Міністерство освіти і науки України
Державний університет «Одеська політехніка»
Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Кордюк Вадим Володимирович,
студент групи РЗ-161

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

Модифікація алгоритму виявлення руху відеокамери
Спеціальність:
125 Кібербезпека

Керівник:
Лебедєва Олена Юріївна,
к.т.н., доцент

Одеса – 2021

Міністерство освіти і науки України
Державний університет «Одеська політехніка»

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій
Кафедра кібербезпеки та програмного забезпечення

Рівень вищої освіти другий (магістерський)
Спеціальність 125 – Кібербезпека
Освітня програма – Кібербезпека

ЗАТВЕРДЖУЮ
Завідувач кафедри КБПЗ
_____
д.т.н., проф. А.А. Кобозєва
____ _____2021р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Кордюку Вадиму Володимировичу

1.Тема роботи: *Модифікація алгоритму виявлення руху відеокамери*
керівник роботи *Лебедєва Олена Юріївна, к.т.н., доцент,*
затверджені наказом ректора Державного університету «Одеська політехніка» від „25” жовтня 2021 р. № 372-в .

2.Зміст роботи*: аналіз проблемної області, постановка задачі, модифікація алгоритму виявлення руху відеокамери, опис програмного продукту, охорона праці*

3. Перелік ілюстративного матеріалу: *слайди презентації.*

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
Охорона праці	Ярова І.А., доцент	Завдання видав	Завдання прийняв

6. Дата видачі завдання "_____"_____20___ р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання	Примітка
1	*Аналіз джерел з теми випускної кваліфікаційної роботи*	*01.09.2021*	*виконано*
2	*Обґрунтування вибору рішення. Збір даних*	*15.01.2021*	*виконано*
3	*Аналіз основних аспектів роботи з відео матеріалами*	*01.10.2021*	*виконано*
4	*Розробка модифікації алгоритму виявлення руху відеокамери*	*10.10.2021*	*виконано*
5	*Розробка програмного додатку, який реалізує модифікацію алгоритму виявлення руху відеокамери*	*17.10.2021*	*виконано*
6	*Підготовка тексту роботи*	*01.11.2021*	*виконано*
7	*Підготовка презентації та доповіді*	*12.11.2021*	*виконано*
8	*Попередній захист*	*26.11.2021*	*виконано*
9	*Нормоконтроль, рецензування*	*15.12.2021*	*виконано*
10	*Занесення роботи в електронний архів*	*18.12.2021*	*виконано*
11	*Допуск до захисту*	*20.12.2021*	*виконано*

Здобувач вищої освіти _____ *Кордюк В.В.*

Керівник роботи _____ *Лебєдева О.Ю.*

ЗАВДАННЯ

на розробку розділу “Охорона праці”

Кордюка Вадима Володимировича, група РЗ-161

Інститут інформаційної безпеки, радіоелектроніки та телекомунікацій

Кафедра кібербезпеки і програмного забезпечення

Тема роботи *Модифікація алгоритму виявлення руху відеокамери*

Зміст розділу:

1. Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів.
2. Аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях.
3. Дослідження психомоторних показників і оцінка сили нервової системи шляхом експрес-діагностики за методикою теппінг-тесту.

Керівник роботи	Консультант з охорони праці
_____ (Лебєдева О.Ю)	_____ (Ярова І.А.)
«____» _____ 2021 р.	«____» _____ 2021 р.

АНОТАЦІЯ

Кваліфікаційна робота на тему “Модифікація алгоритму виявлення руху відеокамери” на здобуття другого (магістерського) рівня вищої освіти за спеціальністю 125 – Кібербезпека, освітня програма – Кібербезпека містить 11 рисунків, 1 таблицю, 53 літературних джерел за переліком посилань. Робота виконана на 61 сторінках тексту.

Метою дипломної роботи є модифікація алгоритму виявлення руху відеокамери шляхом аналізу чотирьох блоків у кожному кадрі цифрового відео.

