МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: 123 «Комп'ютерна інженерія»

Освітньо-професійна програма: «Комп'ютерна графіка і Web-дизайн»

Група: 4КГ-07

Дипломний проект

здобувача освіти денної форми навчання
КГ.07.12.000.ДП

СОКОЛОВА
МАТВІЯ ЄВГЕНОВИЧА

м. Одеса
2024 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Спеціальність: **123 «Комп'ютерна інженерія»**

Освітньо-професійна програма: «**Комп'ютерна графіка і Web-дизайн»**

Група: **4КГ-07**

ПОЯСНЮВАЛЬНА ЗАПИСКА

до дипломного проекту на тему:

Програмна реалізація алгоритму сканування пошкоджень відео-файлів

Проектний матеріал складається з пояснювальної записки на 74 сторінках та графічного (презентаційного) матеріалу на 17 аркушах (слайдах)

Дипломник _____ (Соколов М.Є.)

Керівник _____ (Шувалова І.О.)

Консультанти:

з економічного розділу _____ (Іванченков В.С.)

з розділу охорони праці та техніки безпеки _____ (Чорновол Н.І.)

з нормоконтролю _____ (Петрашова В.І.)

старший консультант _____ (Кривченко Ю.В.)

До захисту допущений

Голова циклової комісії _____ (Кривченко Ю.В.)

Завідувач відділення _____ (Скорнякова О.В.)

Захист «19» 06 2024 р. Протокол ЕК № 3

Оцінка ЕК 5 (відмінно) 92 б.

Секретар ЕК _____

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

Відділення *комп'ютерних систем* Комісія *КТ та ПІ*

Спеціальність *123 «Комп'ютерна інженерія»*

Освітньо-професійна програма *«Комп'ютерна графіка і Web-дизайн»*

ЗАТВЕРДЖУЮ:

Заст. дир. з НВР *Беркань І.В.*

"17" 01 2024 р.

ЗАВДАННЯ

на дипломний проект

Здобувачеві освіти *Соколову Матвію Євгеновичу*

(прізвище, ім'я, по батькові)

1. Тема проекту *Програмна реалізація алгоритму сканування пошкоджень відео-файлів*

затверджена наказом по коледжу від "2" 11 2023 р. № 244-АД-ОД

2. Термін здачі закінченого проекту 10.06.2024

3. Вихідні данні до проекту *Технічна документація з організації щілинної зйомки; технічні характеристики форматів відео-файлів та статичних цифрових зображень; статистичні дані обробки відео-файлів та появи дефектів зображення; програмні засоби обробки відео-зображень; параметри щілинної зйомки та відповідні налаштування системи пошуку дефектних кадрів; загальні вимоги до ергономіки візуального інтерфейсу програмного застосунку у ОС Windows*

4. Зміст розрахунково-пояснювальної записки (перелік питань, які необхідно розробити)

Аналіз методів перевірки цілісності файлів для сканування пошкоджень відео-файлів

Розробка програмного застосунку для сканування пошкоджень відео-файлів

Розробка алгоритмів для сканування пошкоджень відео-файлів

Розробка інтерфейсу застосунку для сканування пошкоджень відео-файлів

Аналіз роботи алгоритму сканування пошкоджень відео-файлів

5. Перелік графічного (презентаційного) матеріалу (з точним зазначенням обов'язкових креслень, кількості слайдів)

Загальні схеми побудови кадрової, скануючої, щілинної зйомки; Схема отримання фото щілинної зйомки; Структурна схема роботи застосунку сканування пошкоджень відео-файлів; Схеми обробки відео-кадрів; Результати щілинної зйомки при різних режимах швидкості руху; Загальна схема просторово-часового алгоритму обробки кадру; Блок-схема процедур та параметрів ПЗ; Блок-схема процесу сканування відео-файлу для пошуку дефектів; Інтерфейс головного вікна застосунку; Ілюстрація визначення дефектів на відео-кадрах

6. Консультанти по проекту, із зазначенням розділів проекту, що їх стосується

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
Основний розділ	*Шувалова І.О.*		
Економічний розділ	*Іванченков В.С.*		
Розділ охорони праці	*Чорновол Н.І.*		
Нормоконтроль	*Петрашова В.І.*		
Старший консультант	*Кривченко Ю.В.*		

7. Дата видачі завдання 15.01.2024

Керівник *Шувалова І.О.* _____
(підпис)

Завдання прийняв до виконання *Соколов М.Є.* _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/р	Назва етапів дипломного проекту	Термін виконання етапів дипломного проекту (роботи)	Відмітка про виконання
1	*Вступ. Постановка мети та задач проектування*	28.04.24	виконано
2	*Аналіз методів перевірки цілісності файлів*	01.05.24	виконано
3	*Огляд засобів для перевірки пошкоджень файлів*	02.05.24	виконано
4	*Аналіз властивостей відео-кодеків та огляд*	05.05.24	виконано
	форматів відео-файлів		
5	*Перевірка пошкоджень відео-файлів*	06.05.24	виконано
6	*Застосування методу щілинної зйомки для*	09.05.24	виконано
	перевірки цілісності відео-файлів		
7	*Розробка алгоритму для пошуку дефектних кадрів*	15.05.24	виконано
8	*Розробка структури програмного застосунку*	17.05.24	виконано
9	*Розробка інтерфейсу застосунку*	20.05.24	виконано
10	*Аналіз отриманих результатів роботи ПЗ для*	25.05.24	виконано
	сканування пошкоджень відео-файлів		
11	*Економічні розрахунки та питання з охорони праці*	02.06.24	виконано
12	*Підготовка графічної частини проекту*	04.06.24	виконано
13	*Підготовка проекту до захисту та тестування ПЗ*	08.06.24	виконано

Дипломник _____
(підпис)

Керівник _____
(підпис)

Формат	Зона	Поз.	Позначення	Назва	Кіл.	Примітка
				Документація		
			КГ 07. 12 000. 00 ДП	Дипломний проект		
А4			КГ 07. 12 000. 00 ДП ПЗ	Пояснювальна записка	1	

Зм.	Арк.	№ докум.	Підпис	Дата	КГ 07. 12 000. 00 ДП		
Розробив		Соколов М.Є.		10.06	Програмна реалізація алгоритму сканування пошкоджень відео-файлів	Літ.	Аркуш 4 / Аркушів 74
Перевірив		Шувалова І.О.		10.06.		Н Д П	
Н. контр.		Петрашова В.І.		10.06		ВСП "ОТФК ОНТУ" гр.4КГ-07	
Затверд.		Кривченко Ю.В.		10.06			

ЗМІСТ

Вступ……7
1 Основний розділ……8
1.1 Аналіз методів перевірки цілісності файлів для сканування пошкоджень відео-файлів……8
1.1.1 Застосування дешифраторів для перевірки пошкоджень файлів……8
1.1.2 Застосування відео-контейнерів……9
1.1.3 Аналіз властивостей відео-кодеків……10
1.1.4 Огляд форматів відео-файлів та їх порівняння……11
1.1.5 Перевірка пошкоджень відео-файлів за допомогою їх декодування……15
1.1.6 Застосування методу щілинної зйомки для перевірки цілісності відео-файлів……18
1.2 Склад програмно-технічних засобів розробки ПЗ……24
1.2.1 Технічні засоби розробки……24
1.2.2 Середовище розробки ПЗ……24
1.3 Реалізація програмного застосунку для сканування пошкоджень відео-файлів ……30
1.4 Розробка алгоритму для пошуку дефектів на відео-кадрах ……38
1.5 Розробка інтерфейсу застосунку ……41
1.6 Аналіз отриманих результатів роботи програмного забезпечення для сканування пошкоджень відео-файлів……44
2 Економічний розділ……49
2.1 Резюме……50
2.2 Визначення трудомісткості розробки програмного забезпечення……50
2.3 Розрахунок ціни програмного продукту……53
3 Розділ охорони праці та техніки безпеки ……55
3.1 Вступ……55
3.2 Аналіз умов праці й забезпечення безпеки при виконанні основних видів робіт на об'єкті дипломного проектування……55

3.3 Розробка заходів з охорони праці.....55
3.3.1 Виробничі приміщення.....56
3.3.2 Мікроклімат робочої зони працівників, вентиляція.....56
3.3.3 Освітлення робочого місця, шум, вібрація.....57
3.3.4 Організація робочого місця користувача ПК.....58
3.4 Пожежна безпека.....58
Висновки.....59
Перелік використаних інформаційних джерел.....60
Додаток А. Текст головного модулю застосунку сканування пошкоджень відео-файлів.....61
Додаток Б. Слайди мультимедійної презентації.....67

ВСТУП

На сьогоднішній день в галузі інформаційних технологій актуальною темою є пошук та впровадження нових можливостей збереження та застосування фото- і video- інформації, що спроможні бути забезпечені рівнем сучасних обчислюваних можливостей.

Популярність застосування video-даних, зйомка, збереження, редагування і поширення файлів через мережу Інтернет стрімко зросла. В всьому світі стали популярними Інтернет-ресурси, що дозволяють будь-якому користувачеві переглядати video-фрагменти і завантажувати свої власні на сервіс youtube і інші. Дуже корисним є реалізації video-роликів задля конференцій, запис важливих заходів, бізнес-зустрічей і поширення їх між всіма, хто їх потребує. Однак проблема контролю цільності video-файлів присутня в усіх подібних застосуваннях та додатках.

Задля покращення ефективності збереження цільності video-файлів і ліквідації зайвого процесу перегляду video-фрагменту на перевірку його цільності і відсутності помилок і артефактів існують спеціальні алгоритми. Подібні алгоритми перевіряння video-файлів, зокрема, розроблений в даній дипломній роботі, являють собою зручний інструмент Скан-пошук будь-якого окремого фрейму із video-файла задля відшукування помилок і ушкоджень.

Поза основу задля власної реалізації алгоритму відшукування пошкоджень на video-фреймах буде взято цифрову щілинну зйомку, що використовує набір з декількох сотень до декількох тисяч окремих фреймів задля реалізації єдиного image. Це image складається із елементів будь-якого із фреймів та має істотний потенціал задля фіксації великого обсягу інформації порівняно із традиційною серійною зйомкою. ІЗ іншої точки зору цифрова щілинна зйомка – це цікава, але малопоширена техніка та на сьогоднішній день існує дефіцит інформаційних ресурсів, присвячених цієї технології, програмних засобам задля її реалізації.

В дипломній роботі аналізуються алгоритми, особливості і специфіка застосування методу відшукування при обробці video-файлів, виконується програмна реалізація алгоритму Скан-пошук ушкоджень video-файлів.

1 ОСНОВНИЙ РОЗДІЛ

1.1 Аналіз методів перевіряння цільності файлів задля Скан-пошук ушкоджень video-файлів

1.1.1 Застосування дешифраторів задля перевіряння ушкоджень файлів

Простого способу дізнатися, чи пошкоджений файл без застосування контрольної суми – не існує. Контрольні суми, зокрема CRC, хеш MD5/SHA-1 дозволяють визначити цілісність і автентичність вихідного файлу [1].

На сьогоднішній день більшість форматів аудіо- і video- контейнерів не містять контрольних сум, отже немає ніяких способів перевірити їх цілісність. Єдине, що можливо зробити – спробувати розшифрувати повний файл та визначити, чи розшифровує декодер несподівані попередження чи помилки.

Декодер чи дешифратор – логічний пристрій, який перетворює код числа, що поступило на вхід, у сигнал на одному із його виходів. Вихідними функціями дешифратора є різноманітні конституенти одиниці. Коли число представлено в вигляді n двійкових розрядів, то дешифратор повинен мати 2^n виходів. Дешифратор довільної складності спроможне бути складений із трьох базових логічних елементів: кон'юнкції, диз'юнкції і інверсії [2].

Розрізняють такі види дешифраторів поза принципом дії:

- послідовні;
- паралельні;
- паралельно-послідовні.

Розрізняють дешифратори першого і другого роду:

- дешифратори першого роду реалізують систему функцій, кожна із яких приймає одиничне значення при відповідному одиничному значенні вхідного слова;
- дешифратори другого роду реалізують систему функцій, кожна із яких приймає одиничне значення при визначених діапазонах вхідного слова.

Поза способом побудови розрізняють:

- лінійні дешифратори. В такому дешифраторі n змінних представляють

сукупність незв'язаних між собою 2^n систем збігу на n входів, кожна із яких реалізує відповідну конституенту одиниці;

– пірамідальні дешифратори. Будуються поза принципом послідовних каскадів: на першому каскаді реалізуються конституенти одиниці задля двох змінних, на $n-1$ реалізуються конституенти одиниці задля n змінних. При цьому на вході із попереднього каскаду отримується вихід.

1.1.2 Застосування video-контейнерів

Велику роль при створені video-контенту грають не тільки параметри розміру, роздільної здатності чи відношення сторін. Потрібно разом з цим брати до уваги формат video-файла із урахуванням стискання (компресії), яка властива різним форматам файлів [3].

Мультимедійний контейнер є форматом файлу чи потоковим форматом, специфікації якого визначають тільки спосіб представлення даних (проте не алгоритм кодування) у межах одного файлу. Він визначає розмір і структуру файлів, що представляються. Медіаконтейнер фактично є метаформатом, адже він зберігає дані та інформацію про те, як саме дані будуть зберігатися всередині файлу. Внаслідок програма, яка здатна коректно ідентифікувати та відкрити файл (зчитати потік), записаний у якому-небудь форматі, внаслідок спроможне бути не здатна декодувати фактичні дані, закодовані в медіа-контейнері.

Контейнер здатний зберігати будь-який тип даних, проте в реальності задля окремих типів файлів існують окремо визначені групи контейнерів. Ці групи налаштовані задля специфічних потреб та інформації, що буде зберігатися у них. Медіа-контейнери є типовим прикладом такої групи файлових контейнерів, котрі створено задля збереження медіа-інформації, яка поділяється на image, video-записи та звукові сигнали.

Задля збереження video-записів медіа-контейнер має не тільки зберігати video- і аудіо- потоки, але та містити мітки задля синхронізації при відтворенні video. В медіа-контейнері спроможне зберігатися декілька потоків одного типу, наприклад фільм, відповідно video-потік із декількома доріжками, відповідно аудіо-потоками, і субтитрами (потік символів).

1.1.3 Аналіз властивостей video-кодеків

Задля ідентифікації і чергування різних типів даних застосовується контейнер файлу. Разом з цим деякі формати контейнерів спроможні зберігати у собі різні типи звукових даних, закодовані деяким кодеком. Кодек є пристроєм чи програмою, яка здатна виконувати перетворення потоків даних чи сигналів. Задля збереження, передачі чи шифрування потоку даних чи сигналу його кодують поза поміччю кодеку, проте задля перегляду чи редагування – декодують. Кодеки дуже часто використовують при програмній обробці video і звуку. В кодеках спроможні використовуватися два види стискання даних. Більшість таких video-кодеків надають перевагу стисненню із витратами, адже це значно зменшує обсяг даних задля збереження чи надсилання файлів. Проте такий вид стискання призводить до зниження якості звуку чи image при відтворенні video-файла. В випадку коли не рекомендована втрата даних, використовують кодеки, що виконують стискання без витрат (lossless), що є корисним при необхідності подальшого редагування video-файла, адже задля редагування потрібна найякісніша роздільна здатність і якість video [4].

Найважливішим поняттям задля вибору формату video-файла є поняття стискання. В “сирих”, відповідно нестиснених video-файлах, як правило, занадто великий обсяг – відеоінформація досить складна та потребує багато місця задля збереження. Замість того, щоб змушувати користувачів працювати із величезними файлами, video-індустрія розробила концепцію стискання файлів, призначену задля зменшення їх розміру. Кожний окремий метод стискання сформований в вигляді кодеку. Чим менші по обсягу файли формує кодек, тим більш ефективним він є. Деякі кодеки разом з цим надають можливість вибору ступеню стискання. Як правило, він пов’язаний із швидкостю відтворення video-файла (найчастіше вимірюється в кілобайтах поза секунду). Чим більше швидкість, тим краще якість файлу, проте з цього збільшується його розмір. Вибір формату файлу і кодеку – це компроміс між розміром файлу і якістю відтворення video- і аудіо-даних.

Різні формати файлів використовують різні форми стискання, отже вибір

кодеку важливий. Формат файлу чи медіа-контейнер – це конкретний спосіб кодування цифрової інформації. Існує безліч різних файлових форматів задля різного застосування, наприклад, формат JPEG задля збереження цифрових фотографій.

1.1.4 Огляд форматів video-файлів і їх порівняння

Застосовується декілька різних файлових форматів задля збереження відеоінформації. В будь-якого із них свої переваги та обмеження, відповідно потрібно ознайомитися із ними до того, як почати створювати video. Загалом медіа-контейнери застосовуються задля збереження медіа-інформації, до якої відноситься аудіо, video і image. Медіа-контейнер – це формат файлу, специфікації якого стосуються лише методу збереження даних у межах одного файлу.

Таблиця 1.1. Види медіа-контейнерів задля збереження зображень

Формат файлу	Розширення	Характеристики
FITS	.fits	Цифровий формат файлів, який застосовується задля збереження, передачі і опрацювання зображень та їх метаданих (електронних таблиць)
TIFF	.tif чи .tiff	Графічний формат файлів, розроблений як один із основних універсальних форматів інсценізації зображень високої якості

Треба зазначити, що медіа-контейнери більш складного типу спроможні підтримувати множинні аудіо- та video-потоки, текстові субтитри, інформацію по розділах, метадані. В деяких випадках заголовок файлу, більшість метаданих та синхронізаційні дані визначаються самим контейнером. Наприклад, є контейнери, створені задля video із низькою роздільною здатність та є контейнери, оптимізовані задля високоякісних файлів, що містять багато потоків високої якості. Частини структури контейнеру video-файла мають різні імена. В RIFF та

PNG їх часто називають шматками, у MPEG-TS їх називають пакетами, проте у JPEG вони називаються «сегменти». Основний контент даних складових частин називається даними чи «корисним навантаженням». В більшості із них кожна складова частина має свій заголовок, у той час як медіа-контейнер TIFF натомість зберігає зміщення, що призводить до складності при збереженні інформації. Модульні складові частини полегшують відновлення інших складових частин в разі псування файлу чи при «випаданні» фреймів. Медіа контейнери задля збереження різних даних представлені в таблицях 1.1 – 1.3.

Таблиця 1.2. Види медіа-контейнерів задля збереження аудіо-даних

Формат файлу	**Розширення**	**Характеристики**
AIFF	*.aif, .aiff чи .aifc*	Спеціальний аудіо-формат файлів задля відтворення і збереження даних звуку на носіях інформації
WAV	*.wav*	Формат файлу-контейнера призначений задля збереження запису оцифрованого аудіо-потоку, підвид RIFF. WAV-контейнер застосовується задля збереження звуку у імпульсно-кодовій модуляції
XMF	*.xmf*	Сімейство пов'язаних музичних форматів файлів, створених задля того, щоб включати у себе різні типи мультимедійних файлів, таких як файли даних MIDI (.MID), файли інструментів (.DLS) та аудіо-хвиль (.WAV)

Окрім того, більшість сучасних медіа-контейнерів пристосована задля збереження усіх чи майже усіх типів медіа-інформації. Найпопулярніші із них представлені в таблиці 1.3. Деякі із них разом з цим спроможні зберігати у собі додаткову інформацію.

Таблиця 1.3. Види медіа-контейнерів задля збереження різних типів даних

Формат файлу	**Розширення**	**Характеристики**
3GP	*.3gpp, .3gpp2*	Файловий формат задля мультимедіа-контейнерів, загальноприйнятих задля мобільних пристроїв
H.264	*.mpg, mp4*	Формат стискання, використовуваний в багатьох файлах MPEG-4, 3GP і QuickTime. Більш ефективний, ніж звичайний кодек MPEG-4
Matroska	*.mkv*	Відкритий проект. Контейнер має можливість зберігати багато аудіо-, video-потоків та субтитрів
MPEG-1	*.mpg, .mpeg*	Початковий формат MPEG. Кодує video низької роздільної здатності
MPEG-2	*.mpg, .mpeg*	Наступна версія формату MPEG, що відтворює аудіо і video кращої якості. Застосовується в телебаченні і деяких супутникових сервісах
MPEG-4	*.mpg, mp4*	Наступна версія формату MPEG, що оптимізована задля високої роздільної здатності і інтернет-video. Спроможне використовувати кодеки H.264 та DivX
MPEG-PS	*.mpg, .mpeg*	Формат потокового video, що застосовується задля потокового відтворення файлів MPEG-1, MPEG-2
QuickTime	*.mov, .qt*	Закритий аудіо- і video- формат задля Apple. Працює як на MacOS, так та на ОС Windows

RealVideo	*.rm, .rt*	Формат, що застосовується в RealPlayer. Формат спроможне зберігати як video- так та аудіо-файли. Має високий ступень стискання, дає на виході video нижчої якості в порівнянні із конкуруючими форматами
WebM	*.webm*	Один із типових форматів video, розроблених спеціально задля збереження матеріалів, що не потребують ліцензій. Розроблений задля застосування із HTML5-video і підтримується більшістю браузерів
Windows Media Video (WMV)	*.wmv*	Закритий файловий формат video Microsoft задля відтворення в Windows Media Player
Audio Video Interleave (AVI)	*.avi*	Формат контейнеру, здатний зберігати дані, стиснені різними кодеками
DivX	*.divx*	Популярний в Інтернет кодек із високою якістю image та високим ступенем стискання. Є підмножиною формату MPEG-4, що підтримує роздільну здатність до 1080p
Digital Video (DV)	*.dv*	Формат із втратами, застосовується в video-камерах. Video із таким форматом поміщуються в контейнери, такі як AVI чи QuickTime
Flash Video	*.flv*	Популярний формат мультимедіа-файлів з Adobe. Застосовується задля відтворення відеороликів

1.1.5 Перевірка ушкоджень video-файлів поза поміччю їх декодування

Як зазначалося в п.1.1.1 можливо скористатися декодерами задля того, щоб перевірити video-файл на псування, проте немає можливості визначити контрольну суму оригінального файлу. Поза поміччю декодування можливо отримати дані, що допоможуть виявити помилки. Це доволі складна операція, проте існують спеціальні фреймворки, котрі дозволяють це зробити. Декодери, у цілому, спроможні виявити помилки при дешифруванні video-файла, відповідно video-файл перевіряється на наявність ушкоджень.

Найрозповсюдженим додатком задля дешифрування video-файлів є FFmpeg. FFmpeg є провідним мультимедійним фреймворком, здатним розшифровувати, кодувати, декодувати, переглядати, фільтрувати і відтворювати майже всі формати незалежно з того, чи були вони розроблені певним комітетом зі стандартів, громадою чи корпорацією. Він разом з цим дуже портативний: FFmpeg компілює, запускає і передає тестувальну інфраструктуру FATE через Linux, Mac OS X, Microsoft Windows, BSD, Solaris, тощо, в різних середовищах побудови, архітектури машин і конфігурацій. Він містить наступні утілити: libavcodec, libavutil, libavformat, libavfilter, libavdevice, libswscale і libswresample, котрі спроможні використовуватися програмами, проте разом з цим ffmpeg, ffplay і ffprobe, котрі спроможні бути використані кінцевими користувачами задля перекодування і відтворення.

Додаток FFmpeg є одним із доступних варіантів безкоштовного софтового забезпечення. Проект знаходиться у стані розроблювання. Задля того, щоб розпочати перевірку і дешифрування, потрібно запустити на виконання наступний код, який першочергово запускає утиліту FFmpeg, вказує параметри, котрі обов'язково потрібні задля процесу дешифрування, вказує файл, шлях до файлу, що перевіряється:

ffmpeg.exe -loglevel -f yuv4mpegpipe -i c: \foo \file_to_test.mkv -> NUL

Наведений вище код спроможне виявити не всі псування, проте лише деякі із них. Цей варіант перевіряння цільності video-файлів не є універсальним, адже спроможне виявити псування, котрі призвели до неспецифічного бітового потоку,

що здатен виявити дешифратор. Цілком можливо, що файл був пошкоджений таким чином, що його помилки не призводять до неспецифічного бітового потоку, але все ще не дають зрозуміти, що в файлі, наприклад, мають місце візуальні артефакти. Наразі достовірно невідомо, чи існує спосіб автоматичного виявлення таких помилок. Найбільша проблема у отже, що задля виявлення помилок потрібно знати, як виглядає файл, і що він собою представляє без ушкоджень. Це потрібно задля порівняння і можливого усунення цих ушкоджень video-файлів.

