фільтрі для кожного пікселя зображення розраховується значення з урахуванням околиці за формулою:

$$P_{x,y} = \begin{pmatrix} R = Ch(Ts, r) \\ G = Ch(Ts, r) \\ B = Ch(Ts, r) \end{pmatrix}. \quad (3.8)$$

де Px,y – піксель, з околицею якого ведеться робота; R, G, B – значення каналів RGB-Спектра; Ts – значення порога, що характеризує можливість обробки; r – ранг околиці; Ch(Ts, r) – функція розрахунку значення каналу в спектрі:

$$Ch(Ts,r)=\frac{\sum_{i=1}^{r}\sum_{j=1}^{r}Sv(i,j)}{\sum_{i=1}^{r}\sum_{j=1}^{r}Cc(i,j)}, \tag{3.9}$$

де Sv(i, j) – функція відсікання значення спектра по порозі; Cc(i, j) – функція вказівки придатності значення спектра по порозі, значення якої визначається за формулою:

$$Sv(i,j)=\begin{cases} sv_{i,j}, & \text{если } |sv_{i,j}-sv_{0,0}|\le Ts, \\ 0, & \text{если } |sv_{i,j}-sv_{0,0}|>Ts, \end{cases}$$

$$Cc(i,j)=\begin{cases} 1, & \text{если } |sv_{i,j}-sv_{0,0}|\le Ts, \\ 0, & \text{если } |sv_{i,j}-sv_{0,0}|>Ts, \end{cases} \tag{3.10}$$

де $sv_{i,j}$ – значення спектра розглянутого колірного каналу; Ts – значення порога.

Реалізація модифікованого фільтра 2d_cleaner виконана з урахуванням обробки по тимчасовій осі декількох кадрів відеопослідовності, що дозволяє зменшити вплив перешкод після стабілізації. У ході експериментів було з'ясовано, що для відеопослідовності, отриманої зі стаціонарної IP-камери, при виконанні такої фільтрації доцільне використання двох або трьох попередніх кадрів.

Нелінійне поліпшення якості зображення

Оскільки в більшості випадків при організації зовнішнього відеоспостереження виникає завдання поліпшення візуальної якості відеопотоку для його кращого сприйняття оператором, в основу модуля поліпшення якості зображень була покладена модифікований алгоритм Multi-Scale Retinex (MSR), що імітує візуальну систему людини [3]. Реалізація алгоритму у вигляді окремого модуля надає можливість через ядро системи робити поліпшення візуальної якості зображень у повноформатному виді тільки для відображуваного на екрані відеопотоку.

MSR-алгоритм стискає динамічний діапазон зображення зі збереженням (збільшенням) локального контрасту в погано і яскраво освітлених областях. Класичний багатомірний MSR-алгоритм [4] є зваженою сумою одномірних SSR-алгоритмів (Single-Scale Retinex) для різних масштабів. Одномірна вихідна функція i-го колірного каналу Ri(x, y, s) обчислюється в такий спосіб:

$$Ri(x, y, s)=\log\{Ii(x, y)\} - \log\{F(x,y,s)*Ii(x,y)\}, \quad (3.11)$$

где Ii(x, y) – вхідна функція i-го колірного каналу по координатах x і y; s – масштабний коефіцієнт; символ * позначає згортку функцій; F(x, y, s) – гауссіан, обумовлений як:

$$F(x, y, s)=Ke–(x2+y2)/s2. \quad (3.12)$$

При цьому коефіцієнт K вибирається таким чином, щоб виконувалася умова

$$\int\int_{W_{x,y}} F(x,y,\sigma)dxdy=1, \quad (3.13)$$

де Wx,y – безліч пікселів, що належать всьому зображенню.

Тоді багатомірна вихідна функція i-го колірного каналу RMi(x, y, w, s) визначається в такий спосіб:

$$R_{Mi}(x,y,w,\sigma)=\sum_{n=1}^{N} w_n R_i(x,y,\sigma_n), \quad (3.14)$$

де w=(w1, w2, …, wm), m=1, 2, …, M – ваговий вектор одномірних вихідних функцій i-го колірного каналу Ri(x, y, s); s=(s1, s2, …, sn), n=1, 2, …, N – вектор масштабів одномірних вихідних функцій. При цьому:

$$\sum_{n=1}^{N} w_n = 1.$$

Розмірність вектора масштабів звичайно вибирається не менше 3. При проведенні експериментів значення склали 13, 87, 180. Ваговий вектор w, як правило, має елементи з рівними значеннями.

Однак класичний алгоритм MSR приводить до перекручування кольору зображення, тому що значення кожної колірний складової пікселя (наприклад в RGB-просторі) заміняється відношенням її вихідного значення до середнього

значення даної колірний складової навколишніх пікселів. Існують кілька рішень даної проблеми. Так, деяке поліпшення результатів спостерігається при переході в інші колірні простори з явним поділом яркістної і відтіночної складових (HIS-, HSV-, HSL-простору). Однак кращий ефект досягається при використанні моделі нормалізованого поділу яркістної і відтіночної складових. Додаткова обробка виконується відповідно до вираження

$$R¢Mi(x, y, w, s, b)=RMi(x, y, w, s)*I?i(x, y, c), \quad (3.15)$$

де I¢i(x, y, c) – нормалізована яскравість, що визначається за формулою:

$$I_i'(x,y,c)=\log\left[1+c\frac{I_i(x,y)}{\sum_{i=1}^{s}I_i(x,y)}\right], \quad (3.16)$$

де c – коефіцієнт, обираний із середини діапазону значень [0…255], c=100-125...

Через характеристики логарифмічної функції MSR-алгоритм робіть деталі зображення більше помітними в тіньових областях, чим у засвічених [3]. Щоб зробіти деталі помітними в засвічених областях, можна застосувати логарифмічну функцію до інвертованого зображення. Будується модифікована логарифмічна функція L(I(x,y)), що залежить від граничного значення Th, обираного користувачем:

$$L(I(x,y))=\begin{cases}k_1\cdot\log(I(x,y)), & \text{если } I(x,y)<Th,\\ -k_2\cdot\log(DR-I(x,y))+\log(DR), & \text{если } I(x,y)\geq Th,\end{cases} \quad (3.17)$$

де:

$$k_1=\frac{\frac{Th}{DR}\cdot\log(DR)}{\log(Th)}, \quad k_2=\frac{\left(1-\frac{Th}{DR}\right)\cdot\log(DR)}{\log(DR-Th)},$$

DR – динамічний діапазон зображення, у цьому випадку DR=255 (для зображень із 8 бітами на колірний канал); k1 і k2 – вагарні коефіцієнти; Th – граничне значення.

Граничне значення Th було обрано рівним 200.

Об'єднання галузей дає класичному алгоритму MSR можливість підвищувати контрастність деталей в областях підвищеної яскравості (для приклада на рисунку 2 область підвищеної яскравості виділена замкнутим контуром). Використання такого виду логарифмічної функції в певній мері підвищує контрастність деталей інформативної області типових розподілів яскравості (60-200), але в значно меншій мері, чим у затінених ділянках і ділянках з високою яскравістю.

Однак вихідного дані MSR-алгоритму з модифікованою логарифмічною функцією перебувають поза динамічним діапазоном зображення [3]. Щоб привести значення до динамічного діапазону зображення [0, 255], зробимо коректування вихідного зображення на основі коригувального коефіцієнта й середнього значення яркостей пікселів обробленого зображення:

$$I_{m}(x,y)=\frac{I_{m}(x,y)-I_{avg}(x,y)}{w_{cor}}+C, \quad (3.18)$$

де Im(x, y) – яскравість пікселя після обробки MSR-алгоритмом з модифікованою логарифмічною функцією; $I_{avg}(x, y)$ – середнє значення яскравості пікселів обробленого зображення; w_{cor} – коригувальний коефіцієнт, вибирається експериментальним шляхом (w_{cor}=0,007), за константу З береться половина динамічного діапазону зображення (C=127,5).

Експериментальні результати

Для оцінки ефективності й доцільності використання запропонованих методів було розроблене експериментальне ПЗ. Перевірка методів виявлення руху й збір статистики здійснювалися з використанням відеоданих з дев'яти IP-камер, розташованих як на жвавих вулицях, так і в місцях з мінімальним рухом. Дослідження проводилося на 57 послідовностях по 1500 кадрів у кожній. Порівняння в розробленому методі руху здійснювалося при однаковому способі обчислення величини руху, відмінність полягало в різних способах формування значення абсолютної різниці, тобто використанням трьох основних підходів.

Найбільшу ефективність запропонований метод показав у колірній моделі RGB з використанням фактора колірного посилення 4. Для з'ясування можливості прискорення роботи алгоритму обчислення величини рухи був проведений ряд експериментів. Для оцінки швидкодії й точності роботи детектора рухи використовувалися кадри різного масштабу, тому що зменшення аналізованого зображення дає істотне збільшення швидкості роботи. У таблиці наведені дані про точність визначення рухи й рівні помилкового визначення руху.

Таблиця 3.1 – Якість детектування руху залежно від масштабу кадрів

Показник якості, %	Масштаб кадрів, %							
	100	70	60	50	40	30	20	10
Точність визначення руху	94,31	95	94,5	93,77	95,01	94,27	88,4	78
Помилкове визначення руху	2,78	3,19	3,6	3,28	5,6	4,1	7,8	23,8

Для оцінки якості стабілізації було проведене тестування методом, запропонованим у роботах зі стабілізації відеоматеріалу [1]. Для вихідної відеопослідовності й стабілізованого відеоматеріалу перебуває різниця між попереднім і поточним кадрами по метриці PSNR. Застосування просторово-тимчасового фільтра 2d_cleaner при стабілізації відеоматеріалу значно поліпшує якість, усуваючи сліди дрібного тремтіння відеопослідовності, від якого не вдалося позбутися при стабілізації. PSNR-значення стабілізованої послідовності помітно вище, що показує меншу різницю між кадрами (більше низьке тремтіння кадру). При застосуванні тимчасового фільтра PSNR-значення відеопослідовності збільшується в межах 30 %. Перевірка модуля поліпшення візуальної якості виконувалася на послідовностях, що містить перекручування, викликані різними природними явищами й умовами освітлення.

На закінчення можна зробити наступні виводи. Пропонована система відеоспостереження з можливістю поліпшення візуальної якості дозволить підвищити ефективність роботи служби охорони завдяки можливостям поліпшення візуальної якості відображуваних даних і стабілізації відеопотоку. Запропоновані детектор руху й формат зберігання відеоданих дозволять більш гнучко управляти переглядом подій без необхідності виконання розрахунку параметрів руху при кожному перегляді відеоархіву. При зміні масштабу від 100 до 30 % фактор руху, що характеризує точність визначення руху, перебуває на схожому рівні, а при масштабі менш 15 % значно погіршується. Використання стабілізації відеопотоку дозволяє підвищити якість знаходження руху в середньому на 2-7 % залежно від ступеня тремтіння камери, оскільки цей фактор значно впливає на роботу алгоритму. Застосування стабілізації відеопотоку також поліпшує можливості візуального спостереження для оператора.

3.2 Розробка структурної схеми

Для розробки системи було вирішено використовувати модульний підхід, оскільки застосування модульної організації при реалізації системи дозволить надалі з мінімальними вкладеннями виконувати модернізацію або розширення системи в цілому. Модульна архітектура також надає більше великі можливості по налагодженню.

Основними модулями пропонованої системи відеоспостереження є модулі поліпшення якості зображення, детектора руху й стабілізації відео. Крім них, з ядром системи зв'язані модулі взаємодії з камерами, із БД і модулі користувальницького інтерфейсу

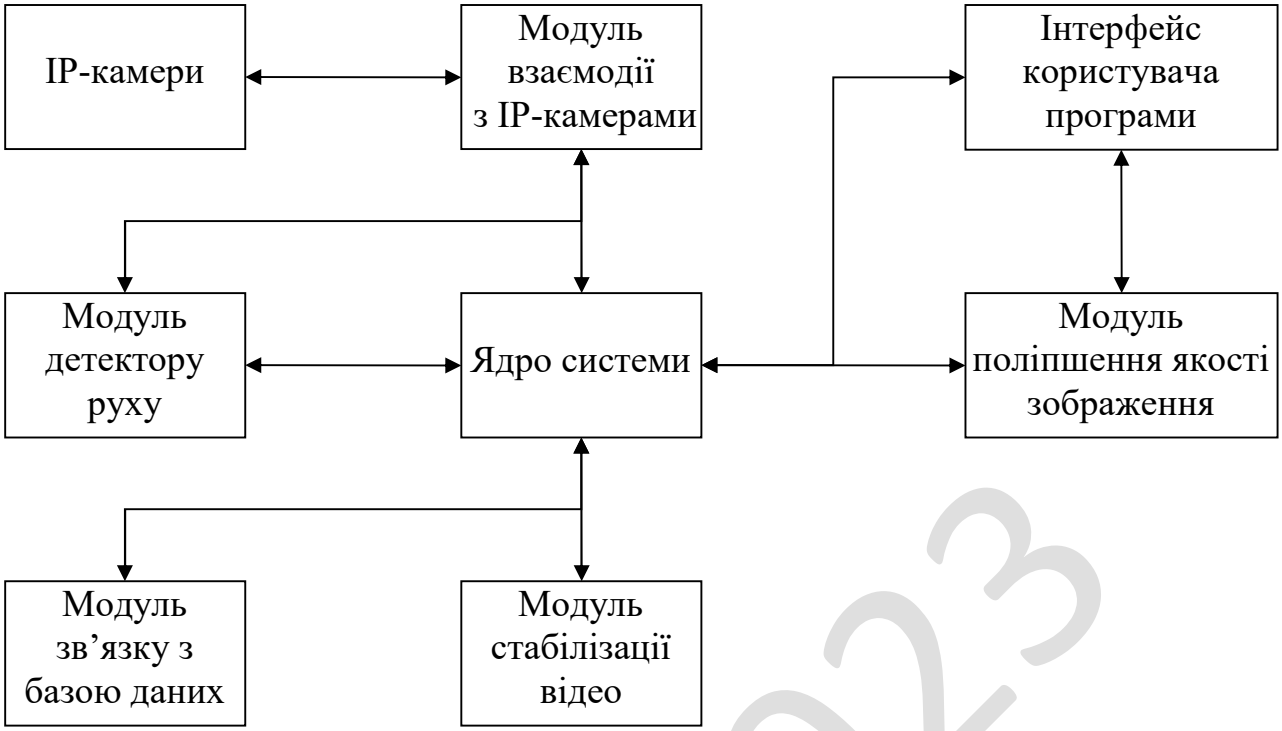

Рисунок 3.1 – Структурна схема системи

При отриманні зображення з веб-камери існує ряд артефактів та іншого, зображення в деяких випадках має вкрай малу якість. При установці на території установ комплексів відео спостережень, фірма звільняє десятки співробітників, що у результаті дає пряму фінансову вигоду і підвищує працездатності колективу.

Але якщо використовуються недосконалі алгоритми роботи це приводить до краху системи безпеки. У зв'язку з швидкими темпами розвитку устаткування відеоспостереження вітчизняного виробництва, внутрішньому ринку систем безпеки потрібні універсальні алгоритми обробки потоків відео інформації.

Розглянемо алгоритми детальніше. Алгоритм збільшення частоти кадрів відео потоку на основі піксельної компенсації руху це алгоритм перетворення частоти кадрів відео потоку, що дозволяє збільшувати швидкість кадрів у довільне ціле число раз. Превагою розробленого методу є його стійкість до складності руху, що досягається за рахунок маскування проблемних ділянок кадру, фільтрації векторів і оцінки точності векторів у кожному пікселі кадру.

Висока точність інтерпольованих кадрів також досягається за рахунок обробки накладень і піксельно-векторного поля. Останнє обчислюється з використанням оригінального методу восьмикратного підвищення векторного поля.

Також використовується алгоритм відстеження точкових особливостей у відео послідовностях це алгоритм виявлення й відстеження точкових особливостей у відео послідовностях ґрунтуються на припущенні, що різкість зображень змінюється плавно протягом всієї послідовності.

Алгоритм може відслідковувати мережні відео послідовності, отримані за допомогою ручних відеокамер із системою автоматичного підстроювання фокусування (WEB-камери), у яких часто присутні значні перепади різкості між сусідніми кадрами. Зміна різкості та придушення ефекту тремтіння кадру – основане на алгоритмі придушення тремтіння кадру, що будує модель руху.

Програмні можливості: Можливість обробки всіх видів вхідних відео потоків інформації за допомогою використання WDM драйверів, таких пристроїв як веб-камера, відеокамера і т.п.; Блок ведення статистики роботи програми; Блок обробки відео даних (з обробкою); Система автоматичного збереження відео потоку.

Характеристики ПЗ:

– Малий розмір програмного забезпечення (оптимізований код).

– Розширена довідкова система.

– Перевірка підключення веб-камери до хост-контролера USB.

– Модульна структура програмного забезпечення.

– Трансляція відео потоку у глобальну мережу при необхідності.

Відеопотік може бути оброблятись як у режимі динамічному (on-line трансляція) так і у статичному вигляді (off-line відео запис). Програма фільтрує, компенсує та коректує відео сигнал.

Розроблені алгоритми крім реалізованого ПЗ можливо застосовувати в інших програмах за допомогою підключення DLL файлу у стороні розробки. В

системах відео спостереження на заводах, продовольчих магазинах самообслуговування, на територіях паркінгів, книгарнях. В комплексах відео спостереження, систем правоохоронних органів початкового рівня. Як комплекс вивчення характеристик і властивостей відео і аудіо сигналу у вищих учбових закладах. В домашніх системах вуличного або кімнатного відео спостереження. Для передачі масивів інформації через глобальну мережу Інтернет. Як комплекс вивчення алгоритмів обробки зображень.

Програмне забезпечення, який виконує стиснення називається кодек (кодер/декодер). Стиснення зі швидкістю до 1:500 може бути досягнута. У результаті цифровий потік 1 і 0 підрозділяється на помічені пакети, які потім передаються через цифрові мережі будь-то (як правило, ISDN або IP). Використання аудіо-модемів в лінії передачі дозволяють за використання POTS або Plain Old Телефонна система, в деяких низькошвидкісних додатків, таких як відеотелефонія, тому що вони перетворюють цифрових імпульсів в / з аналогової хвилі в звуковому діапазоні спектра .

Інші компоненти, необхідні для системи VTC включають в себе:

– Відео-вхід: відео камери або веб-камери.

– Відео вихід: комп'ютерний монітор, телевізор або проектор.

– Аудіо вхід: мікрофони, CD / DVD-плеєр, касетний плеєр, або будь-якого іншого джерела аудіо вихід перед підсилювача.

– Аудіо вихід: як правило динаміки пов'язаних з пристроєм відображення або по телефону.

– Передача даних: аналоговий або цифрової телефонної мережі, локальної мережі або Інтернет.

Існують два основних види систем VTC. Виділені системи мають всі необхідні компоненти упаковані в одну частину обладнання, як правило, консоль з високою якістю з дистанційним управлінням відеокамерою. Ці камери можна керувати на відстані, щоб пересунути вліво і вправо, нахил вгору і вниз, і масштаб. Вони стали відомі як камери PTZ. Консоль містить всі електричні

інтерфейси, керуючого комп'ютера, а також програмного забезпечення або апаратного кодека. Все направлений мікрофон, підключеним до консолі, а також ТБ-монітор з гучномовцями та / або відео-проектор.

3.3 Розробка функціональної схеми

На рисунку 3.4 зображена функціональна схема системи. Нижче розглянемо її більш докладно. Для того, щоб зрозуміти, як функціонально працює система на рисунках 3.2 та 3.3 наведені відповідно наступні алгоритми:

– Опис алгоритму збільшення частоти кадрів відео потоку на основі піксельної компенсації руху та подальше придушення ефекту тремтіння кадру у VideoStream.

– Опис алгоритмів придушення тремтіння кадру у VideoStream з використанням Image-Based Rendering.

Наведемо спершу опис цих алгоритмів.