У роботі було розроблено модифікацію алгоритму виявлення руху відеокамери та реалізовано програмний додаток зі зручним інтерфейсом, за допомогою якого користувач може аналізувати обране відео, та з відкритим кодом. По завершенню роботи з обраним відео програма створює графік руху відеокамери. Розроблена модифікація алгоритму може бути використана як складова частина комплексних систем обробки відеофайлів для охоронних систем.

Були проведені експерименти та наводяться висновки про використання метрик для оцінки подібності блоків, що аналізуються.

ЦИФРОВЕ ВІДЕО, ЦИФРОВЕ ЗОБРАЖЕННЯ, ТРЕКІНГ, ВИЯВЛЕНЯ РУХУ КАМЕРИ

ANNOTATION

Qualification work on "Modification of the algorithm for detecting the movement of the video camera" to obtain the second (master's) level of higher education in 125 - Cybersecurity, educational program - Cybersecurity contains 11 pictures, 1 tables, 53 literature sources by reference. The work is done on 61 pages of text.

The aim of the thesis is to modify the algorithm for detecting the movement of the video camera by analyzing the four blocks in each frame of digital video.

The work developed a modification of the algorithm for detecting the movement of the video camera and implemented a software application with a user-friendly interface through which the user can analyze the selected video, and open source. After working with the selected video, the program creates a schedule of the camcorder. The developed modification of the algorithm can be used as an integral part of complex video file processing systems for security systems.

Experiments were conducted and conclusions were drawn on the use of metrics to assess the similarity of the blocks under analysis.

DIGITAL VIDEO, DIGITAL IMAGE, TREKKING, CAMERA MOVEMENT DETECTION

ЗМІСТ

ВСТУП

Від швидкого прогресу в технології автономних систем та засобів для відеоспостереження до більш повсякденних функцій допомоги оператору в сучасних системах безпеки, відеоспостереження майбутнього все більше покладається на досягнення в області комп’ютерного зору для більшої безпеки та зручності. У той же час постачальники транспортної інфраструктури та послуг розширюють свою залежність від комп’ютерного зору для підвищення безпеки та ефективності перевезень. Таким чином, комп’ютерний зір допомагає вирішувати критичні проблеми на багатьох кінцях нашого повсякденного життя — на рівні споживача, а також на рівні постачальника інфраструктури.

Досягнення в області комп’ютерного зору відіграють вирішальну роль у вирішенні проблем в системах безпеки все більш ефективними способами. Привабливість комп’ютерного зору для цих цілей передусім пов’язана з економічною ефективністю цих технологій, а також із широким спектром застосувань, які може підтримувати комп’ютерний зір.

Розширені системи допомоги водієві (ADAS) розгортаються в постійно зростаючій кількості, але в міру зростання обчислювальної потужності в комп'ютерних системах і в міру того, як зв'язок між засобом безпеки і інфраструктурою стає більш надійним, ці системи почнуть змінювати свою роль з надання допомоги, для полегшення прийняття рішень, оскільки це стосується безпеки.

Крім того, необхідні нові й оптимізовані підходи до обробки відео, щоб об’єднати інформацію, отриману камерами. Виявлення руху відеокамери є однією з основ обробки та аналізу відео, і зазвичай виконується шляхом оцінки моделі фону сцени, а потім будь-яке відхилення від моделі фону розглядається як рухомий кадр. Моделі виявлення руху відеокамери є дуже привабливими, оскільки дозволяють точно оцінити фонову модель за наявності змінних фону та поступової зміни освітлення.

Аналіз останніх публікацій показав, що для більшості сучасних розробок, пов’язаних з рішенням задач машинного зору залишається проблема з отриманням та обробкою візуальної інформації, тому тема роботи є актуальною.

Метою роботи є розробка модифікації алгоритму виявлення руху відеокамери шляхом аналізу чотирьох блоків у кожному кадрі відео.

Для досягнення поставленої мети необхідно вирішити наступні задачі:

– Проаналізувати стан сучасних розробок в області трекінгу;
– Проаналізувати та виявити слабкі місця розробленого алгоритму виявлення руху відеокамери;
– Розробити основні кроки модифікації алгоритму виявлення руху відеокамери;
– Розробити програмний продукт для розробленої модифікації алгоритму виявлення руху відеокамери; провести експерименти, спрямовані на порівняння показника візуальної якості трекінгу.