Наявність неповних чи обрізаних video-файлів є наступною проблемою при перевірці цільності video-файлів. Можливість виявлення неповних/усічених файлів сильно залежить з індивідуального формату файлу. З іншого боку, формати, такі як MP4, мають глобальну структуру даних із точними положеннями усіх фреймів/пакетів, отже, достатньо легко виявити, коли їх не вистачає [4].

Застосунок HandBrake є інструментом задля конвертування video в майже будь-який формат, використовуючи великий набір найновіших існуючих кодеків (рис. 1.1). Це безкоштовний додаток для платформ (Windows, MacOS, Linux).

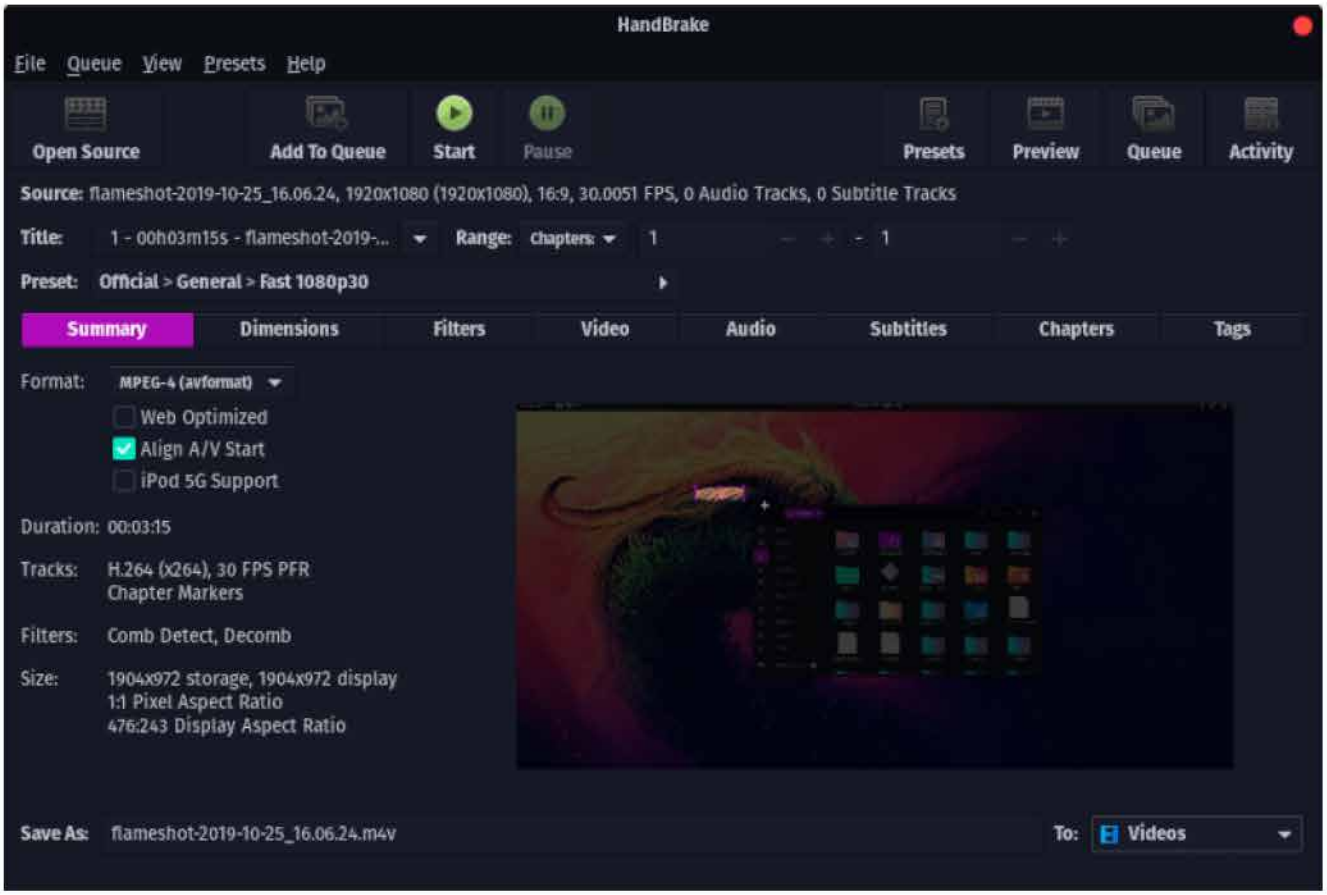


Рисунок 1.1. Інтерфейсу застосунку HandBrake

Застосунок HBBatchBeast є безкоштовним додатком задля HandBrake і FFmpeg/FFprobe задля Windows, macOS і Linux із акцентом на багаторазове перетворення пакетного екземпляра Handbrake (включаючи рекурсивне Скан-пошук папок і їх перегляд). Структура папки призначення зберігається такою ж, як та структура вихідної папки.

Медіа-файли у папках разом з цим конвертуються. Задля кожної папки можливо відстежувати декілька папок та встановлювати різні пресети перетворення. Існує разом з цим функція перевіряння стану здоров'я, яка спроможне сканувати пошкоджені video файли поза поміччю функції "*--scan*" *Handbrakes*, хоча це не завжди є точним інструментом.

Вказаний застосунок створений задля перегляду папок та файлів і рекурсивного перекодування папок. В програмі є вікно, куди можливо ввести власноруч заданий код, проте, коли просто ввести "*--scan*", програма не буде робити перекодування, проте просто оцінить, пошкоджений файл чи ні. Коли він пошкоджений, буде видано помилку. Нижче наведені ілюстрації, що показують інтерфейс даного застосунку задля налаштувань і результатів Скан-пошук файлу.

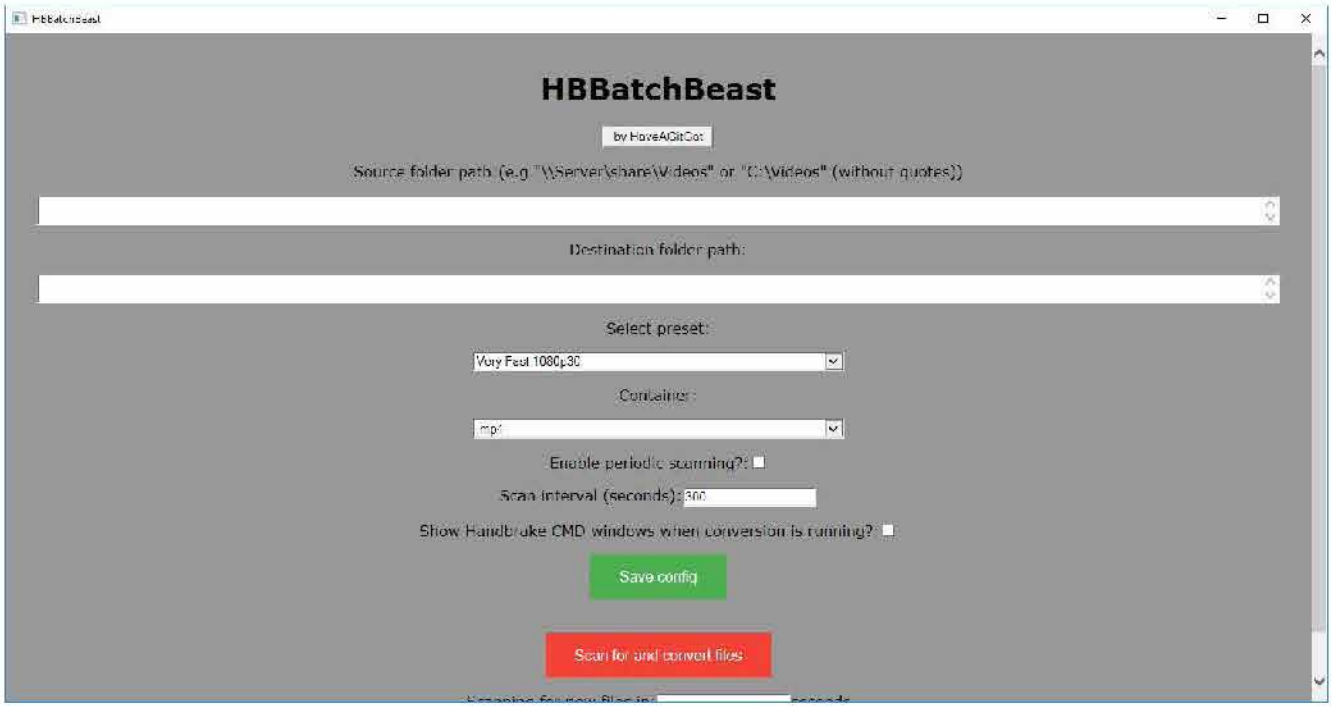


Рисунок 1.2. Інтерфейс налаштування застосунку HBBatchBeast задля Скан-пошук video-файлів

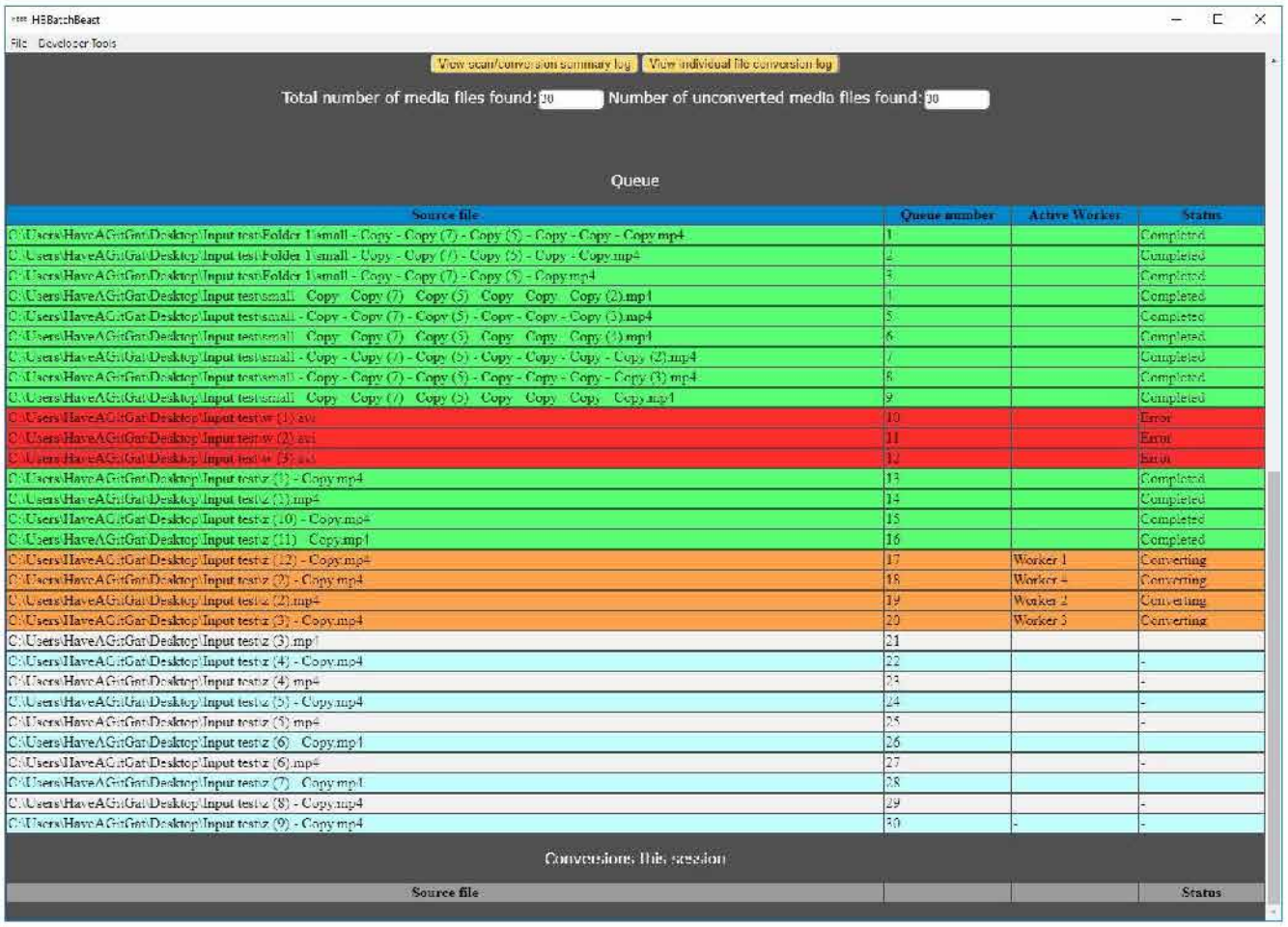


Рисунок 1.3. Перегляд результатів Скан-пошук в застосунку HBBatchBeast

Вказаний застосунок дуже швидко сканує бібліотеку video-файлів – Скан-пошук більше ніж тисячі video-файлів відбувається поза декілька хвилин та виявляє купу пошкоджених файлів. Варто зазначити, що це Скан-пошук разом з цим не є стовідсотково точним. Разом з цим недоліком застосунку є те, що він вказує на помилки, коли назва файлу містить спеціальні символи, такі як "%".

1.1.6 Застосування методу щільової фотометрії задля перевіряння цільності video-файлів

В залежності з геометрії побудови фотознімка можливо виділити три види фотозйомки: кадрова, скануюча і щілинна зйомка.

При використанні кадрової фотометрії усі точки фрейму фіксуються у один момент часу (при спрацюванні затвора). Такий знімок має єдиний центр проекції і строгу геометрію побудови image. Схему побудови кадрової фотометрії показано на рис. 1.4.

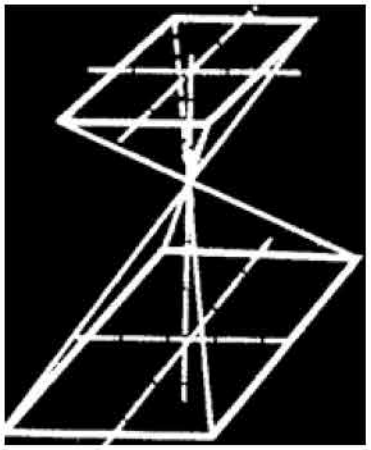


Рисунок 1.4. Загальна схема побудови кадрової фотометрії

Недоліком такого методу є недостатня оперативність, адже потрібно чекати, поки буде знято та зафіксовано на носієві інформації все image. Задля реалізації панорамних знімків із різних зон, знятих окремо, ці окремі кадри потрібно точно поєднувати, що потребує досить багато часу [9].

В скануючих камерах побудова image як по рядку, так та по фрейму відбувається попіксельно, відповідно кожна точка фіксується у окремий момент часу і має свій центр проекції. Схему побудови скануючої фотометрії наведено на рис. 1.5.

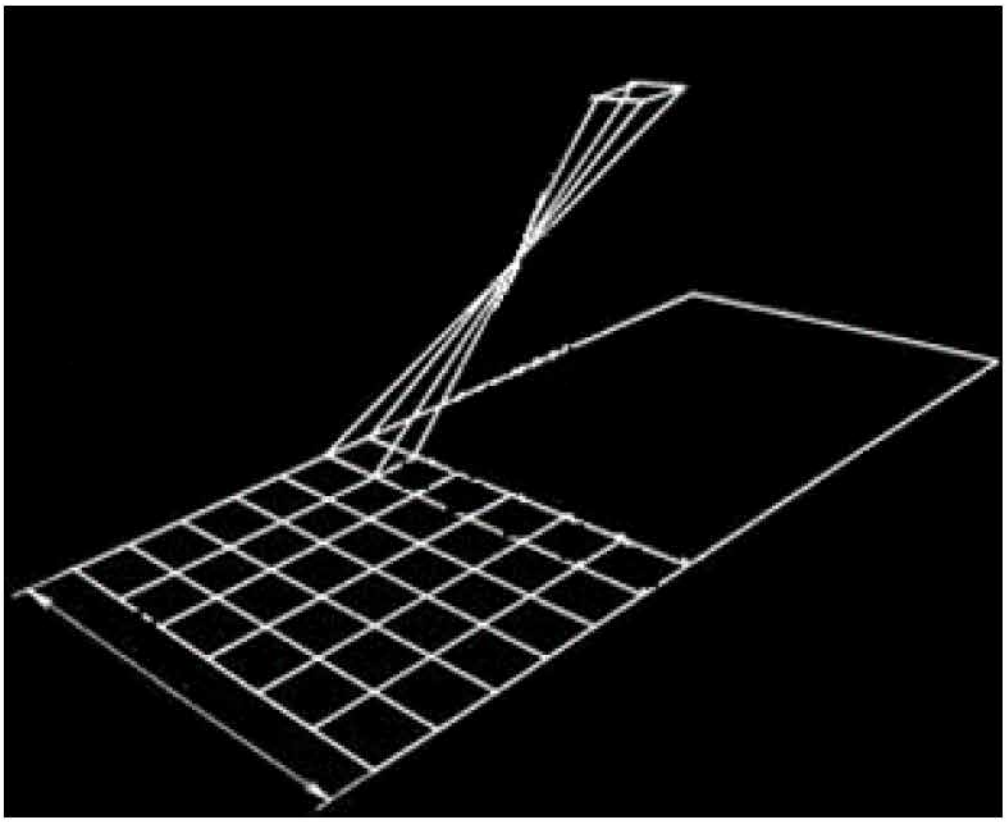


Рисунок 1.5. Загальна схема побудови скануючої фотометрії

Треба зазначити, що поза своїми геометричними властивостями скануючий знімок, що складається із окремих елементів, поступається кадровому. Однак, скануюча зйомка, на відміну з кадрової, має великі можливості завдяки використанню вузьких знімальних зон задля отримання image у широких спектральних діапазонах. Крім того, вона забезпечує швидку передачу інформації і можливість подання знімка у цифровому вигляді, що дозволяє використовувати задля його тематичної опрацювання комп'ютерні технології [10].

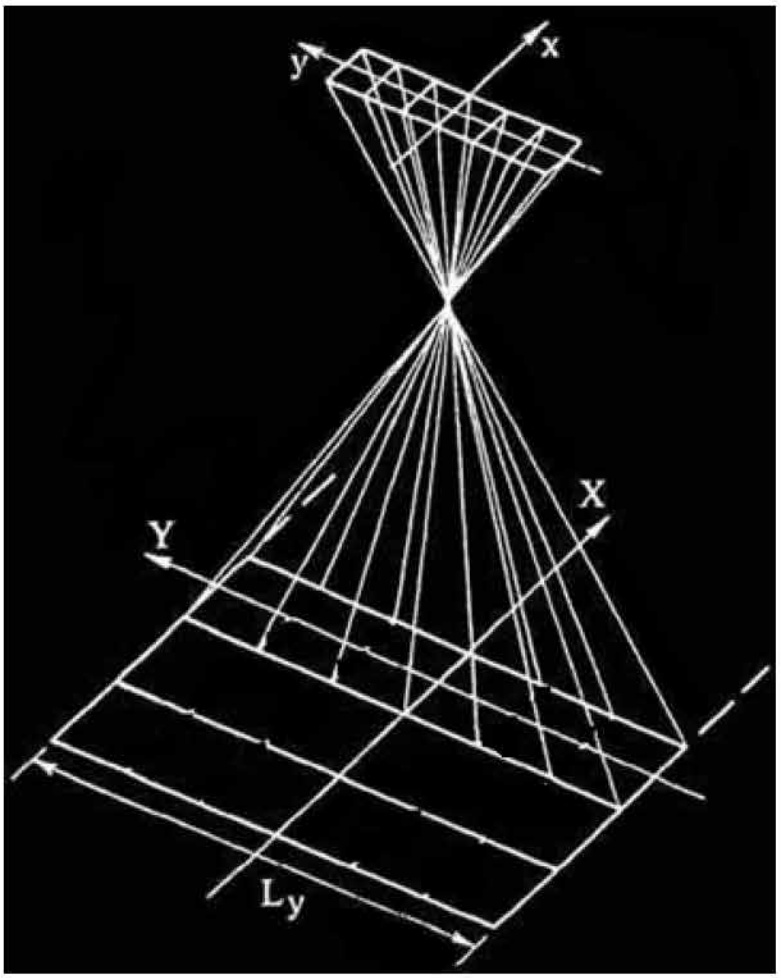


Рисунок 1.6. Загальна схема отримання щільової фотографії

При щілинній зйомці image отримують послідовно рядок поза рядком (рис. 1.6). В площині image встановлюється непрозорий екран із вузькою щіллю, через яку image реєструється на фотоплівці. Роль щілі спроможне разом з цим відігравати лінійка фотоприймачів, що перетворює у електричний сигнал один рядок у просторі предметів. Скан-пошук простору предметів проекцією експозиційної щілі (чи проекцією лінійки фотоприймачів) здійснюється переміщенням уздовж координати X знімальної камери.

Треба зазначити, що щілинна камера робить фотографії шириною у піксель, проте потім склеює вертикальні смужки в ціле image. У результаті можливо бачити, що відбувалося в фокусі щілі протягом певного часу. Разом з цим реалізації image спроможне здійснюватися поза рахунок переміщення об'єкту фотометрії відносно фотокамери, що відповідає створенню ширококутних панорам. При фотографуванні таким способом рухомих об'єктів кінцевим результатом буде image, де все нерухоме – розмите, проте рухоме відображається в вигляді, наближеному до звичного, проте містить аномалії. Техніка реалізації щілинних знімків потребує спеціальних вимог, котрі важко реалізувати. Знімки, що відповідають методу щільової фотометрії, можливо отримати поза поміччю спеціальних щілинних камер, котрі не виробляються серійно [11].

Окрім технічних засобів щільової фотометрії існують програмні додатки, такі як „Slit-Scan Movie Maker" (рис. 1.7) задля MAC OS, програма „After Effects" з Adobe і інші, що призводять до ефектів, схожих на отримувані способом щільової фотометрії.

Рисунок 1.7. Інтерфейс застосунку Slit-Scan Movie Maker (головне вікно)

Програмний застосунок After Effects з Adobe (рис. 1.8) дає змогу розробляти вражаючі анімовані ефекти поза поміччю плагіну SlitScan. Ця програма використовує графічну обробку image, використовуючи основні принципи щільової фотометрії.

Рисунок 1.8. Інтерфейс застосунку After Effects (головне вікно)

Обидві зазначені вище програми є платними та входять до складу цілого софтового комплексу, що не є зручним при використанні і призводить до застосування значних системних ресурсів. Разом з цим існують розважальні додатки задля ОС Android і iOS, котрі використовують камеру пристрою, щоб створювати image, що відповідають методу щільової фотометрії. Їх головний недолік –недостатня функціональність і невелика кількість налаштувань. Схожі ефекти можливо отримати поза поміччю застосунку „VideoCube“ з Microsoft Research, який, нажаль, підтримує лише формат нестиснутих video-файлів та застосовує інші поняття і параметри налаштування, відмінні з методу щільової фотометрії. На рис. 1.9. наведено інтерфейс застосунку „VideoCube“ [12].

На базі принципів отримання щілинного знімка можливо визначити методи його цифрового моделювання. Пропонується замість механічного способу реалізації щілинних фотографії використовувати програмну обробку знятого із

великою кількістю фреймів поза секунду video. Нестача фреймів поза секунду призведе до часткової пікселізації фотографії, надлишок – тільки до зайвої ширини одержуваної фотографії, однак це легко усунути у будь-якому графічному редакторі.