Розроблений алгоритм дозволяє збільшувати частоту кадрів VideoStream в довільне ціле число разів. Перетворення частоти проводиться за рахунок збільшення загального числа кадрів VideoStream – додавання нових (інтерпольованих) кадрів. Нові кадри вставляються у вихідну відео послідовність таким чином, що число інтерпольованих кадрів між будь-якими двома сусідніми вихідними кадрами одне і те ж і пропорційно кратності.

Розглянемо як він працює. Одна ітерація алгоритму починається з обчислення карти відео векторів далі проводиться сегментація даних та обчислення інтерпольованих кадрів. З правого боку описано основну формулу алгоритму.

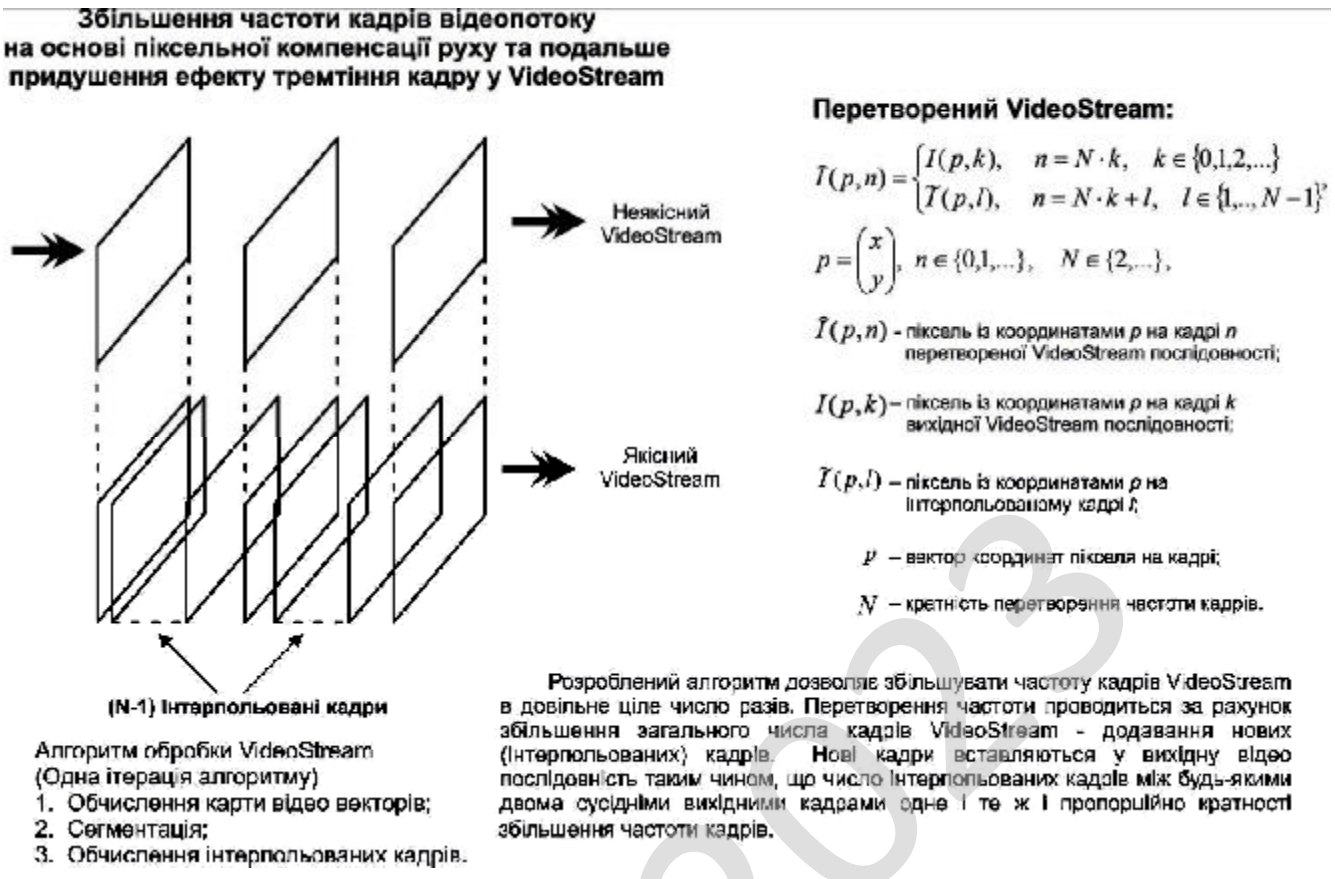

Рисунок 3.2 – Опис алгоритму збільшення частоти кадрів відео потоку на основі піксельної компенсації руху та подальше придушення ефекту тремтіння кадру у VideoStream

На рисунку 3.3 наведено опис додаткових алгоритмів які використовувались при розробці магістерського проекту – придушення ефекту тремтіння кадру у VideoStream з використанням Image-Based Rendering.

Ефект тремтіння кадру виникає через неякісну лінію зв'язку з мережею Інтернет. Спочатку проходить визначення яскравості пікселя з обчисленням наведеним у формулі.

Далі проходе розрахунок схеми розташування локальних сусідів пікселя та схеми розташування глобальних сусідів пікселя з обчисленням координат поточного пікселя на інтерпольованому кадрі з врахуванням вектора руху у наведеній формулі.

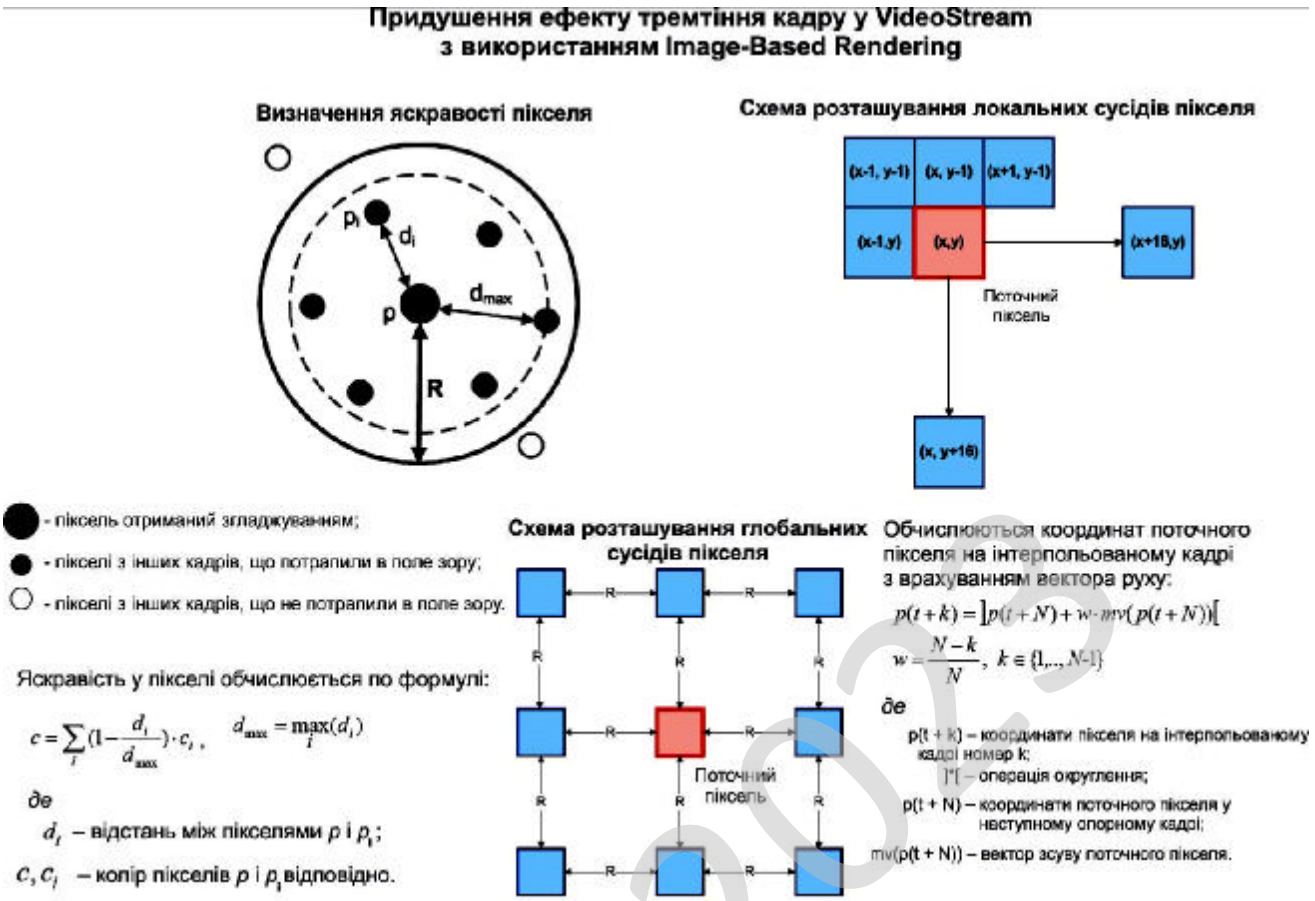

Рисунок 3.3 – Опис алгоритмів придушення тремтіння кадру

Основне призначення системи це обробка відео потоку застосовуючи алгоритм піксельної компенсації руху.

Функціональна схема розробленої системи зображена на рисунку 3.4. З рисунку видно, що розроблена система працює на алгоритмі збільшення частоти кадрів відео потоку на основі піксельної компенсації руху та подальше придушення ефекту тремтіння кадру у VideoStream.

Програмний продукт, забезпечує універсальну систему обробки вхідного потоку, забезпечує гарантовано стійкій відео сигнал з мінімальною кількістю шумів та з максимальною різкістю зображення яка ґрунтується на нових алгоритмах обробки відеозображень та частково модернізованих існуючих.

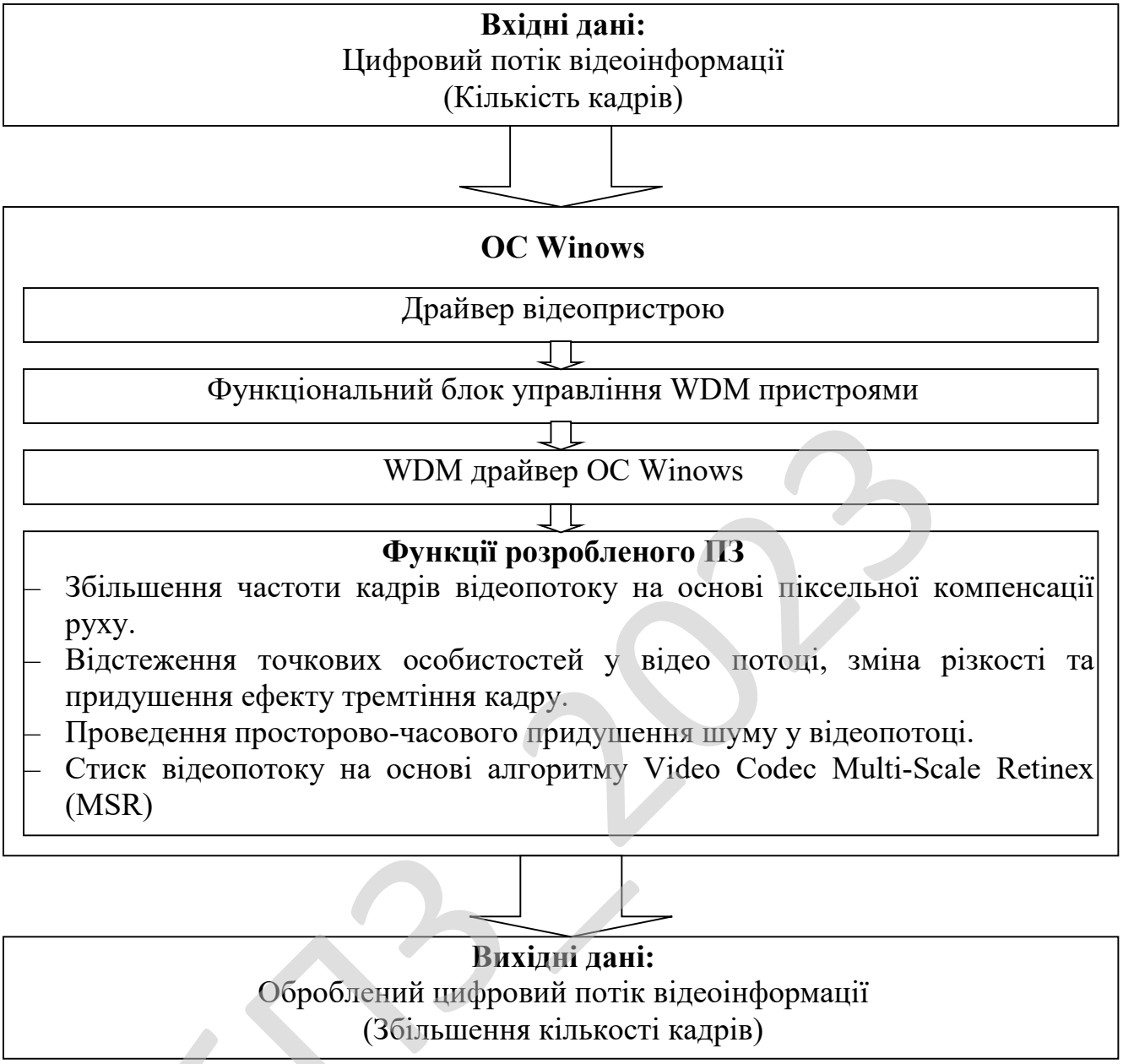

Рисунок 3.4 – Функціональна схема системи

Розглянемо її, спочатку ми отримуємо вхідні дані – цифровий потік відео інформації як він є, тобто неякісний, нечіткий. Він має N кількість кадрів.

До ОС Windows 8 він потопляє через підсистему драйверів (WINDOWS DRIVER MODEL).

Спочатку до драйверу відео пристрою потім до блоку управління WDM пристроями далі до WDM драйверу операційної системи.

Після проходження такої ланки розроблене ПЗ може його обробляти що і

проходить.

Розроблене ПЗ приймає відео потік та застосовує наступні алгоритми:

– Збільшення частоти кадрів VideoStream на основі піксельної компенсації руху.

– Відстеження точкових особливостей у VideoStream, зміна різкості та придушення ефекту тремтіння кадру.

– Проведення просторово-часового придушення шуму у VideoStream.

– Стиск відео потоку на основі алгоритму Video Codec Multi-Scale Retinex (MSR).

В кінцевому випадку на виході ми отримуємо оброблений відеопотік зі збільшеною кількістю кадрів N+A.

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

3.4 Розробка діаграми процесів

Діаграма взаємодії процесів системи, розробленої у результаті виконання магістерського проектування, наведена на рисунку 3.5.

Після початку роботи розробленого ПЗ ми потрапляємо до головного вікна ПЗ звідки можемо потрапити до налаштувань ПЗ та через модуль захисту ПЗ до вікна відображення VideoStream.

Далі за допомогою бібліотек обробки VideoStream та трансляції провести захоплення VideoStream, стабілізацію VideoStream та кінцеве підвищення якості VideoStream

Програмний продукт, забезпечує універсальну систему обробки вхідного потоку, забезпечує гарантовано стійкій відео сигнал з мінімальною кількістю шумів та з максимальною різкістю зображення яка ґрунтується на нових алгоритмах обробки відеозображень та частково модернізованих існуючих.

Пропонована програмна система призначена для поліпшення якості відеоматеріалу, отриманого в складних умовах освітлення, при організації відеоспостереження об’єктів.

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.

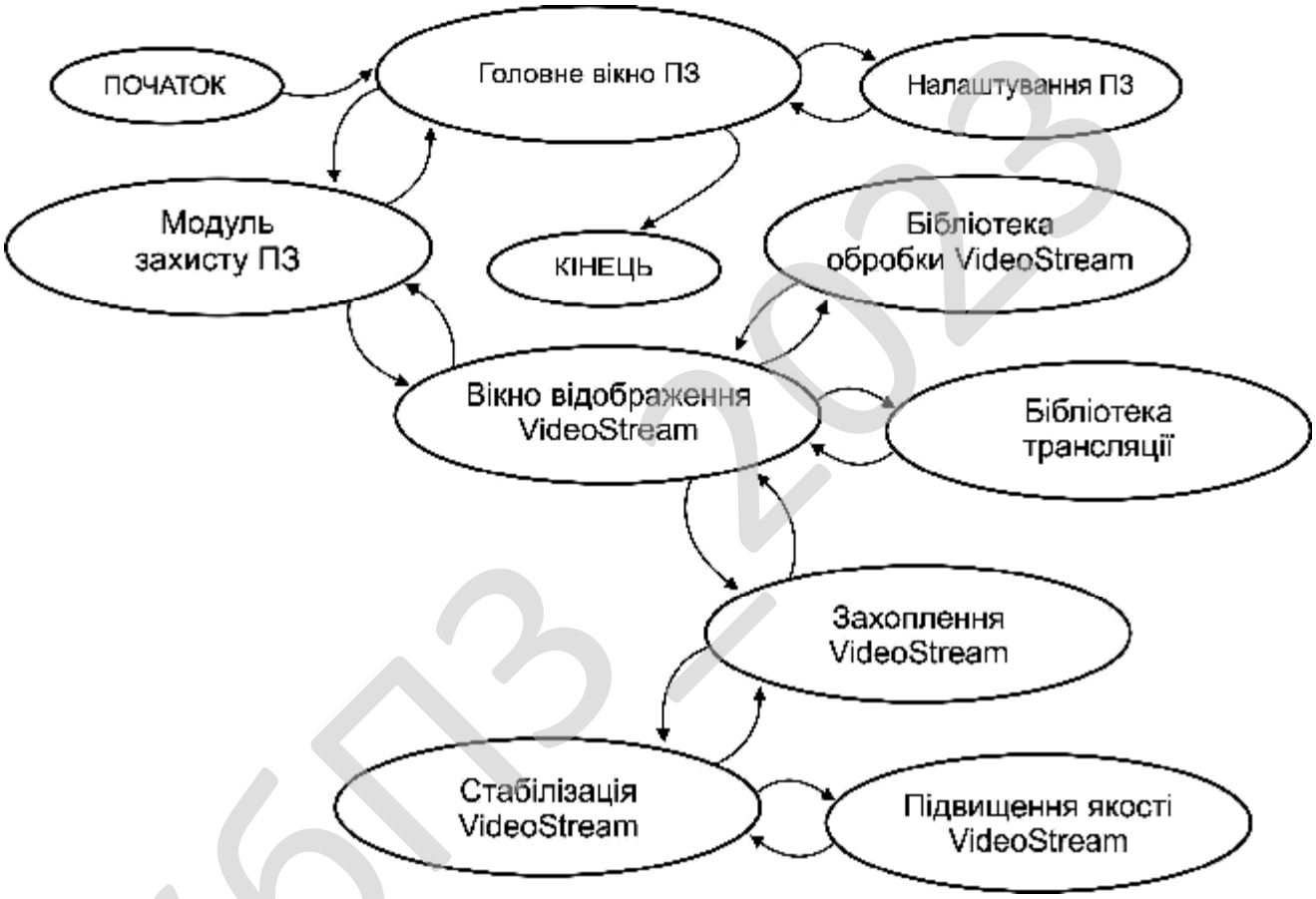

Рисунок 3.5 – Діаграма взаємодії процесів

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:

– Ініціалізація ПЗ.

– Підключення бібліотеки стабілізації відеопотоку.

– Підключення бібліотеки підвищення якості відеопотоку.

– Запит – є доступ до мережі?

– Запит – WDM пристрій знайдено?

– Файл налаштувань присутній?

– Налаштування значень VideoStream.

– Запит – WDM пристрій знайдено?

– Підпрограма стабілізації та підвищення якості відеопотоку.

– Проведення просторово-часового придушення шуму.

– Швидкісний тест кадру VideoStream.

– Стиск VideoStream.

– Трансляція VideoStream.

– Запит WM_CLOSE?

– Виведення вікна завершення роботи ПЗ.

– Відключення WDM пристрою.

– Виникли помилки при виконанні програми?

– Виведення повідомлення з тілом помилки.

– Відключення додаткових модулів.

– Звільнення ресурсів програми.