Під показником візуальної якості трекінгу будемо розуміти 2D-зображення вектору руху камери.

Об'єкт дослідження – оцінка руху відеокамери в цифровому відео.

Предмет дослідження – методи виявлення руху відеокамери.

Наукова новизна одержаних результатів полягає в наступному. Визначено слабкі місця попередньо розробленого алгоритму виявлення руху відеокамери. Було розроблено модифікацію алгоритму виявлення руху відеокамери шляхом аналізу чотирьох блоків у кожному кадрі відео. За результатами експериментів наводяться висновки про використання метрик для оцінки подібності блоків та кількості блоків, що аналізуються.

Практичне значення отриманих результатів. Практична цінність роботи полягає в розробці та реалізації програмного додатку для виявлення руху відеокамери зі зручним інтерфейсом та відкритим кодом. Розроблений алгоритм може бути використаний як складова частина комплексних систем відеоспостереження.

У вступі обумовлена актуальність теми, сформульована мета, задачі, об’єкт та предмет досліджень.

В першому розділі розглянуті основні поняття та визначення що до вибраної теми, розглянуті існуючі програмні засоби які дозволяють вирішувати таке завдання, та була визначена актуальність створення та модифікації даного алгоритму.

В другому розділі розглянуті поняття трекінгу відео, його види та існуючі алгоритми. Розглянуто створений алгоритм виявлення руху відеокамери.

В третьому розділі розглянуто створений програмний додаток та основні моменту роботи з інтерфейсом програми. Наведена якісна оцінка ефективності модифікованого алгоритму, та проведено порівняння показника візуальної якості трекінгу.

В четвертому розділі проведено аналіз умов праці і вибір заходів і засобів захисту від небезпечних виробничих факторів, аналіз техногенних небезпек і вибір заходів і засобів повного забезпечення безпеки у надзвичайних ситуаціях, визначення та необхідна кількості первинних засобів пожежогасіння.

В висновках описується отримані результати роботи.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Основні поняття

Питання безпеки будинку, офісу, підприємства дуже актуальне у сучасному світі. У зв'язку з цим виникає життєва важлива потреба в охоронних підприємствах, службах безпеки та охоронцях. Але технічний прогрес не стоїть на місці, і на допомогу охоронцям надходить відеоспостереження. Системи відеоспостереження стають невід'ємною частиною охоронних комплексів. Завдяки таким системам охорона може своєчасно відреагувати на дії зловмисників та припинити їх. За останнє десятиліття відеоспостереження набуло широкого поширення і сьогодні залишилося мало об'єктів, не обладнаних системами відеоспостереження.

Відеоаналітика – технологія, що використовує методи комп'ютерного зору для автоматизованого отримання різних даних на підставі аналізу послідовності зображень, що надходять із відеокамер у режимі реального часу або з архівних записів. Відеоаналітика є програмним забезпеченням (ПЗ) для роботи з відеоконтентом. В основі програмного забезпечення лежить комплекс алгоритмів машинного зору, що дозволяють вести відеомоніторинг та проводити аналіз даних без прямої участі людини. Алгоритми відеоаналітики можуть бути інтегровані в різні бізнес-системи, які найчастіше використовуються у відеоспостереженні та інших сферах безпеки.

Відстеження відео – це процес виявлення об'єкта, що рухається, з часом за допомогою камери. Він має безліч застосувань, у тому числі: взаємодія людини з комп'ютером, безпека та спостереження, відеозв'язок і стиснення, доповнена реальність, управління трафіком, отримання медичних зображень і т. д. Відстеження відео може зайняти багато часу через велику кількість даних, що містяться у відео. Ще більше ускладнює ситуацію можлива необхідність використання методів розпізнавання об'єктів для відстеження, що є складним завданням.