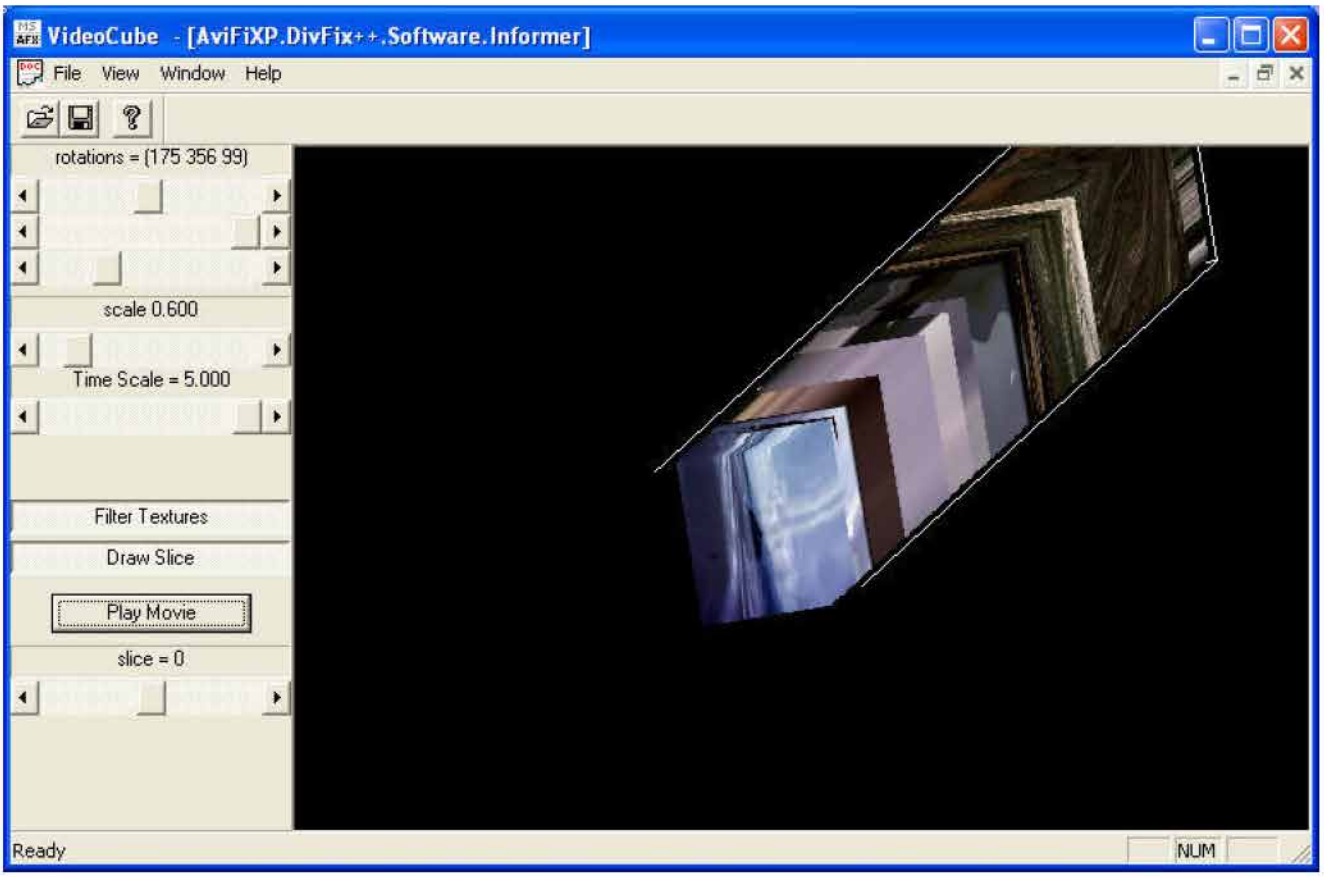


Рисунок 1.9. Інтерфейс застосунку VideoCube (головне вікно)

На рис. 1.10 схематично представлені принципи опрацювання video задля отримання цифрових фотографій, аналогічні методу щільової фотометрії.

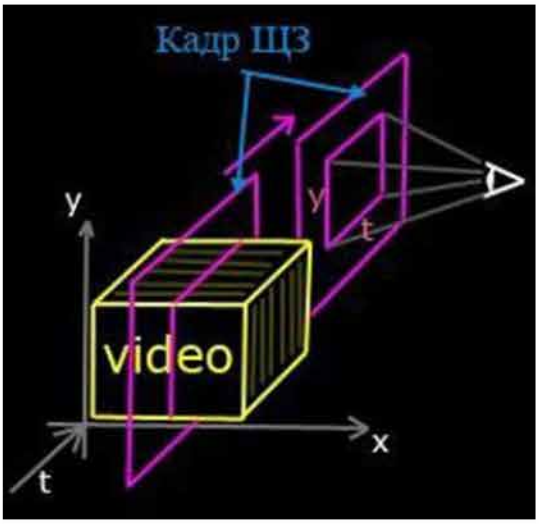


Рисунок 1.10. Загальна схема отримання із відеоряду щільової фотометрії

1.2 Склад програмно-технічних засобів розроблювання ПЗ

Згідно технічному завданню на виконання дипломної роботи, програмне забезпечення задля Скан-пошук ушкоджень video-файлів має бути складене мовою C++ чи мовою Delphi в системі програмування Embarcadero RAD Studio під ОС сімейства Windows. В якості мови програмування обрано саме мову Delphi, яка найкраще підходить задля рішення подібних прикладних задач.

1.2.1 Технічні засоби розроблювання

При виборі технічних засобів розроблювання софтового забезпечення задля Скан-пошук ушкоджень video-файлів найбільшу роль грає чинник швидкодії роботи ПК, оскільки саме з нього залежить час розроблювання ПЗ, проте відповідно витрати на розробку та його собівартість.

Швидкість функціонування додатків на ПК у основному визначається наступними параметрами:

- Об'ємом оперативної пам'яті (ОП);
- Швидкодією процесора;
- Об'ємом відеопам'яті (ВП).

Виходячи із вимог, що пред'являються до використовуваних програмних засобів розроблювання (Embarcadero RAD Studio), мінімальні значення вищеперелічених параметрів складають: ОП – 4 Гб, процесор – із частотою 2,5 ГГц, ВП – 512 Мб, ОС – Windows 10.

1.2.2 Середовище розроблювання ПЗ

Embarcadero RAD Studio – середовище швидкої розроблювання додатків (RAD) фірми Embarcadero Technologies, яке працює під ОС Windows. Поточна версія Embarcadero RAD Studio 10.3 Rio об'єднує Delphi та C++ Builder в єдине інтегроване середовище розроблювання:

- RAD Studio Professional – підходить задля індивідуальних розробників та невеликих груп, що створюють настільні та мобільні додатки;
- RAD Studio Enterprise – підходить задля груп розробників, що створюють

клієнт-серверні чи багаторівневі додатки;

- RAD Studio Architect – підходить задля корпоративних клієнтів, що створюють бази даних чи веб-додатки.

Задля виконання розроблювання софтового продукту віртуального частотоміру найбільш доцільним є застосування саме версії RAD Studio Professional.

Embarcadero RAD Studio представляє собою набір засобів розроблювання додатків, який дозволяє створювати додатки із графічним призначеним задля користувача інтерфейсом задля Windows, Mac OS X, .NET, PHP і веб-рішень. До складу входять:

- Embarcadero Delphi – дає можливість створювати повнофункціональні додатки задля Windows та Mac OS X;
- Embarcadero C ++ Builder – це середовище C++, яке повністю відповідає концепції швидкої розроблювання додатків, об'єднує засоби ANSI C++ та багатофункціональну розширювану інфраструктуру візуальних компонентів;
- Embarcadero Prism – являє собою крос-платформне рішення задля розроблювання та Delphi-подібну мову програмування задля швидкої розроблювання додатків .NET, Mono, ASP.NET та додатків, створюваних в рамках парадигми Data Driven Design (орієнтованих на роботу із визначеними наборами даних) задля Windows, Linux та Mac OS X;
- Embarcadero RadPHP – спрощує реалізації веб-додатків на PHP завдяки наявності візуальних засобів проектування інтерфейсів, редактора, відладчика, засобів підключення до баз даних та інтегрованої бібліотеки повторно використовуваних класів компонентів. Компоненти RadPHP дозволяють робити веб-інтерфейси в стилі iOS та Android;
- ER / Studio Developer Edition – допомагає проектувальникам баз даних аналізувати, документувати та повторно використовувати дані та надає засоби зворотного проектування, аналізу і оптимізації баз даних;
- InterBase SMP Developer Edition – надає розробникам крос-платформну

базу даних задля реалізації та тестування задля вбудованих додатків та додатків задля малих та середніх поза розмірами підприємств.

RAD Studio включає у себе широкий набір додаткових програм:

- InstallAware Express – надає засоби, що дозволяють користувачам, котрі не мають навичок програмування та розроблювання сценаріїв, створювати складні настановні пакети;
- Rave Reports компанії Nevrona – набір рішень задля реалізації звітів;
- FireMonkey;
- TeeChart Standard компанії Steema – компоненти задля реалізації діаграм;
- VCL задля веб-рішень (IntraWeb) компанії Atozed Software – платформа веб-додатків RAD;
- FinalBuilder Embarcadero Edition – служить задля автоматизації процесу складання;
- CodeSite Express – засоби ведення журналу задля збірки програм;
- AQTime Standard компанії SmartBear – реалізації профілів продуктивності;
- Beyond Compare Text Compare – порівняння файлів вихідного коду;
- RemObjects Internet Tools та Oxfuscator – додаткова функціональність задля веб-розроблювання та обфускації коду в Delphi Prism.

Основні нові можливості Rad Studio 10 Seattle є такими.

Розробники на Delphi та C++ Builder спроможні швидко відновити свої VCL- та FMX-додатки та у повній мірі скористатися засобами ОС Windows 10. Підтримуються компоненти Windows 10 та «рідні» API та компоненти WinRT / UWP, елементи інтерфейсу Windows 10 VCL (рис. 1.11). Разом з цим оновлена підтримка Windows 10 FMX. В C++ Builder 10 із'явився перший в світі компілятор C++ на основі CLANG задля Windows та мобільних платформ із розширеннями RAD PME, що забезпечують швидку розробку задля Windows та інших платформ. Він підтримує тісну інтеграцію із VCL задля Windows та кросплатформними структурами FMX, мова C++11 та керування пам'яттю на основі ARC (автоматичного підрахунку посилань) задля C++, проте має зворотну сумісність.

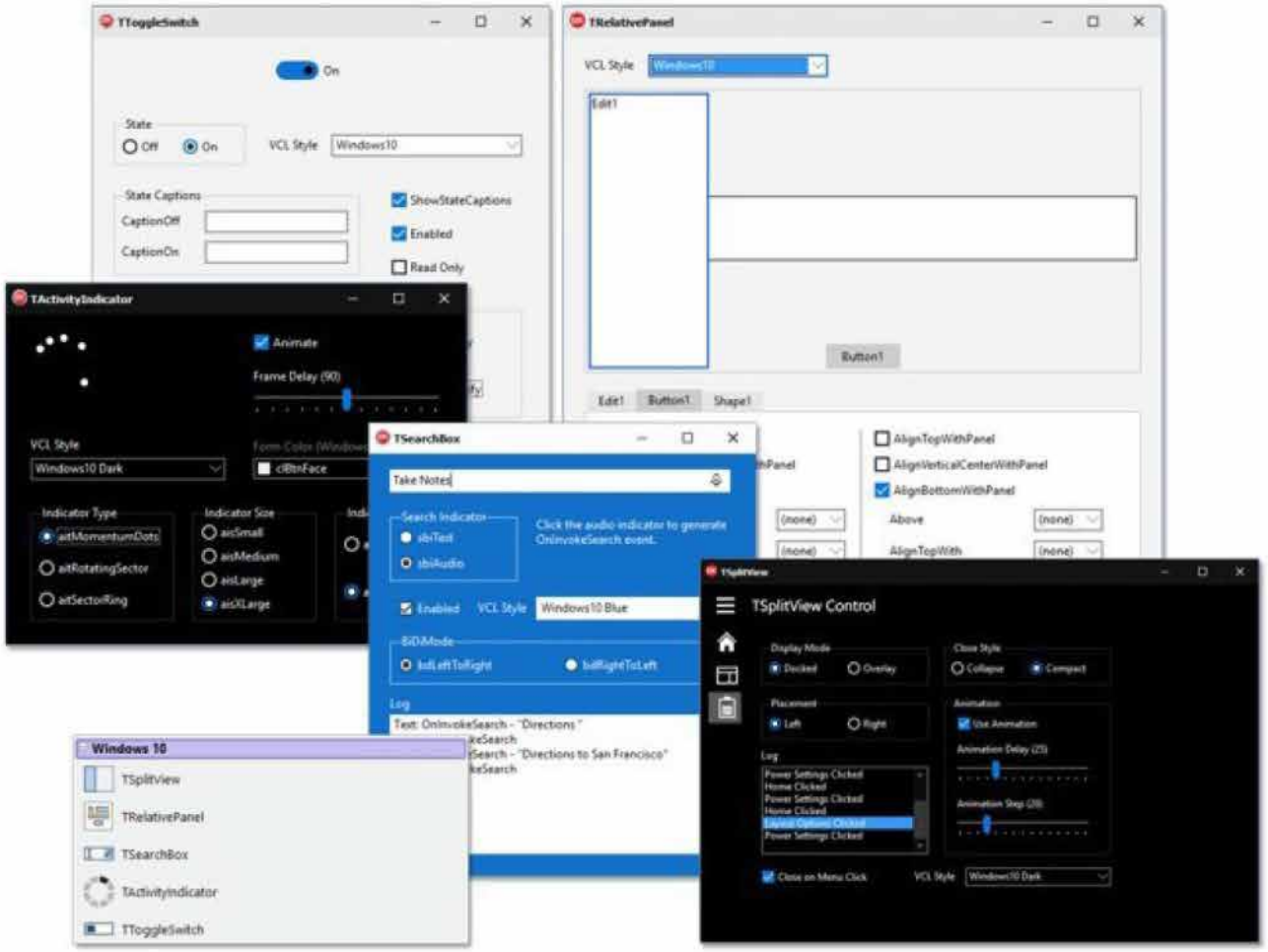


Рисунок 1.11. Елементи інтерфейсу Windows 10 VCL в RAD Studio

Новий компілятор C++ Builder (рис. 1.12) робить версію RAD Studio 10 обов'язковим оновленням задля розробників на C++ та нових розробників, котрі переходять на C++ із інших мов та наборів інструментів – Java, Objective-C, C #, Xcode, Xamarin та Visual Studio.

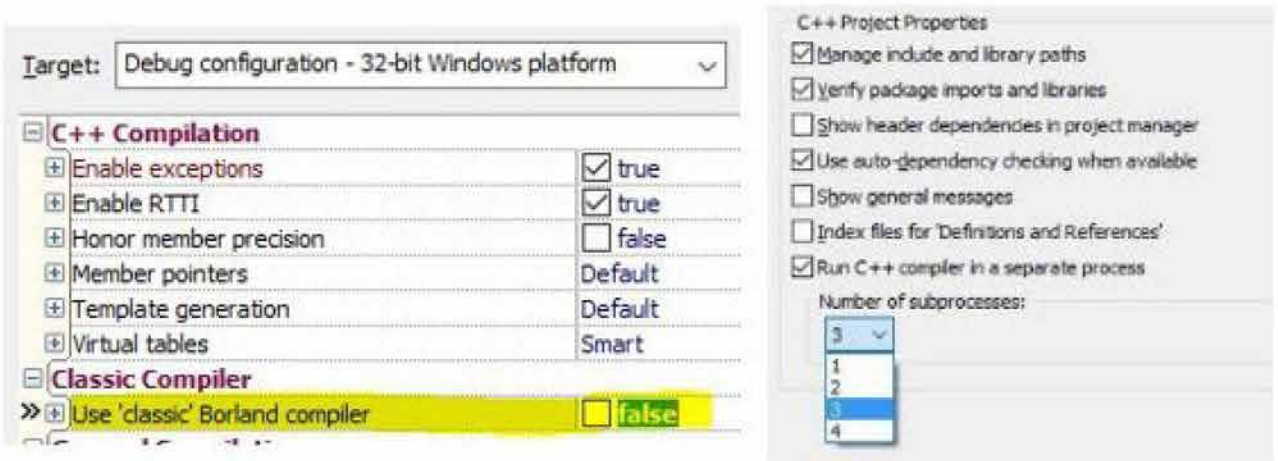


Рисунок 1.12. Налаштування компілятору та проекту C++Builder в RAD Studio

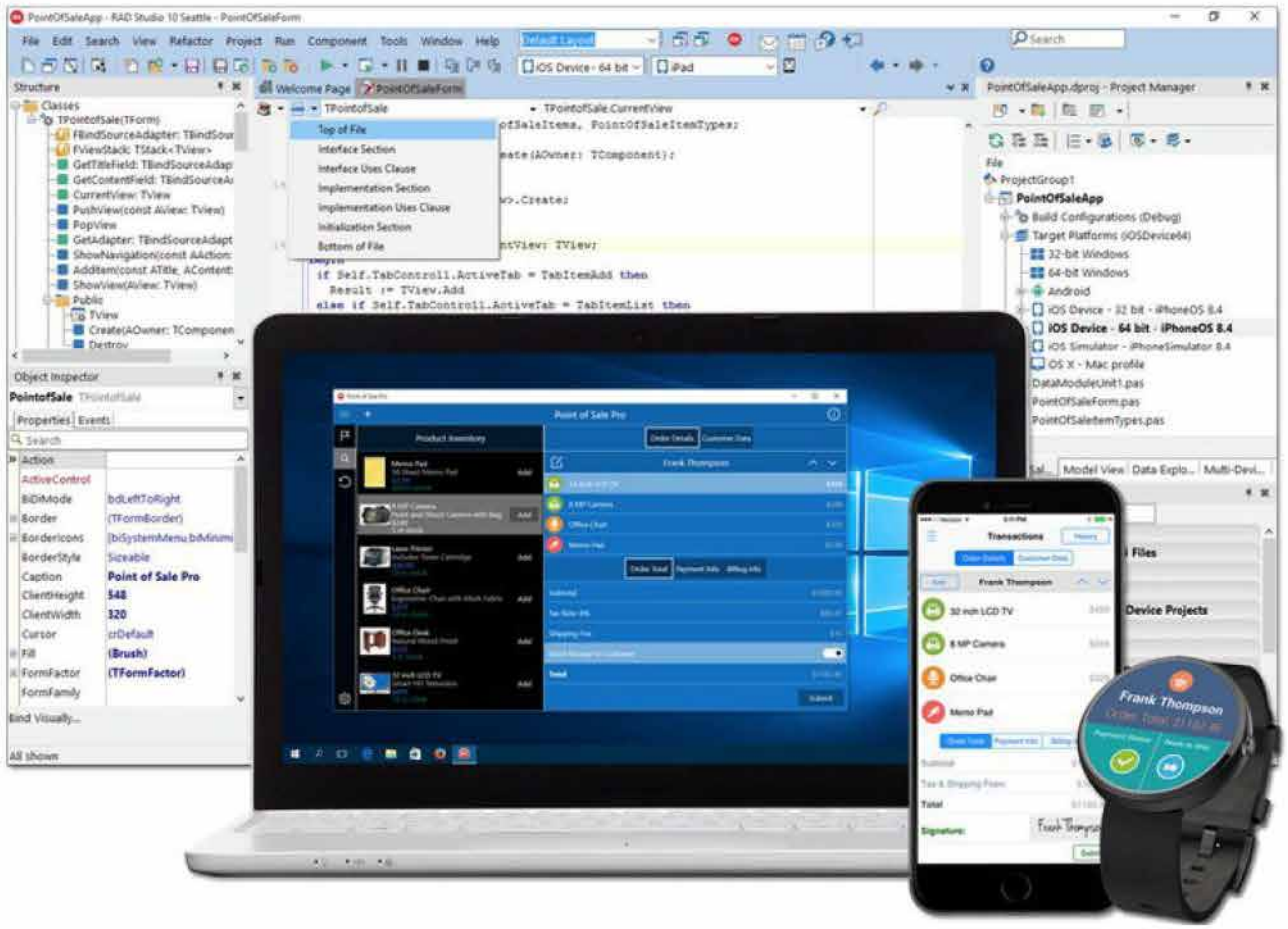


Рисунок 1.13. Інтеграція засобів розроблювання RAD Studio різних платформ

ІЗ непомітних задля користувача змін можливо відзначити перероблену в даній версії архітектуру системи керування продуктами і збірками, яка дозволила фактично подвоїти підтримуваний розмір проектів та підвищити стабільність та продуктивність при роботі із великими проектами, особливо задля декількох платформ (рис. 1.13). Оновлення до цієї версії стане у нагоді розробникам, котрі стикалися із обмеженнями ресурсів та іншими проблемами у крупних проектах.