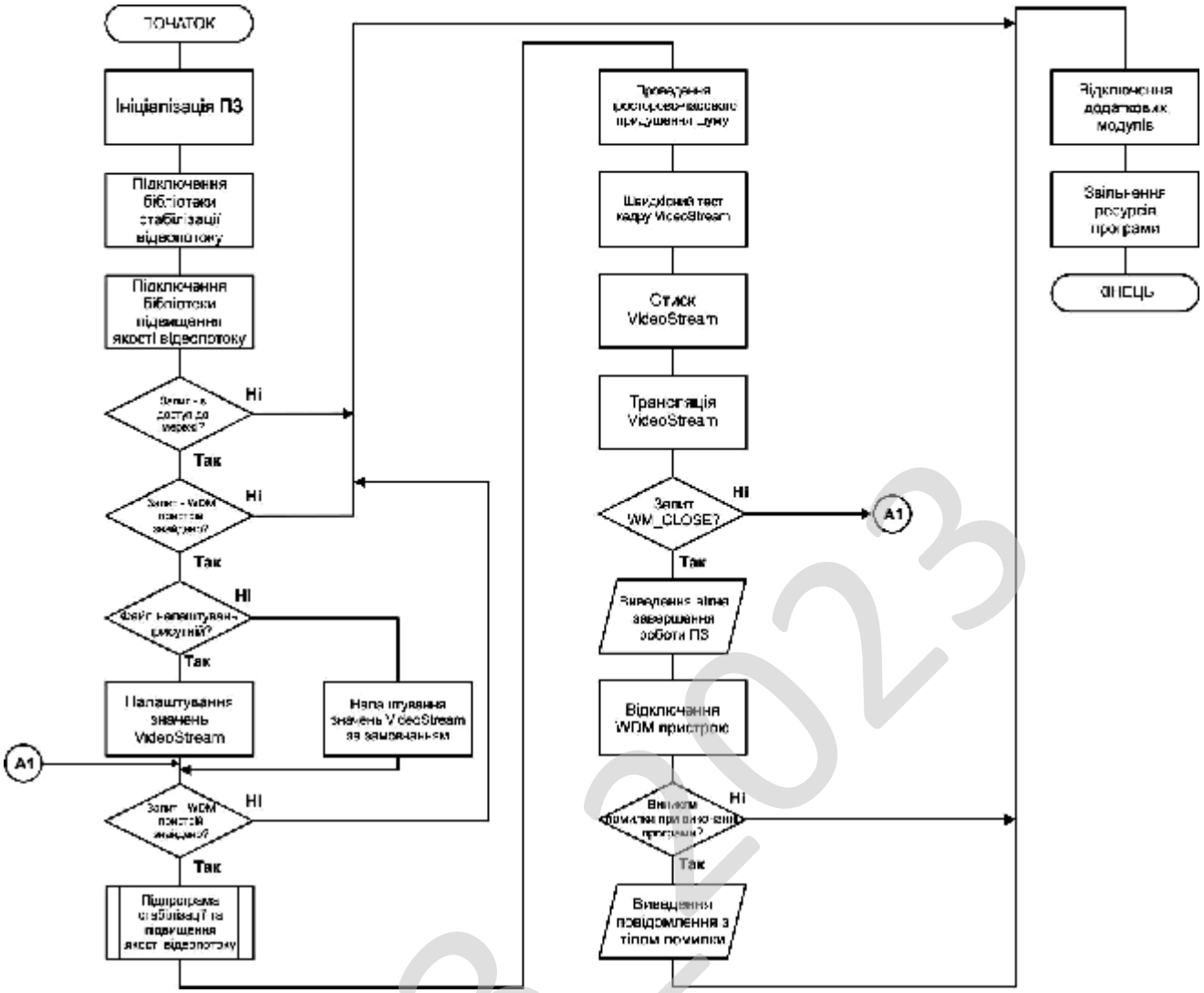

Рисунок 4.1 – Блок-схема основної програми

На рисунку 4.2 наведено блок-схему підпрограми стабілізації та підвищення якості відеопотоку. Її робота складається з виконання наступних кроків:

– Завантаження потокової VideoStream інформації.

– Конвертування кадрів VideoStream інформації.

– Аналіз кадру VideoStream інформації.

– Формування корегую чого пакету.

– Стек заповнений?

– Очищення стека.

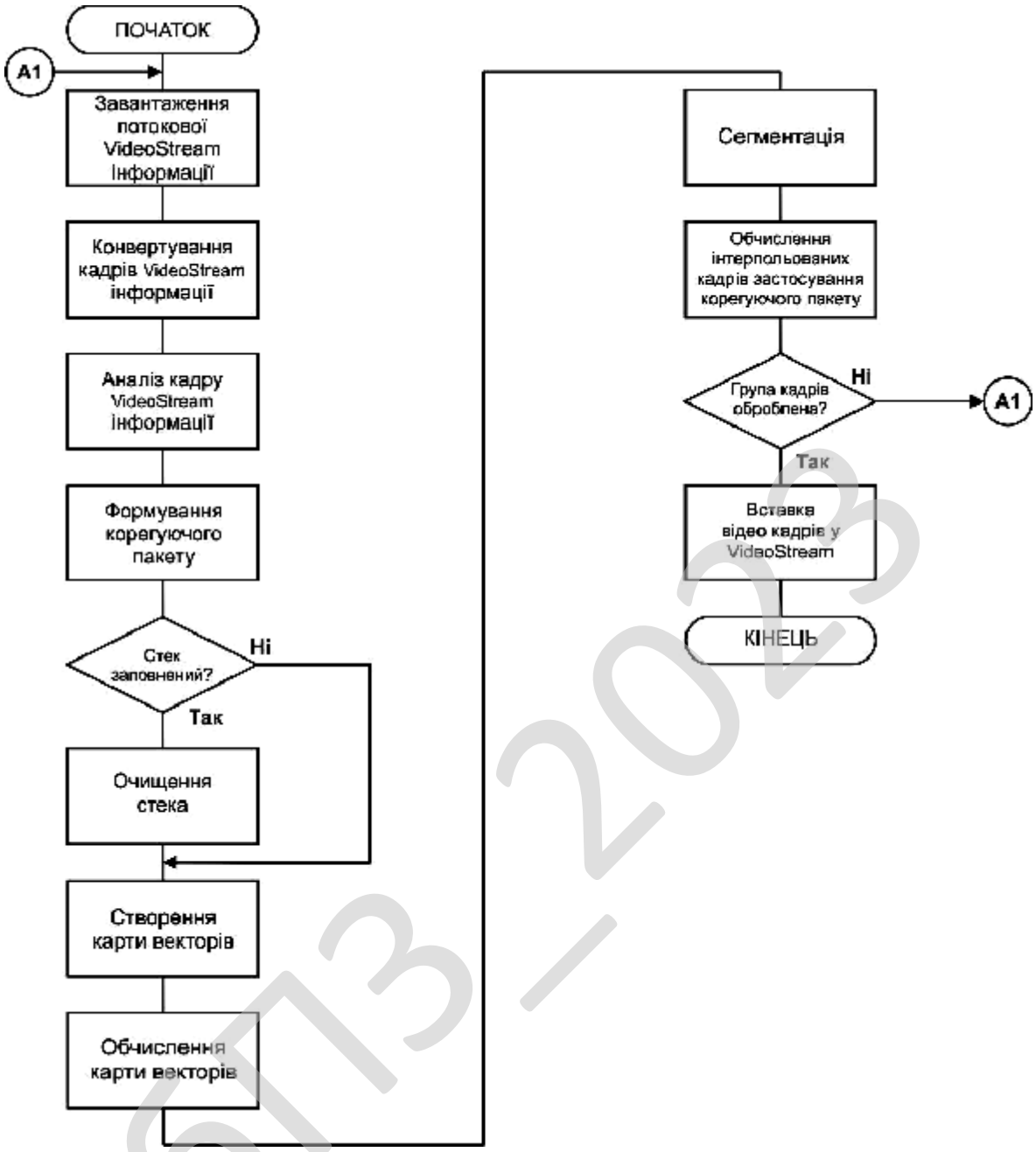

Рисунок 4.2 – Блок-схема підпрограми стабілізації та підвищення якості відеопотоку

– Створення карти векторів.

– Обчислення карти векторів.

– Сегментація.

– Обчислення інтерпольованих кадрів застосування корегуючого пакету.

– Група кадрів оброблена?

– Вставка відео кадрів у VideoStream.

Опис алгоритмів функціонування системи

Розрахунок алгоритму взаємодії з потоком відео інформації. При складанні блок-схем програмного забезпечення і напрацювання алгоритмів взаємодії з відео потоком я зіткнувся з масою проблем, які вимагали напрацювання процедур і функцій над основною проблематикою розглянемо частину написаного вихідного коду для захвату декількох потоків відео інформації та їх обробки.

```
//Підключення бібліотек
       #include <vcl.h>
       #include <vfw.h>
       #include <Clipbrd.hpp>
       #pragma hdrstop
//---------------------------------------------------------------------
       __fastcall TForm1::TForm1(TComponent* Owner): TForm(Owner)
       {        pCB = Clipboard ();
              Buffer = new Graphics::TBitmap;
//Графічний буфер
       }
//---------------------------------------------------------------------
//Підключаємо першу Web камеру
       void __fastcall TForm1::Button1Click(TObject *Sender)
       {
//Створення потоку
hWndCLeft   =   capCreateCaptureWindow   ("",   WS_CHILD,   this->Left,   this->Top,
 this-  >Width, this->Height, this->Handle, 11011);
// Підключення Web камери
       capDriverConnect (hWndCLeft, 0);
// Встановлення параметрів таймеру для оновлення кадрів Web камери
      Timer1->Interval = cseTime->Value;
// Включення таймера
              Timer1->Enabled = true;
              Canvas->Pen->Width = 4;
              Canvas->Pen->Color = clRed;
// Заголовок
              Caption = IntToStr (Buffer->Width) +":"+IntToStr (Buffer->Height);}
//---------------------------------------------------------------------
//Підключаємо другу камеру
       void __fastcall TForm1::Button2Click(TObject *Sender)
// Створення потоку
```

```
        hWndCRight = capCreateCaptureWindow ("", WS_CHILD, this->Left, this->Top,
                                             this->Width,              this->Height,
                                             this->Handle, 11011);
// Підключення Web камери
        capDriverConnect (hWndCRight, 0);
        }
        //Обробка виклику події таймера
        void __fastcall TForm1::Timer1Timer(TObject *Sender)
        {
                capGrabFrame (hWndCLeft);
                if (capEditCopy (hWndCLeft)) {
    Buffer->LoadFromClipboardFormat(CF_BITMAP, pCB->GetAsHandle (CF_BITMAP), 0);
                        int Hmin;
                        int x = 10, y = 10;
                        int z=0;
                        for (int j = 0; j < Buffer->Height; j++) {
        for (int i = 0; i < Buffer->Width; i++) {
        if (i == 0 && j == 0) Hmin = (int) Buffer->Canvas->Pixels[j][i];
        else if (Hmin < (int) Buffer->Canvas->Pixels[j][i]) {
        Hmin = (int) Buffer->Canvas->Pixels[j][i];
                                                x = j;
                                                y = i;
                                                z++;
                                        }
                                }
                        }
Caption = IntToStr (Buffer->Width) + " : " + IntToStr (Buffer->Height)+ " - " +
                    IntToStr (x) + " : " + IntToStr (y);
                        Buffer->Canvas->Pen->Color=0x000000ff;
                        Buffer->Canvas->Pen->Style=psSolid;
                        Buffer->Canvas->Ellipse(10 + x, 10 + y, x + 20, y + 20);
                        Canvas->Draw (10, 10, Buffer);
                }
        }
//---------------------------------------------------------------------------
        void __fastcall TForm1::Button3Click(TObject *Sender)
        {//Ручне відключення від Web камери 1
                capDriverDisconnect (hWndCLeft);
        //Ручне відключення від Web камери 2
                capDriverDisconnect (hWndCRight);}
//---------------------------------------------------------------------------
        void __fastcall TForm1::FormClose(TObject *Sender, TCloseAction &Action)
```

```
        {//Автоматичне відключення від Web камери 1
                capDriverDisconnect (hWndCLeft);
// Автоматичне відключення від Web камери 2
                capDriverDisconnect (hWndCRight);}
//-------------------------------------------------------------
```

Будь-яка вертикальна синхронізація, що виявлена в межах 64 ряду VWINDOW, скине лічильники, що керують сигналом VACTIVE і місцем розташування наступної очікуваної вертикальної синхронізації.

Повне блокування (настроювання якого ±1 такт) досягнуті 12 рядків коли фактичний горизонтальний імпульс придбаний. Це відбувається незалежно від вертикального циклу.

Коли вертикальне блокування досягнуто, 64 ряд VWINDOW відкриє 32 ряду до наступного очікуваної вертикальної синхронізації. Якщо вертикальна синхронізація виявлена вперше, ніж VWINDOW відкривається, це буде ігноруватися. VRESET буде виводитися в очікуваній вертикальній синхронізації місці розташування, чи був вертикальна синхронізація виявлений чи ні.

Якщо ніяка вертикальна синхронізації не виявлена в межах VWINDOW, вертикальна синхронізації, виявлена до наступного VWINDOW буде прийнятий як законний, і вертикальний вибір годині буде скинутий, щоб синхронізувати це до цієї вертикальної синхронізації. VRESET буде виводитися і лічильники, що керують сигналом VACTIVE, місце розташування наступного очікуваного вертикальної синхронізації, і VWINDOW буде скинуто.

Тому, що лічильник VWINDOW скинутий, VWINDOW залишиться відкритим для 32 рядків після вертикальної синхронізації.

Будь-яка додаткова вертикальна виявлена синхронізації, у тієї годину як VWINDOW відкритий, також скине лічильники, що керують сигналом VACTIVE і місцем розташування наступного очікуваного вертикальної синхронізації.

Припущення, що рівень DC є тим самим, вертикальне блокування, буде досягнуте в 1-2 полях.

Зсуви DC між двома сигналами через незафіксоване відео можуть розширювати(продовжувати) вертикальне блокування (чим більший зсув DC, тим довше блокування година).

Для швидкого вертикального блокування рекомендується, щоб усі введення були DC, відновлена до загального(звичайному) рівню DC до переключення. Це може бути легко виконано, просто використовуючи діод, щоб фіксувати, щоб заснувати.

Горизонтальне блокування здійснене як цифровий Phase Lock Loop (PLL – Цикл Блокування Стадії), без скидання. Якщо горизонтальна синхронізація виявлена, і HWINDOW відкритий, HWINDOW закритий, і сигнал помилки з генерований заснованим на розходженні між виявленим горизонтальної синхронізації місцем розташування і його очікуваним місцем розташування. Цей сигнал помилки поданий потім, щоб коректувати наступне очікуване горизонтальної синхронізації місце розташування. HWINDOW відкриє 12 тактів до наступного очікуваної горизонтальної синхронізації, і не буде закривати поки горизонтальна синхронізація не буде виявлена. HRESET буде виводитися в кожнім очікуваному горизонтальній синхронізації місці розташування.

Обробка обчислених пікселів

Ціль даної операції – видалення пікселів, обчислених з використанням неправильних векторів руху. Такі пікселі звичайно розташовані невеликими групами, оточеними порожніми пікселями. Цей етап є модифікацією звичайної морфологічної операції, що зветься звуженням.

На даному етапі проводиться видалення невеликих окремих груп пікселів з інтерпольованого кадру. Значення в точках інтерпольованого кадру, що перебувають в позиціях пікселів, будуть обчислені на наступних етапах. Обробка проводиться по пікселям інтерпольованого кадру зверху вниз, ліворуч праворуч.

Для кожного порожнього пікселя обчислюється його значення. Спосіб обчислення залежить від типу блоку, у якому перебуває піксель наступного опорного кадру з тими ж координатами, що й поточний піксель.

Для обчислення значення пікселя необхідно визначити його вектор зсуву.

Пошук вектора зсуву виробляється в кілька етапів. Спочатку пошук виконується серед векторів з малої околиці даного пікселя. Потім, якщо підходящий вектор не був знайдений, пошук виробляється серед векторів руху більше вилучених пікселів. Якщо після цього підходящий вектор руху не був знайдений, використається альтернативний вектор.

Більш детально алгоритм обчислення значення пікселя виглядає в такий спосіб:

1. Формування локального набору кандидатів формується набір векторів зсуву, кожний вектор якого є вектором – кандидатом поточного пікселя. Вектора даного набору вибираються з пікселів, розташованих у позиціях, зазначених на рисунку 4.5. Очевидно, що для формування набору кандидатів використаються тільки обчислені піксели.

2. Пошук вектора в наборі кандидатів: з набору кандидатів вибирається вектор з найменшою локальною помилкою. Якщо помилка цього вектора менше заданого порога, то даний вектор вибирається в якості поточного й виконується крок 5.

3. Додавання глобальних кандидатів у набір: у набір кандидатів додаються вектора обчислених пікселів, що відстоять від поточні на заданій відстані R. Ця відстань збільшується при кожному виконанні даного кроку. Якщо дана відстань стає більше заданого порога то обробка переходить на крок 4, інакше – на крок 2.

4. Обчислення альтернативного вектора: якщо набір кандидатів порожній, то як поточний вектор вибирається нульовий, інакше – вектор з набору з найменшою локальною помилкою.

5. Обчислення значення пікселя виробляється з використанням вектора, обраного в якості поточного. Поточний вектор і його локальна помилка зберігаються в спеціальних масивах, і обробка переходить до наступного пікселю.

Варто помітити, що в кожний момент часу набір кандидатів містить тільки «вірні» вектора руху, що значно збільшує точність передачі руху всього алгоритму.

Розглянемо розроблену методику повернення зображень на заданий програмістом кут. Рішення цього завдання не так тривіально, як може здатися на перший погляд. Особливо якщо враховувати той факт, що піксель (найменша складова зображення) насправді не є точкою. Якщо бути точним, то це квадрат зі сторонами 1x1. Виходячи із цього можна сформулювати алгоритм, за допомогою якого й можна реалізувати обертання.

Представимо, що зображення – це матриця пікселів. Тоді при обертанні зображення відбувається накладення вихідної матриці й поверненої на заданий кут. А оскільки одиничний піксель може бути тільки одного певного кольору, то необхідно правильно розподілити ці самі кольори.

Далі, щоб хоч якось розбавити теорію, представлена її графічна інтерпретація.

На лівому рисунку представлене вихідне зображення. На середньому малюнку видне, як відбувається перетинання матриць вихідного й повернутого зображень. Праворуч видний кінцевий результат роботи алгоритму – відбувається пропорційний розподіл кольорів по точкам піксельной матриці. Також можна зрівняти результати обертання рисунка за допомогою алгоритму й програми Photoshop.

Прототип функції

```
BITMAP aarot::rotate(HBITMAP src, double rotation,
                     aar_callback callbackfunc, int bgcolor,
                     bool autoblend)
```

Аргументи функції:

1. Scr: шлях до вихідного зображення.

2. Rotation: число градусів, на які необхідно повернути зображення (проти годинникової стрілки).

3. Callbackfunc (Callbackptr): покажчик на функцію callback. Ця функція стежить за тим, на який кут уже було повернене зображення.

4. Bgcolor: колір фону зображення, що де обертається, не накладається на вихідне.

5. Autoblend: повинні чи края зображення, що обертається, бути змішаним з кольором фону, заданим у змінній Bgcolor.

Використання функції Callback. Дана функція має такий вигляд:

```
bool Aarotcallbackfunc(double percentdone)
{
   ...
}
```

де percentdone відсоток виконання програми по обертанню зображення (0 це 0%, 1 це 100%).