Мета відстеження відео – пов'язати цільові об'єкти у послідовних відеокадрах. Асоціація може бути особливо складною, коли об'єкти швидко рухаються щодо частоти кадрів . Ще одна ситуація, що ускладнює проблему, - це коли об'єкт, що відстежується, з часом змінює орієнтацію. Для цих ситуацій системи відеоспостереження зазвичай використовують модель руху, яка описує, як зображення мети може змінитися при різних можливих рухах об'єкта.

Трекінгом називається визначення розташування об'єкта, що рухається в часі за допомогою камери. Алгоритм аналізує кадри відео і видає положення цільових об'єктів, що рухаються, відносно кадру.

Основна проблема в трекінгу полягає у зіставленні положень цільового об'єкта на послідовності кадрів, якщо об'єкт рухається швидко щодо частоти кадрів. Таким чином, системи трекінгу зазвичай використовують модель руху, яка описує як може змінюватися зображення цільового об'єкта при різних його рухах.

1.2 Підходи до оцінки руху

У різних публікаціях запропоновано такі підходи до оцінки руху: рекурсивні растрові алгоритми, алгоритми частотної області, алгоритми оптичного потоку та методи відповідності блоків. Наведемо їх короткий порівняльний аналіз.

Рекурсивні растрові алгоритми використовують ітеративне вдосконалення оцінки руху для окремих елементів растрових зображень, які виконуються градієнтними методами, для передбачення рекурсивного усунення кожного елемента растру від його сусідніх елементів. Ці алгоритми мають більшу обчислювальну складність і менше піддаються налаштуванню, тому їх важко використовувати в апаратних засобах. Частотні методи оцінки руху застосовуються для глобальної оцінки руху кадру (Global Motion). Найбільш відомим із цих методів є фазовий метод кореляції на основі перетворення Фур'є. Існуючі алгоритми визначення глобального руху кадру використовуються як для

стиснення, так і для обробки відеоматеріалів, наприклад, задачах побудови панорамного зображення або стабілізації відео з тремтіння кадру.

Алгоритми визначення глобального руху кадру можна розділити на три групи:

– Алгоритми, які використовують особливі точки (Feature Points), наприклад, алгоритм 2D Ridge Motion;
– Алгоритми, які використовують вектори руху (Motion Vector);
– Алгоритм глобального пошука.

Алгоритми з кожної групи мають характерні сильні і слабкі сторони. Так, застосування апарату особливих точок потребує значних тимчасових витрат на вибір спеціальних точок (Feature Points Selection) та на їх відстеження (Feature Points Tracking). Перевагою цих алгоритмів є висока надійність визначення зсувів.

Використання векторів руху (Motion Estimation) дає нижчу надійність визначення зсувів, ніж особливі точки. Але алгоритми визначення векторів руху мають велику перевагу – високу швидкість.

Під алгоритмами глобального пошуку розуміються алгоритми, що визначають глобальний рух по всьому кадру без застосування проміжних перетворень. Методи оптичного потоку гарантують високу точність для сцен з невеликими усуненням, але допускають збої, коли усунення є великими. Крім того, ці методи страждають від апертурної проблеми, оскільки кожна околиця пікселів може мати різний рух на зображенні.

Алгоритми відповідності блоків оцінюють рух на підставі прямокутних блоків та обчислюють один вектор руху для кожного блоку. Ці алгоритми є найбільш підходящими для такої апаратної реалізації через їх продуману модифікацію і простоту.

У базовому методі порівняння блоків (Block Matching Algorithm, BMA) кожний кадр розділений на блоки, що складаються з локальних блоків інтенсивності та кольоровості. Зазвичай для ефективного кодування оцінка руху виконується тільки на кожному блоці яскравості. Для кожного блоку на поточному кадрі здійснюється пошук відповідного, тобто найменш спотвореного:

блоку на наступному кадрі та записується його зміщення (або вектор руху). Зазвичай кодується різниця між поточним та наступним кадром. Таким чином, замість вихідного блоку інтенсивності можуть бути передані вектор руху та величина помилки, що вийшла, та забезпечує стиснення даних та усунення між кадрової надмірності. Підсумовування кадру та вектора руху дає точну копію слідуючого кадру. Однак, яскравість сцени може змінюватися за часом, тому деякі моделі враховують зміну яскравості. Подібне коригування може покращити результат стабілізації, якщо не приймати до уваги таких локальних змін яскравості, як тіні.