Процедури розроблювання, тестування і складання документації були серйозно доопрацьовані (рис. 1.14). Тепер вони краще підходять задля реалізації додатків на декількох мовах задля платформ, підтримуваних новою версією RAD Studio. Інші нові можливості Rad Studio (10.2 Tokyo і 10.3 Rio):

- підтримка паралельної компіляції C++;
- підтримка C++ 17 в C ++ Builder задля 64-бітової версії Windows;
- підтримується розробка додатків задля ОС iOS, MacOS, Android, Linux (рис.1.15);

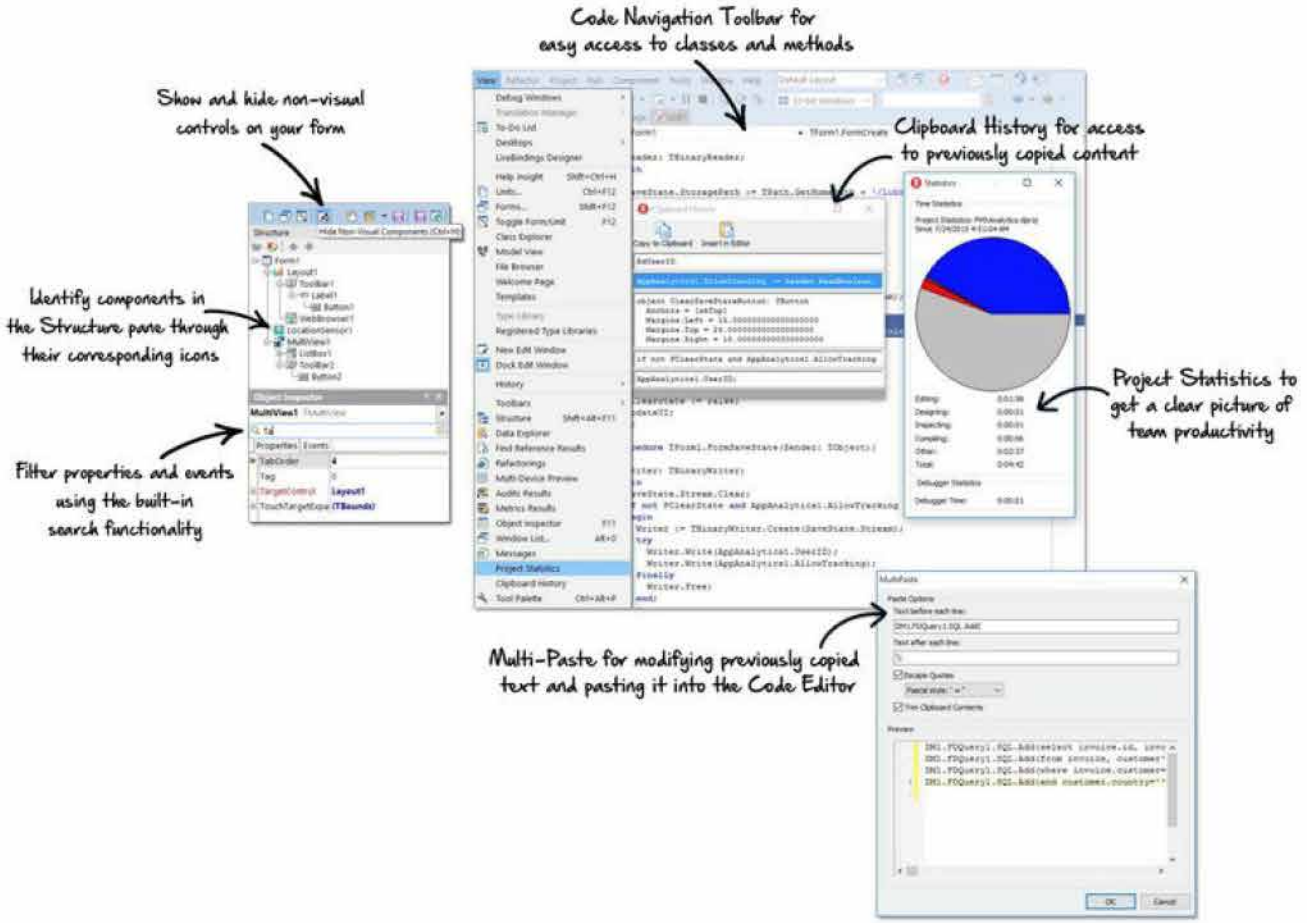


Рисунок 1.14. Ілюстрація можливостей системи документації в RAD Studio

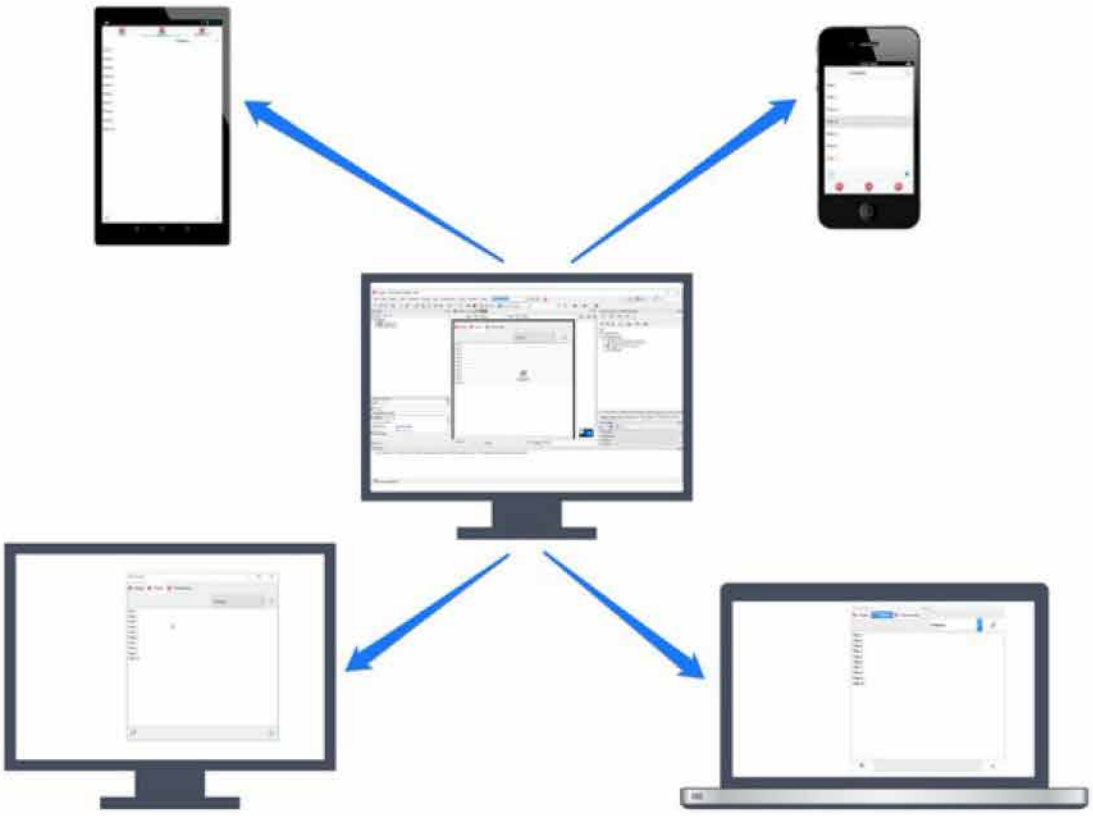


Рисунок 1.15. Ілюстрація можливостей в RAD Studio

- підтримка модульного тестування DUnitX задля Android та iOS;
- підтримка DirectX 12;
- підтримка виклику API WinRT;
- підтримка FireDAC задля бази даних NoSQL MongoDB;
- нова поведінка MultiView;
- включена підтримка СУБД MariaDB.

1.3 Реалізація софтового застосунку задля Скан-пошук ушкоджень video-файлів

Задля перевіряння video-файлів на псування зазвичай використовують дешифратори, проте такий процес є складним і потребує теоретичні знання із дешифрування і знання деяких фреймворків.

Спеціальних програмних засобів задля перевіряння video-файлів на цілісність в відкритому доступі не існує. Задля того, щоб дізнатися, чи є в файлі пошкоджені кадри, його можливо передивитися в будь-якому video-плеєрі. Проте це складно зробити поза короткий час, коли video-файл є повноцінним фільмом великої тривалості.

Саме отже пропонується використовувати щілинну зйомку задля опрацювання video в якості способу перевіряння цільності video-файлів і відшукування пошкоджень на video-фреймах.

На рис. 1.16 наведено структурну схему роботи створюваного застосунку задля відшукування пошкоджень на video-фреймах.

При виконанні відшукування пошкоджень на video-фреймах в створюваному додатку виконується моделювання щільової фотометрії із вертикальною щіллю. При цьому один стовпець точок, що відповідає положенню віртуальної щілі, витягується із будь-якого оригінального фрейму video та розміщуються у порядку збільшення індексу (горизонтальної координати) стовпця послідовно один поза одним.

На рис. 1.17 наведено схему, що пояснює методики та ефекти, отримані поза поміччю методу щільової фотометрії.

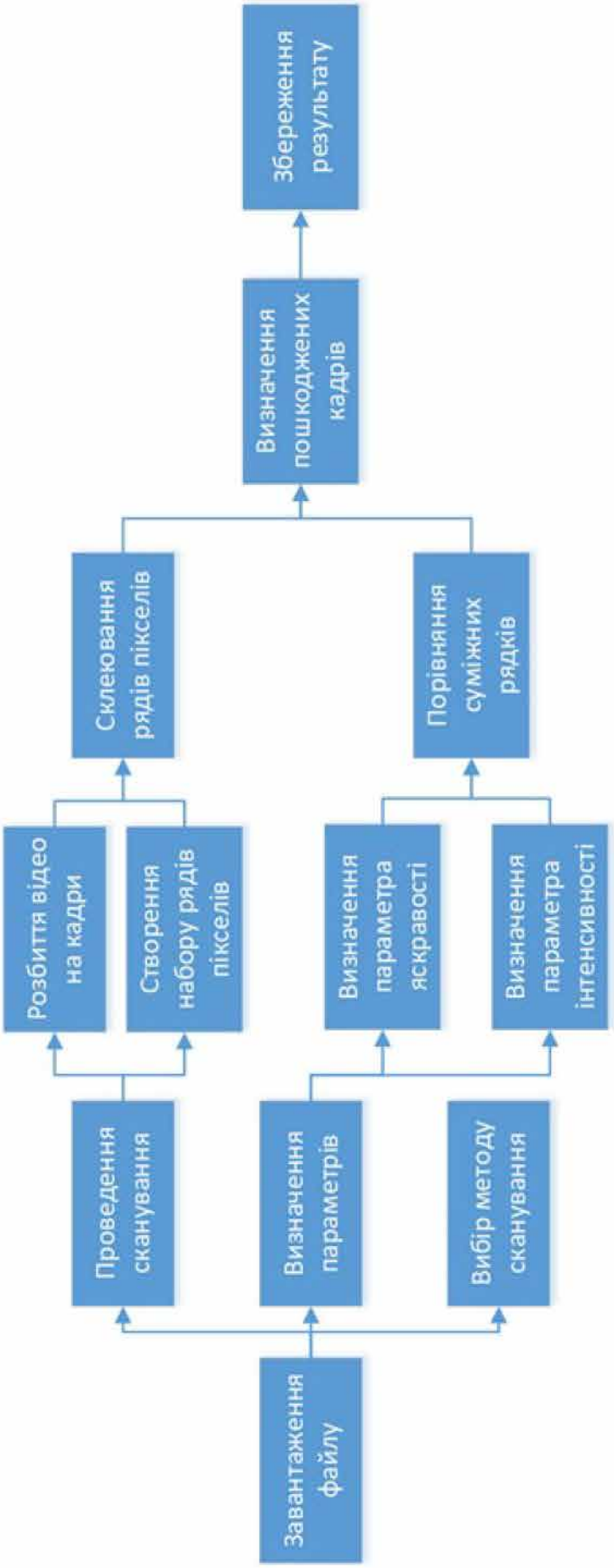


Рисунок 1.16. Структурна схема роботи застосунку задля відшукування пошкоджень на video-фреймах

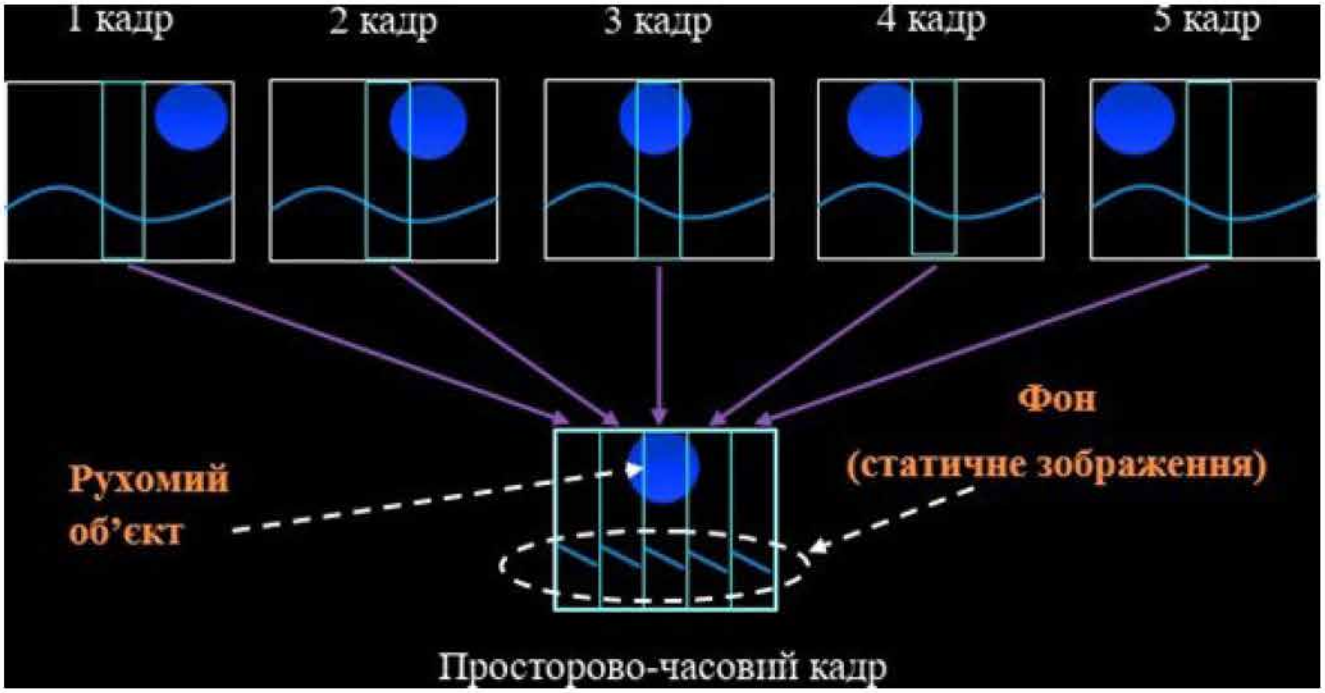


Рисунок 1.17. Схема опрацювання video-фреймів, створених способом щільової фотометрії із вертикальною статичною щіллю

В даному проекті реалізацію софтового забезпечення задля відшукування пошкоджень на video-фреймах виконано мовою Delphi в інтегрованому середовищі розроблювання Embarcadero RAD Studio.

Процес реалізації просторово-часового фрейму із нерухомою щіллю програмно реалізовано так:

```
for i:=Frame_start to Frame_last do
begin
InputFrame.Position:=i;
for j:=1 to InputImage.Height do
OutpuImage.Canvas.Pixels[i-Frame_start,j]:=
InputImage.Canvas.Pixels[Slit_Position,j];
end;
```

В наведеному коді Frame_start – початковий кадр video; Slit_Position – позиція щілі; Frame_last – кінцевий кадр video.

Image, отримані поза даним алгоритмом, показані на рис.1.18. Задля опрацювання використовувалось video, яке було зняте автором даної роботи поза поміччю смартфону Samsung Galaxy S10.

Рисунок 1.18. Image, створені поза поміччю щільової фотометрії

Кожен кадр оброблюваного video представляє собою різні моменти часу, отже отримуване image – це image подій, що відбулися на протязі певного часу. Довжина image дорівнює кількості оброблених фреймів. Статичні (нерухомі) предмети відображаються смужками (фон), проте рухомі об'єкти із'являються в вигляді, близькому до оригінального, але, можливо, із деякими відрізненнями. При створенні фотографій способом щільової фотометрії із вертикальною статичною щіллю, із будь-якого фрейму оброблюваного video витягується один вертикальний стовпець точок, що відповідає віртуальній щілині, але задля будь-якого фрейму положення щілі має різні значення та залежить з швидкості руха щілі. Наприклад, коли швидкість руха щілі дорівнює 1, стовпчик задля наступного фрейму береться на одну позицію лівіше, ніж із попереднього фрейму та так задля усіх фреймів (рис. 1.19).

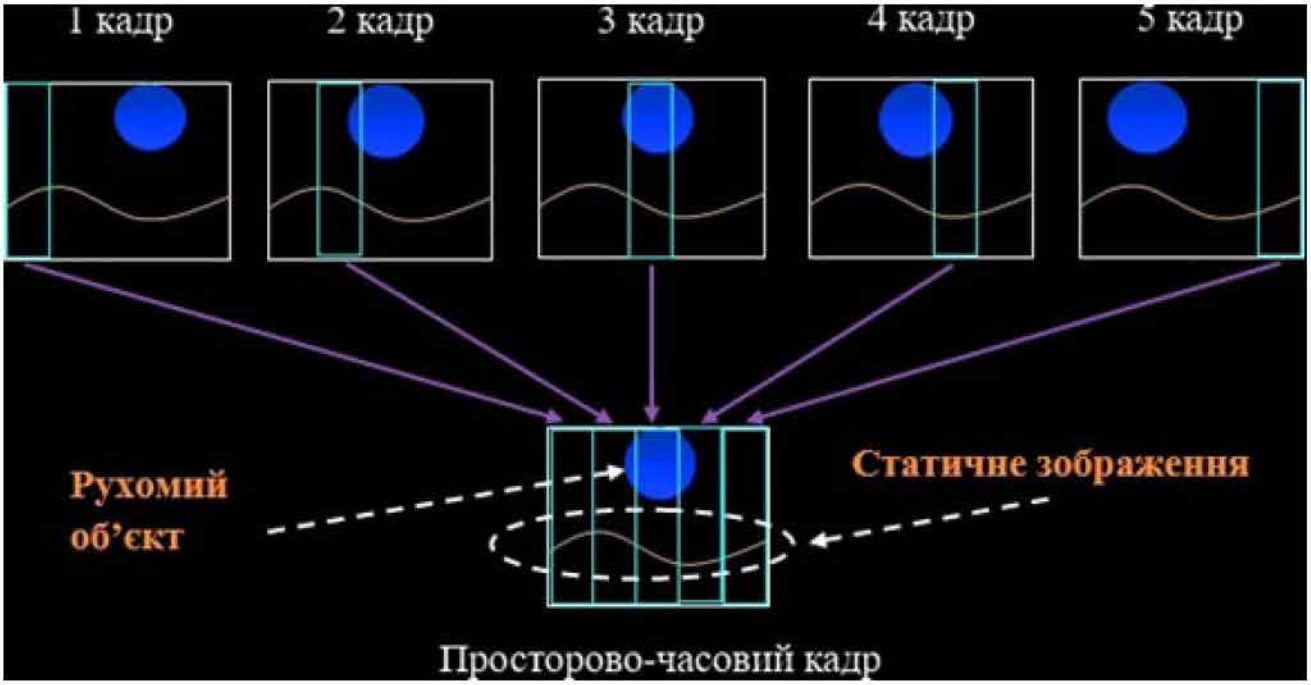


Рисунок 1.19. Схема опрацювання video-фреймів, створених способом щільової фотометрії із вертикальною рухомою щіллю

Процес реалізації просторово-часового фрейму із рухомою щіллю програмно реалізовано так:

```
for i:=Frame_start to Frame_last do
begin
InputFrame.Position:=i;
Slit_Position:=Slit_Position+V;
```

```
for j:=1 to InputImage.Height do
OutputImage.Canvas.Pixels[iFrame_start,j]:=
InputImage.Canvas.Pixels[Slit_Position,j];
end;
```

де Frame_start – початковий кадр video; Slit_Position – позиція щілі; Frame_last – кінцевий кадр video.

Однією із особливостей щільової фотометрії із вертикальною рухомою щіллю є те, що на фотографіях із'являються image нерухомих предметів, котрі у статичній щільової фотометрії відображались смугами. Величина ефекту залежить з швидкості зміни положень щілі у двох сусідніх кадрах – швидкості руха щілі. Коли V=1, де V – швидкість руха щілі (рис. 1.20), довжина статичного предмету відповідає оригінальній, проте коли 0<V<1 (рис 1.21 і рис 1.22), статичні предмети на фотографії витягуються, проте рухомі об'єкти залишаються незмінними.

Рисунок 1.20. Результат щільової фотометрії при швидкості руха щілі, рівної 1

Наведене на рис.1.20 image є результатом фото, отриманого поза поміччю розробленого софтового застосунку, використовуючи метод рухомої вертикальної щілі. Швидкість руха щілі на цьому зображенні дорівнює 1, відповідно кожний наступний стовпець точок береться послідовно, кадр поза кадром, та ширина

отриманого image дорівнює ширині сканованого video.

Рисунок 1.21. Результат щільової фотометрії при швидкості руха щілі, рівної 0.1

Наведене на рис.1.21 image отримане шляхом опрацювання того ж самого video, що та попереднє, способом із рухомою вертикальною щіллю. Швидкість руха була зменшена, відповідно статична частина image подекуди дублювалась. Image вийшло ширшим в порівнянні із попереднім, проте video-фрагмент був однаковий. На рис. 1.22 наведено результат обробку цього ж video-файла, проте швидкість руха щілі становила 0.5, отже image вийшло не таким широким, як на рис.1.21.

Рисунок 1.22. Результат щільової фотометрії при швидкості руха щілі, рівної 0.5

Треба зазначити, що задля моделювання щільової фотометрії із горизонтальною рухомою щіллю виконуються ті ж самі операції, що та при

щілинній зйомці із вертикальною рухомою щіллю, але замість стовпчиків використовуються рядки, що теж представляють собою аналог щілі. Незважаючи на те, що така методика отримання фотографій схожа на попередні, ефект буде носити зовсім інший характер. Поза поміччю вказаного методу можливо отримувати цікаві ефекти при зйомці рухомих об'єктів, що обертаються навколо осі, коли предмет чи його частини рухаються із змінною швидкістю (рис. 1.23).

Рисунок 1.23. Результати щільової фотометрії із горизонтальною рухомою щіллю

Пояснення ефектів, що отримуються при обробці video таким способом, представлене на рис. 1.24, де наведена схема опрацювання video-фреймів, створених способом щільової фотометрії video-фреймів, створених способом щільової фотометрії із горизонтальною рухомою щіллю: статичне оригінальне image не змінюється, рухомі об'єкти змінюють свою форму, але не змінюють уявлення про них. Зазвичай такий ефект будують на основі послідовно знятих фреймів статичного предмету, знятого під різними кутами до площини камери. Предмет чи сцена фотографується через певний крок по куту, проте потім фотографії склеюються у фото-редакторі із використанням фільтру згладжування. Такий процес займає доволі багато часу. Щілинна зйомка при меншій технічній складності спроможне дати той самий ефект, але поза набагато менший час.

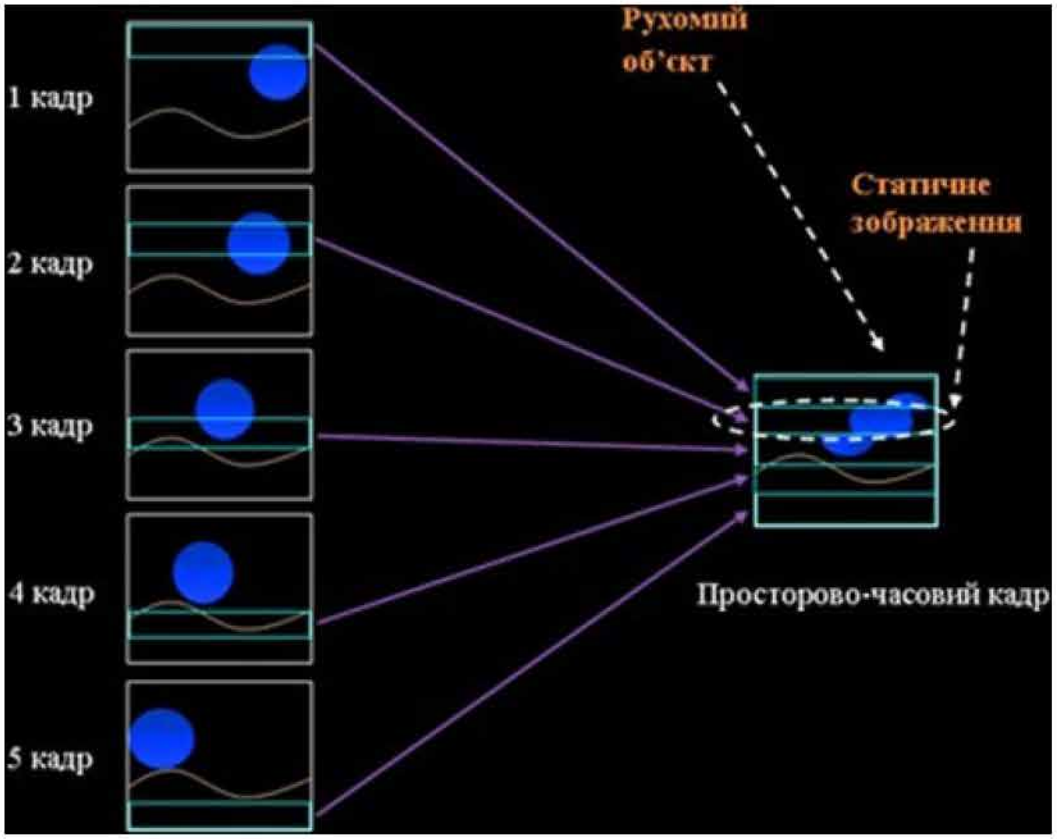


Рисунок 1.24. Схема опрацювання video-фреймів, створених способом щільової фотометрії із горизонтальною рухомою щіллю

1.4 Розробка алгоритму пошуку пошкоджень на video-фреймах

Існуючий підхід до вирішення проблеми видалення пошкоджень (ушкоджень) базується на припущенні, що артефакт присутній лише на одному кадрі. Відповідно, коли деякі пікселі не виявлені на попередньому чи наступному кадрі – цей піксель належить до дефекту. Розроблений метод опрацювання video і його Скан-пошук способом щільової фотометрії заснований на застосуванні просторових, часових чи просторово-часових алгоритмів опрацювання фрейму video-файла. Метод, заснований на припущенні, що артефакт має невелику площину, називається просторовим. Алгоритм використовує деякі операції математичної морфології. Дефектами на кадрах буде вважатися різниця величин, яке перевищує фіксовану деяку величину. При малих значеннях цієї величини, можливе виявлення помилкових спрацювань, викликаних шумом. Перевагою даного методу є простота в обчисленні. Основний недолік – неможливість виявлення артефактів великого розміру на темних і світлих плямах, котрі погано виражені.