Якщо ж функція callback() поверне true (що-небудь, крім 0), то робота алгоритму буде негайно завершена. Розглянемо реалізований алгоритм.

```
HBITMAP aarot::rotate(HBITMAP src, double rotation, aar_callback callbackfunc, int bgcolor, bool autoblend)
{
    polyoverlap = new aar_pnt[16];
    polysorted = new aar_pnt[16];
    corners = new aar_pnt[4];
    double dx[] = {0.0, 1.0, 1.0, 0.0};
// координати
    double dy[] = {0.0, 0.0, 1.0, 1.0};
    for (int i = 0; i < 4; i++)
    {
        corners[i].x = dx[i];
        corners[i].y = dy[i];
    }
//Обертання в градусах [0, 360)
    int mult = (int)rotation / 360;
    if (rotation >= 0)
        rotation = rotation - 360.0 * mult;
```

```
            else
                rotation = rotation - 360.0 * (mult - 1);
//Розрахунок значень cos і sin, які будуть використовуватися протягом
// всієї програми
            coss = aar_cos(rotation);
            sins = aar_sin(rotation);
           HBITMAP res = dorotate(src, rotation, callbackfunc, bgcolor, autoblend);
            delete [] polyoverlap;
            delete [] polysorted;
            delete [] corners;
            return res;
// повернення результату
        }
// виклик функції з різними вхідними параметрами №1
        HBITMAP aarot::rotate(HBITMAP src, double rotation, aar_callback
callbackfunc)
        {
            return rotate(src, rotation, callbackfunc, 0x00FFFFFF, true);
        }
// виклик функції з різними вхідними параметрами №2
        HBITMAP aarot::rotate(HBITMAP src, double rotation, aar_callback
callbackfunc, int bgcolor)
        {
            return rotate(src, rotation, callbackfunc, bgcolor, true);
        }
// виклик функції з різними вхідними параметрами №3
        HBITMAP aarot::rotate(HBITMAP src, double rotation, aar_callback
callbackfunc, bool autoblend)
        {
            return rotate(src, rotation, callbackfunc, 0x00FFFFFF, autoblend);
        }
```

Далі розглянемо, як саме алгоритм використовувався у формі програми.

```
        int WINAPI Winmain (HINSTANCE hthisinstance, HINSTANCE hprevinstance, LPSTR
lpszargument, int nfunsterstil)
        {
//Завантаження зображення яке буде обертатися
        HBITMAP hsrcbmp = (HBITMAP)Loadimage(NULL, IMAGENAME, IMAGE_BITMAP, 0, 0,
LR_DEFAULTSIZE | LR_LOADFROMFILE);
        int starttime = Gettickcount();
// Замір часу
        hrotbmp = aarot::rotate(hsrcbmp, DEGREES, callbackfunc, 0xffffff, true);
```

```
cout << "Повертання виконано за " << Gettickcount() - starttime << " за
мілісекунд" << endl;
            Deleteobject(hsrcbmp);
//видалення об'єкту
        }
// Одержуємо інформацію про зображення, для створення правильного розміру
// вікна для малювання
            Getobject(hrotbmp, sizeof(BITMAP), &bitmapinfo);
            HWND hwnd;
            MSG messages;
            WNDCLASSEX wincl;
            wincl.hinstance = hthisinstance;
            wincl.lpszclassname = "Rotator";
            wincl.lpfnwndproc = Windowprocedure;
            wincl.style = CS_DBLCLKS;
            wincl.cbsize = sizeof (WNDCLASSEX);
            wincl.hicon = Loadicon (NULL, IDI_APPLICATION);
            wincl.hiconsm = Loadicon (NULL, IDI_APPLICATION);
            wincl.hcursor = Loadcursor (NULL, IDC_ARROW);
            wincl.lpszmenuname = NULL;
            wincl.cbclsextra = 0;
            wincl.cbwndextra = 0; wincl.hbrbackground = (HBRUSH) COLOR_BACKGROUND;
            if (!Registerclassex (&wincl))
                return 0;
hwnd = Createwindowex (0, "Rotator", "Повернення зображення",
WS_OVERLAPPEDWINDOW,  CW_USEDEFAULT,  CW_USEDEFAULT,
bitmapinfo.bmwidth + 8, bitmapinfo.bmheight + 34, HWND_DESKTOP, NULL,
hthisinstance, NULL);
            Showwindow (hwnd, nfunsterstil);
            while (Getmessage (&messages, NULL, 0, 0))
            {
                Translatemessage(&messages);
                Dispatchmessage(&messages);
            }
            return (int)messages.wparam;//повернення
        }
```

Алгоритми машинної графіки можна розділити на два рівні: нижній і верхній. Група алгоритмів нижнього рівня призначена для реалізації графічних примітивів (математичних розрахунків, ліній, кіл, заповнень і т.п.).

Ці алгоритми або подібні їм відтворені в графічних бібліотеках мов високого рівня або реалізовані апаратний в графічних процесорах робочих станцій.

Серед алгоритмів нижнього рівня можна виділити наступні групи.

– Найпростіші в значенні математичних методів і відмінні простотою реалізації, що використовуються. Як правило, такі алгоритми не є як найкращими за об'ємом виконуваних обчислень або необхідним ресурсам пам'яті.

– Алгоритми з складнішими математичними передумовами, що відрізняють їх більшу ефективність.

– До третьої групи слід віднести алгоритми, які можуть бути без великих ускладнень реалізовані апаратний (допускаючи розпаралелювання, рекурсивні, реалізовувані в найпростіших командах). В цю групу можуть потрапити і алгоритми, представлені в перших двох групах.

– Нарешті, до четвертої групи можна віднести алгоритми із спеціальним призначенням (наприклад, для усунення сходового ефекту, тремтіння кадру та інші).

До алгоритмів верхнього рівня відносяться в першу чергу алгоритми видалення невидимих ліній і поверхонь. Задача видалення невидимих ліній і поверхонь продовжує залишатися центральною в машинній графіці..

До задачі видалення невидимих ліній і поверхонь примикає задача побудови (замальовування) півтонових (реалістичних) зображень, тобто обліку явищ, пов'язаних з кількістю і характером джерел світла, обліку властивостей поверхні тіла (прозорість, заломлення, віддзеркалення світла).

Проте, при цьому не слід забувати, що виведення об'єктів в алгоритмах верхнього рівня забезпечується примітивами, що реалізовують алгоритми нижнього рівня, тому не можна ігнорувати проблему вибору і розробки ефективних алгоритмів нижнього рівня.

Для різних областей застосування машинної графіки на перший план можуть висуватися різні властивості алгоритмів.

Для наукової графіки велике значення має універсальність алгоритму, при цьому швидкодія може відходити на другий план.

Для систем моделювання, відтворюючих об'єкти, що рухаються, швидкодія стає головним критерієм, оскільки потрібно генерувати зображення практично в реальному масштабі часу.

Особливості растрової графіки зв'язані з тим, що звичайні зображення, з якими стикається людина в своїй діяльності (креслення, графіки, карти, художні картини і т.п.), реалізовані на площині, що складається з нескінченного набору крапок.

Екран же растрового дисплея представляється матрицею дискретних елементів, що мають конкретні фізичні розміри.

При цьому число їх істотно обмежено. Тому не можна провести точну лінію з однієї крапки в іншу, а можна виконати тільки апроксимацію цієї лінії з відображенням її на дискретній матриці (площини).

Таку площину також називають цілочисельними гратами, растровою площиною або растром.

Ці грати представляються квадратною сіткою з кроком 1. Відображення будь-якого об'єкту на цілочисельні грати називається розкладанням його в растр або просто растровим уявленням.

4.2 Захист розробленого програмного забезпечення

Захист розробленого програмного забезпечення буде відбуватися за допомогою CRYPTON – алгоритм симетричного блочного шифрування (розмір блоку 128 біт, ключ довжиною до 256 біт), розроблений південнокорейським криптологом Чьо Лім Хун з південнокорейської компанії Future Systems, яка з кінця 1980-х років працює на ринку забезпечення мереж і захисту інформації.

Алгоритм був розроблений в 1998 році в якості шифру – учасника конкурсу AES. Як зізнавався автор, конструкція алгоритму спирається на алгоритм SQUARE[1]. В алгоритмі Crypton немає традиційних для блочних шифрів мережі Фейстеля. Основу даного шифру становить так звана SP-мережа (повторювана циклова функція, що складається із замін-перестановок, орієнтована на розпаралелену нелінійну обробку всього блоку даних). Крім високої швидкості, перевагами таких алгоритмів є полегшення дослідження стійкості шифру до методів диференціального та лінійного криптоаналізу, що є на сьогодні основними інструментами розтину блочних шифрів. На конкурс AES була представлена версія алгоритму Crypton v0.5. Однак, як казав Чьо Лім Хун, йому не вистачало часу для розробки повної версії. І вже на першому етапі конкурсу AES в ході аналізу алгоритмів, версія Crypton v0.5 була замінена на версію Crypton v1.0. Відмінність нової версії від первинної полягала в зміні таблиці замін та в модифікації процесу розширення ключа.

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

Шифрування

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

– Таблична заміна γ;

– Лінійне перетворення π;

– Байтова перестановка τ;

– Операція σ.

Таблична заміна γ

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

Лінійне перетворення π

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

Байтова перестановка τ

Дана перестановка перетворює найпростішим чином рядок даних у стовпець.

Операція σ

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

5 МЕТОДИКА ВПРОВАДЖЕННЯ СИСТЕМИ В ПРОМИСЛОВУ ЕКСПЛУАТАЦІЮ

Розглянемо детально інтерфейс. Після запуску програми на екрані з’явиться головне вікно, яке зображене на рисунку 5.1. З нього видно, що інтерфейс користувача програми складається з таких логічних блоків:

– Меню ПЗ.

– Блоку виведення зображення.

– Функціональних кнопок: On/Off стабілізацію; On/Off підвищення якості зображення; Налаштування джерела відеосигналу; On/Off стиску відеопотоку підвищення; Налаштування; Авторське право.

– Інформації обробкт відеопотоку: Метод стабілізації та підвищення якості відеопотоку; Якість сигналу; Підвищення частоти кадрів; Частота кадрів відеопотоку; Стиск відео потоку; Нелокальне усереднення для придушення шуму.

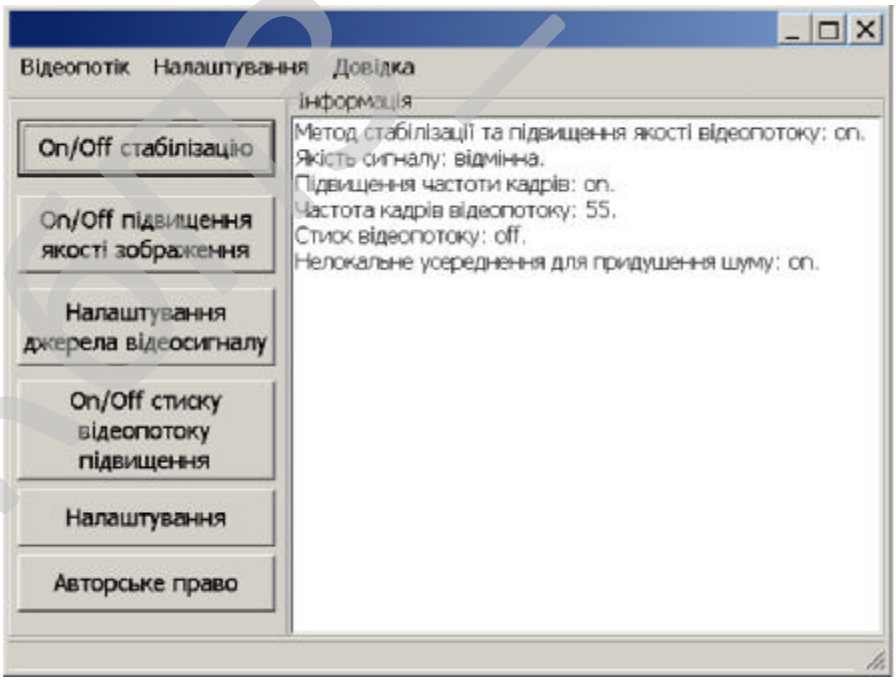

Рисунок 5.1 – Головне вікно програми

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.

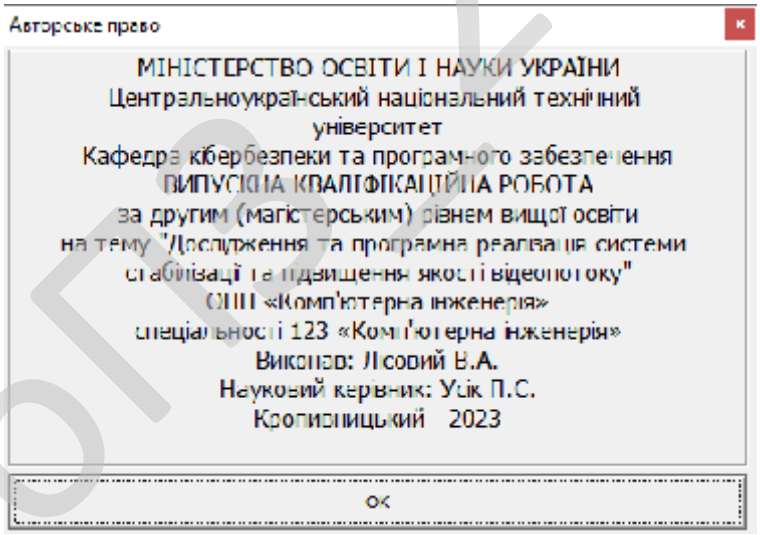

Рисунок 5.2 – Довідка розробника

6 НАУКОВА НОВИЗНА

У випускній кваліфікаційній роботі за другим (магістерським) рівнем вищої освіти розроблено програмне забезпечення, яке призначено для системи стабілізації та підвищення якості відеопотоку.

Метою розробки є дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку.

Об'єктом дослідження є процес стабілізації та підвищення якості відеопотоку.

Предметом дослідження є методи стабілізації та підвищення якості відеопотоку.

Методи дослідження базуються на методах комп'ютерної графіки, методах математичної статистики, методах розробки програмного забезпечення.

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

– Удосконалено метод стабілізації та підвищення якості відеопотоку.

– Розроблено вітчизняний продукт стабілізації та підвищення якості відеопотоку, 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 60 днів (три місяці).

В магістерській роботі було проведено дослідження та виконана програмна реалізація системи стабілізації та підвищення якості відеопотоку.

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

а) невеликі системні потреби;

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

в) зручність у користуванні;

г) 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	19
3. Запланований термін розробки, днів	Fpq	60 (3 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

Продовження таблиці 7.1

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

Продовження таблиці 7.1

1	2	3
28. Досвід роботи з мовою і інструментами середовища розробки (1-6)	–	2
29. Досвід роботи з програмними інструментами розробки (1-6)	–	3
30. Розробка ПЗ для декількох серверів одночасно (1-6)	–	2
31. Вимоги до дотримання встановленого графіка робіт (1-6)	–	2
32. Вартість ПЗ у розробника (НМА), грн.	–	19000
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.:

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

де: A – коефіцієнт Боема, A = 2,45;

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

B – показник ступеня, що визначається співвідношенням:

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

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

$$B = 1{,}01+0{,}001(2{,}43+3{,}64+3{,}38+3{,}95+2{,}73) = 1{,}027.$$

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

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

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

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

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

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

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

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

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

S – коефіцієнт стиснення (або подовження) графіка робіт %, що дозволяє коректувати терміни розробки ПЗ згідно встановленим вимогам. Вибираємо в межах (25…350)%. Приймаємо S =80 %

$$T_{РП} = 0{,}3 \cdot 2{,}66 \cdot 9{,}37^{0{,}33+0{,}2(1{,}026-1{,}01)} \cdot 65 = 109 \text{ люд/день.}$$

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	109	Ф 7.1-7.4
Впровадження	13	Д13
Всього	150	–

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.

$$Ч = \frac{150 \cdot 1}{60 - 5} = 2,75 \text{ ставки.}$$

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	6	540	9
Монітор	60	6	360	6
Клавіатура	30	6	180	3
Маніпулятор «мишка»	30	6	180	3
Принтер матричний	60	0	0	0,0
Принтер лазерний	120	1	120	2
Принтер струминний	60	1	60	1
Сканер	20	1	20	0,33
Концентратор-маршрутизатор	30	1	30	0,5
Кабельні господарства ЛОМ на 1 м.п.	2,5	300	750	12,5
Копіювальний апарат	140	1	140	2,33
Усього за рік:			$З_ч$	39,66

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

Виходячи з цього фонд робочого часу інженерів-електронщиків складає:

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

$$Ф_{др}^{с} = \frac{39,66 \cdot 3}{1,2} = 99,15 \text{ год.}$$

Визначаємо необхідну кількість ставок штатного персоналу сектора ТО:

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

$$Ч_{ел} = 99,15/(60 \cdot 8) = 0,2 \text{ ставки.}$$

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

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

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

Посада	Вид роботи	Час	К-ть штатних одиниць
Адміністратор загальної мережі, аналітик	Адміністрування локальної мережі, поштового та серверу DNS (ОС FreeBSD), маршрутизатора Cisco, доменного контролеру Windows Server 2022, серверу доступу ADSL (ОС Linux), налаштування ADSL, VPN PPPoE, Frame Relay, Wi-Fi	2	0,5
	Налаштування і конфігурування базової станції безпровідного зв'язку (CMTS)	0,5	
	Розробка та впровадження проектів з організації зв'язку між віддаленими об'єктами, ЛОМ	0,5	
	Забезпечення цілодобової роботи зв'язку клієнтів до мережі Інтернет	1	
Всього		4	

Продовження таблиці 7.4

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

Складемо штатний розклад виконавців.

Таблиця 7.5 – Штатний розклад виконавців

Посада	Кількість ставок	Середньомісячний оклад, грн.	Всього за період розробки, грн.
Керівник (ІТ-менеджер)	1	20164	60492
Продакт-менеджер	0,25	15000	11250
Інженер-програміст	2,75	17000	140250
Інженер-електронщик	0,2	15000	9000
Інженер-системотехнік	0,25	15000	11250
Адміністратор мережі	0,5	15000	22500
Системний програміст	0,25	15000	11250
Дизайнер WEB	0,25	15000	11250
Інженер-верстальник	0,25	15000	11250
Бухгалтер-економіст	0,5	15000	22500
Всього за період розробки	$R_{сп}$ = 6,2	-	$Ф_{роб}$ = 310992

Розрахуємо середньоденну зарплату одного виконавця:

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

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

$$З_{сд} = \frac{310992}{6,2 \cdot 60} = 836 \; грн.$$

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 400…1600 $у.о./м^2$. Враховуючи, що курс складає 1 у.о. = 38 грн. приймаємо для розрахунку вартість одного метра квадратного рівною 20000 грн./м^2. На кожне робоче місце у середньому потрібно 8 $м^2$. З урахуванням цього:

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

Вартість передавальних пристроїв складає 10% від вартості будівель, і у даному випадку вона складе: 128000 грн.

Балансова вартість інвентарю розраховується за нормою 3500 грн. на одне робоче місце. Тобто:

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

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

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

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

Специфікація на обчислювальну техніку наведена в таблиці 7.7.