Деякими авторами пропонується статистичний метод оцінки руху, заснований на моделюванні блоків зображення з використанням розподілу Гауса. У цьому випадку оптимізація виконується за допомогою алгоритму максимізації очікування (ЕМ-алгоритму), заснованого на ітеративної оптимізації параметрів моделі (апріорної ймовірності, векторів значень і ко варіативних матриць), і розширеного відстані Махаланобіса, що застосовується для оцінки відповідності між блоками та пошуку найближчих блоків на сусідніх кадрах.

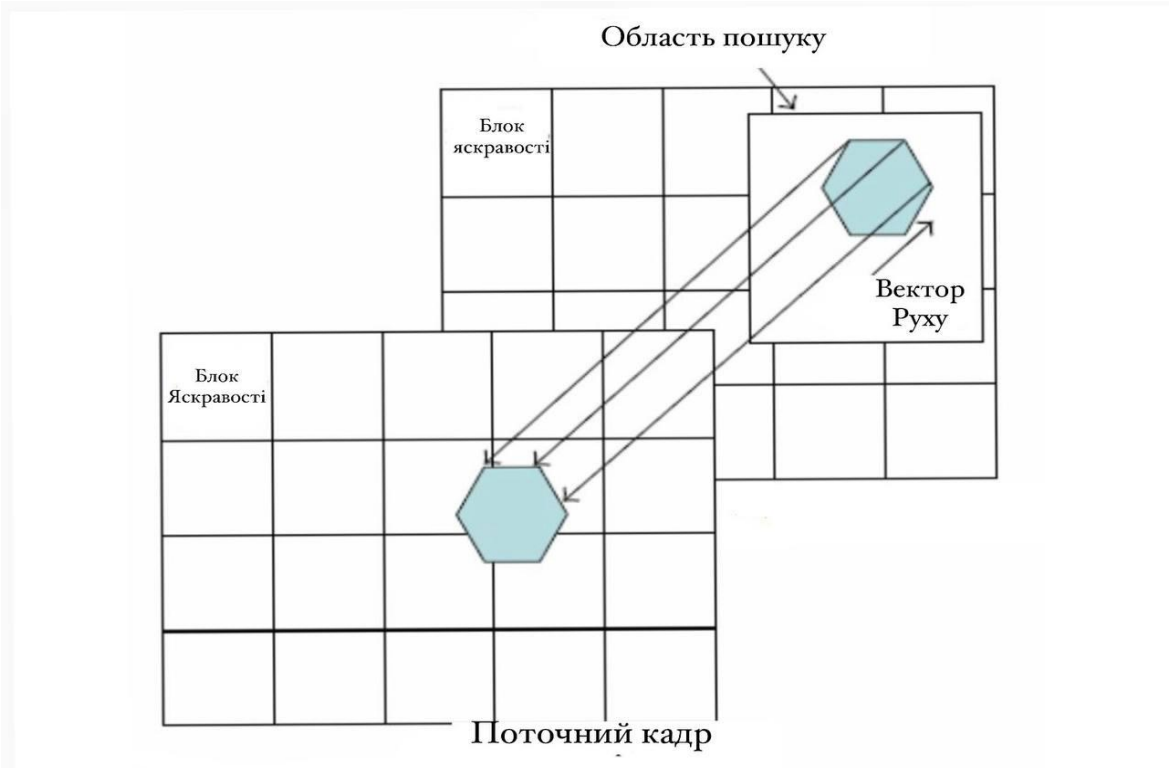


Рисунок 1.1 – Схема реалізації методу відповідності блоків

Розглянемо основну ідею алгоритму стабілізації відеоматеріалів на основі модифікованого методу відповідності блоків.