Метод, заснований на припущенні, що абсолютна різниця пошкодженого пікселя із пікселями із попереднього і наступного фреймів перевищує задане порогове значення, тоді як різниці інтенсивності пікселя перевищують інтенсивність сусіднього пікселя, називається часовим. Цей алгоритм дозволяє доволі якісно діяти в ситуаціях, коли об'єкт перекривається іншим чи раптово зникає із фрейму, адже різниця між цими кадрами буде великою тільки у одному часовому напрямку. Недоліком цього алгоритму є те, що не враховується рух між кадрами і об'єктів в камері. Метод, заснований на аналізі послідовного рядку/стовбця із n точок, взятих із трьох послідовних фреймів, називається просторово-часовим (рис.1.25).

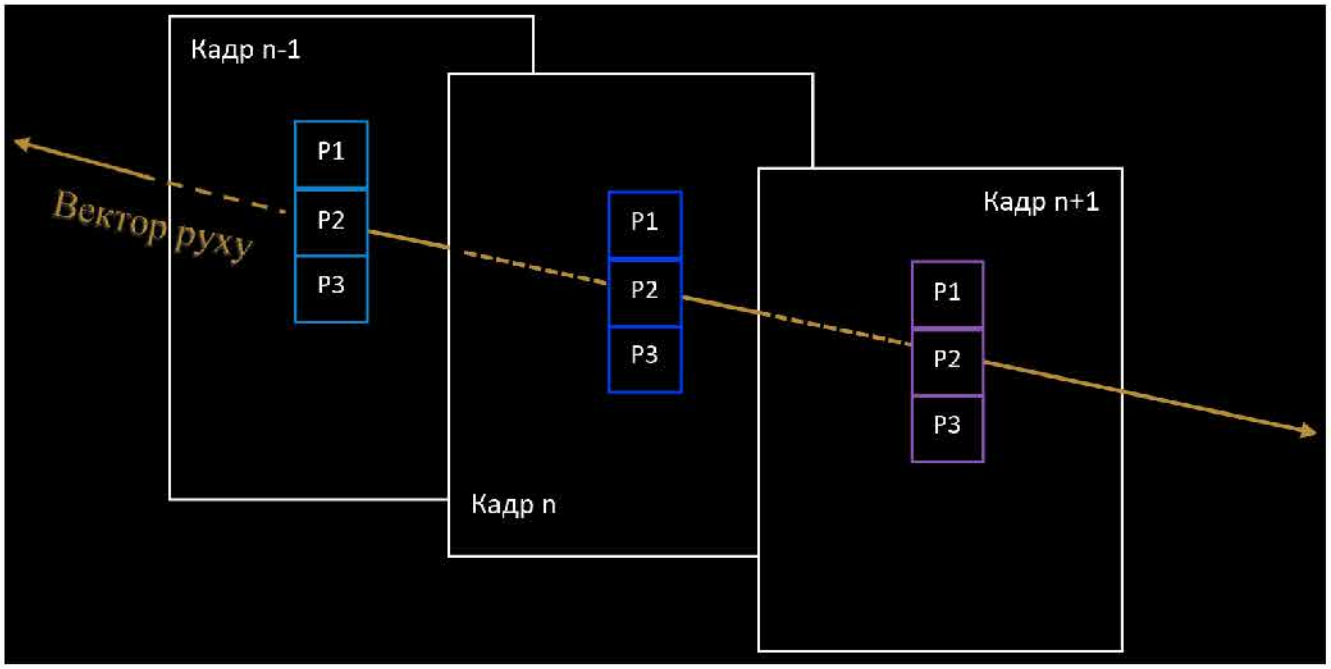


Рисунок 1.25. Загальна схема просторово-часового опрацювання фрейму

Просторово-часовий метод перевіряння полягає у отже, що дефектні пікселі дуже відрізняються по яскравості і інтенсивності в загальному розподілені яскравості image. Задля роботи алгоритму сканується перший кадр video і записується стовпчик точок із заданими початковими координатами відносно осі X чи B. Параметри кольору будь-якого із точок у отриманому ряді точок записується в відповідний масив даних (матрицю). Потім із будь-якого наступного фрейму зчитується і записується і сама інформація. При цьому координати по осі X чи B не змінюються, проте змінюється значення часу, відповідно матриця даних поповнюється.

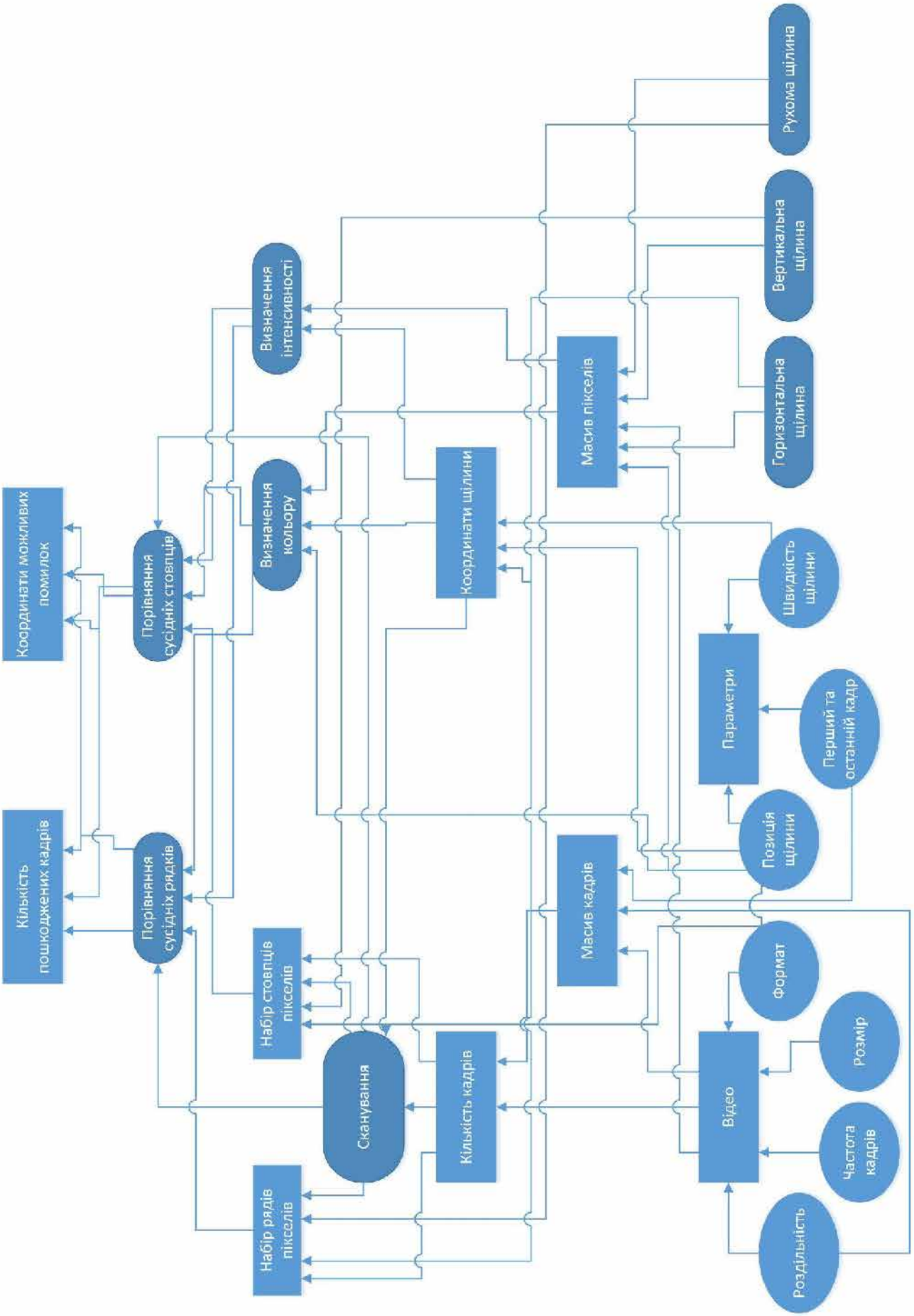


Рисунок 1.26. Блок-схема процедур і параметрів софтового забезпечення задля відшукування пошкоджень на video-фреймах

Потім відбувається порівняння будь-якого стовпця точок із сусідніми. ІЗ допусканням невеликих відхилень програма перевіряє ідентичність отриманих сусідніх стовпців, відповідно порівнюються стовпці *n-1*, *n* і *n+1*, потім *n*, *n+1* і *n+2*, тощо. Коли усі значення відповідають заданим параметрам – на video не виявлено ушкоджень. Коли навпаки – програма відтворює на головному вікні отримане image із'єднаних послідовно рядів точок аби можливо було побачити пошкоджені кадри. В випадку відсутності псування можливим є сценарій, поза яким артефакти не займали увесь кадр, проте лише на деяких невеликих частинах фрейму було помітно візуальне псування точок. Отже після вдалого Скан-пошук video потрібно ще раз впевнитися, що увесь кадр цілий. Саме отже алгоритм Скан-пошук video-файлів було удосконалено. Задля уникання невиявлених артефактів на зображенні пропонується зробити декілька сканувань video із зміною початкової координати. Таких сканувань можливо робити безліч, обмежуючись лише роздільною здатністю сканованого video-файла. Блок-схема процедур і параметрів софтового забезпечення задля відшукування пошкоджень на video-фреймах зображена на рис. 1.26. Блок-схема процесу Скан-пошук video-файла задля відшукування пошкоджень на video-фреймах наведена на рис. 1.27.

1.5 Розробка інтерфейсу застосунку

Програмне забезпечення задля відшукування пошкоджень на video-фреймах буде складатися із головного застосунку (exe-файл) і допоміжних файлів-бібліотек. Застосунок розроблено задля аналізу video-файлів поза поміччю щільової фотометрії. Розробку виконано в візуальному режимі на мові Delphi в середовищі розроблювання Embarcadero RAD Studio 10.3. Інтерфейс головного вікна розробленого софтового засобу дозволяє власноруч отримувати image, створені таким способом (рис. 1.28). В налаштуваннях програми є можливість виставити позицію щілі, швидкість руха щілі, обирати горизонтальну чи вертикальну щілинну зйомку, проте разом з цим статичну чи рухому. Одним із головних параметрів опрацювання video є вибір початкового і останнього фрейму video-файла, котрі будуть оброблятися. Отримані поза поміччю додатку кадри зберігаються в вигляді файлів *.bmp.

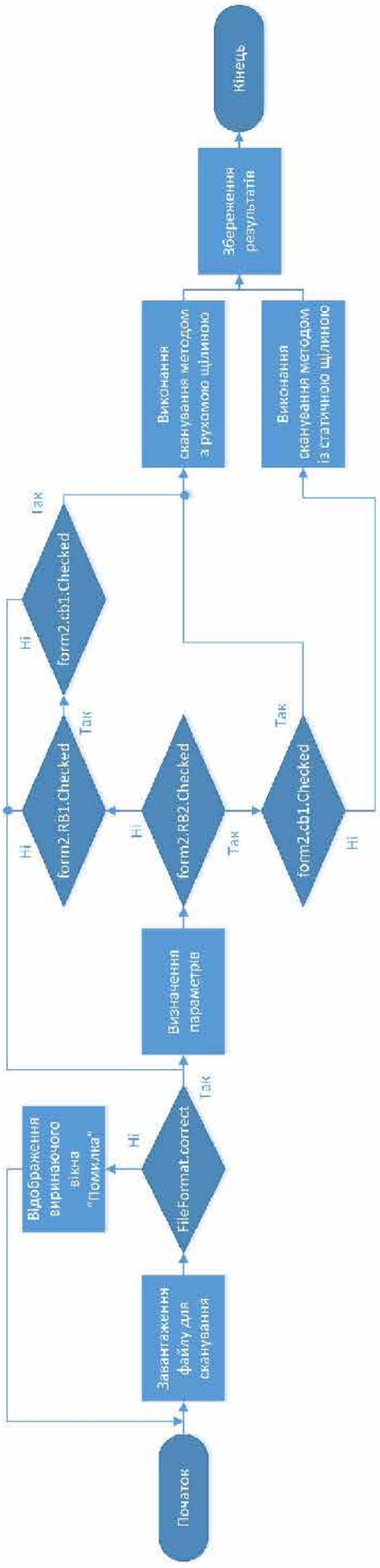


Рисунок 1.27. Блок-схема процесу відшукування пошкоджень

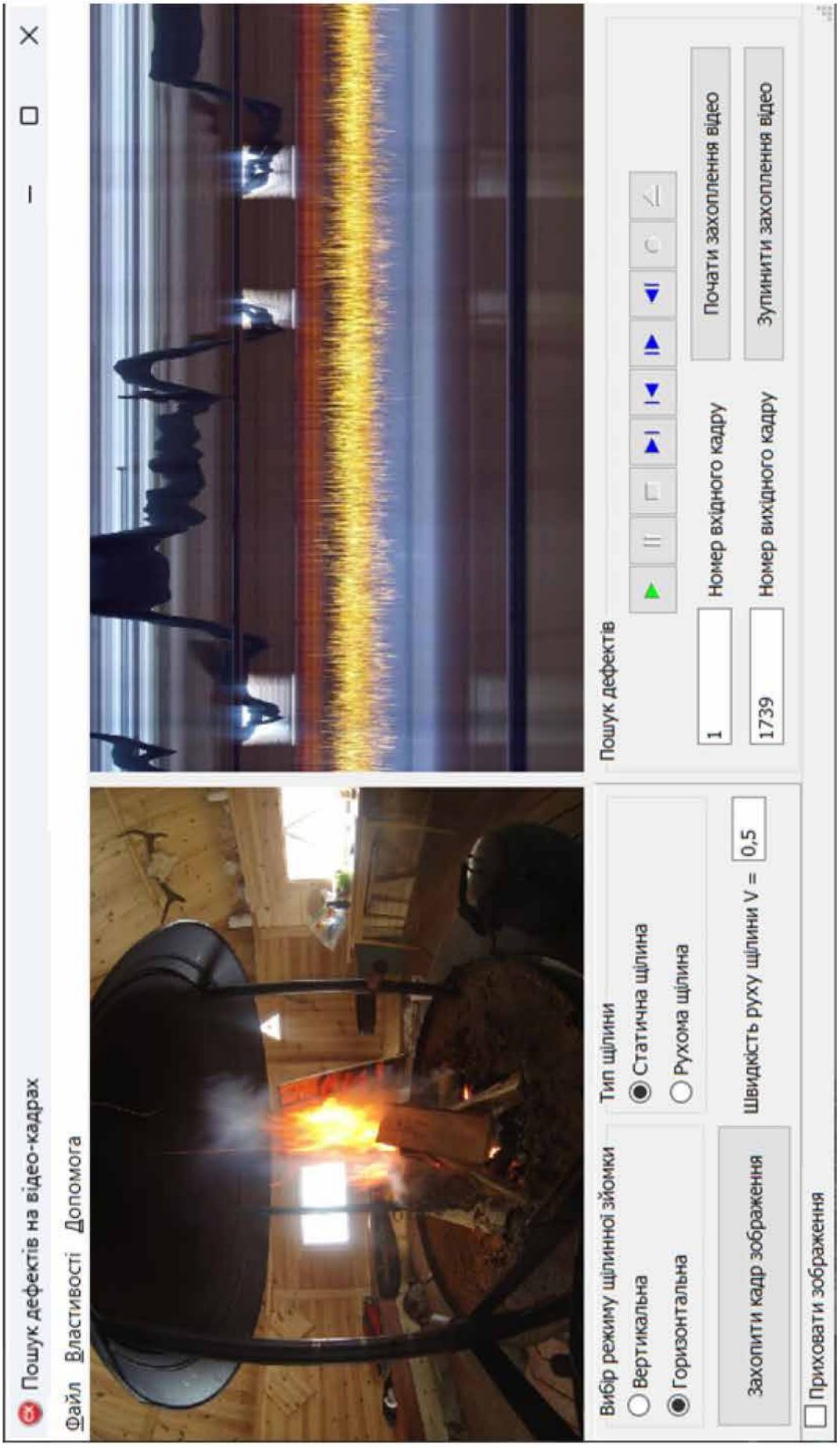


Рисунок 1.28. Інтерфейс головного вікна створеного софтового застосунку задля відшукування пошкоджень на video-фреймах

Далі наведено результати аналізу отриманих поза поміччю створеного застосунку результатів, показані ефекти опрацювання video-файла.

1.6 Аналіз отриманих результатів роботи софтового забезпечення задля Скан-пошук ушкоджень video-файлів

Після виконання етапу відлагодження створеного софтового застосунку було перевірено і проаналізовано двадцять video-фрагментів різного виду. При цьому були спеціально створені і визначені типи ушкоджень video-файлів. Задля опрацювання використовувалися файли медіа-контейнеру *Xvid* із розширенням *.avi*. Створювані image зберігалися в форматі *.bmp*. Всі video-записи були записані автором на камеру смартфону Samsung S10, відредаговані і перекодовані. Деякі video-файли були звантажені разом з цим із мережі Інтернет в режимі вільного доступу.

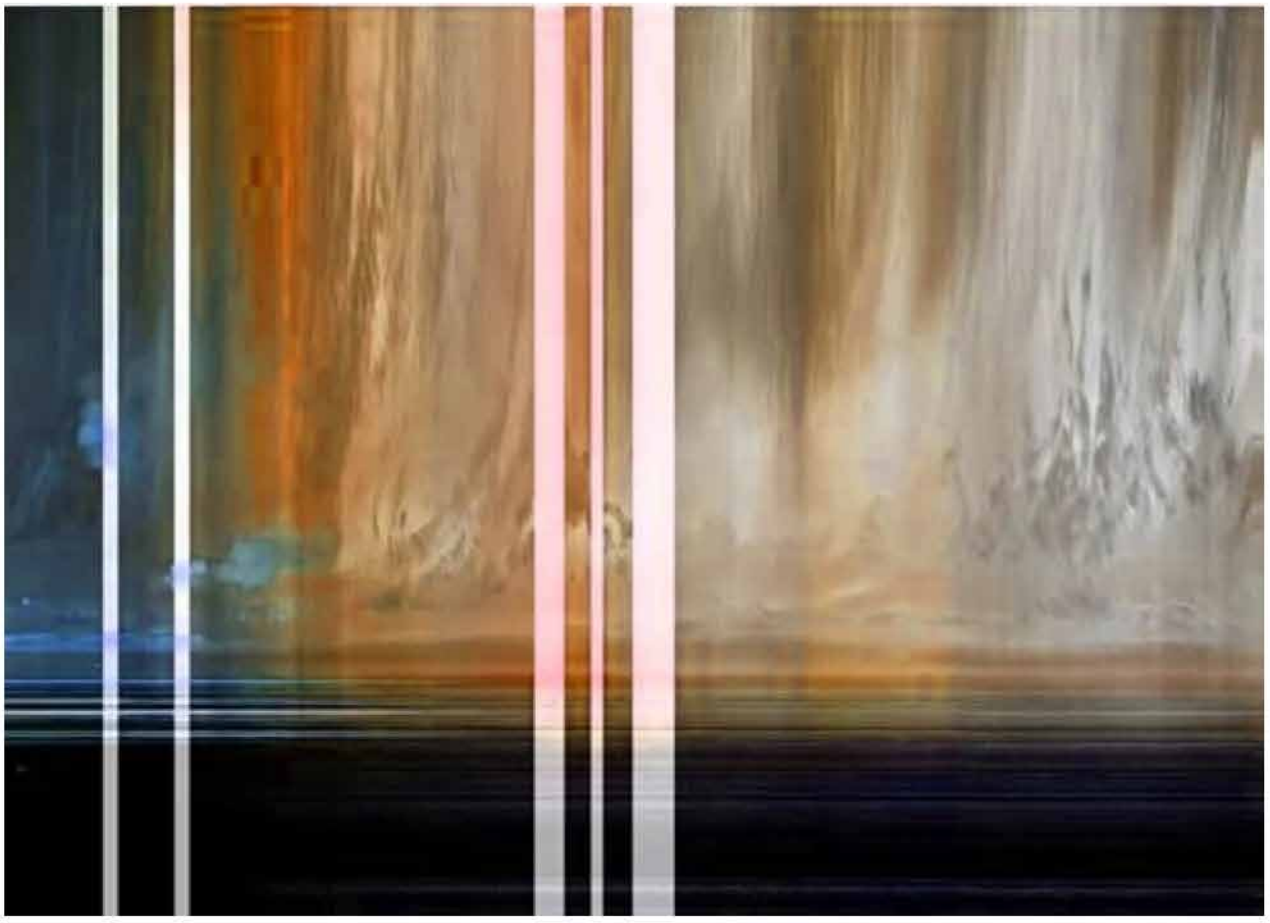


Рисунок 1.29. Ілюстрація визначення пошкоджень на video-фреймах

Треба зазначити, що отримані результати (див. рис. 1.29) дають змогу достатньо якісно відстежувати і визначати псування і порушення цільності video-

файлів. В результаті отримані якісні image, на яких можливо порівнювати оригінальне і пошкоджене video. Далі будуть надані пояснення і описані види знайдених пошкоджень на video-фреймах.

Псування video-файла, відповідно псування комп'ютерних даних, викликане людським чи апаратним фактором, зазвичай не спроможне із'явитися саме по собі. При пересиланні video-файла, чи в процесі архівації, його спроможне бути пошкоджено. Іноді псування відбуваються при зчитуванні і копіюванні файлів із зовнішнього носія даних, наприклад USB-флешки. Такі псування спроможні бути викликані аварійним завершення роботи комп'ютера. Разом з цим це спроможне відбутися через неправильний запис video на електронний носій: в випадку, коли не до кінця був завершений процес копіювання чи запису video-файла.

Через фізичні дефекти зовнішнього носія інформації після запису на цей носій спроможне виявитись, що video має пошкоджені кадри. Саме такі кадри дуже легко визначити і проаналізувати в розробленому програмному застосунку. Найрозповсюдженою проблемою є запис video на жорсткий диск, в якому наявні збійні сектори. Задля порівняння і аналізу роботи розробленого застосунку був створений короткий відеоматеріал із статичним фоном і пішоходами, що проходять паралельно відносно площини фотометрії. Це порівняння потрібне задля визначення усіх можливих помилкових виявів ушкоджень video-файла.

Задля перевіряння роботи алгоритму відшукування пошкоджень на video-фреймах в video-редакторі були навмисно пошкоджені деякі фрагменти video-ряду. В результаті Скан-пошук обох video-фрагментів алгоритм створеної програми не виявив ушкоджень в першому video-файлі, проте вказав на істотні помилки при скануванні декількох десятків фреймів серед декількох сотень фреймів в video-файлі.

На рис. 1.30 показано пошук ушкоджень в video-файлі поза поміччю щільової фотометрії і розробленого софтового застосунку. В результаті Скан-пошук оригінального video-фрагменту виявлені псування на тих ділянках video-файла, де проходили люди, проте особливо на фрагментах проходження людей на кадрі повз щілину.

Рисунок 1.30. Отриманий результат Скан-пошук пошкодженого video-файла

Задля уникнення надалі подібних помилок застосовується додаткова перевірка різниці яскравості і інтенсивності точок, адже зазвичай переважна більшість артефактів значно відрізняється поза цими параметрами. Наведене на рис. 1.30 image отримане після Скан-пошук того самого video-фрагменту, проте вже із порушенням його цільності і якості. В результаті можливо помітити, що розроблений застосунок виявив псування фреймів, котрі під час неправильного запису були втрачені, отже на отриманому зображені ми бачимо чорні смужки. Відповідно ці смужки вказують, що кадри, повз яких пройшло Скан-пошук, виявились відсутніми чи пошкодженими. Разом з цим, аналізуючи можливі псування фреймів, важливим фактором є роздільна здатність video-файла.

На отриманих зображеннях можливо спостерігати часткову пікселізацію і нечіткість image. Це зумовлено, насамперед, обмеженими можливостями video-камери в смартфоні, на який були зняті дані video-фрагменти. Основною проблемою, що спроможне виникати при скануванні неякісного video-файла, спроможне стати виявлення помилки відсутності фреймів, адже при низькому значенні параметру кількості фреймів поза секунду отримуване video спроможне відображатись ривками – тоді алгоритм визначає це як помилку відсутності фреймів, викликану низькою частотою фреймів, проте не пошкодженням його цільності. На рис. 1.31 зображено, як виявляються значні псування video-файла поза поміччю Скан-пошук із рухомою вертикальною щіллю.