Дані по оптовій ціні на обладнання та комплектуючі вибирались по прайсу фірми Supercomp за 25.10.23 – джерело https://supercomp.kiev.ua/

Таблиця 7.6 – Специфікація

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Персональний комп’ютер		10947
Системний блок		7347
Процесор	INTEL Celeron G5905 (BX80701G5905 1200, 2 ядра, 2 потоки, 3.5 GHz, TDP – 5 Вт, 14nm, box	-
Системна плата	GIGABYTE H470M H сокет – 1200, DDR4 64 ГБ, 3200 MHz, LAN – 1 Гбіт/с, D-Su (VGA), HDMI, 1 x M.2 2280, 4 x Sata 6. Gb/s, Micro-ATX	-
Відеокарта	Intel UHD Graphics 610	-
Жорсткий диск	SSD M.2 2280 240GB Apace (AP240GAS2280P4-1) 240 GB, 3D TLC, M.2 PCI Express 3.0 x4	-
Жорсткий диск	SSD 2.5" 240GB Mibran (MI2.5SSD/SP240GB) Серия – Spider, 24 GB, 3D TLC NAND, 2.5", SATA III (6Gb/s)	
Оперативна пам’ять	DDR4 8GB 2400 MHz Patriot	-
DVD-привод	DVDRW Pioneer DVR-TD10RS SATA Slir Black Bulk (DVR-TD10RS)	-
Корпус	Vinga CS104B-500W класіка, Miditowe ATX, Micro – ATX, Mini – ITX, PSU – 50 Вт	-

Продовження таблиці 7.6

Найменування комплектуючої або обладнання	Тип	Оптова ціна
Кулер	–	–
Кардрідер внутрішній	USB 2.0 Card reader STORM CR-35U1A4-E int. 3.5", 1*USB2.0+AUDIO+1394, multi: A Type Cards, black	220
інше	Клавіатура, мишка	Подару-нок
Монітор	22" TFT, ASUS VW223D (5ms, 300/3000: 1 170/160, D-SUB, Wide)	3600
Принтер лазерний	Canon i-SENSYS LBP6030W	2700
Принтер струминний	Epson Stylus Photo P50 (C11CA45341) + USE cable	5500
Копіювальний апарат	Canon i-SENSYS MF217W with Wi-Fi	5965

Таблиця 7.7 – Балансова вартість обчислювальної техніки

Найменування обчислювальної техніки	Кількість, шт.	Ціна за одиницю, грн.	Витрати на транспортування, монтаж та випробовування.	Загальна вартість, грн.
Персональні комп’ютери	8	10947	8757,6	96333,6
Принтер лаз.	2	2700	540	5940
Принтер струм.	1	5500	550	6050
Сканери	-	-	-	0
Копіюв. апарат	1	5965	596,5	6561,5
Всього	–	–	–	114885,1

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 7.8 – 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
Група 4			
3. Обчислювальна техніка	114885	-	-
Всього по групі	114885	50	57442,5
Група 5, 6			
4. Вимірювальні пристрої	5190	25	1297,5
5. Транспортні засоби	97500	20	19500
6. Господарський інвентар	28000	25	7000
Всього по групі	130690	-	27797,5
7. Нематеріальні активи	19000	10	1900
Разом	$К_р$ = 1672575		$А_р$ = 157540

Примітка: вартість автомобіля взята по даним автобазару Авто-PIA, джерело https://auto.ria.com/uk/auto_daewoo_lanos_ 33568345.html, складає 97500 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.

$$З_о = 836 \cdot 150/19 = 6600 \text{ грн.}$$

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, %.

$$З_д = 6600 \cdot 10 \cdot 0{,}01 = 660 \text{ грн.}$$

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

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

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

$$С_{оц} = 0{,}01 \cdot 22(6600+660) = 1597 \text{ грн.}$$

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

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

де: $Н_г$ – загальногосподарські витрати, %.

$$Г_{осп} = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

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 0,5 пачки паперу на період розробки. Тоді, враховуючи, що вартість пачки паперу складає $Ц_п$=210 грн., визначаємо вартість паперу за період розробки:

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

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

Згідно прийнятих норм по комплектації до вартості запам'ятовуючих пристроїв входить вартість CD/DVD дисків. Їх кількість дорівнює кількості коробочних версій запропонованого продукту (приймаємо 5):

$$З_{М2} = \sum Ц_д, \quad (7.17)$$

де: $Ц_д$ – вартість дисків CD/DVD: CDR box – 21,4 грн./шт., DVD-R box – 35 грн./шт.

$$З_{М2} = 5 \cdot 21,4 = 107 \text{ грн.}$$

Згідно норм одноразовій заправці підлягають усі друкуючі пристрої і становить:

$$З_{М3} = \sum Ц_з., \quad (7.18)$$

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

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

$$З_М = (105 + 107 + 1702)/19 = 101 \text{ грн.}$$

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

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

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

$$O_п = 6600 \cdot 15 \cdot 0{,}01 = 990 \text{ грн.}$$

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	$З_о$	6600
2. Додаткова зарплата виконавців	$З_д$	660
3. Відрахування на соціальні потреби	$С_{оц}$	1597
4. Загальногосподарські витрати	$Г_{осп}$	990
5. Витрати на матеріали	$З_М$	101
6. Освоєння нових операційних систем, мов програмування	$О_п$	990
7. Амортизація основних фондів	$А_м$	2073
8. Повна собівартість програмного забезпечення	$С_п$	13011
9. Плановий прибуток	$П_р$	6505,5
10.Ціна підприємства $Ц_п = С_п + П_р$	$Ц_п$	19516,5
11.Податок на додану вартість $ПДВ = 0.01 \cdot Н_{дв} \cdot Ц_п$	$ПДВ$	3903,3
12.Відпускна ціна програмної продукції $Ц = Ц_п + ПДВ$	$Ц$	23419,8

Визначимо витрати на амортизацію основних фондів з урахуванням загальної річної суми амортизаційних відрахувань та кількості екземплярів програм ($N_е$ = 19 прим.):

$$А_м = \frac{А_р \cdot N_{міс}}{N_е \cdot 12}, \qquad (7.20)$$

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

$$A_м = 157540 \cdot 3/(19 \cdot 12) = 2073 \text{ грн.}$$

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

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

$$C_п = 6600+660+1597+990+101+990+2073 = 13011 \text{ грн.}$$

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

Для даного програмного забезпечення рівень рентабельності складає 50%.

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

де: $P_п$ – рівень рентабельності, %.

$$П_р = 0{,}01 \cdot 50 \cdot 13011 = 6505{,}5 \text{ грн.}$$

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	–	23420
Всього капітальних витрат	–	23420

7.7 Визначення експлуатаційних витрат

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

Таблиця 7.11 – Розрахунок експлуатаційних витрат у споживача програмної продукції

Найменування статей витрат	Позначення	Сума витрат за варіантами, грн.	
		Базовий	Новий
1. Витрати на технічне обслуговування	$З_р$	85888	21472
2. Витрати на електроенергію	$З_{ел}$	5320	4560
3. Витрати на амортизацію	$З_{ам}$	0	5855
Всього витрат за рік	$І$	91208	31887

Витрати на профілактичні роботи:

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

де: $Т_р$ – кількість годин обслуговування кожного комп'ютера за рік, *год.*;

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

Після купівлі нового програмного забезпечення кількість профілактичних годин робіт зменшилася з 800 годин на рік до 200 годин на рік, тому витрати на технічне обслуговування зменшилися з:

$$З_{р\,баз} = 800 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 85888\ грн,$$

до:

$$З_{р\,нов} = 200 \cdot 80 \cdot 1{,}1 \cdot 1{,}22 = 21472\ грн.$$

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 ($Ц_{ел}$):

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

$$З_{ел\ баз} = 0{,}35 \cdot 4000 \cdot 3{,}8 = 5320 \text{ грн.}$$

$$З_{ел\ нов} = 0{,}3 \cdot 4000 \cdot 3{,}8 = 4560 \text{ грн.}$$

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

Таблиця 7.12 – Розрахунок амортизаційних відрахувань

Групи основних фондів	Норма амортизації %	Балансова вартість, грн., за варіантами		Сума відрахувань, грн за варіантами	
		Базовий	Новий	Базовий	Новий
Програмна продукція	25	–	23420	–	5855
Всього відрахувань	-	–	23420	–	5855

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:

$$E_в = (Ц_п - С_п) \cdot N_е - \sum_{i=1}^{m} E_{р_m} \cdot К_{р_m}, \quad (7.25)$$

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

$$Ев = (19516 - 13011) \cdot 19 - (0{,}05 \cdot 1408000 + 0{,}5 \cdot 114885 +$$
$$+0{,}25 \cdot 33190 + 0{,}2 \cdot 97500 + 0{,}1 \cdot 19000) \cdot 3/12 = 84210 \text{ грн.}$$

Визначимо період окупності додаткових капітальних вкладень у виробника програмної продукції:

$$Т_в = \frac{К_р^*}{(Ц_п - С_п) \cdot N_е}, \quad (7.26)$$

де: K_p^* – балансова вартість основних фондів розробника без врахування вартості ОФ третьої групи, так як їх строк служби на порядок більший ніж період розробки ПЗ.

$$T_в = \frac{264575}{(19516 - 13011) \cdot 19 \cdot 12 / 3} = 0,54 \text{ років}.$$

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.	19
2. Повна собівартість розробленої програми	Грн.	13011
3. Ціна розробленої програми	Грн.	19516
4. Плановий прибуток від реалізації розробленої програми	Грн.	6505
5. Рентабельність програмної продукції	%	50
6. Об’єм додаткових капітальних вкладень у виробника програмної продукції	Грн.	1672575
7. Загальний прибуток від реалізації програмної продукції	Грн.	123595
8. Величина економічного ефекту при виготовлені програмної продукції	Грн.	84210
9. Період окупності додаткових капітальних вкладень у виробника програмної продукції	Років.	0,54
10.Об’єм додаткових капітальних вкладень у споживача програмної продукції	Грн.	23520
11.Величина економічного ефекту у користувача програмної продукції	Грн.	53466
12.Період окупності додаткових капітальних вкладень у користувача програмної продукції	Років	0,4

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.

$$E_{сп} = (91208\text{-}31887)\text{-}0{,}25 \cdot 23420 = 53466 \text{ грн.}$$

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

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

$$T_{сп} = \frac{23420}{91208 - 31887} = 0{,}4 \text{ року}.$$

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 польців та м'язи рук) при роботі з клавіатурою мають теж істотне навантаженням [2]. До шкідливих факторів, які впливають на робитників 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 визначимо фактори, які можуть викликати професійне захворювання і впливають на працездатність програміста,

8.2 Шкідливі і небезпечні фактори при роботі з комп'ютером

Електронно-обчислювальна машин (ЕОМ) та інше обладнання є джерелами небезпеки ураження електричним струмом. Так як робота програміста характеризується істотним зоровим навантаженням, то вимагає належного освітлення. У приміщенні, в якому працюють люди (у т.ч. програмісти) необхідно створити належний мікроклімат, праметри якого регламентуються, Державними санітарними правилами і нормами, зокрема ДСанПіН 3.3.2.007-98 [2].

При роботі з використанням ЕОМ відзначають наступні небезпечні та шкідливі фактори:

– ризик виникнення надзвичайних ситуацій природного або штучного характеру на об'єкті або території.

– ризик виникнення пожежі;

– негативний вплив на органи зору людини;

– ризики ураження електричним струмом;

– недостатня, або надмірна освітленість робочого місця;

– електромагнітні (у т.ч. високочастотні) електромагнітні випромінювання (коливання);

– несприятливі мікрокліматичні умови;

– нервово-емоційна напруженість праці;

– інтелектуальні навантаження;

– монотонність праці;

– невідповідність ергономічних показників робочого місця діючим вимогам;

– шум;

– статичні навантаження на кістково-м’язовий апарат.

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	3,4
Довжина	3,5
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	11,9
Обсяг, V	$м^3$	не менше 20.0	35,7

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

У зазначеному приміщенні працє 1 людина. За даними, які наведено у табл. 8.1 та табл. 8.2, можна зробити висновок, що площа та об’єм 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] та 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/с	Температура, °С	Воло-гість%	Швидкість повітря, м/с
Холодна	22-24	40-60	0,1	22-23	45-60	0,12
Тепла	23-25	50-70	0,1	23-25	52-70	0,1

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 значень;

Таким чином можна припустити, що основною причиною можливого 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: 3,4 м.; довжина –3,5 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 *[9]*:

$$F=ESKZ/n,$$

де: *F* – світловий потік, що розраховується, Лм;

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

S – площа освітлюваного приміщення (у нашому випадку *S*=3,4 х 3,5 = 11,9 $м^2$);

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, $S = 11{,}9\ м^2$;

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

A – ширина приміщення, $A = 3{,}4\ м$;

B – довжина приміщення, $B = 3{,}5\ м$.

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 [6])*. Підставимо всі значення у формулу, визначемо світловий потік: $F=20312$ *Лм*.

Будемо використовувати лампи *TL-F 12 36W LED 6000K IP65*, світловий потік яких $F_л = 3000$ Лм.

Число ламп визначається по формулі:

$$N=F/F_л$$

де: F – світловий потік,

$F_л$ – світловий потік одного *світильника*.

Підставимо всі значення у формулу та визначемо індекса приміщення: N= *20312/* 3000=6,77 шт.

Приймаємо необхідну кількість світильників 7 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. Охорона праці. Ч. 1. Захисне заземлення: метод. вказ. до викон. розрахунків з викор. персон. ЕОМ ІВМ сумісного типу / Кіровоград. ін-т с.-г. машинобуд.; [укл. О. В. Оришака, Є. К. Солових, В. О. Оришака]. – Кіровоград: КІСМ, 1997. – 20 с. Режим доступу до ресурсу: http://dspace.kntu.kr.ua/jspui/handle/123456789/4358 (дата звернення: 16.06.2023).

6. 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

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

8. 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).

9. 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 стабілізації та підвищення якості відеопотоку.

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 стабілізації та підвищення якості відеопотоку.

Рішення даного завдання полягало у вирішенні наступних задач:

– Був проведений огляд існуючих систем стабілізації та підвищення якості відеопотоку.

– Досліджена система стабілізації та підвищення якості відеопотоку.

– На основі отриманих результатів досліджень створена програмна реалізація системи стабілізації та підвищення якості відеопотоку.

Розроблені під час виконання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти алгоритми дозволяють успішно вирішувати завдання стабілізації та підвищення якості відеопотоку.

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 Builder C++. 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 CRYPTON.

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

Розроблена програма має реальний економічний ефект від її впровадження у виробництво у сумі 53466 грн. З урахуванням вартості розробки програми та обладнання, строк окуплення становить 0,4 роки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Лісовий В.А. Дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку // Збірник праць молодих науковців ЦНТУ. – Вип. 14. – Кропивницький: ЦНТУ, 2023.

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

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

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

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

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

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

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

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

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

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

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

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

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

15. 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,

16. 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.

17. 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

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

19. Smirnov O., Kuznetsov A., Lokotkova I., Kuznetsova T., Florov S., Lebid O. «Using Orthogonal Signals to Hide Information in Images». 4 IEEE International Conference on Advanced Information and Communication Technologies (AICT) – 2021, Lviv, Ukraine, September 21-25, 2021. P. 255-260.

20. 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.

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

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

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

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

25. 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.

26. 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.

27. 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.

28. 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.

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

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

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., 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.

35. 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.

36. 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.

37. 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.

38. 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.

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

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

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

42. 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.

43. 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.

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

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

46. 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).

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

48. Дрєєва Г.М., Смірнов О.А., Дрєєв О.М. Метод генерування фрактальноподібної числової послідовності на основі скінченного автомату для моделювання трафіку у мережі. Центральноукраїнський науковий вісник. Технічні науки. № 1(32). с. 173-183, 2019.

49. Смірнов О.А., Кавун С.В., Коваленко О.В., Дрєєв О.М. Мережні інформаційні технології. Навчальний посібник – Кіровоград: РВЛ КНТУ, 2016. – 159 с.

50. Смірнов О.А., Дрєєв О.М. Порівняння бітових щільностей при використанні різних методів кодування інформації. Збірник наукових праць "Системи обробки інформації". – Випуск 2 (118). т.2. – Х.: ХУПС – 2014. – С. 64-67

Додаток А
(обов'язковий)

Технічне завдання

Зміст

1 Найменування та область застосування

Це технічне завдання розповсюджується на дослідження та програмну реалізацію системи стабілізації та підвищення якості відеопотоку.

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. № 34-13 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 стабілізації та підвищення якості відеопотоку.

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 стабілізації та підвищення якості відеопотоку;

– 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 Builder C++.

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 повинні бути розглянуті шкідливі і небезпечні фактори при роботі з комп’ютером.

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 – 96 аркушів.

10 Етапи розробки

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

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

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

10.4 Побудова схем взаємодії даних.

10.5 Створення прототипу ПЗ.

10.6 Віднаходження ПЗ, аналіз отриманих результатів.

10.7 Робота над питанням охорони праці і техніки безпеки.

10.8 Розрахунок з техніко-економічного обґрунтування.

10.9 Оформлення пояснювальної записки і виконання робіт по графічній частині.

11 Порядок контролю та приймання

11.1 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на попередній захист 10.12.2023 р.

11.2 Подання випускної кваліфікаційної роботи за другим (магістерським) рівнем вищої освіти на захист 19.12.2023 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

_____ Усік П.С.

Дослідження та програмна реалізація системи стабілізації та підвищення якості відеопотоку