Під час розгляду відео послідовності, містить об'єкти, що рухаються, вектор зміщення кожного об'єкта в площині зображення оцінюється методом повного пошуку відповідності блоки (FSBMA). Поточний кадр розділений на матриці – макроблоки, які включають відповідний блок і його найближчих сусідів на попередньому кадрі. Це дозволяє створити вектор руху макроблок від одного місця до іншого на попередньому кадрі. Таке зміщення, обчислене всіма макроблоками представляє оцінку руху на поточному кадрі. Область пошуку для відповідності макроблоку обмежена до p пікселів (де p є пошуковим параметром) на всі боки на попередньому кадрі. Збільшення значення p дозволяє знаходити більший рух, але робить процес пошуку руху найбільш ресурсоємним.

Відповідність одного макроблока іншому полягає в виведенні функції вартості. При цьому макроблок з найменшим значенням функції вартості відповідає поточному блоку. Функція вартості визначається розширеним відстанню Махаланобіса для зважених компонентів розподіл Гауса.

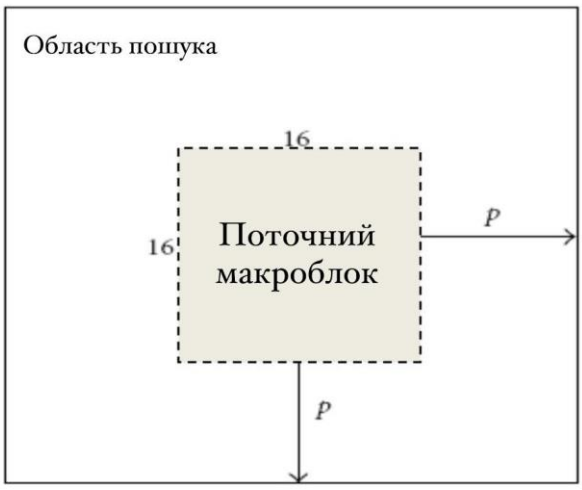


Рисунок 1.2 – Область пошуку розмірів 16 пікселів і зсуву в *p* пікселів

Ця відстань поділена на гаусові інтервали між попереднім та поточними блоками, компоненти сильних, середніх та слабких ваг. Область пошуку розмірів 16 пікселів і зсуву в p пікселів. При моделюванні сумішшю двох розподілів Гауса функція вартості визначається розширеною відстанню Махаланобіса між компонентами сильних (d1) та слабких ваг (d3).

Розглянемо методику блокової оцінка руху. Схема блокової оцінки руху містить наступну послідовність дій. Зображення ділиться на не перекриваються блоки пікселів, значення інтенсивності яких визначається як $□_□(□, □)$, де $□, □$ – координати пікселя, n –номер кадру. Для кожного блоку в невеликій околиці $−□□ < □□ < +□□$ та $−□□ < □□ < +□□$ шукаються найбільш схожі блоки на наступному кадрі $□_□ + 1(□ + □□, □ + □□)$.

Подібність блоків визначається мінімізацією функції помилки e, відповідно до використовуваної метрикою. Зазвичай застосовуються три метрики: сума абсолютних різниць (SAD), сума квадратичних відхилень (SSD) та середнє значення квадратів різниць (MSD):

$$□□□(□_□, □_□) = \sum_{□=1}^{□} \sum_{□=1}^{□} |□_{□+1}(□, □) − □_□(□ + □_□, □ + □_□)|$$

$$□□□(□_□, □_□) = \sum_{□=1}^{□} \sum_{□=1}^{□} (□_{□+1}(□, □) − □_□(□ + □_□, □ + □_□))^2$$

$$□□□\left(□_□, □_□ = \frac{1}{□_{□□□} * □_{□□□}}\right)$$

$$\sum_{□=1}^{□} \sum_{□=1}^{□} (□_{□+1}(□, □) − □_□(□ + □_□, □ + □_□))^2$$

де $□_{□□□}$– кількість аналізованих блоків. Результатом даного етапу є розраховані для кожного блоку зображення локальні вектори рухи (LMV).

Після знаходження локальних векторів руху потрібно визначити, чи вони описують рух камери або деяких об'єктів у кадрі. Для цього будується нечітка

модель, аналогічна описаної у роботі, на виході якої визначається міра достовірності векторів руху.