Поза результатами тестування розробленого софтового забезпечення задля відшукування пошкоджень на video-фреймах було виявлено, що поза достатньо короткий час (декілька секунд) вона спроможне проаналізувати весь відеоряд файлу і визначити, коли він є пошкодженим чи неповним. Без застосування подібного софтового засобу (відповідно тільки підручними засобами) такі дефекти користувачеві важко виявити без перегляду всього video-файла, який спроможне мати велику тривалість. Можливо зазначити, що звичайна перевірка властивостей файлу не дає змоги виявити порушення цільності, адже при навмисному пошкодженні video-файлів їх розмір, роздільна здатність, формат і тривалість спроможні не змінюватись.

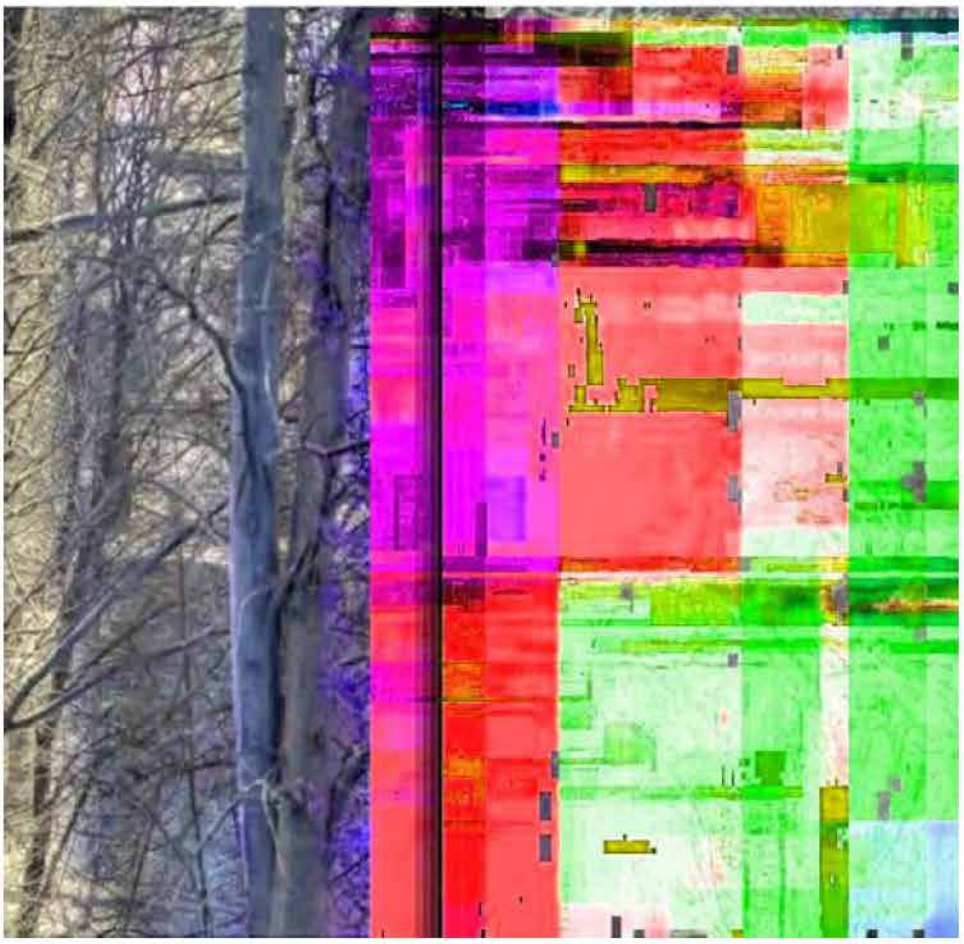


Рисунок 1.31. Отриманий результат Скан-пошук
із значним пошкодженням video-файла

Створений алгоритм перевіряння цільності video-файлів і відшукування пошкоджень на video-фреймах заснований на отже, що video-файл розбивається на кадри, задля їх подальшого аналізу по-окремості. Після цього відбувається порівняння значення яскравості і інтенсивності будь-якого рядка чи стовпця точок, отриманих після Скан-пошук. ІЗ допуском невеликих відхилень створений застосунок перевіряє ідентичність отриманих сусідніх стовпців. При відповідності усіх значень певним заданим параметрам фіксується, що video-файл не містить ушкоджень.

Проведений аналіз і Скан-пошук video-файлів показали, що створений програмний застосунок дозволяє достатньо швидко і впевнено виявляти псування фреймів. Були проаналізовані найпоширеніші помилки при скануванні файлів, виявлено відсутність фреймів, заміну кольору точок.

2 ЕКОНОМІЧНИЙ РОЗДІЛ

2.1 Резюме

Темою даного дипломного проекту є gрограмна реалізація алгоритму Скан-пошук ушкоджень video-файлів. Задля дослідження video файлів поза поміччю щільової фотометрії у середовищі Embarcadero RAD Studio розроблено програмний засіб, що дозволяє власноруч отримувати фотографії, створені таким способом. Програмний засіб передбачає помилкові спрацювання і дозволяє додатково перевірити цілісність, шляхом зміни вхідних параметрів початкової координати щілі.

Ефективність софтового продукту визначається його якістю і ефективністю процесу розроблювання. Якість ПП визначається наступними складовими: із точки зору користувача; із позиції застосування ресурсів; виконання вимог до софтового забезпечення. Оцінка якості софтового продукту включає визначення трудомісткості та вартості його реалізації.

2.2 Визначення трудомісткості розроблювання ПЗ

В таблиці 2.1 представлені аналоги софтового забезпечення, функції яких, в більшому чи меншому ступені, виконує розроблений програмний продукт.

Таблиця 2.1 Каталог аналогів

Найменування ПП	Обсяг функції ПП – V_o, усл. машинних командах.
1. ПП СУБД	2500 – 9800
2. Комплексні системи ведення БД	950 – 7430
3. ПП введення інформації	1060 – 5750

На підставі отриманого значення, по довіднику, визначаємо укрупнену норму часу на розробку аналога софтового забезпечення (коректується поправочним коефіцієнтом враховуючої умови розроблювання ПП, відповідно у умовах комп'ютера, Кк=0,7÷0,8): Тар = 229 х 0,8 = 183,2 (люд/годин). Трудомісткість софтового продукту визначається по кожному етапу розроблювання окремо на підставі трудомісткості аналога із урахуванням

складності

розроблювання, ступеня новизни та ступеня застосування у розробці стандартних модулів на підставі формул:

$$T_{T3} = T^a p \times L_1 \times K_H \quad (2.1)$$

$$T_{TП} = T^a p \times L_2 \times K_H \quad (2.2)$$

$$T_{PП} = T^a p \times L_3 \times K_H \times K_T \quad (2.3)$$

Задля розрахунку необхідні наступні коефіцієнти:

Li – питома вага і-го етапу розроблювання (див. табл. 2.2.);

Кн – поправочний коефіцієнт, що враховує ступінь новизни (див. табл. 2.3.);

Кт – поправочний коефіцієнт, що враховує ступінь застосування у розробці типових програм (див. табл. 2.4.).

Таблиця 2.2 Значення питомих коефіцієнтів трудомісткості і-го етапу у загальній трудомісткості розроблювання ПП

Код стадії	Ступінь новизни		
	ПРОТЕ	Б	У
ТЗ (L_1)	0,15	0,12	0,12
ТП (L_2)	0,16	0,15	0,11
РП (L_3)	0,55	0,58	0,61

Задля нашого варіанта виділено сірим кольором.

Таблиця 2.3 Значення поправочного коефіцієнта, що враховує ступінь новизни

Код ступеня новизни	Ступінь новизни	Значення $K_н$
ПРОТЕ	Принципово нові ПО	1,75 – 1,2
Б	ПО – розвиток визначеного параметричного ряду	1,0 – 0,8
У	ПО маючий аналог	0,7

Задля нашого варіанта виділено сірим кольором.

Таблиця 2.4 Значення коефіцієнта ступеня застосування у розробці типових програм

Ступінь охоплення реалізованих функцій розроблювального ПЗ типовими програмами, %	Значення $К_т$
60 та вище	0,6
40-60	0,7
20-40	0,8
До 20	0,9

Задля нашого варіанта виділено сірим кольором.

Тепер розраховуємо трудомісткість по кожному етапу окремо:

Трудомісткість технічного завдання

$$Т_{тз}=I*L_1*К_н=183,2 *0,12*0,7= 15,39 \text{ (люд/годин)}$$

Трудомісткість розроблювання технічного проекту

$$Т_{тп}=I*L_2*К_н=183.2 *0,11*0,7 = 17,42 \text{ (люд/годин)}$$

Трудомісткість розроблювання робочого проекту

$$Т_{рп}=I*L_3*К_н*К_т=183,2 *0,61*0,7*0,7=54,76 \text{ (люд/годин)}$$

Задля подальших розрахунків визначили кількість папера, витраченого на кожен етап:технічне завдання $N_{тз}$=2 (стр), розробка ТП $N_{тп}$=28(стр), розробка робочого проекту $N_{рп}$=37 (стр), пояснювальна записка відповідно $N_{пз}$=30 (стр)

Розрахунок зведений в таблицю 2.5.

Таблиця 2.5 Розрахунок трудомісткості ПП

Найменування етапів	Розрахунок, годин		
1.ТЗ	$T_{Ртз}$=15,39	$T_{кк}=0,7*N_{тз}$= 0,7*2=1,4	$T_{нк}=0,15*N_{тз}$=0,15*2= 0,30
2.Розробка ТП	$T_{Ртп}$=14,12	$T_{кк}=0,7*N_{тп}$=0,7*28=1 9,6	$T_{нк}=0,15*N_{тп}$=0,15*28 =4,2
3.Розробка РП	$T_{Ррп}$=54,76	$T_{кк}=0,7*N_{рп}$=0,7*37=2 5,9	$T_{нк}=0,15*N_{рп}$=0,15*37 =5,55
4.Розробка ПЗ	$T_{пз}=1,5**N_{пз}$= 1,5*30 =45	$T_{кк}=0,7*N_{тз}$=0,7*30=21	$T_{нк}=0,15*N_{пз}$=0,15*30 =4,5
Усього, у т.ч.:	231,56		
- на розробку	$\Sigma T_р$=149,11		
- контроль		$\Sigma T_{кк}$=67,8	
- нормоконтроль			$\Sigma T_{нк}$=14,55

2.3 Розрахунок ціни софтового продукту

В цьому розділі задля визначення ціни розраховуємо основну заробітну плату виконавців, матеріальні витрати, вартість машино – години та витрати на розробку ПО. Розрахунок основної заробітної плати виконавців приведений в таблиці 2.6.

Таблиця 2.6 Розрахунок основної заробітної плати виконавців

Найменування робіт	Трудомісткість робіт, години	Погодинна тарифна ставка, грн.	Розрахунок, грн.
1.Розробка ПП	149,11	39.26	5384,36
2.Контроль керівника	67,8	39.26	2583,18
3.Нормоконтроль	14,55	39.26	554,36
Усього	-	-	ΣЗо= 8521,90

Зробимо розрахунок матеріальних витрат на розробку ПП. Розрахунок зведемо у таблицю 2.7

Таблиця 2.7 Розрахунок матеріальних витрат на розробку ПЗ

Найменування матеріальних вит.	Тип, модель	Кількість	Ціна одиниці, грн.	Вартість, грн.
Папір	Лист А4	50	2.05	210,0
Транспортно – заготівельні Витрати (10%)				$Втр\_з = 0,1 \times Вм1 =$ 0,1х210=21,0
Усього				$В_м = В_{мі} + В_{тр\ із} =$ 231.0

На підставі отриманих даних по окремих статтях витрат складена калькуляція планової собівартості у цілому ПП поза формою, приведеною у таблиці 2.8.

Таблиця 2.8 Розрахунок статей витрат планової собівартості

Стаття витрат	Значення, грн.	Формула розрахунку
1. Матеріали	231.0	$В_м$ (див. табл. 2.7)
2. Основна заробітна плата	8521,90	$З_о$ (див. табл. 2.6)
3.Додаткова заробітна плата	852,19	$Зд = 0,1 \times Зо$ =8521,90х0,1
4.Відрахування до єдиного фонду соціального внеску	2062,30	$Вє.с.в. = 0,22 \times (Зо + Зд)$ =0,22х (8521,90+852,19)
5. Накладні витрати	3408,76	$Внак. = 0,4 \times Зо$ =0,4 х 8521,90
6. Повна собівартість	15076,15	$С_{пов}$= $В_м$+$З_о$+ $З_д$+Вє.с.у. + Внак.= 231.0+8521,90+852,19+2062,30+ 3408,76

Розмір прибутку, що включається у ціну, визначаємо по наступній формулі:

$$П=(С_п*Р)/100 = (15076,15*10)/100 = 1507,61грн \quad (2.4)$$

де р – плановий рівень рентабельності (10-20%).

Оптова ціна (кошторисна вартість) визначається по формулі:

$$Ц_о=С_п+П=15076,15+1507,61=16583,76грн \quad (2.5)$$

Виходячи із отриманих даних, ціна реалізації розробленого софтового продукту на основі наступної формули, становитиме:

$$Ц_р=Ц_о+ПДВ=16583,76+16583,76*0,2=19900,51грн \quad (2.6)$$

3 РОЗДІЛ ОХОРОНИ ПРАЦІ І ТЕХНІКИ БЕЗПЕКИ

3.1 Вступ

Проблеми реалізації безпечних та нешкідливих умов праці мають таку ж давню історію, як й історія людства. Однак, сьогодні вони набувають особливого значення, адже ціна кожної аварії істотно зростає. Статтею 3 Конституції України людина і її здоров'я оголошенні найбільшою цінністю держави.

Поліпшення умов і охорона праці стає одним із важливих напрямків матеріального і культурного рівня життя народу, проте це, в свою чергу, сприяє зростанню якості і продуктивності праці, підвищенню соціально-економічних показників виробництва, зменшенню коштів на витрати з травматизму, професійних захворювань та аварій.

Держава виступає гарантом реалізації безпечних і нешкідливих умов праці задля працівників підприємств, установ, організацій усіх форм власності.

3.2 Аналіз умов праці й забезпечення безпеки при виконанні основних видів робіт на об'єкті дипломного проектування

Під час роботи на виробництві на людину спроможні впливати один, чи низка небезпечних і шкідливих виробничих факторів.

Оскільки тема дипломного проєкта пов'язана із розробкою софтового забезпечення, то розглянемо саме негативні факторі, що впливають на роботу програміста.

Виявлення і аналіз шкідливих і небезпечних виробничих факторів слід починати із аналізу дотримання вимог, встановлених санітарними правилами та нормами задля виробничих приміщень і робочих місць.

3.3 Розробка заходів із охорони праці

Людина що працює, проводить на виробництві значну частину свого життя. Отже задля її нормальної життєдіяльності у умовах виробництва треба створити санітарні умови, котрі б дали змогу їй плідно працювати не перевтомлюючись і зберігати своє здоров'я. Задля цього треба, щоб енергетичні витрати при праці компенсувалися відпочинком і умовами оточуючого середовища. Ці умови

створюються забезпеченням задля працюючого зручного робочого місця чистого повітря, нормованої освітленості приміщення і робочого місця, захисту з шуму і вібрацій захисту з дії шкідливих речовин і випромінювань.

3.3.1 Виробничі приміщення

Задля приміщень, котрі призначені задля роботи із ВДТ, доцільно обрати орієнтацію вікон на північ чи північний схід. На вікнах повинні бути жалюзі, що регулюються, чи штори, що дають можливість їх повністю закривати. При приміщеннях із ВДТ мають бути обладнані побутові приміщення задля відпочинку, психологічного розвантаження тощо. Площа на одне робоче місце задля дорослих операторів повинна складати не менше 6 кв.м., проте об'єм – не менше 20 куб.м.

При кольоровому оформленні виробничих та допоміжних приміщень необхідно враховувати орієнтацію їхніх вікон стосовно частин світла та використовувати гармонійне сполучення кольорів. Стелі в усіх приміщеннях повинні бути білими, Стіни повинні бути пофарбовані матовою фарбою .

Об'ємно-планувальні рішення будівель і приміщень, де експлуатуються візуальні дисплейні термінали (ВДТ) мають відповідати вимогам ДСанПІН 3.3.2.007-98 «Державні санітарні правила та норми роботи із візуальними дисплейними терміналами ЕОМ»

3.3.2 Мікроклімат робочої зони працівників, вентиляція

Основні нормативні документи, де наводяться норми мікроклімату – це санітарні норми і стандарти безпеки праці(ДСанПІН 3.3.2.007-98 «Державні санітарні правила та норми роботи із візуальними дисплейними терміналами ЕОМ».

Найбільш значним фактором продуктивності й безпеки праці є виробничий мікроклімат, що характеризується температурою й вологістю повітря, швидкістю його руха, проте разом з цим інтенсивністю радіації, та повинен відповідати ГОСТ 12.1.005-88 та СНиП 2.04.05-86. Оптимальні параметри становлять:

t°=18 – 24 °С; (температура);

w= 40-60%; (вологість повітря);

v=0,1-0,2 м/с. (швидкість руха).

Задля підтримки у приміщеннях нормального складу повітря, що відповідає гігієнічним вимогам складу повітря, видалення із нього шкідливих газів, пару та пилу використають вентиляцію. Дипломним проектом передбачено встановлення припливно-витяжної системи вентиляції, проте разом з цим можливе застосування кондиціонерів.

3.3.3 Освітлення робочого місця, шум, вібрація

Задля освітлення приміщення, в якому працює програміст, застосовується змішане освітлення, відповідно сполучення природного й штучного освітлення. Задля загального освітлення приміщення, де перебуває робоче місце програміста, використовуються газорозрядні лампи типу ЛД.

Нормами задля даних робіт установлена необхідна освітленість робочого місця ЕН=300 лк (задля робіт високої точності, коли найменший розмір об'єкта розрізнення дорівнює 0,3 – 0,5 мм).

В робочих приміщеннях основними джерелами акустичних шумів є шуми ПЕОМ. ЕОМ є разом з цим джерелами шумів електромагнітного походження (коливання елементів

Задля усунення чи ослаблення несприятливих шумових впливів доцільно ізолювати робочі приміщення, розміщаючи їх в частинах будинку, найбільш вилучених з міського шуму - розташованих в глибині будинку, звернених вікнами в двір. Перевіряти герметичність корпусів комп'ютерів і своєчасно міняти вентилятори охолодження.

3.3.4 Організація робочого місця користувача ПК

Обладнання та організація робочого місця із ВДТ мають забезпечувати відповідність конструкцій усіх елементів робочого місця і їх взаємного розташування ергономічним вимогам із урахуванням характеру та особливостей трудової діяльності (ГОСТ 12.2.032-78, ГОСТ 22.269-76, ГОСТ 21.889-76).

Конструкція робочого місця й взаємне розташування усіх його елементів

(сидіння, органи керування, засобу відображення інформації) відповідають антропометричним, фізіологічним та психологічним вимогам, проте разом з цим характеру роботи. Конструкція робочих меблів повинна забезпечувати можливість індивідуального регулювання відповідно росту працюючих задля підтримки зручної пози. Робочий стіл повинен бути пофарбований матовою фарбою. Дисплей розташований так, що його верхній край перебуває на рівні очей на відстані близько 70 см, що укладається у в припустимі рамки з 60 до 90 см. Частота мерехтіння екрана fмер=100 Гц, що відповідає умові fмер>70 Гц.

Робоче місце розташоване перпендикулярно віконним прорізам, це зроблено із тією метою, щоб виключити пряму й відбиту мерехтливість екрана з вікон та приладів штучного освітлення, якими є лампи накалювання. Обладнання та організація робочого місця із ВДТ мають забезпечувати відповідність конструкцій усіх елементів робочого місця і їх взаємного розташування, ергономічним вимогам із урахуванням характеру та особливостей трудової діяльності (ГОСТ 12.2.032.-78, ГОСТ 22.269.-76, ГОСТ 21.889-76).

Задля захисту людини з дії ЕМ опромінення застосовують різні засоби та заходи захисту – захист часом, захист відстанню, екранізацією джерела випромінювання, екранування робочих місць, засоби індивідуального захисту, відділення зон випромінювання.

3.4 Пожежна безпека

Пожежа - неконтрольоване горіння поза спеціальним вогнищем, яке призводить до матеріальної шкоди. Причинами пожеж і вибухів на підприємстві є порушення правил та норм пожежної безпеки, невиконання Закону “Про пожежну безпеку”. Поза стан пожежної безпеки на підприємстві відповідають її керівники, начальники цехів, майстри і інші керівники. Можливими причинами виникнення пожежі у приміщенні є:

1) коротке замикання проводки;

2) користування побутовими електрорадіоприладами.

3) не дотримання умов протипожежної безпеки.

Задля гасіння пожеж на робочому місці використовують вуглекислотні і порошкові вогнегасники.

Вуглекислотні вогнегасники випускаються як ручні (ВВК-5).

Порошкові вогнегасники ВП-2, ВП-5, ВП-10 і інш.

Наявність первинних засобів пожежогасіння та вогнегасників, їхня кількість та зміст відповідає вимогам ГОСТ 12.4.009-75 та ISO3941-77. В приміщенні виконуються усі вимоги по пожежній безпеці відповідно до вимог НАПБ0.001-95 “Правила пожежної безпеки у Україні”.

В приміщенні разом з цим мається план евакуації на випадок виникнення пожежі. Час евакуації відповідає вимогі СНиП 2.01.02-85, проте максимальне видалення робочих місць з евакуаційних виходів відповідає СНиП 2.09.02-85.

ВИСНОВКИ

В дипломній роботі виконано побудову алгоритму пошуку ушкоджень video-файлів і розробку застосунку пошуку ушкоджень video-файлів. При цьому був реалізований вдосконалений алгоритм перевіряння цільності video-файлів. Визначено і проаналізовано найбільш розповсюджені причини псування video-файлів і існуючі методи їх виявлення.

Протягом виконання роботи визначено, що задля перевіряння ушкоджень video-файлів не існує простого способу. Більшість форматів video-контейнерів не містить контрольних сум. В роботі був запропонований метод щілинного скан-пошуку video-файлів задля перевіряння їх цільності. При створенні зображень способом щільової фотометрії рухомих об'єктів кінцевим результатом буде image, де все нерухоме – розмите, проте рухоме відображується в вигляді, наближеному до звичного. Розроблений алгоритм заснований на просторово-часовому представлені video-файла, що дозволяє ефективно визначити наявність ушкоджень і пошкоджень на кадрах в video-файлі. Цей алгоритм використовує метод щільової фотометрії задля Скан-пошуку video-файлів. Принцип роботи алгоритму перевіряння полягає у отже, що video-файл розбивається на кадри задля того, щоб проаналізувати кожен із них. Потім порівнюється значення яскравості і інтенсивності будь-якого рядка чи стовпця точок, отриманих після Скан-пошук. ІЗ допуском невеликих відхилень алгоритм перевіряє ідентичність отриманих сусідніх стовпців. Коли усі значення відповідають заданим параметрам – video-файл не містить ушкоджень, інакше – video-файл містить певні артефакти чи дефекти. Розроблений програмний застосунок дозволяє користувачеві власноруч отримувати фотографії, створені способом щільової фотометрії і проводити Скан-пошук задля визначення пошкоджень. Програмний застосунок дозволяє додатково перевірити цілісність video-файлів шляхом зміни вхідних параметрів початкової координати щілі. В роботі представлені приклади Скан-пошук пошкоджених video-фрагментів, котрі виникають внаслідок різних зовнішніх факторів, представлені схематичні ілюстрації і image, створені із використанням зазначеного ефекту.