Лістинг програми

Код документу 12

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

Загальна кількість аркушів: 37

Літера: РП

Кропивницький – 2023 року

Файл обробки палітри кадру відео потоку - PalitraVideo.cpp

```
//-------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В’ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-------------------------------------------------------------------
#include <afxwin.h> // Використання стандартних бібліотек
#include "gmlcommon.h"
#include <windows.h>
#include "gmlmath.h"
#include "gmlimage.h"
#include "gmlimagefilter.h"
#include "gmlimagecomposer.h"
#include "filters/gmlfilterrgb2bgr.h"
#include "filters/gmlfilterrepres.h"
#include <memory.h>

//Параметри палітри
static const int YUV2RGB[] =
{
TAP(1.1644),
TAP(1.596),
TAP(0),
-TAP(222.9184),
TAP(1.1644),
-TAP(0.815),
-TAP(0.39),
TAP(135.6096),
TAP(1.1644),
TAP(0),
TAP(2.016),
-TAP(276.6784)
};
// Конвертація YUV420 TO BGR
static void ConvertYUV420_TO_BGR(int width,
                         int height,
                         BYTE* Y,
                         BYTE* U,
                         BYTE* V,
                         BYTE** in_pLineArray)
{
int i, j;
assert(((width | height) & 7) == 0);
width /= 2;
BYTE* pCurLine, * pNextLine;
for (i = 0; i < height; i += 2, Y += 4 * width,U += width, V += width)
{
pCurLine = in_pLineArray[height - 1 - i];
pNextLine = in_pLineArray[height - 1 - i - 1];
for (j = 0; j < width; j++, pCurLine += 6, pNextLine += 6)
{
int r0, g0, b0, r, g, b;
PROCESS_UV(j, r0, g0, b0);
PROCESS_Y(j * 2, r, g, b, r0, g0, b0);
SAVE_RGB(0, r, g, b, pCurLine);
PROCESS_Y(j * 2 + 1, r, g, b, r0, g0, b0);
SAVE_RGB(3, r, g, b, pCurLine);
PROCESS_Y(j * 2 + width * 2, r, g, b, r0, g0, b0);
SAVE_RGB(0, r, g, b, pNextLine);
PROCESS_Y(j * 2 + width * 2 + 1, r, g, b, r0, g0, b0);
SAVE_RGB(3, r, g, b, pNextLine);
}
}
}
VideoImage::Image() : m_lock_count(0), m_created(false), m_line_array(NULL)
{
ASSERT(sizeof(m_format) == sizeof(int));
```

```
}
VideoImage::~Image()
{
ASSERT(m_lock_count == 0);
}
VideoImage::Image(Image& in_Image)
{
this->CreateCopy(in_Image);
}
bool const VideoImage::operator ==(Image const & Other)
{
return AlmostEqual(Other, 0);
}

void VideoImage::CreateCopy(const Image& rSrcBmp)
{
if (&rSrcBmp != this && rSrcBmp.Created())
{
// Створення пустого поля зображення
InternalCopy(rSrcBmp);
m_created = rSrcBmp.m_created;
}
}
void VideoImage::CreateFilteredCopy(const Image& rSrcBmp,
                         const ImageFilter& rFilter)
{
rFilter.Apply(&rSrcBmp, this);
m_created = true;
}
void VideoImage::CreateComposition(const std::vector<Image*>& src,
                        const ImageComposer& comp)
{
comp.Apply(src, this);
}

void VideoImage::Create(int width,
             int height,
             FORMAT format,
             REPRES repres,
             ORIENT orient)
{
if (Created() &&
width == GetWidth() &&
height == GetHeight() &&
format == GetFormat() &&
repres == GetRepres() &&
orient == GetOrient())
return;
if (Created())
FreeMembers();
m_created = InternalCreate(width, height, format, repres, orient);
ASSERT(m_created);
}
bool VideoImage::Created() const
{
return m_created;
}
// Очищення
void VideoImage::Clear()
{
if (Created())
{
FreeMembers();
m_created = false;
}
}
// Використання фільтру
void VideoImage::ApplyFilter(const ImageFilter& Filter)
{
```

```
Filter.ApplyInPlace(this);
}
void VideoImage::Decompose(const ImageDecomposer& decomposer,
                std::vector<Image*>& out_res)
{
decomposer.Apply(this, out_res);
}
void VideoImage::Lock(bool bReadable, bool bWriteable)
{
ASSERT(m_lock_count >= 0);
ASSERT(bReadable || bWriteable);
m_lock_count++;
}
// Розблокування
void VideoImage::Unlock()
{
m_lock_count--;
ASSERT(m_lock_count >= 0);
}
//Розрахунок
bool VideoImage::AlmostEqual(const Image& Bmp, int epsilon) const
{
if (GetWidth() != Bmp.GetWidth() ||
GetHeight() != Bmp.GetHeight() ||
HasAlpha() != Bmp.HasAlpha() ||
GetBitsPerPixel() != Bmp.GetBitsPerPixel())
return false;
if (m_Resolution != Bmp.GetResolution())
return false;
PLBYTE ** ppLines1 = GetLineArray();
PLBYTE ** ppLines2 = Bmp.GetLineArray();
int y,x;
for (y=0; y<GetHeight(); y++)
for (x=0; x<GetWidth(); x++)
switch (GetBitsPerPixel())
{
case 8:
if (abs (ppLines1[y][x] - ppLines2[y][x]) > epsilon)
return false;
break;
case 24:
if (ppLines1[y][x*3+PL_RGBA_RED] != ppLines2[y][x*3+PL_RGBA_RED] ||
ppLines1[y][x*3+PL_RGBA_GREEN] != ppLines2[y][x*3+PL_RGBA_GREEN] ||
ppLines1[y][x*3+PL_RGBA_BLUE] != ppLines2[y][x*3+PL_RGBA_BLUE])
return false;
break;
case 32:
if (abs (ppLines1[y][x*4+PL_RGBA_RED] - ppLines2[y][x*4+PL_RGBA_RED]) > epsilon
||
abs (ppLines1[y][x*4+PL_RGBA_GREEN] - ppLines2[y][x*4+PL_RGBA_GREEN]) > epsilon
||
abs (ppLines1[y][x*4+PL_RGBA_BLUE] - ppLines2[y][x*4+PL_RGBA_BLUE]) > epsilon)
return false;
if (HasAlpha() &&
abs (ppLines1[y][x*4+3] - ppLines2[y][x*4+3]) > epsilon)
return false;
break;
default:
PLASSERT (false);
}
  if (GetBitsPerPixel() == 8)
  {
  int i;
  PLPixel32 * pPal1 = GetPalette();
  PLPixel32 * pPal2 = Bmp.GetPalette();
  for (i=0; i<255; i++)
  {
  if (abs (pPal1[i].Get() - pPal2[i].Get()) > epsilon ||
  abs (pPal1[i].Get() - pPal2[i].Get()) > epsilon ||
```

```
  abs (pPal1[i].Get() - pPal2[i].Get()) > epsilon)
  return false;
  }
  }
return true;
}
void VideoImage::InitLocals(int width,
                 int height,
                 FORMAT format,
                 REPRES repres,
                 ORIENT orient)
{
m_width = width;
m_height = height;
m_format = format;
m_repres = repres;
m_orient = orient;
InitLineArray();
}
void VideoImage::InternalCopy(const Image& rSrcBmp)
{
bool bCompatible;
if (Created() &&
rSrcBmp.GetWidth() == GetWidth() &&
rSrcBmp.GetHeight() == GetHeight() &&
rSrcBmp.GetFormat() == GetFormat() &&
rSrcBmp.GetRepres() == GetRepres() &&
rSrcBmp.GetOrient() == GetOrient())
bCompatible = true;
else
{
FreeMembers();
bCompatible = InternalCreate(rSrcBmp.GetWidth(),
                          rSrcBmp.GetHeight(),
                          rSrcBmp.GetFormat(),
                          rSrcBmp.GetRepres(),
                          rSrcBmp.GetOrient());
};
if (!bCompatible)
{
ASSERT(bCompatible);
m_created = false;
return;
}
const_cast<Image&>(rSrcBmp).Lock(true, false);
Lock(false, true);
BYTE** pSrcLines = rSrcBmp.GetLineArray();
BYTE** pDstLines = GetLineArray();
int LineLen = GetWidth() * GetElemSize();
for (int y = 0; y < GetHeight(); y++)
{
memcpy(pDstLines[y], pSrcLines[y], LineLen);
}
m_created = rSrcBmp.m_created;
Unlock();
const_cast<Image&>(rSrcBmp).Unlock();
}
//Зміна орієнтації зображення
bool VideoImage::ChangeOrientation(ORIENT new_orient)
{
if (m_orient != new_orient)
InternalChangeOrientation(new_orient);
ASSERT(m_orient == new_orient);
return (m_orient == new_orient);
}
bool VideoImage::ChangeFormat(FORMAT new_format)
{
if (m_format != new_format)
{
```

```
if (m_format != F_RGB &&
  m_format != F_BGR ||
  new_format != F_RGB &&
  new_format != F_BGR)
{
ASSERT("Непідтримується" == 0);
return false;
}
ApplyFilter(gml::FilterRGB2BGR());
m_format = new_format;
}
ASSERT(m_format == new_format);
return (m_format == new_format);
}
bool VideoImage::ChangeRepres(REPRES new_repres)
{
if (m_repres != new_repres)
ApplyFilter(gml::FilterRepres(new_repres));
ASSERT(m_repres == new_repres);
return (m_repres == new_repres);
}
bool VideoImage::CreateFromDIB(struct tagBITMAPINFO* in_pDIB, BYTE* in_pcData)
{
int iWidth, iHeight;
ORIENT Orient;
/// створення зображення
iWidth = in_pDIB->bmiHeader.biWidth;
iHeight = abs(in_pDIB->bmiHeader.biHeight);
if (in_pDIB->bmiHeader.biHeight < 0)
{
iHeight = -in_pDIB->bmiHeader.biHeight;
Orient = O_TOPLEFT;
}
else
{
iHeight = in_pDIB->bmiHeader.biHeight;
Orient = O_BOTTOMLEFT;
}
Create(iWidth, iHeight, F_BGR, R_BYTE, Orient);
if (!Created())
return false;
if (!in_pcData)
in_pcData = ((BYTE *) in_pDIB) + in_pDIB->bmiHeader.biSize;
switch (in_pDIB->bmiHeader.biCompression)
{
case BI_RGB:
{
  switch (in_pDIB->bmiHeader.biBitCount)
  {
    case 24:
      {
        int iLineLen = (iWidth * 3 + 3) & -4;
        for (int i = 0; i < GetHeight(); i++, in_pcData += iLineLen)
          memcpy(GetLineArray()[i],
                 in_pcData,
                 GetElemSize() * GetWidth());
      }
      break;
    case 8:
      {
        int iLineLen = (iWidth + 3) & -4;
        if (!in_pDIB->bmiHeader.biClrUsed)
          in_pcData += 255 * sizeof(RGBQUAD);
        else
in_pcData += (in_pDIB->bmiHeader.biClrUsed - 1) * sizeof(RGBQUAD);
        for (int i = 0; i < GetHeight(); i++, in_pcData += iLineLen)
          for (int j = 0; j < GetHeight(); j++)
          {
 GetLineArray()[i][j * 3 + 0] = in_pDIB->bmiColors[in_pcData[j]].rgbBlue;
```

```
GetLineArray()[i][j * 3 + 1] = in_pDIB->bmiColors[in_pcData[j]].rgbGreen;
GetLineArray()[i][j * 3 + 2] = in_pDIB->bmiColors[in_pcData[j]].rgbRed;
          }
      }
      break;
    default:
      return false;
      break;
  };
}
break;
case MAKEFOURCC_LOCAL('Y','V','1','2'):
case MAKEFOURCC_LOCAL('I','4','2','0'):
{
  BYTE* Y = in_pcData,
           * U = Y + iWidth* iHeight,
           * V = U + iWidth* iHeight / 4;
  ConvertYUV420_TO_BGR(iWidth, iHeight, Y, U, V, GetLineArray());
}
break;
default:
return false;
break;
}
return true;
}
// Обробка
void VideoImage::PaintImage(CDC* in_pDC,
                 CPoint dst_org,
                 CSize dst_size,
                 DWORD dwRop)
{
unsigned char storage[sizeof(BITMAPINFOHEADER) + 256 * 4];
BITMAPINFOHEADER* bmih = (BITMAPINFOHEADER*) storage;
char* src_data = 0;
char* dst_data = 0;
CSize src_size;
int channels = 0, depth = 0;
HBITMAP hbmp = 0;
HDC hdc = 0;
if (!Created())
return;
if (GetRepres() != R_BYTE)
{
return;
}
else
depth = 8;
switch (GetFormat())
{
case F_L:
channels = 1;
break;
case F_RGB:
case F_BGR:
channels = 3;
break;
default:
return;
}
src_size.cx = GetWidth();
src_size.cy = GetHeight();
bmih->biSize = sizeof(*bmih);
bmih->biWidth = src_size.cx;
bmih->biHeight = -src_size.cy;
bmih->biPlanes = 1;
bmih->biBitCount = (unsigned short) (channels * depth);
bmih->biCompression = BI_RGB;
bmih->biSizeImage = 0;
```

```
bmih->biXPelsPerMeter = 0;
bmih->biYPelsPerMeter = 0;
bmih->biClrUsed = 0;
bmih->biClrImportant = 0;
if (channels == 1)
{
int i;
RGBQUAD* palette = (RGBQUAD*) (storage + sizeof(BITMAPINFOHEADER));
for (i = 0; i < 256; i++)
palette[i].rgbBlue = palette[i].rgbRed = palette[i].rgbGreen = (unsigned char)
i;
}
hdc = CreateCompatibleDC(0);
hbmp = CreateDIBSection(hdc,
                    (BITMAPINFO *) bmih,
                    DIB_RGB_COLORS,
                    (void * *) &dst_data,
                    0,
                    0);
int iYStart, iYEnd, iYStep;
if (GetOrient() == O_TOPLEFT)
{
iYStart = 0;
iYEnd = GetHeight();
iYStep = 1;
}
else
{
iYEnd = -1;
iYStart = GetHeight() - 1;
iYStep = -1;
}
if (hbmp && in_pDC)
{
HGDIOBJ hold = SelectObject(hdc, hbmp);
int dst_step = (src_size.cx* channels + 3) & -4;
int y;
for (y = iYStart; y != iYEnd; y += iYStep, dst_data += dst_step)
{
memcpy(dst_data, GetLineArray()[y], GetWidth() * channels);
}

VERIFY(SetStretchBltMode(in_pDC->m_hDC, COLORONCOLOR));
VERIFY(StretchBlt(in_pDC->m_hDC,
                dst_org.x,
                dst_org.y,
                dst_size.cx,
                dst_size.cy,
                hdc,
                0,
                0,
                src_size.cx,
                src_size.cy,
                dwRop));
SelectObject(hdc, hold);
}
if (hbmp)
DeleteObject(hbmp); // видалення об’єкту
if (hdc)
DeleteDC(hdc);
}
```

Файл проекту програми - Project.cpp

```
//---------------------------------------------------------------------
#include <vcl.h> //Використання бібліотеки vcl
#pragma hdrstop // директива компілятора
//---------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//---------------------------------------------------------------------
USEFORM("main.cpp", MainForm); // Використання форми MainForm
USEFORM("GraphConfig.cpp", GraphConfig); // Використання форми GraphConfig
USEFORM("Videosteam.cpp", Videosteam); // Використання форми Videosteam
USEFORM("VideoFile.cpp", VideoFile); // Використання форми VideoFile
USEFORM("VideoImage.cpp", VideoImage); // Використання форми VideoImage
//---------------------------------------------------------------------
WINAPI WinMain(HINSTANCE, HINSTANCE, LPSTR, int)
{
        try
        {
Application->Initialize();// Ініціалізація
Application->CreateForm(__classid(TVideoFile), &VideoFile); //створення форми
Application->CreateForm(__classid(TMainForm), &MainForm); //створення форми
Application->CreateForm(__classid(TVideosteam), &Videosteam); //створення форми
Application->CreateForm(__classid(TGraphConfig), &GraphConfig);//створення форми
Application->Run(); // початор ротоби процесу
        }
        catch (Exception &exception) // перевірка на помилки
        {
                Application->ShowException(&exception);
        }
        catch (...)
        {
                try
                {
                        throw Exception("");
                }
                catch (Exception &exception)
                {
                        Application->ShowException(&exception);
                }
        }
        return 0;//повернення коду завершення роботи ПЗ
}
//---------------------------------------------------------------------
```

Файл головного вікна програми - main.cpp

```
#include <vcl.h> //Використання бібліотеки vcl
#pragma hdrstop // директива компілятора
#include "main.h"
//---------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//---------------------------------------------------------------------
#pragma package(smart_init) // директива компілятора
#pragma link "DSPack" // директива компілятора
#pragma resource "*.dfm" // директива компілятора
TMainForm *MainForm;
TSysDevEnum *SysDev;
//---------------------------------------------------------------------
__fastcall TMainForm::TMainForm(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------------------
//Обробник створення форми
void __fastcall TMainForm::FormCreate(TObject *Sender)
{
  SysDev = new TSysDevEnum(CLSID_VideoInputDeviceCategory);
  if (SysDev->CountFilters > 0) {
    int i;
    TMenuItem *Device;
    for(i = 0; i < SysDev->CountFilters; i++) {
      Device = new TMenuItem(Devices);
      Device->Caption = SysDev->Filters[i].FriendlyName;
      Device->Tag = i;
      Device->OnClick = DevicesClick;
      Devices->Add(Device);
    }
  };
}
//---------------------------------------------------------------------
// Обробник пошуку підключеного обладнання
void __fastcall TMainForm::DevicesClick(TObject *Sender)
{
FilterGraph->ClearGraph();
FilterGraph->Active = false;
Filter->BaseFilter->Moniker = SysDev->GetMoniker(((TMenuItem *)Sender)->Tag);
  FilterGraph->Active = true;
  ICaptureGraphBuilder2 *Graph = NULL;
  IBaseFilter *SourceFilter = NULL;
  IBaseFilter *VideoFilter = NULL;
  CheckDSError(FilterGraph->QueryInterface(IID_ICaptureGraphBuilder2, &Graph));
CheckDSError(VideoWindow->QueryInterface(IID_IBaseFilter, &VideoFilter));
  CheckDSError(Filter->QueryInterface(IID_IBaseFilter, &SourceFilter));
  Graph->RenderStream(&PIN_CATEGORY_PREVIEW, NULL, SourceFilter, NULL,
VideoFilter);
  FilterGraph->Play();
  Graph->Release();
  VideoFilter->Release();
  SourceFilter->Release();
}
//---------------------------------------------------------------------
//Обробник видалення форми
void __fastcall TMainForm::FormDestroy(TObject *Sender)
{
  delete SysDev;
}
//---------------------------------------------------------------------
// Завершення по запиту
void __fastcall TMainForm::FormCloseQuery(TObject *Sender, bool &CanClose)
{
```

```
  FilterGraph->Active = false;
}
//---------------------------------------------------------------------
```

Файл головного вікна програми - main.h

```
#include <Classes.hpp> // Використання стандартних бібліотек: Classes; Controls;
                       // StdCtrls; Forms; Menus
#include <Controls.hpp>
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include "DSPack.hpp" // Додаткова бібліотека обробки відео потоку
#include <Menus.hpp>
//---------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//---------------------------------------------------------------------
class TMainForm:public TForm // Об'ява класу TMainForm
{
__published:
        TVideoWindow *VideoWindow;
        TMainMenu *MainMenu;
        TMenuItem *Devices;
        TFilterGraph *FilterGraph;
        TFilter *Filter;
        //Обробник створення форми
        void __fastcall FormCreate(TObject *Sender);
        void __fastcall DevicesClick(TObject *Sender);
        //Обробник видалення форми
        void __fastcall FormDestroy(TObject *Sender);
        void __fastcall FormCloseQuery(TObject *Sender, bool &CanClose);
private:
public:
        __fastcall TMainForm(TComponent* Owner);
};
//---------------------------------------------------------
extern PACKAGE TMainForm *MainForm;
//---------------------------------------------------------
#endif
```

Файл вікна налаштувань - GraphConfig.cpp

```
//---------------------------------------------------------
#include <vcl.h> //Використання бібліотеки vcl
#pragma hdrstop // директива компілятора

#include "GraphConfig.h"
//---------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//---------------------------------------------------------------------
#pragma package(smart_init) // директива компілятора
#pragma link "CSPIN" // директива компілятора
#pragma resource "*.dfm"// директива компілятора
//---------------------------------------------------------
bool EditGraphConfig(TGraphConfig* config)
{
  bool result = false;
  TFGraphConfig* f = new TFGraphConfig(Application);
  try {
    f->cbWantPreview->Checked = config->WantPreview;
    f->cbWantCapture->Checked = config->WantCapture;
    f->cbWantBitmaps->Checked = config->WantBitmaps;
    f->cbWantDV->Checked = config->WantDV;
    f->cbDoPreallocCaptureFile->Checked = config->DoPreallocFile;
    f->cbUseTempFile->Checked = config->UseTempFile;
    f->cbWant->Checked = config->Want;
    f->cbVComp->Checked = true;
    f->cbAComp->Checked = true;
    f->feCaptureFile->Text = config->CaptureFileName;
    f->feTempFile->Text = config->TempCaptureFileName;
    f->cbDVResolution->ItemIndex = int(config->DVResolution);
    f->cbPixelFormat->ItemIndex = int(config->PixelFormat);
    f->sePreallocSize->Value = config->PreallocFileSize;
    for (int i = 0; i<f->cbVSources->Items->Count; i++)
      if (f->cbVSources->Items->Strings[i]==config->VCapSource) {
        f->cbVSources->ItemIndex = i;
        break;
      };
    for (int i = 0; i<f->cbASources->Items->Count; i++)
      if (f->cbASources->Items->Strings[i]==config->ACapSource) {
        f->cbASources->ItemIndex = i;
        break;
      };
    for (int i = 0; i<f->cbVComps->Items->Count; i++)
      if (f->cbVComps->Items->Strings[i]==config->VComp) {
        f->cbVComps->ItemIndex = i;
        break;
      };
    for (int i = 0; i<f->cbAComps->Items->Count; i++)
      if (f->cbAComps->Items->Strings[i]==config->AComp) {
        f->cbAComps->ItemIndex = i;
        break;
      };
    f->cbVComp->Checked = f->cbVComps->ItemIndex >=0;
    f->cbAComp->Checked = f->cbAComps->ItemIndex >=0;
    result = (f->ShowModal()==mrOk);

    if (result) {
      config->Clear();
      config->WantPreview = f->cbWantPreview->Checked;
      config->WantCapture = f->cbWantCapture->Checked;
      config->WantBitmaps = f->cbWantBitmaps->Checked;
      config->WantDV = f->cbWantDV->Checked;
      config->DoPreallocFile = f->cbDoPreallocCaptureFile->Checked;
      config->UseTempFile = f->cbUseTempFile->Checked;
      config->Want = f->cbWant->Checked;
```

```
      config->PreallocFileSize = f->sePreallocSize->Value;
      config->CaptureFileName = f->feCaptureFile->Text;
      config->TempCaptureFileName = f->feTempFile->Text;
      if (f->cbDVResolution->ItemIndex>=0)
config->DVResolution= TDVResolution(f->cbDVResolution->ItemIndex);
      if (f->cbPixelFormat->ItemIndex>=0)
config->PixelFormat= TPixelFormat(f->cbPixelFormat->ItemIndex);
      config->VCapSource = f->cbVSources->Text;
      config->ACapSource = f->cbASources->Text;
      if (f->cbVComp->Checked) config->VComp = f->cbVComps->Text;
      if (f->cbAComp->Checked) config->AComp = f->cbAComps->Text;
      config->VCompState = "";
    }
  }
  __finally {
    delete f;
  }
  return result;
}
//---------------------------------------------------------
__fastcall TFGraphConfig::TFGraphConfig(TComponent* Owner)
        : TForm(Owner)
{
}
//---------------------------------------------------------
// Обробник кнопки
void __fastcall TFGraphConfig::cbWantPreviewClick(TObject *Sender)
{
  cbWantBitmaps->Enabled = cbWantPreview->Checked;
}
//---------------------------------------------------------
void __fastcall TFGraphConfig::cbWantClick(TObject *Sender)
{
  cbASources->Enabled = cbWant->Checked;
}
//---------------------------------------------------------
void __fastcall TFGraphConfig::cbWantCaptureClick(TObject *Sender)
{
  feCaptureFile->Enabled = cbWantCapture->Checked;
  cbDoPreallocCaptureFile->Enabled= cbWantCapture->Checked;
  cbUseTempFile->Enabled = cbWantCapture->Checked;
}
//---------------------------------------------------------
void __fastcall TFGraphConfig::cbUseTempFileClick(TObject *Sender)
{
  feTempFile->Enabled = cbUseTempFile->Checked;
}
//---------------------------------------------------------
void __fastcall TFGraphConfig::cbVCompClick(TObject *Sender)
{
  cbVComps->Enabled = cbVComp->Checked;
}
//---------------------------------------------------------
//Обробник створення форми
void __fastcall TFGraphConfig::FormCreate(TObject *Sender)
{
  TStringList* s;
  s = GetVideoDevicesList(true);
  try {
    cbVSources->Items->Assign(s);
  }
  __finally {
    delete s;
  }
  s = GetDevicesList(true);
  try {
    cbASources->Items->Assign(s);
  }
  __finally {
```

```
    delete s;
  };
  s = GetVideoCompressorsList(true);
  try {
    cbVComps->Items->Assign(s);
  }
  __finally {
    delete s;
  };
  s = GetCompressorsList(true);
  try {
    cbAComps->Items->Assign(s);
  }
  __finally {
    delete s;
  };
}
//---------------------------------------------------------
//Застосування фільтру
void __fastcall TFGraphConfig::cbDoPreallocCaptureFileClick(
      TObject *Sender)
{
  sePreallocSize->Enabled = cbDoPreallocCaptureFile->Checked;
}
```