Для кожного локального вектору руху розраховується міра помилки двома способами:

– Евклідова відстань (величина вектору) $E' = (e1, e2, \dots, en)$, де $1 - n$ – номер вектору;
– Кут нахилу вектору щодо горизонталі $C' = (c1, c2, \dots, cn)$.

Проводиться розрахунок середньої помилки □□ − □□□□□□ □□□□□. Для кожного елемента □′, □′ виконується розрахунок помилки відхилення:

$$□□ = □□/□□, □□ = □□/□□ \ ,$$

де □□ та □□ – медіанні значення □′ і □′ відповідно.

Для покращення оцінки руху застосовуються сигмоїдальні функції приналежності, що дозволяє більш точно відокремити рух об'єктів у кадрі від усунення камери. При не професійній зйомці на відео послідовності часто виникають шуми різного виду, які погіршують можливість оцінки руху та подальшої стабілізації. Тому запропоновано використовувати різні параметри для не зашумлених та зашумлених відео послідовностей. Помилка відхилення di подається на вхід функції Приладдя, її значення відображається на різних класах точності: високий, середній, низький. Нижчі значення похибки відображаються на найкращому класі, високі значення похибки – на гіршому. Будуються сигмоїдної функції приналежності, представлений вираз для розрахунку значень кращого класу:

$$f(x; a, b) = \begin{cases} 1, x \le a, \\ 1 - 2\left(\frac{x-a}{b-a}\right)^2, a \le x \le \frac{a+b}{2}, \\ 2\left(\frac{x-b}{b-a}\right)^2, \frac{a+b}{2} \le x \le b, \\ 0, x \ge b. \end{cases}$$

Рекомендується використовувати значення □ = *0,5* та □ = *1,5* для опису класу «високий» не зашумлених відео послідовностей і □ = *0,75* та b=1,75 для зашумлених відеопослідовностей. Застосовується модель нечіткого висновку. Сугено-Канга , щоб визначити якість вектора. Модель нечіткого виведення TSK досить проста, тому що вона є компактним та обчислювально ефективним уявленням, яке може бути реалізовано з використанням адаптивних методів для побудови нечітких моделей. Представлені чотири різні нечіткі набори: відмінний, добрий, середній, поганий. Кожному з чотирьох класів відповідає константа: (*1,0.75,0.5,0*) для не зашумлених та (*1,0.85,0.65,0*) – для зашумлених відео послідовностей. Вихідне значення визначається на основі обох вхідних даних відповідно до мінімального значення.

Результат нечіткої моделі – це міра правдоподібності вектору руху, що знаходиться в діапазоні [0; 1]. Налаштування параметрів функції приналежності можна регулювати, яким чином значення помилки □ впливають на вихідне значення правдоподібності вектора. Коли обчислені заходи правдоподібності векторів, вибираються лише найкращі 60% значень, які передаються на вхід методу розрахунку параметрів стабілізації руху.

Глобальний рух між сусідніми кадрами можна оцінити за допомогою двовимірної лінійної моделі, яка є гарним компромісом між точністю і обчислювальною складністю. Ця модель описує міжкадровий рух за допомогою чотирьох різних параметрів, а саме: два напрями руху, кут повороту та коефіцієнт збільшення, які пов'язують координати точки (□□, □□) у кадрі з точкою (□□, □□) у наступному кадрі перетворенням:

$$\begin{cases} x_f = x_i \lambda cos\Theta - y_j \lambda sin\Theta + T_x, \\ y_f = x_i \lambda sin\Theta + y_j \lambda cos\Theta + T_y \end{cases}$$

де λ – параметр збільшення; θ – кут повороту; □□, □□ – модулі вектора руху за напрямками □, у відповідно.

Оскільки вектори руху, отримані з фону зображення, повинні бути дуже схожі за величиною та напрямком, використовується механізм кластеризації для класифікації поля руху на два кластери, що відповідають фону та передньому плану зображення. Глобальний рух, викликаний рухом камери, оцінюється у процесі кластеризації, що складається з двох етапів.