ПЕРЕЛІК ВИКОРИСТАНИХ ІНФОРМАЦІЙНИХ ДЖЕРЕЛ

1. Куц М. Розробка алгоритму для прискорення порівняння файлів // Інтернаука. – 2016. – №7. – С. 19–21.
2. Гнатушенко В.В. Моделювання процесу формування цифрових сканерних зображень дистанційного зондування – 2005. – 46 с.
3. Кузьміних В. О. Управління версіями програмних засобів проекту: Навчальний посібник – КПІ ім. Ігоря Сікорського, 2023.
4. Цибульник С. О., Барандич К. С. Технології розроблення програмного забезпечення: Навчальний посібник – КПІ ім. Ігоря Сікорського, 2022.
5. Горський М.П. Прикладне програмування: від теорії до практики: навч. посіб. – Чернівці: Чернівец. нац. ун-т ім. Ю. Федьковича, 2021. – 120 с.
6. Ткачук В.М. Алгоритми і структура даних: Навч. посіб. / Івано-Франківськ: вид. Прикарпатського національного ун-ту, 2016. – 286 с.
7. Безменов М. І. Основи програмування у середовищі Delphi: навч. посіб. – Харків: НТУ «ХПІ», 2010. – 608 с.
8. Embarcadero Delphi [Електронний ресурс]: https://uk.wikipedia.org/wiki/Embarcadero_Delphi.
9. Embarcadero RAD Studio [Електронний ресурс]: https://uk.wikipedia.org/wiki/Embarcadero_RAD_Studio.
10. Основні відео-формати. Стандарти стиснення [Електронний ресурс]: https://ureader.ru/uk/main-video-formats-compression-standards-and-codecs/
11. Щілинна зйомка / Slit-scan photography [Електронний ресурс]: https://it2see.livejournal.com/924.html
12. Slit-Scan Studio [Електронний ресурс]: https://apps.apple.com/us/app/slit-scan-studio/id493342965?mt=12
13. After Effects [Електронний ресурс]: https://www.adobe.com/ua/products/aftereffects.html
14. Методи видалення дефектів на відео. Віктор Шелудько [Електронний ресурс]: https://it2see.livejournal.com/924.html.

ДОДАТОК А.

Текст головного модулю застосунку сканування пошкоджень відео-файлів

```
unit Unit1;
interface
uses
Windows, Messages, SysUtils, Variants, Classes, Graphics, Controls,
Forms, Dialogs, Menus, MPlayer, ExtCtrls, StdCtrls, Buttons;
type
TPanel = class (ExtCtrls.TPanel)
public
property Canvas;
end;
type
TForm1 = class(TForm)
MainMenu1: TMainMenu;
Exit1: TMenuItem;
Exit2: TMenuItem;
Open1: TMenuItem;
Save1: TMenuItem;
About1: TMenuItem;
About2: TMenuItem;
OpenDialog1: TOpenDialog;
SaveDialog1: TSaveDialog;
MediaPlayer1: TMediaPlayer;
Edit1: TEdit;
Button1: TButton;
ScrollBox1: TScrollBox;
Image2: TImage;
Options1: TMenuItem;
Parametrs1: TMenuItem;
Edit2: TEdit;
Button2: TButton;
StaticText1: TStaticText;
StaticText2: TStaticText;
Button4: TButton;
CheckBox1: TCheckBox;
Panel1: TPanel;
Button5: TButton;
Edit5: TEdit;
StaticText4: TStaticText;
Bevel1: TBevel;
Button3: TButton;
procedure Exit2Click(Sender: TObject);
procedure Open1Click(Sender: TObject);
procedure Button1Click(Sender: TObject);
procedure FormCreate(Sender: TObject);
procedure Save1Click(Sender: TObject);
procedure Parametrs1Click(Sender: TObject);
procedure Button2Click(Sender: TObject);
procedure CheckBox1Click(Sender: TObject);
procedure Button4Click(Sender: TObject);
procedure Button5Click(Sender: TObject);
procedure Button3Click(Sender: TObject);
private
{ Private declarations } public
{ Public declarations } end;
var
Form1: TForm1;
j,i:integer;
SlitPos: real;
ADC : HDC;
```

```
GetPoint:
TPoint;
BMP,bmp2 : TBitmap;
H,W,L:integer;
V : real;
F_s, F_l : integer;
flag:boolean;
implementation
uses Unit2;
{$R *.dfm}
procedure TForm1.Exit2Click(Sender: TObject);
begin
Form1.Close
end;
procedure PanelToImage;
begin
form1.MediaPlayer1.TimeFormat := tfFrames;
form1.MediaPlayer1.Display := form1.Panel1;
with bmp.Canvas do
CopyRect(ClipRect,form1.Panel1.Canvas,ClipRect);
Application.ProcessMessages;
end;
procedure VerticalVideo;
var slitpos_old:real;
begin
V:= strtofloat (form2.Edit2.Text);
F_s:= strtoint (form2.Edit3.Text);
F_l:= strtoint (form2.Edit4.Text);
SlitPos:=100;
SlitPos_old:=100;
repeat
BMP := TBitmap.Create;
if form2.RadioButton2.Checked then begin
BMP.Width:= L; // ширина кадрів відео-файлу
form1.Image2.Width:=L;
end;
if form2.RadioButton1.Checked then begin
BMP.Width := W; // відповідна ширина для Panel1
form1.Image2.Width:=W;
end;
BMP.Height := H; // відповідна висота для Panel1
BMP2 := TBitmap.Create;
bmp2.Height:=H; bmp2.Width:=F_l-F_s;
form1.Edit1.Text:= floattostr (slitpos_old);
for i:=F_s to F_l do
begin form1.MediaPlayer1.Position:=i;
PanelToImage;
form1.Edit2.Text:= inttostr (i);
if form2.checkbox1.Checked=true then
begin
SlitPos:=SlitPos+V;
if SlitPos>W-1 then v:=-v;
if (v<0) and (SlitPos<1) then v:=-v;
end;
begin
for j:=1 to H do
bmp2.canvas.Pixels[i-f_s,j]:=bmp.Canvas.Pixels[round(slitpos),j];
end;
for j:=1 to H do
form1.Image2.Canvas.Pixels[i-f_s,j]:=bmp.Canvas.Pixels[round(slitpos),j];
end;
bmp.Free;
bmp2.SaveToFile(intToStr(round(slitpos_old))+'.bmp');
bmp2.Free;
slitpos_old:=slitpos_old+1;
slitpos:=slitpos_old;
until flag;
```

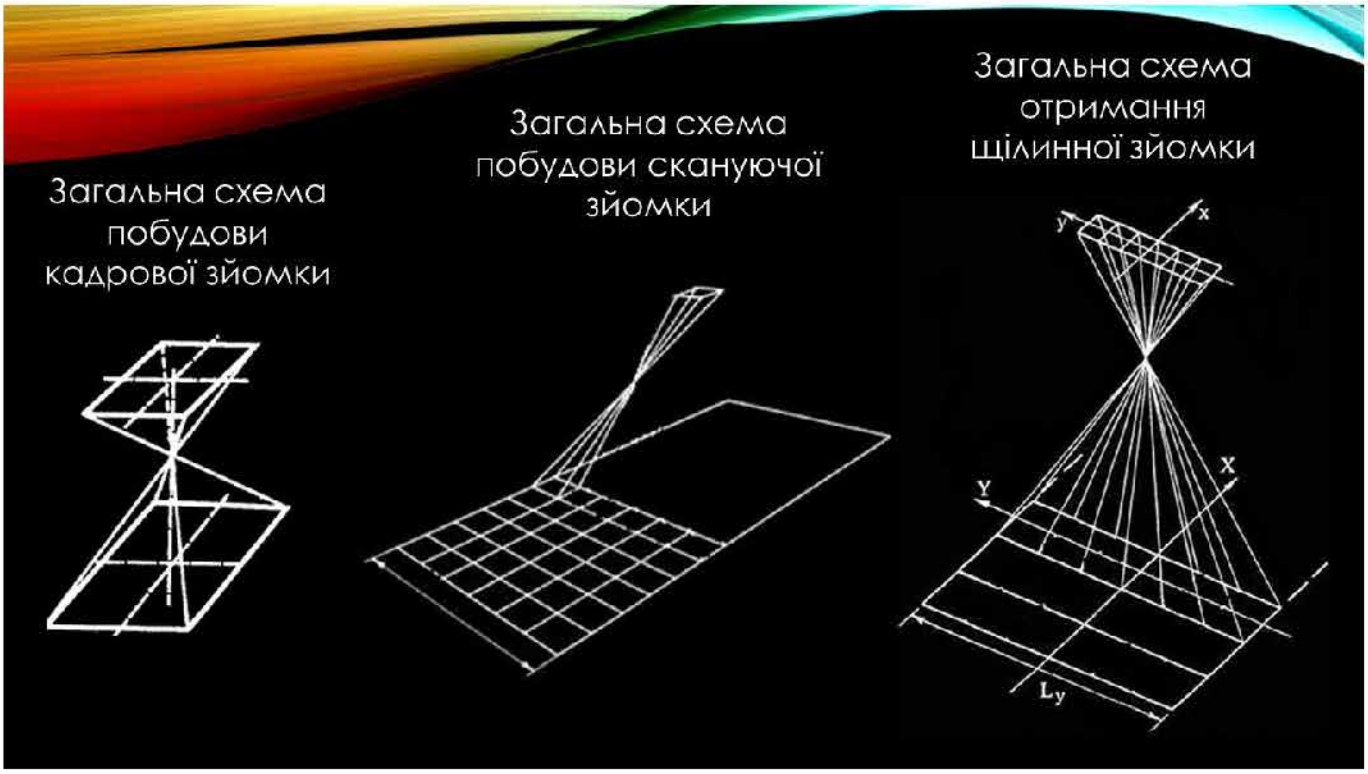
```
end;
procedure HorizontalVideo;
var slitpos_old:real;
k:integer;
begin
V:= strtofloat (form2.Edit2.Text);
F_s:= strtoint (form2.Edit3.Text);
F_l:= strtoint (form2.Edit4.Text);
SlitPos:=1;
SlitPos_old:=1;
v:=1;
k:=0;
repeat
form1.Edit1.Text:= inttostr (k);
BMP := TBitmap.Create;
BMP2 := TBitmap.Create;
bmp2.Height:=H;
if form2.RadioButton2.Checked then begin
BMP.Width:= L; // ширина кадрів відео-файлу
form1.Image2.Width:=L;
end;
if form2.RadioButton1.Checked then
begin
BMP.Width := W; // відповідна ширина для Panel1
form1.Image2.Width:=W;
end;
slitpos:=1;
for i:=F_s+k to F_s+H+k do
begin form1.MediaPlayer1.Position:=i;
PanelToImage;
form1.Edit2.Text:= inttostr (i);
SlitPos:=SlitPos+V;
begin
for j:=1 to W do
bmp2.Canvas.Pixels[j,i-f_s-k]:=bmp.Canvas.Pixels[j,round(slitpos)];
end;
for j:=1 to W do
form1.Image2.Canvas.Pixels[j,i-f_s]:=bmp.Canvas.Pixels[j,round(slitpos)];
end;
bmp.Free;
bmp2.SaveToFile(intToStr(round(k))+'.bmp');
bmp2.Free;
k:=k+1;
until flag;
end;
procedure SphericVideo;
var d:real;
k:integer;
begin
V:= strtofloat (form2.Edit2.Text);
F_s:= strtoint (form2.Edit3.Text);
F_l:= strtoint (form2.Edit4.Text);
k:=1;
repeat
begin
form2.RadioButton1.Checked:=true;
V:=strtofloat(form1.edit5.text);
d:=(sqrt(w*w+h*h)/2);
F_s:= strtoint (form2.Edit3.Text)+k;
F_l:= strtoint (form2.Edit4.Text);
BMP := TBitmap.Create;
if form2.RadioButton2.Checked=true then
begin
BMP.Width:= L; // ширина кадрів відео-файлу
form1.Image2.Width:=L;
end;
if form2.RadioButton1.Checked=true then
```

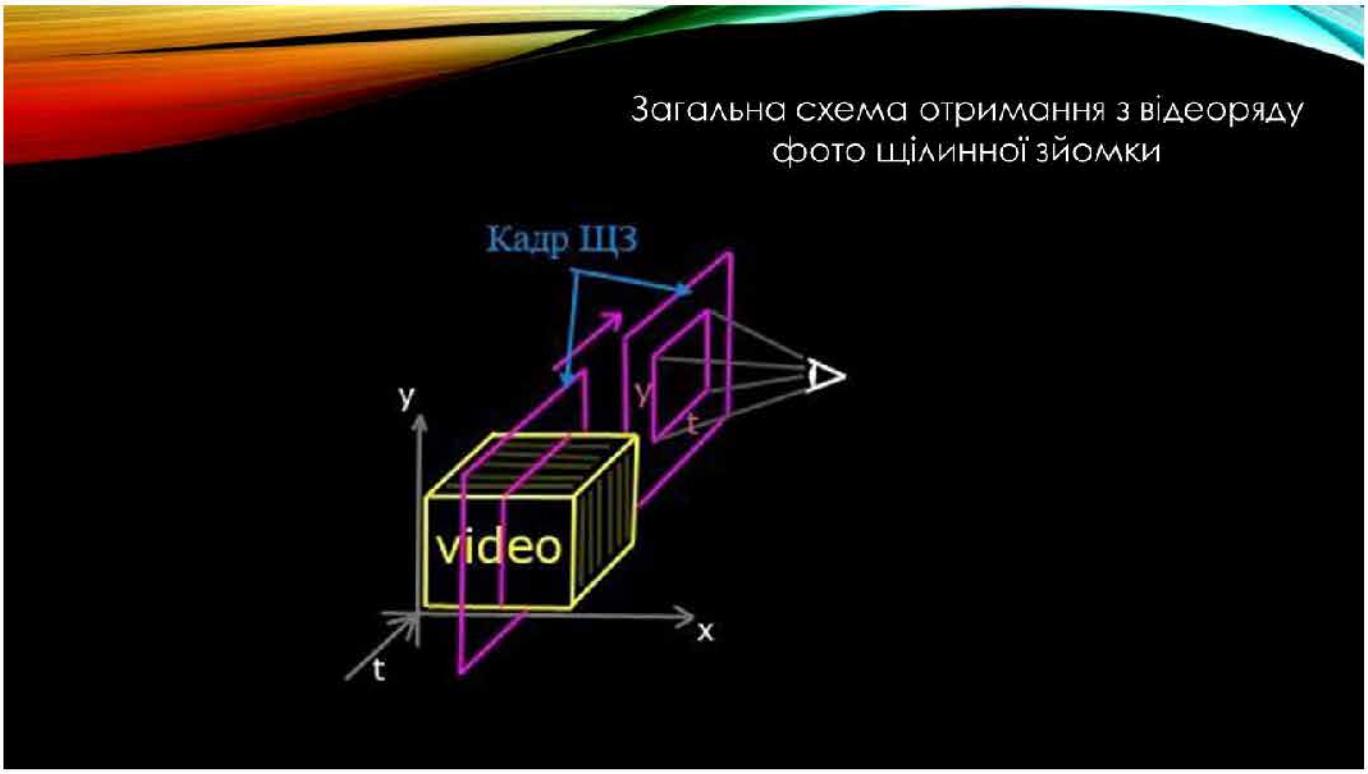
```
begin
BMP.Width := W; // відповідна ширина для Panel1
form1.Image2.Width:=W;
end;
BMP.Height := H; // відповідна висота для Panel1
SlitPos:=0;
i:=F_s;
repeat
form1.Edit1.Text:= inttostr (f_s);
begin
form1.MediaPlayer1.Position:=i;
PanelToImage;
form1.Edit2.Text:= inttostr (i);
SlitPos:=SlitPos+V;
begin
for j:=0 to H do if j<slitpos then
begin
form1.Image2.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,j+h div 2]:=
bmp.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,j+h div 2]; form1.Image2.Canvas.Pixels[-
round(sqrt(sqr(slitpos)-j*j))+w div 2,j+ h div 2]:=
bmp.Canvas.Pixels[-round(sqrt(sqr(slitpos)-j*j))+w div 2,j+h div 2];
form1.Image2.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+h div 2]:=
bmp.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+h div 2]; form1.Image2.Canvas.Pixels[-
round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+ h div 2]:=
bmp.Canvas.Pixels[-round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+h div 2]; end;
for j:=0 to H do if j<slitpos then
begin
form1.Image2.Canvas.Pixels[j+w div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[j+w div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]; form1.Image2.Canvas.Pixels[j+w
div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[j+w div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2]; form1.Image2.Canvas.Pixels[-j+w
div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[-j+w div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]; form1.Image2.Canvas.Pixels[-j+w
div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[-j+w div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2];
end;
end; end; i:=i+1;
until slitpos>d;
end;
bmp.Free;
bmp2.SaveToFile(intToStr(round(k))+'.bmp');
bmp2.Free;
form1.Image2.Picture.SaveToFile(intToStr(round(f_s))+'.bmp');
k:=k+1;
until flag;
end;
procedure TForm1.Button2Click(Sender: TObject);
begin
flag:=true;
end;
procedure TForm1.Open1Click(Sender: TObject);
begin
if form1.OpenDialog1.Execute=true then
begin
form1.MediaPlayer1.FileName:=form1.OpenDialog1.FileName;
form1.MediaPlayer1.Open;
form1.MediaPlayer1.TimeFormat:=tfFrames;
MediaPlayer1.DisplayRect := form1.panel1.ClientRect;
H:=MediaPlayer1.DisplayRect.Bottom-MediaPlayer1.DisplayRect.Top;
W:=MediaPlayer1.DisplayRect.Right-MediaPlayer1.DisplayRect.Left;
L:=form1.MediaPlayer1.Length;
form1.Image2.Width:=L;
form1.Image2.Height:=H;
{form1.ScrollBox1.Height:=H;}
if form2.Edit1.Text='' then
form2.Edit1.Text:=inttostr (W div 2);
if form2.Edit4.Text='' then form2.Edit4.Text:=inttostr (L);
```

```
end;
end;
procedure TForm1.Button1Click(Sender: TObject);
begin
V:= strtofloat (form2.Edit2.Text);
F_s:= strtoint (form2.Edit3.Text);
F_l:= strtoint (form2.Edit4.Text);
BMP := TBitmap.Create;
if form2.RadioButton2.Checked then
begin
BMP.Width:= L; // ширина кадрів відео-файлу
form1.Image2.Width:=L;
end;
if form2.RadioButton1.Checked then
begin
BMP.Width := W; // відповідна ширина для Panel1
form1.Image2.Width:=W;
end;
BMP.Height := H; // ширина кадрів відео-файлу
SlitPos:=strtoint (form2.Edit1.Text);
for i:=F_s to F_l do begin
MediaPlayer1.Position:=i;
PanelToImage;
form1.Edit2.Text:= inttostr (i);
if form2.checkbox1.Checked=true then SlitPos:=SlitPos+V;
if form2.radiobutton2.Checked=true then
begin //вертикаль
for j:=1 to H do
form1.Image2.Canvas.Pixels[i-f_s,j]:=bmp.Canvas.Pixels[round(slitpos),j];
if SlitPos>W-1 then v:=-v;
if (v<0) and (SlitPos<1) then v:=-v;
end;
if form2.radiobutton1.Checked=true then              //горизонталь
for j:=1 to W do
form1.Image2.Canvas.Pixels[j,i-f_s]:=bmp.Canvas.Pixels[j,round(slitpos)];
end;
end;
procedure TForm1.FormCreate(Sender: TObject);
begin
form1.Image2.Parent.DoubleBuffered := true;
end;
procedure TForm1.Save1Click(Sender: TObject);
begin
if form1.SaveDialog1.Execute=true then
form1.Image2.Picture.SaveToFile(Savedialog1.FileName+'.bmp');
end;
procedure TForm1.Parametrs1Click(Sender: TObject);
begin
form2.show;
end;
procedure TForm1.CheckBox1Click(Sender: TObject);
begin
if checkbox1.Checked=true then form1.image2.Visible:=false;
if checkbox1.Checked=false then form1.image2.Visible:=true;
end;
procedure TForm1.Button4Click(Sender: TObject);
begin
if form2.RadioButton1.Checked then horizontalvideo;
if form2.RadioButton2.Checked then verticalvideo;
end;
procedure TForm1.Button5Click(Sender: TObject);
var
d:real;
begin
V:=strtofloat(edit5.text);
d:=(sqrt(w*w+h*h)/2);
F_s:= strtoint (form2.Edit3.Text);
```

```
F_l:= strtoint (form2.Edit4.Text);
BMP := TBitmap.Create;
if form2.RadioButton2.Checked=true then
begin
BMP.Width:= L; // ширина кадрів відео-файлу
form1.Image2.Width:=L;
end;
if form2.RadioButton1.Checked=true then
begin
BMP.Width := W; // відповідна ширина для Panel1
form1.Image2.Width:=W;
end;
BMP.Height := H; // відповідна висота для Panel1
SlitPos:=0;
i:=F_s; repeat
begin
MediaPlayer1.Position:=i;
PanelToImage;
form1.Edit2.Text:= inttostr (i);
SlitPos:=SlitPos+V;
begin
for j:=0 to H do if j<slitpos then
begin
form1.Image2.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,j+h div 2]:=
bmp.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,j+h div 2]; form1.Image2.Canvas.Pixels[-
round(sqrt(sqr(slitpos)-j*j))+w div 2,j+ h div 2]:=
bmp.Canvas.Pixels[-round(sqrt(sqr(slitpos)-j*j))+w div 2,j+h div 2];
form1.Image2.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+h div 2]:=
bmp.Canvas.Pixels[round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+h div 2]; form1.Image2.Canvas.Pixels[-
round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+ h div 2]:=
bmp.Canvas.Pixels[-round(sqrt(sqr(slitpos)-j*j))+w div 2,-j+h div 2];
end;
for j:=0 to H do if j<slitpos then
begin
form1.Image2.Canvas.Pixels[j+w div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[j+w div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]; form1.Image2.Canvas.Pixels[j+w
div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[j+w div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2]; form1.Image2.Canvas.Pixels[-j+w
div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[-j+w div 2,round(sqrt(sqr(slitpos)-j*j))+h div 2]; form1.Image2.Canvas.Pixels[-j+w
div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2]:=
bmp.Canvas.Pixels[-j+w div 2,-round(sqrt(sqr(slitpos)-j*j))+h div 2];
end;
end;
end;
i:=i+1;
until slitpos>d;
end;
procedure TForm1.Button3Click(Sender: TObject);
begin
sphericvideo;
end;
end.
```

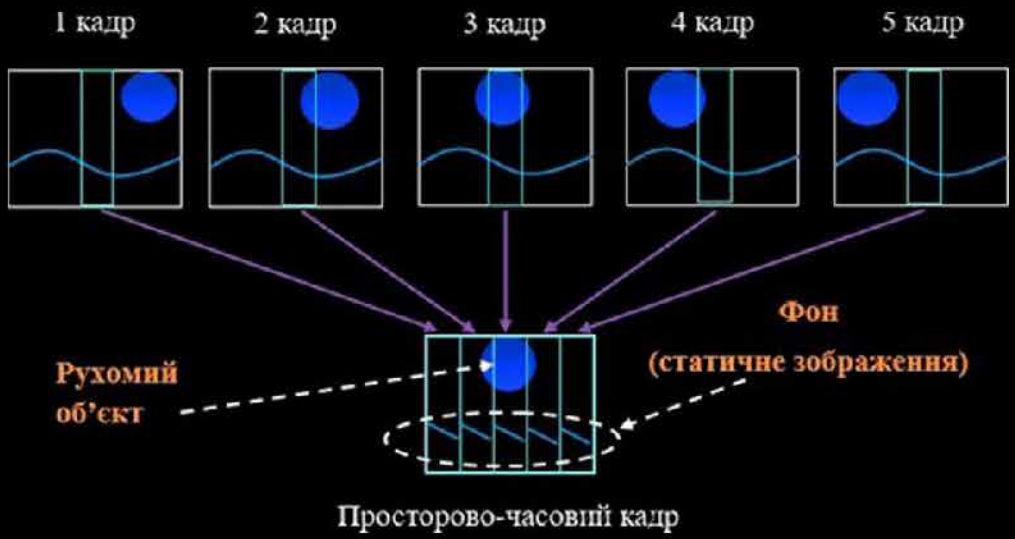
ДОДАТОК Б. Слайди мультимедійної презентації