Файл вікна налаштувань - GraphConfig.h

```
//---------------------------------------------------------
#include <Classes.hpp> // Використання стандартних бібліотек: Classes; Controls;
#include <Controls.hpp> // StdCtrls; Forms; Mask
#include <StdCtrls.hpp>
#include <Forms.hpp>
#include <Mask.hpp>
#include "CSPIN.h"
#include "VCap.hpp"
//---------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//---------------------------------------------------------------------
class TFGraphConfig:public TForm // Об'ява класу TFGraphConfig
{
__published:
        TLabel *Label3;
        TLabel *Label4;
        TLabel *Label6;
        TLabel *Label5;
        TLabel *Label7;
        TLabel *Label8;
        TCheckBox *cbWantPreview;
        TCheckBox *cbWantCapture;
        TCheckBox *cbWantBitmaps;
        TCheckBox *cbWantDV;
        TCheckBox *cbDoPreallocCaptureFile;
        TCheckBox *cbUseTempFile;
        TGroupBox *GroupBox1;
        TLabel *Label1;
        TLabel *Label2;
        TComboBox *cbVSources;
        TComboBox *cbASources;
        TGroupBox *GroupBox2;
        TComboBox *cbVComps;
        TComboBox *cbAComps;
        TCheckBox *cbVComp;
        TCheckBox *cbAComp;
        TCheckBox *cbWant;
        TComboBox *cbDVResolution;
        TComboBox *cbPixelFormat;
        TButton *Button1;
        TButton *Button2;
        TEdit *feCaptureFile;
        TEdit *feTempFile;
        TCSpinEdit *sePreallocSize;
        void __fastcall cbWantPreviewClick(TObject *Sender);
        void __fastcall cbWantClick(TObject *Sender);
        void __fastcall cbWantCaptureClick(TObject *Sender);
        void __fastcall cbUseTempFileClick(TObject *Sender);
        void __fastcall cbVCompClick(TObject *Sender);
        //Обробник створення форми
        void __fastcall FormCreate(TObject *Sender);
void __fastcall cbDoPreallocCaptureFileClick(TObject *Sender);
private:    // User declarations
public:     // User declarations
        __fastcall TFGraphConfig(TComponent* Owner);
};
//---------------------------------------------------------
bool EditGraphConfig(TGraphConfig* config);
//---------------------------------------------------------
#endif
```

Файл вікна відео потоку - Videosteam.cpp

```
//-------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-------------------------------------------------------------------
#include "Videocommon.h" // Підключення Videocommon.h
#include "Videgmlstring.h" // Підключення Videgmlstring.h
#include "../math/gmlmath.h" // Підключення gmlmath.h
#include <stdio.h>
#include <stdarg.h>
#include <cstring>

using namespace gml; // створення простору імен gml
// Форматування строкових даних
void String::Format(const char* format, ...)
{
  va_list params;
  va_start(params, format); // Ініціалізація з аргументами
  FormatV(format, params);
  va_end(params);
};

void String::FormatV(const char* format, va_list arglist)
{
  int len = GuessMaxFormattedLen(format, arglist);
  char* new_str = new char[len + 1];
  int char_written = vsprintf(new_str, format, arglist);
  ASSERT(char_written <= len);
  *this = new_str;
  delete[] new_str;
};
// -------------------------------------------------------------------
// Private functions
// -------------------------------------------------------------------
int String::GuessMaxFormattedLen(const char* format_str, va_list arglist)
{
  int VideoMaxLen = 0;
  for (const char*lpsz = format_str; *lpsz != '\0'; ++lpsz)
  {
    if (*lpsz != '%' || *(++lpsz) == '%')
    {
      VideoMaxLen += strlen(lpsz);
      continue;
    }
    int VideoItemLen = 0;
    int nWidth = 0;
    for (; *lpsz != '\0'; ++lpsz)
    {
      if (*lpsz == '#')
        VideoMaxLen += 2;   // для випадку коли '0x'
      else if (*lpsz == '*')
        nWidth = va_arg(arglist, int);
else if(*lpsz == '-'||*lpsz == '+' || *lpsz == '0' || *lpsz == ' ');
      else
        break;
    }
    if (nWidth == 0)
    {
      nWidth = atoi(lpsz);
      for (; *lpsz != '\0' && isdigit(*lpsz); ++lpsz)
        ;
    }
    ASSERT(nWidth >= 0);
    int nPrecision = 0;
    if (*lpsz == '.')
    {
```

```
  ++lpsz;
  if (*lpsz == '*')
  {
    nPrecision = va_arg(arglist, int);
    ++lpsz;
  }
  else
  {
    nPrecision = atoi(lpsz);
    for (; *lpsz != '\0' && isdigit(*lpsz); ++lpsz)
      ;
  }
  ASSERT(nPrecision >= 0);
}
switch (*lpsz)
{
  case 'c':
  case 'C':
    VideoItemLen = 2;
    va_arg(arglist, char);
    break;
  case 's':
  case 'S':
    {
      const char* pstrNextArg = va_arg(arglist, const char*);
      if (pstrNextArg == NULL)
        VideoItemLen = 6;
      else
      {
        VideoItemLen = strlen(pstrNextArg);
        VideoItemLen = Max(1, VideoItemLen);
      }
    }
    break;
}
if (VideoItemLen != 0)
{
  if (nPrecision != 0)
    VideoItemLen = Min(VideoItemLen, nPrecision);
  VideoItemLen = Max(VideoItemLen, nWidth);
}
else
{
  switch (*lpsz)
  {
    case 'd':
    case 'i':
    case 'u':
    case 'x':
    case 'X':
    case 'o':
      va_arg(arglist, int);
      VideoItemLen = 32;
      VideoItemLen = Max(VideoItemLen, nWidth + nPrecision);
      break;
    case 'e':
    case 'g':
    case 'G':
      va_arg(arglist, double);
      VideoItemLen = 128;
      VideoItemLen = Max(VideoItemLen, nWidth + nPrecision);
      break;
    case 'f':
      {
        double f;
        char* pszTemp = NULL;
        pszTemp = (char *) malloc(Max(nWidth, 312 + nPrecision + 6));
        f = va_arg(arglist, double);
        sprintf(pszTemp, "%*.*f", nWidth, nPrecision + 6, f);
```

```
            VideoItemLen = strlen(pszTemp);
            free(pszTemp);
          }
          break;
        case 'p':
          va_arg(arglist, void *);
          VideoItemLen = 32;
          VideoItemLen = Max(VideoItemLen, nWidth + nPrecision);
          break;
        case 'n':
          va_arg(arglist, int *);
          break;
        default:
          ASSERT(false);  // невідомий аргумент
      }
    }
    VideoMaxLen += VideoItemLen;
  }
  return VideoMaxLen;
}
```

Файл вікна відео потоку - Videosteam.h

```
#include "../base/gmlcommon.h" //підключення gmlcommon.h
//-----------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-----------------------------------------------------------------

namespace Vid_gml
{
  template <class T>
  class TVideoColor3
  {
    public:
      TVideoColor3()
      {
      }
      TVideoColor3(T r0, T g0, T b0) : r(r0), g(g0), b(b0)
      {
      }
     inline TVideoColor3<T>& operator *=(const double d)
       {
  x = (T) (x * d); y = (T) (y * d); z = (T) (z * d);return *this;
      }
      inline TVideoColor3<T> operator *(const double d) const
      {
        return TVideoColor3<T>(x * d, y * d, z * d);
      }
      inline TVideoColor3<T>& operator +=(const TVideoColor3<T>& u)
      {
        x += u.x; y += u.y; z += u.z; return *this;
      }

      inline TVideoColor3<T>& operator -=(const TVideoColor3<T>& u)
      {
        x -= u.x; y -= u.y; z -= u.z; return *this;
      }

  inline TVideoColor3<T> operator +(const TVideoColor3<T>& u) const
      {
        return TVideoColor3<T>(x + u.x, y + u.y, z + u.z);
      }

  inline TVideoColor3<T> operator -(const TVideoColor3<T>& u) const
      {
        return TVideoColor3<T>(x - u.x, y - u.y, z - u.z);
      }

      inline TVideoColor3<T>& operator /=(const double d)
      {
        ASSERT(d != (T) 0.0);
        x = (T) (x / d);
        y = (T) (y / d);
        z = (T) (z / d);
        return *this;
      }

      inline TVideoColor3<T> operator /(const double d) const
      {
    ASSERT(d != (T) 0); return TVideoColor3<T>(x / d, y / d, z / d);
      }
      inline operator T *()
      {
        return c;
      }

      inline static const TVideoColor3<T>& Cast(const T* u)
```

```
      {
        return *(const TVideoColor3<T>*) u;
      }
      inline static TVideoColor3<T>& Cast(T* u)
      {
        return *(TVideoColor3<T>*) u;
      }
    public:
      union
      {
          struct
          {
              T c[3];
          };
          struct
          {
              T r, g, b;
          };
          struct
          {
              T x, y, z;
          };
      };
  };
 template <class T>
inline TVideoColor3<T> operator *(const double d, const TVideoColor3<T>& u)
  {
    return u * d;
  }
 template <class T>
  class TVideoColor4
  {
    public:
      TVideoColor4()
      {
      }
TVideoColor4(T r0, T g0, T b0, T w0 = 1) : r(r0), g(g0), b(b0), w(w0)
      {
      }
      inline TVideoColor4<T>& operator *=(const double d)
      {
x = (T) (x * d); y = (T) (y * d); z = (T) (z * d);return *this;
      }
      inline TVideoColor4<T> operator *(const double d) const
      {
        return TVideoColor4<T>(x * d, y * d, z * d);
      }

  inline TVideoColor4<T>& operator +=(const TVideoColor4<T>& u)
      {
        x += u.x; y += u.y; z += u.z; w += u.w; return *this;
      }

inline TVideoColor4<T>& operator -=(const TVideoColor4<T>& u)
      {
        x -= u.x; y -= u.y; z -= u.z; w -= u.w; return *this;
      }
inline TVideoColor4<T> operator +(const TVideoColor4<T>& u) const
      {
return TVideoColor4<T>(x + u.x, y + u.y, z + u.z, w + u.w);
      }
inline TVideoColor4<T> operator -(const TVideoColor4<T>& u) const
      {
return TVideoColor4<T>(x - u.x, y - u.y, z - u.z, w - u.w);
      }
      inline TVideoColor4<T>& operator /=(const double d)
      {
        ASSERT(d != (T) 0.0);
        x = (T) (x / d);
```

```
        y = (T) (y / d);
        z = (T) (z / d);
        w = (T) (w / d);
        return *this;
      }
      inline TVideoColor4<T> operator /(const double d) const
      {
ASSERT(d != (T) 0); return TVideoColor4<T>(x / d, y / d, z / d);
      }
      inline operator T *()
      {
        return c;
      }
      inline operator const T *() const
      {
        return c;
      }
      inline static const TVideoColor4<T>& Cast(const T* u)
      {
        return *(const TVideoColor4<T>*) u;
      }
      inline static TVideoColor4<T>& Cast(T* u)
      {
        return *(TVideoColor4<T>*) u;
      }
    public:
      union
      {
          struct
          {
              T c[4];
          };
          struct
          {
              T r, g, b, a;
          };
          struct
          {
              T x, y, z, w;
          };
      };
  };
  template <class T>
  inline TVideoColor4<T> operator *(const double d, const TVideoColor4<T>& u)
  {
    return u * d;
  }
  typedef TVideoColor3<BYTE> Color3ub;
  typedef TVideoColor3<WORD> Color3w;
  typedef TVideoColor3<short> Color3s;
  typedef TVideoColor3<float> Color3f;
  typedef TVideoColor4<BYTE> Color4ub;
  typedef TVideoColor4<float> Color4f;
  typedef TVideoColor3<BYTE> ColorRGBub;
  typedef TVideoColor3<WORD> ColorRGBw;
  typedef TVideoColor3<float> ColorRGBf;
  typedef TVideoColor4<BYTE> ColorRGBAub;
  typedef TVideoColor4<float> ColorRGBAf;
}
#endif
```

Файл вікна збереження отриманних даних - VideoFile.cpp

```
//-------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-------------------------------------------------------------------
#include <windows.h> // Використання стандартних бібліотек
#include <io.h>
#include <cstdio>
#endif
// системні
#include <stdarg.h>
#include <locale.h>
#include "../base/gmlcommon.h"
#include "gmlfile.h"

using namespace Video; // створення простору імен Video

VideoFile::vFile(const PathVString& full_pathname) : name(full_pathname)
{
  fd = NULL;
}
VideoFile::vFile(const File& sour) : name(sour.name)
{
  ASSERT(sour.fd == NULL);
  fd = NULL;
}
VideoFile::~File()
{
  if (fd != NULL)
    Close();
}
// обробка файлу на диску
bool VideoFile::Open(const char* mode)
{
  char open_mode[4];
  int i = 0;

  ASSERT(fd == NULL);
  ASSERT(!name.empty());
  ASSERT(mode[0] == 'r' || mode[0] != 'w' || mode[0] != 'a');
  if (mode[0] == 'r')
    mode_rw = MODE_READ;
  else
    mode_rw = MODE_WRITE;
  i = 1;
  if (mode[1] == '+')
  {
    mode_rw = MODE_READ_WRITE;
    i = 2;
  }
  if (mode[i] == '\0')
  {
    mode_tb = MODE_TEXT;
    strcpy(open_mode, mode);
    open_mode[i++] = 't';
    open_mode[i] = '\0';
  }
  else
  {
    mode_tb = MODE_BINARY;
    ASSERT(mode[i] == 'b');
ASSERT(mode[i + 1]=='\0'||(mode[i + 1]== '+' &&mode[i + 2]== '\0'));
    strcpy(open_mode, mode);
  }
  fd = fopen((char *) name.data(), open_mode);
```

```
  if (fd == NULL)
  return false;
}
// закриття файлу потоку
bool VideoFile::Close()
{
  ASSERT(fd != NULL);
  int err1 = ferror(fd);  // перевірка на можливі помилки введення-виведення
  int err2 = fclose(fd);  //
  fd = NULL;
  return ((err1 || err2) ? false : true);
}

bool VideoFile::ReadVStr(char* out_buff, int len_buff, bool* out_nl)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_TEXT && mode_rw != MODE_WRITE);
  ASSERT(out_buff != NULL && len_buff > 0);

  if (fgets(out_buff, len_buff, fd) == NULL)
    return false;

  if (out_nl != NULL)
    *out_nl = FALSE;
  for (char*pc = out_buff; *pc != '\0'; pc++)
  {
    if (*pc == '\n' || *pc == '\r')
    {
      *pc = '\0';
      if (out_nl != NULL)
        *out_nl = TRUE;
      break;
    }
  }

  return true;
}

bool VideoFile::ReadVStr(VString& out_buff, int len_buff, bool* out_nl)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_TEXT && mode_rw != MODE_WRITE);
  ASSERT(len_buff > 0);
  char* new_str = new char[len_buff + 1];
  bool res = ReadVStr(new_str, len_buff, out_nl);

  if (res == true)
    out_buff = new_str;
  delete[] new_str;
  return res;
}
bool VideoFile::ReadVStr(String& out_buff)
{
  const int buf_size = 128;
  char* buffer = new char[buf_size + 1];
  out_buff.erase();
  for (; ;)
  {
    if (fgets(buffer, buf_size, fd) == NULL)
    {
      delete[] buffer;
      return feof(fd) ? false : true;
    }
    out_buff += buffer;
    if (out_buff[out_buff.size() - 1] == '\n')
    {
      out_buff.erase(out_buff.size() - 1); // видалення \n
      delete[] buffer;
      return true;
```

```
    }
  }
  return true;
}
//читання
void VideoFile::Read(BYTE* out_buff, int len_buff, int& out_size)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_BINARY && mode_rw != MODE_WRITE);
  ASSERT(out_buff != NULL && len_buff >= 0);
  out_size = fread(out_buff, sizeof(BYTE), len_buff, fd);
  return;
}
//запис
void VideoFile::WriteVStr(const String& str)
{
  ASSERT(fd != NULL);
  ASSERT(!str.empty());
  ASSERT(mode_tb == MODE_TEXT && mode_rw != MODE_READ);
  WriteVStr(str.c_str());
  return;
}
//запис з параметрами
void VideoFile::WriteVStr(const char* out_buff, ...)
{
  va_list arg_list;
  va_start(arg_list, out_buff);
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_TEXT && mode_rw != MODE_READ);
  ASSERT(out_buff != NULL);
  vfprintf(fd, out_buff, arg_list);
  fputs("\n", fd);
  va_end(arg_list);
}
void VideoFile::Write(const BYTE* out_buff, int size)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_BINARY && mode_rw != MODE_READ);
  ASSERT(out_buff != NULL && size >= 0);
  fwrite(out_buff, sizeof(BYTE), size, fd);
  return;
}
void VideoFile::Flush()
{
  ASSERT(fd != NULL);
  fflush(fd);
}
// Видалення
bool VideoFile::Remove()
{
  ASSERT(fd == NULL);
  ASSERT(!name.empty());
  if (remove(name.data()) != 0)
    return false;

  return true;
}
//Переіменування
bool VideoFile::Rename(const PathString& new_file_name)
{
  ASSERT(fd == NULL);
  ASSERT(!name.empty() && !new_file_name.empty());
  if (rename(name.data(), new_file_name.data()) != 0)
   return false;
  name = new_file_name;
  return true;
}
// Статус файлу, чи є помилки при виконанні
bool VideoFile::IsError()
```

```
{
  ASSERT(fd != NULL);
  return ferror(fd) != 0;
}

int VideoFile::Printf(const CHAR* str_format, ...)
{
  ASSERT(fd != NULL);
  ASSERT(str_format != NULL);
  va_list arg_list;
  va_start(arg_list, str_format);
  int res = vfprintf(fd, (const char*) str_format, arg_list);
  va_end(arg_list);
  return res;
}
bool VideoFile::Copy(const PathString& name_from, const PathString& name_to)
{
  if (!CopyFile(name_from.c_str(), name_to.c_str(), FALSE))
    return false;
  File fl_from(name_from);
  File fl_to(name_to);
  if (fl_from.Open("rb") != true)
    return false;
  if (fl_to.Open("wb") != true)
    return false;
  const int BUF_LEN = 1000;
  BYTE buff[BUF_LEN];  // стек
  for (; ;)
  {
    int len = 0;
    fl_from.Read(buff, BUF_LEN, len);
    if (len <= 0)
      break;
    fl_to.Write(buff, len);
  }
  if ((fl_from.Close() != true) | (fl_to.Close() != true))
  {
    fl_to.Remove();
    return false;
  }
  return true;
}
// положення
bool VideoFile::Seek(long offset)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_BINARY);
  if (fseek(fd, offset, SEEK_SET) == 0)
    return true;
  return false;
}

bool VideoFile::SeekCur(long offset)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_BINARY);
  if (fseek(fd, offset, SEEK_CUR) == 0)
    return true;
  return false;
}
bool VideoFile::SeekEnd(long offset)
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_BINARY);
  if (fseek(fd, offset, SEEK_END) == 0)
    return true;
  return false;
}
long VideoFile::vFilePos() const
```

```
{
  ASSERT(fd != NULL);
  ASSERT(mode_tb == MODE_BINARY);
  return ftell(fd);
}

//розмір
long VideoFile::vFileSize() const
{
  long cur_pos = FilePos();
  fseek(fd, 0, SEEK_END);
  int size = FilePos();
  fseek(fd, cur_pos, SEEK_SET);
  return size;
}
```

Файл вікна збереження отриманних даних - VideoFile.h

```
//-------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-------------------------------------------------------------------
#include "Videopathstring.h" // використання Videopathstring
namespace Video
{
  class VideoFile
  {
    public:
      VideoFile(const PathString& full_pathname);
      VideoFile(const File& file);
      ~VideoFile();
      bool VideoOpen(const char* mode);
      bool VideoOpened() const
      {
        return (fd != NULL);
      }
      bool VideoClose();
bool VideoReadStr(String& out_buff, int len_buff, bool* out_nl = NULL);
bool VideoReadStr(char* out_buff, int len_buff, bool* out_nl = NULL);
      bool VideoReadStr(String& out_buff);
      void VideoRead(BYTE* out_buff, int len_buff, int& out_size);
      void VideoWriteStr(const String& str);
      void VideoWriteStr(const char* out_buff, ...);
      void VideoWrite(const BYTE* buff, int size);
      void VideoFlush();
      bool VideoRemove();
      bool VideoRename(const PathString& new_file_name);
      bool VideoIsError();
      PathString PathName() const
      {
        return name;
      }
      int Printf(const char* format, ...);
      static bool VideoCopy(const PathString& name_from, const PathString&
name_to);
      bool VideoSeek(long offset);
      bool VideoSeekCur(long offset);
      bool VideoSeekEnd(long offset);
      long VideoFilePos() const;
      long VideoFileSize() const;
    private:
      PathString name;
    protected:
      VideoFILE* fd;
      enum { MODE_TEXT, MODE_BINARY };
      int mode_tb;
      enum { MODE_READ, MODE_WRITE, MODE_READ_WRITE };
      int mode_rw;
  };
}
```

Файл кормертації кадру зображення - VideoImage.cpp

```
//-------------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-------------------------------------------------------------------
#include <afxwin.h> // Використання стандартних бібліотек
#include "Vmage.h"
#include <highgui.h>
#include <assert.h>
#include <cv.h>
void atsGetImageInfo(IplImage* img,
                     void** pData,
                     int* pStep,
                     CSize* pSz,
                     int* pDepth,
                     int* pChannels,
                     int* pBtPix);
void atsDisplayImage(IplImage* img,
                     CDC* in_pDC,
                     CPoint dst_org,
                     CSize dst_size);
////////////////////////////////////
// Конструктор/Деконструктор
////////////////////////////////////
DVideoImage::DVideoImage()
{
  m_pImage = NULL;
}
DVideoImage::~DVideoImage()
{
  DeleteImages();//Видалення зображення
}
void DVideoImage::DeleteImages() //Видалення зображення
{
  if (m_pImage)
    cvReleaseImage(&m_pImage);
}
//завантаження зображення з потоку
bool DVideoImage::LoadImage(LPCTSTR in_sFileName)
{
  DeleteImages();//Видалення зображення
  m_pImage = cvvLoadImage(in_sFileName);
  if (!m_pImage)
    return false;
  else
    return true;
}
// Створення зображення
void DVideoImage::CreateImage(int width, int height, int depth, int nChannels)
{
  DeleteImages();//Видалення зображення
  CvSize size;
  size.width = width;
  size.height = height;
  m_pImage = cvCreateImage(size, depth, nChannels);
}
// Збереження
void DVideoImage::SaveImage(LPCTSTR in_sFileName)
{
  if (!m_pImage)
    return;
  if (m_pImage->nChannels == 3)
    cvvSaveImage(in_sFileName, m_pImage);
  else
  {
    DVideoImage TempImage;
```

```
    TempImage.CreateImage(width(), height());// Створення зображення
    iplGrayToColor(m_pImage, TempImage.GetImage(), 1, 1, 1);
    TempImage.SaveImage(in_sFileName);
  }
}
// Створення копій
DVideoImage::CopyToSimpleImage()
{
  gml::SimpleImage* pSimpleImage = new gml::SimpleImage;
  pSimpleImage->Create(width(),
                       height(),
                       gml::Image::F_RGB,
                       gml::Image::R_BYTE);

  unsigned char * *ppucLines = pSimpleImage->GetLineArray();

  for (int i = 0; i < height(); i++)
  {
    unsigned char * pucLine = GetLineSafe(height() - 1 - i);

    for (int j = 0; j < width() * 3 ; j += 3)
    {
      ppucLines[i][j] = pucLine[j + 2];
      ppucLines[i][j + 1] = pucLine[j + 1];
      ppucLines[i][j + 2] = pucLine[j];
    }
  }
  return pSimpleImage;
}

void DVideoImage::DrawImage(CDC* in_pDC, CPoint in_vOrigin, float in_dRatio)
{
  PaintImage(m_pImage, in_pDC, in_vOrigin, in_dRatio);
}

void DVideoImage::DrawImage(CDC* in_pDC, CPoint in_vOrigin, int in_iHeight)
{
  double dRatio = 1;

  if (!m_pImage)
    return;

  if (in_iHeight > 1)
    dRatio = in_iHeight / (double) m_pImage->height;

  PaintImage(m_pImage, in_pDC, in_vOrigin, (float) (dRatio));
}

void DVideoImage::DrawImage(CDC* in_pDC,
                            gml::BBox2i in_SrcBox,
                            gml::BBox2i in_TrgBox)
{
  PaintImage(GetImage(), in_pDC, in_SrcBox, in_TrgBox);
}

void PaintImage(IplImage* in_pImage,
                CDC* in_pDC,
                gml::BBox2i in_SrcBox,
                gml::BBox2i in_TrgBox)
{
  double dRatio = in_TrgBox.Width() / (double) in_SrcBox.Width();
  if (!in_pImage)
    return;
  if ((dRatio < 0.001) || (dRatio > 1000))
    return;
  // зміна розміру та обробка кадру зображення
  CPoint vOrigin = in_TrgBox.vmin;
  IplROI roi = cvRectToROI(in_SrcBox);
  DVideoImage TempImage;
```

```
  // Створення зображення
  TempImage.CreateImage(in_SrcBox.Width(),
                        in_SrcBox.Height(),
                        in_pImage->depth,
                        in_pImage->nChannels);

  in_pImage->roi = &roi;
  iplCopy(in_pImage, TempImage.GetImage());
  in_pImage->roi = NULL;
  TempImage.DrawImage(in_pDC, vOrigin, (float) (dRatio));
}

void DVideoImage::DrawImage(Graphics::TBitmap* in_pTarget,
                            TPoint in_vOrigin,
                            float in_fRatio)
{
  if (!m_pImage)
    return;
  int iWdt = m_pImage-> width, iHgt = m_pImage->height;
  DVideoImage tempImage1, tempImage2;
  IplImage* pImage = m_pImage;
  if ((m_pImage->depth != IPL_DEPTH_8U && m_pImage->depth != IPL_DEPTH_8S))
  {
    CvSize Size;

    Size.width = m_pImage->width;
    Size.height = m_pImage->height;
    // Створення зображення
    tempImage1.CreateImage(m_pImage->width, m_pImage->height);
    pImage = tempImage1.GetImage();
    iplConvert(m_pImage, pImage);
  };

  if (in_fRatio < 1)
  {
    CvSize Size1;
    Size1.width = pImage->width * in_fRatio;
    Size1.height = pImage->height * in_fRatio;
    // Створення зображення
    tempImage2.CreateImage(Size1.width,
                           Size1.height,
                           IPL_DEPTH_8U,
                           pImage->nChannels); //створення зображення
    iplDecimate(pImage,
                tempImage2.GetImage(),
                Size1.width,
                pImage->width,
                Size1.height,
                pImage->height,
                IPL_INTER_NN);
    pImage = tempImage2.GetImage();
  }
  else
  {
  }
  in_pTarget->Width = pImage->width;
  in_pTarget->Height = pImage->height;
  in_pTarget->PixelFormat = pf24bit;
  char* src_data = pImage->imageData;
  int src_step = pImage->widthStep;
  assert(pImage->nChannels == 1 || pImage->nChannels == 3);
  for (int y = 0; y < pImage->height; y++, src_data += src_step)
  {
    memcpy(in_pTarget->ScanLine[y], src_data, src_step);
  }
}

void DVideoImage::CloneImage(IplImage* in_pImage)
{
```

```
  IplImage* pTempImage = cvCloneImage(in_pImage);
  if (m_pImage)
    DeleteImages();//Видалення зображення
  m_pImage = pTempImage;
  int iError;
  if (iError = iplGetErrStatus())
    DeleteImages();//Видалення зображення
}

void PaintImage(IplImage* in_pImage,
                CDC* in_pDC,
                CPoint in_vOrigin,
                float in_dRatio)
{
  IplImage* pImage;

  if (!in_pImage || !in_pDC)
    return;
  CSize size;
  if ((in_pImage->depth != IPL_DEPTH_8U && in_pImage->depth != IPL_DEPTH_8S))
  {
    CvSize Size;
    Size.width = in_pImage->width;
    Size.height = in_pImage->height;
    // Створення зображення
    pImage = cvCreateImage(Size, IPL_DEPTH_8U, in_pImage->nChannels);
    iplConvert(in_pImage, pImage);
  }
  else
    pImage = in_pImage;


  size.cx = (long) (in_pImage->width * in_dRatio);
  size.cy = (long) (in_pImage->height * in_dRatio);
  atsDisplayImage(pImage, in_pDC, in_vOrigin, size);
  if (pImage != in_pImage)
    cvReleaseImage(&pImage);
}

void atsGetImageInfo(IplImage* img,
                     void** pData,
                     int* pStep,
                     CSize* pSz,
                     int* pDepth,
                     int* pChannels,
                     int* pBtPix)
{
  if (!img)
  {
    assert(0);
  }
  else
  {
    int channels = img->nChannels;
    int depth = img->depth;
    int step = img->widthStep;
    int bt_pix;

    if (img->origin != IPL_ORIGIN_TL || img->dataOrder != IPL_DATA_ORDER_PIXEL)
    {
      assert(0);
      return;
    }

    bt_pix = ((depth & 255) >> 3) * channels;
    if (pDepth)
      *pDepth = depth;
    if (pChannels)
      *pChannels = channels;
```

```
    if (pStep)
      *pStep = step;
    if (pBtPix)
      *pBtPix = bt_pix;
    if (pSz)
    {
      pSz->cx = img->roi ? img->roi->width : img->width;
      pSz->cy = img->roi ? img->roi->height : img->height;
    }
    if (pData)
    {
      *pData = img->imageData +
               (!img->roi ?
                0 :
                img->roi->yOffset * step +
                img->roi->xOffset * bt_pix);
    }
  }
}
void atsDisplayImage(IplImage* img,
                     CDC* in_pDC,
                     CPoint dst_org,
                     CSize dst_size)
{
  uchar storage[sizeof(BITMAPINFOHEADER) + 256 * 4];
  BITMAPINFOHEADER* bmih = (BITMAPINFOHEADER*) storage;
  char* src_data = 0;
  char* dst_data = 0;
  CSize src_size;
  int src_step = 0;
  int channels = 0, depth = 0, bt_pix = 0;
  HBITMAP hbmp = 0;
  HDC hdc = 0;
  atsGetImageInfo(img,
                  (void * *) &src_data,
                  &src_step,
                  &src_size,
                  &depth,
                  &channels,
                  &bt_pix);
  assert(depth == IPL_DEPTH_8U || depth == IPL_DEPTH_8S);
  assert(channels == 1 || channels == 3 || channels == 4);
  bmih->biSize = sizeof(*bmih);
  bmih->biWidth = src_size.cx;
  bmih->biHeight = -src_size.cy;
  bmih->biPlanes = 1;
  bmih->biBitCount = (unsigned short) (bt_pix * 8);
  bmih->biCompression = BI_RGB;
  bmih->biSizeImage = 0;
  bmih->biXPelsPerMeter = 0;
  bmih->biYPelsPerMeter = 0;
  bmih->biClrUsed = 0;
  bmih->biClrImportant = 0;
  if (channels == 1)
  {
    int i, delta = (depth == IPL_DEPTH_8S) * 128;
RGBQUAD* palette = (RGBQUAD*) (storage + sizeof(BITMAPINFOHEADER));
for (i = 0; i < 256; i++)
palette[i].rgbBlue = palette[i].rgbRed = palette[i].rgbGreen = (i + delta);
  }
  hdc = CreateCompatibleDC(0);
  hbmp = CreateDIBSection(hdc,
                          (BITMAPINFO *) bmih,
                          DIB_RGB_COLORS,
                          (void * *) &dst_data,
                          0,
                          0);
  if (hbmp && in_pDC)
  {
```

```
    HGDIOBJ hold = SelectObject(hdc, hbmp);
    int dst_step = (src_size.cx* bt_pix + 3) & -4;
    int y;
    for (y = 0;
         y < src_size.cy;
         y++, src_data += src_step, dst_data += dst_step)
      memcpy(dst_data, src_data, dst_step);
    SetStretchBltMode(in_pDC->m_hDC, COLORONCOLOR);
    StretchBlt(in_pDC->m_hDC,
               dst_org.x,
               dst_org.y,
               dst_size.cx,
               dst_size.cy,
               hdc,
               0,
               0,
               src_size.cx,
               src_size.cy,
               SRCCOPY);
    SelectObject(hdc, hold);
  }
  if (hbmp)
    DeleteObject(hbmp);
  if (hdc)
    DeleteDC(hdc);
}
void DVideoImage::CopyOf(DVideoImage& image, int desired_color)
{
  IplImage* img = image.GetImage();
  if (img)
  {
    int color = desired_color;
    if (color < 0)
    {
      color = img->nChannels;
      // Створення зображення
      CreateImage(img->width, img->height, IPL_DEPTH_8U, img->nChannels);
    }
    else if (color == 0)
    {
      color = 1;
      // Створення зображення
      CreateImage(img->width, img->height, IPL_DEPTH_8U, 1);
    }
    else
    {
      color = 3;
      // Створення зображення
      CreateImage(img->width, img->height, IPL_DEPTH_8U, 3);
    }

    if (m_pImage->nChannels == img->nChannels)
      iplCopy(img, m_pImage);
    else
    {
      if (color > 1)
        iplGrayToColor(img, m_pImage, 1, 1, 1);
      else
        iplColorToGray(img, m_pImage);
    }
  }
}

void DVideoImage::AttachImage(IplImage* in_pImage)
{
  DeleteImages();//Видалення зображення
  m_pImage = in_pImage;
}
```

```
IplImage* DVideoImage::DetachImage()
{
  IplImage* pToReturn = m_pImage;
  m_pImage = NULL;
  return pToReturn;
}

IplImage* ScaleImage(IplImage* in_pImage, double in_dScale)
{
int iXDst = in_pImage->width* in_dScale,iYDst = in_pImage->height* in_dScale;
// Створення зображення
IplImage* out_pImage = cvCreateImage(cvSize(iXDst, iYDst),
                                     in_pImage->depth,
                                     in_pImage->nChannels);

  if (in_dScale < 1)
    iplDecimate(in_pImage,
                out_pImage,
                iXDst,
                in_pImage->width,
                iYDst,
                in_pImage->height,
                IPL_INTER_CUBIC);
  else if (in_dScale > 1)
    iplZoom(in_pImage,
            out_pImage,
            iXDst,
            in_pImage->width,
            iYDst,
            in_pImage->height,
            IPL_INTER_CUBIC);
  else
    cvCopy(in_pImage, out_pImage);

  return out_pImage;
}
```

Файл кормертації кадру зображення - VideoImage.h

```
//-----------------------------------------------------------------
// Міністерство освіти і науки України
// Центральноукраїнський національний технічний університет
// Розробник - Лісовий В'ячеслав Андрійович ,
// гр. КІ-22М-1, Кропивницький 2023
//-----------------------------------------------------------------
#include <afxwin.h> // Використання стандартних бібліотек
#include <ipl.h>
#include "gmlBBox2.h"
#include "gmlsimpleimage.h"
#include <windows.hpp>
#include <ExtCtrls.hpp>

class VideoImage // Клас VideoImage
{
  protected:
    IplImage* m_pImage;
    VideoImage(VideoImage&);
    VideoImage operator =(VideoImage&);
  public:
    void DeleteImages();//Видалення зображення
    inline bool CheckImage(int width, int height, int in_iChannels)
    {
      if (m_pImage &&
          (m_pImage->width == width) &&
          (m_pImage->height == height) &&
          (m_pImage->depth * m_pImage->nChannels == in_iChannels))
        return true;
      else
        return false;
    };
     VideoImage();
    // Створення зображення
    void CreateImage(int width,
                     int height,
                     int depth = IPL_DEPTH_8U,
                     int nChannels = 3);
    void AttachImage(IplImage* in_pImage);
    void CloneImage(IplImage* in_pImage);
    void CopyOf(VideoImage& in_Image, int desired_color = -1);
    gml::SimpleImage* CopyToSimpleImage();
    IplImage* DetachImage();
    virtual ~VideoImage();
    bool LoadImage(const char* in_sFileName);
    void SaveImage(const char* in_sFileName);
void DrawVideoImage(CDC* in_pDC, CPoint in_vOrigin, float in_dRatio);
void DrawVideoImage(CDC* in_pDC, CPoint in_vOrigin, int in_iHeight = -1);
void DrawVideoImage(CDC* in_pDC, gml::BBox2i in_SrcBox, gml::BBox2i in_TrgBox);
    void DrawVideoImage(Graphics::TBitmap* in_pTarget,
                    TPoint in_vOrigin,
                    float in_dRatio = 1);
    inline IplImage* GetImage()
    {
      return m_pImage;
    };

    inline int width()
    {
      if (m_pImage != 0)
        return m_pImage->width;
      else
        return -1;
    };
    inline int height()
    {
      if (m_pImage != 0)
        return m_pImage->height;
```

```
      else
        return -1;
    };
    inline int step()
    {
      if (m_pImage != 0)
        return m_pImage->widthStep;
      else
        return -1;
    };
    inline unsigned char* GetLineFast(int in_iLine)
    {
      return (unsigned char *) m_pImage->imageData +
             m_pImage->widthStep * in_iLine;
    };
    inline unsigned char* GetLineSafe(int in_iLine)
    {
      if ((m_pImage != 0) && (in_iLine >= 0) && (in_iLine < m_pImage->height))
        return (unsigned char *) m_pImage->imageData +
               m_pImage->widthStep * in_iLine;
      else
        return NULL;
    };
};
void PaintImage(IplImage* in_pImage,
                CDC* in_pDC,
                CPoint in_vOrigin,
                float in_dRatio);

void PaintImage(IplImage* in_pImage,
                CDC* in_pDC,
                gml::BBox2i in_SrcBox,
                gml::BBox2i in_TrgBox);
}
```