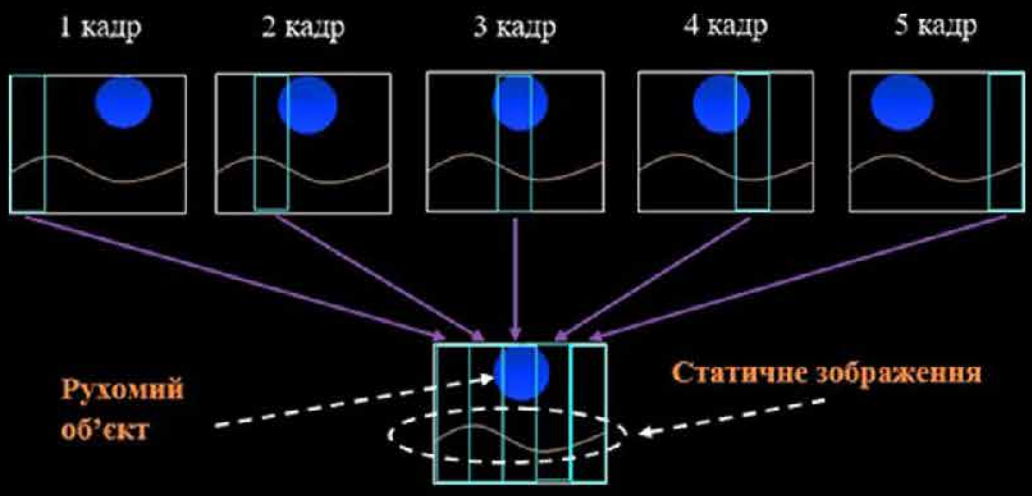
Структурна схема роботи застосунку для сканування пошкоджень відео-файлів

Схема обробки відео-кадрів, створених методом щілинної зйомки з вертикальною статичною щілиною



Зображення, створені за допомогою вертикальної щілинної зйомки з нерухомою щілиною

Схема обробки відео-кадрів, створених методом щілинної зйомки з вертикальною рухомою щілиною

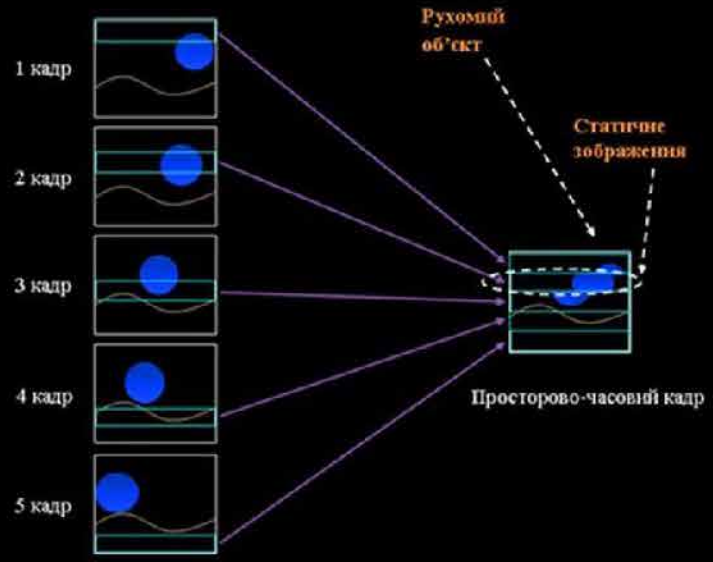


Результат щілинної зйомки при швидкості руху щілини, рівної 0.1

Результат щілинної зйомки при швидкості руху щілини, рівної 0.5

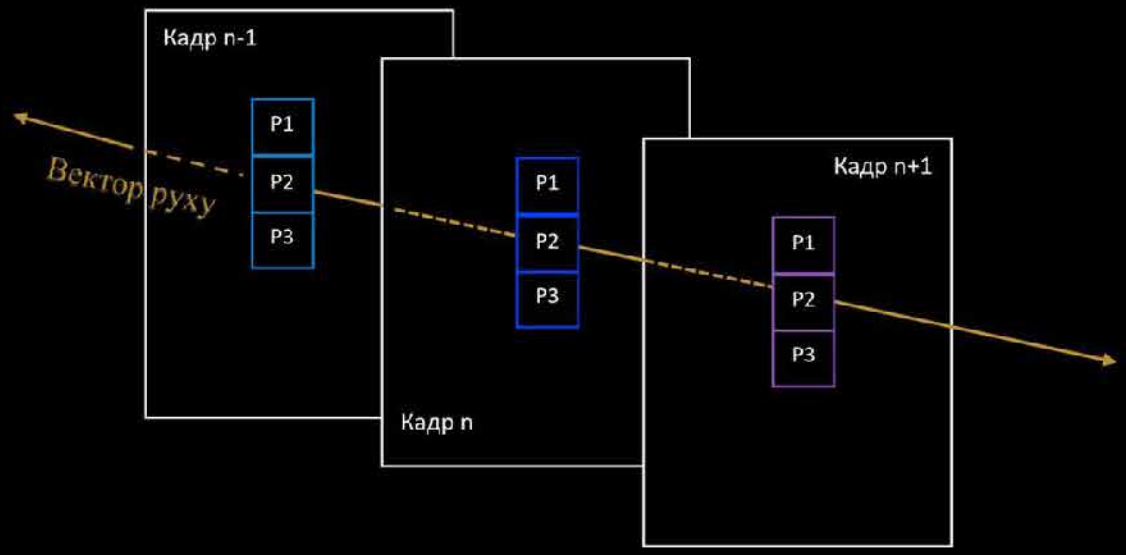
Результат щілинної зйомки при швидкості руху щілини, рівної 1

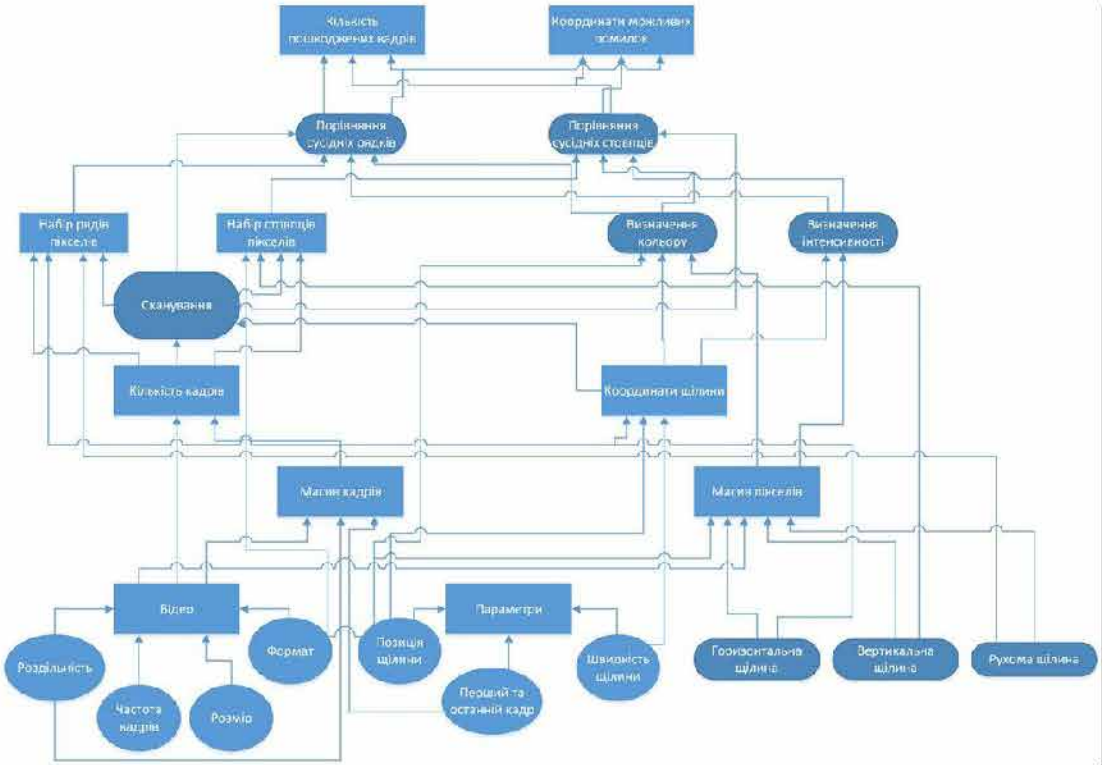
Схема обробки відео-кадрів, створених методом щілинної зйомки з горизонтальною рухомою щілиною



Результати щілинної зйомки з горизонтальною рухомою щілиною

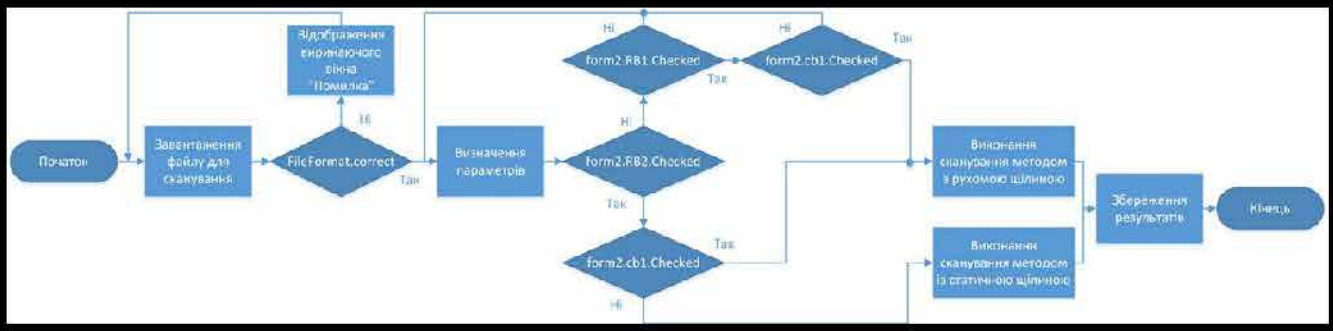
Загальна схема просторово-часового алгоритму обробки кадру



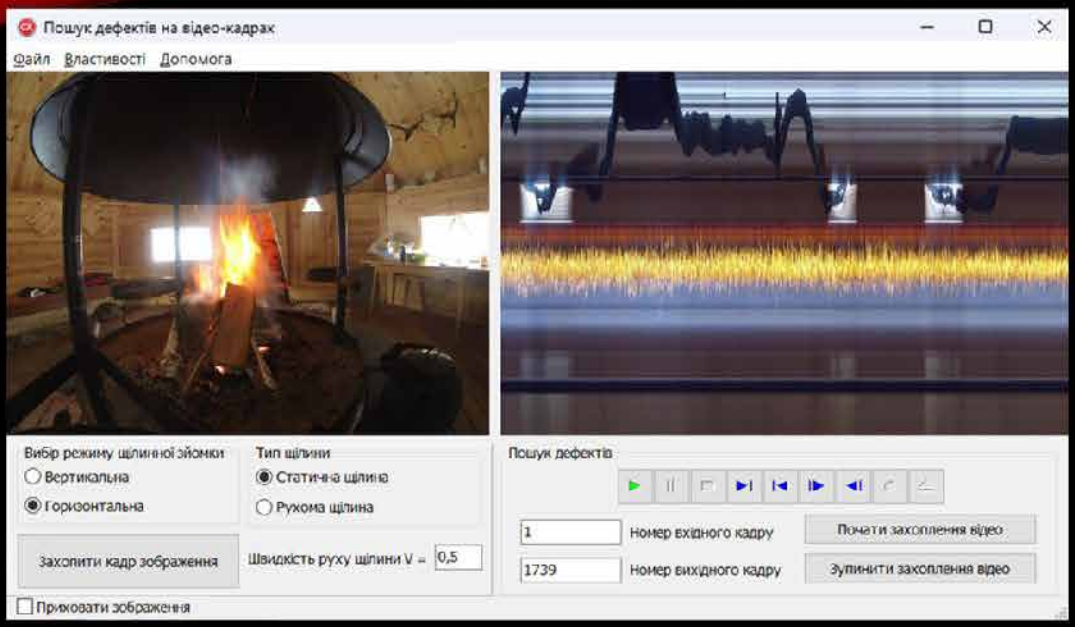


Блок-схема процедур та параметрів сканування пошкоджень відео-файлів

Блок-схема процесу сканування відео-файлу для пошуку дефектів

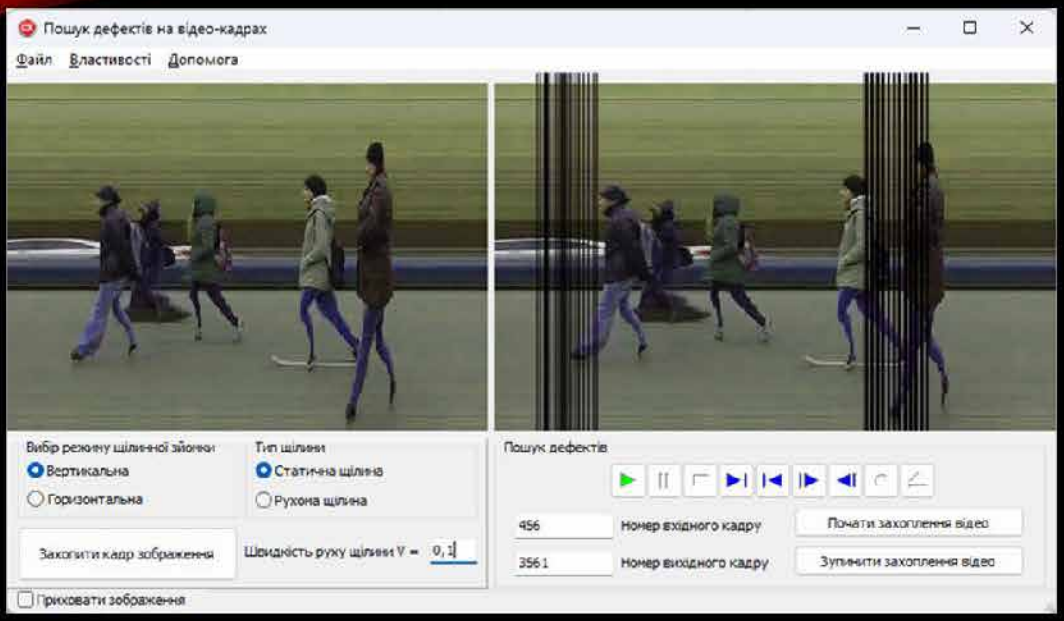


Інтерфейс головного вікна застосунку для пошуку дефектів на відео-кадрах

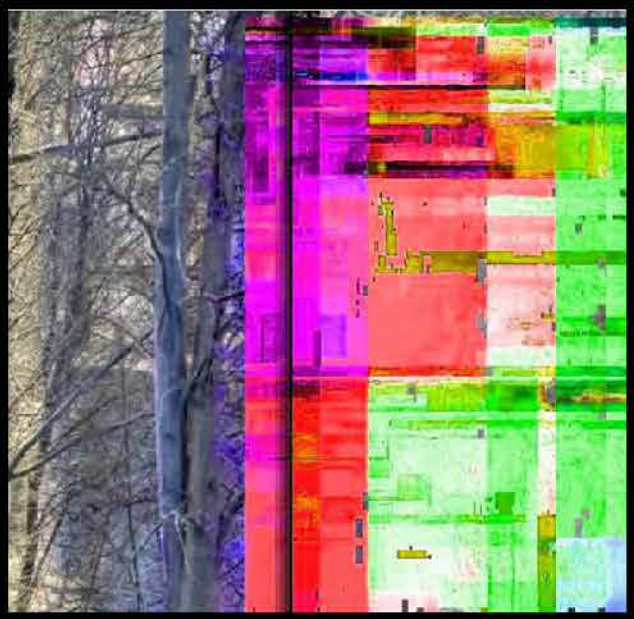


Ілюстрація визначення дефектів на відео-кадрах

Отриманий результат сканування пошкодженого відео-файлу



Отриманий результат сканування із значним пошкодженням відео-файлу



ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

ВІДГУК

керівника на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Соколову Матвію Євгеновичу

(прізвище, ім`я та по батькові)

Спеціальність: *123 «Комп'ютерна інженерія»*

Освітня програма: *«Комп'ютерна графіка і Web-дизайн»*

Тема дипломного проекту: *Програмна реалізація алгоритму сканування пошкоджень відео-файлів*

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ

а) обсяг і якість виконання проекту (графічного матеріалу і розрахунково-пояснювальної записки) *Дипломний проект виконано відповідно технічному завданню. Пояснювальна записка до дипломного проекту містить 74 сторінки. У пояснювальній записці виконано огляд та аналіз методів пошуку дефектів на відео-кадрах, розроблено алгоритмічне та програмне забезпечення, що реалізує пошук дефектів на відео-кадрах. Графічна частина складається з 17 слайдів, оформлених у вигляді презентації, передбачених технічним завданням. Якість виконання пояснювальної записки та слайдів добра.*

б) самостійність роботи над проектом: *Протягом виконання дипломного проекту здобувач освіти Соколов Матвій поступово та послідовно виконував всі етапи, проявив ініціативу в створенні загальної концепції та реалізації роботи. Всі роботи здобувач освіти виконував самостійно, з оглядом на рекомендації керівника.*

в) теоретична підготовка випускника (випускниці): *Здобувач освіти Соколов Матвій під час роботи над дипломним проектом вивчив достатньо багато літературних та інтернет-джерел за даною тематикою.*

Вважаю, що теоретична підготовка дипломника достатня і він готовий до захисту проекту.

г) вміння розв'язувати виробничі та конструкторські питання *Під час виконання дипломного проекту здобувач освіти Соколов Матвій показав вміння організовано працювати над поставленим завданням, застосовувати знання у галузі програмування та математики, розробляти, встановлювати та налаштовувати спеціалізоване програмне забезпечення, користуючись сучасними комп'ютерними програмними засобами, такими як Embarcadero RAD Studio, HandBrake, HBBatchBeast, Slit-Scan Movie Maker, Microsoft PowerPoint, After Effects, VideoCube та ін.*

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові керівника дипломного проекту *Шувалова Ірина Олегівна*

Місце роботи і посада керівника дипломного проекту *ВСП «Одеський технічний фаховий коледж ОНТУ», викладач циклової комісії комп'ютерних технологій та програмної інженерії*

Підпис

«10» 06 2024 р.

ВСП «ОДЕСЬКИЙ ТЕХНІЧНИЙ ФАХОВИЙ КОЛЕДЖ ОНТУ»

РЕЦЕНЗІЯ

на дипломний проект здобувача (здобувачки) освіти

відділення комп'ютерних систем

Соколову Матвію Євгеновичу

(прізвище, ім'я та по батькові)

Спеціальність *123 «Комп'ютерна інженерія»*

Освітня програма *«Комп'ютерна графіка і Web-дизайн»*

Керівник дипломного проекту (роботи) *Шувалова Ірина Олегівна*

(прізвище, ім'я та по батькові)

Тема дипломного проекту (роботи) *Програмна реалізація алгоритму сканування пошкоджень відео-файлів*

Обсяг розрахунково-пояснювальної записки *74* сторінок

Обсяг графічної (презентаційної) частини *17* аркушів (слайдів)

ХАРАКТЕРИСТИКА ДИПЛОМНОГО ПРОЕКТУ (РОБОТИ)

а) заключення про ступінь відповідності виконаного дипломного проекту завданню

Представлений на рецензію дипломний проект відповідає затвердженій темі та виконаний відповідно технічному завданню. Дипломний проект присвячений проблемі сканування пошкоджень відео-файлів та складається з пояснювальної записки, додатку з програмним кодом та мультимедійної презентації, що містить приклади роботи програми.

б) характеристика виконання кожного розділу дипломного проекту

Пояснювальна записка складається з основного розділу (аналізу предметної області, проектування застосунку, реалізації застосунку, тестування застосунку), економічного розділу, розділу охорони праці та додатків. Перелічені розділи поетапно охоплюють розробку, виконані докладно та обґрунтовано. Розділ охорони праці містить загальну інформацію та вимоги до техніки безпеки оператора КТ. Економічний розділ проекту містить розрахунок витрат на НДР та реалізацію проекту.

в) оцінка якості виконання пояснювальної записки та графічної частини дипломного проекту

Графічна частина складається з 17 слайдів мультимедійної презентації, виконаної у програмному продукті MS PowerPoint, які містять ілюстративні схеми, скріншоти роботи програмного застосунку, передбачені технічним завданням. Пояснювальна записка виконана акуратно та у відповідності до норм. Якість виконання графічної частини проекту та пояснювальної записки відмінна, розробку виконано у повному обсязі.

г) перелік позитивних якостей дипломного проекту *Реалізований алгоритм представляє практичний інтерес. Розглянутий метод дозволяє реалізувати ефективне сканування пошкоджень відеокадрів. Визначено та проаналізовано найбільш розповсюджені причини пошкодження відео-файлів та існуючі методи їх виявлення.*

д) основні недоліки дипломного проекту *1. Бажано було б реалізувати у програмі роботу з різними видами відеофайлів;*

2. Не зайвим було б розглянути додаткові приклади пошкоджень відеофайлів, де є різні артефакти

Оцінка розрахункової частини *Відмінно*

Оцінка графічної частини *Відмінно*

Загальна оцінка *Відмінно*

Прізвище, ім'я, по батькові рецензента *к.т.н. Селіванова Алла Віталіївна*

Місце роботи і посада рецензента *Одеський національний технологічний університет, декан факультету комп'ютерної інженерії, програмування та кіберзахисту*

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ ОДЕСЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНОЛОГІЧНИЙ УНІВЕРСИТЕТ ФАКУЛЬТЕТ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ, ПРОГРАМУВАННЯ ТА КІБЕРЗАХИСТУ

Підпис: _____

« 17 » червня 2024 р.

UNICHECK
by Turnitin

Ім'я користувача:
Катерина Григорівна Краснокутська

ID перевірки:
1016245645

Дата перевірки:
12.05.2024 20:40:32 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
12.05.2024 20:41:38 EEST

ID користувача:
100011688

Назва документа: 4КГ-07_Матвій_Соколов

Кількість сторінок: 53 Кількість слів: 9340 Кількість символів: 67724 Розмір файлу: 2.78 MB ID файлу: 1016030388

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

23.3%
Схожість

Найбільша схожість: 8.43% з Інтернет-джерелом (https://ela.kpi.ua/bitstream/123456789/34322/1/Kishka_bakalavr.pdf)

23.3% Джерела з Інтернету	443	Сторінка 55

Не знайдено джерел з Бібліотеки

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Замінені символи	28

Підозріле форматування	11 сторінок

ДОЗВІЛ
НА РОЗМІЩЕННЯ
ВИПУСКНОЇ КВАЛІФІКАЦІЙНОЇ РОБОТИ
(ДИПЛОМНОГО ПРОЕКТУ)
В ЕЛЕКТРОННОМУ РЕПОЗИТАРІЇ ВСП «ОТФК ОНТУ»

Ми, що нижче підписалися,

Соколов Матвій Євгенович,
здобувач освіти гр. 4КГ-07, та

Шувалова Ірина Олегівна,
керівник дипломного проекту,

не заперечуємо щодо розміщення електронного варіанту пояснювальної записки до дипломного проекту фахового молодшого бакалавра на тему:

«Програмна реалізація алгоритму сканування пошкоджень відео-файлів» (автор роботи – Соколов М.Є., керівник роботи – Шувалова І.О.)

виконаного у ВСП «Одеський технічний фаховий коледж Одеського національного технологічного університету» в 2024 році, у повному обсязі в електронному репозитарії ВСП «ОТФК ОНТУ» для вільного доступу через мережу Інтернет.

Несемо відповідальність за ідентичність електронного та друкованого варіантів випускної кваліфікаційної роботи і даємо згоду на обробку персональних даних.

Виконавець _____ / Соколов М.Є. /

Керівник _____ / Шувалова І.О. /

«10» червня 2024 р.