Перший етап – побудова гістограми H, яка містить достовірні локальні вектор руху (*Valid LMV*). Значення □□,□ збільшується на одиницю щоразу, коли зустрічається локальний вектор із відповідними координатами □□□(□, □).

Другий етап – вибір значення глобального вектора руху. За винятком випадку, коли у сцені переважає великий об'єкт, що рухається, блок кластера, що відповідає фону зображення, має максимальне кількість голосів. Максимальне значення цього кластера вибирається як глобальний вектор руху (*GMV*).

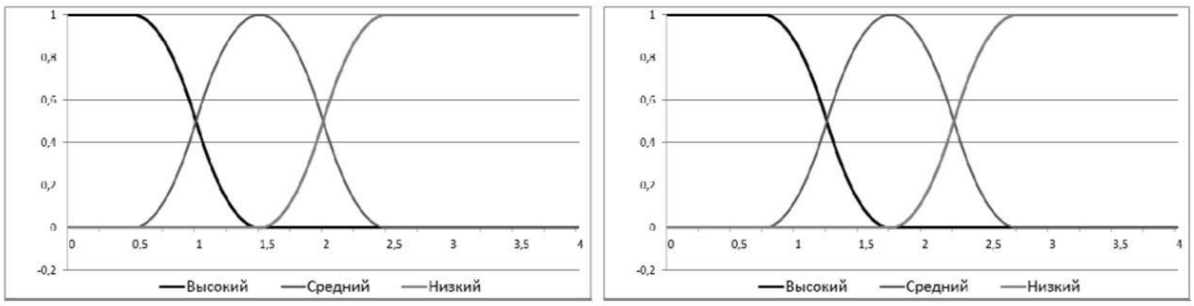


Рисунок 1.3 – Сигмоїдна нечітка функція приналежності для не зашумлених та зашумлених відео послідовностей.

2. РОЗРОБКА МОДИФІКАЦІЇ АЛГОРИТМУ ВИЯВЛЕННЯ РУХУ КАМЕРИ.

2.1 Цифрові відео та зображення та їх представлення

Відео – електронна технологія формування, запису, обробки, передачі, зберігання і відтворення рухомого зображення, заснована на принципах телебачення, а також аудіовізуальний твір, записаний на фізичному носії (відеокасеті, карті пам'яті тощо) [1].

Відеозапис – електронна технологія запису візуальної інформації, представленої у формі відеосигналу або цифрового потоку відеоданих, на фізичний носій з метою збереження цієї інформації і можливості подальшого її відтворення і відображення на пристрої виведення [1].

Існує два основні класи форматів запису відеозображення: аналогові і цифрові [2].

Аналогові формати для відео зйомки історично розроблялися на основі телевізійного стандарту. Цей сигнал є складеним (композитним) і утворюється в результаті складання сигналу яскравості і двох модульованих цветоразностных сигналів. Останні два називають сигналом колірності. Із-за об'єднання цих елементів в одному сигналі якість композитного відео далеко від досконалості. В результаті ми маємо неточну передачу кольору, недостатньо «чисту» картинку і інші чинники втрати якості.

Якість цифрових відеоматеріалів перевищує якість аналогових, цифрові записи не «старіють», вимоги до майстерності оператора при зйомці цифровою камерою нижчі, а самим, мабуть, головною перевагою цифрових форматів є зручність монтажу знятого матеріалу.

Цифрове відео є впорядкованим набором зображень, або кадрами [3]. Саме тому стосовно нього часто використовують термін відеопослідовність. При послідовному виведенні цих кадрів на екран створюється імітація «живого» зображення. Подібний принцип використовується в кінематографії. Для здобуття злитого, плавного руху без ривків, використовується особливість людського зору. Людський зір володіє інертністю, тобто око продовжує бачити предмет ще деякий

час після того, як предмет перемістився або зник з поля зору в інше місце. Таким чином, якщо змінювати кадри на екрані з достатньою швидкістю, людина бачитиме наступний кадр в той час, як попередній ще не потухнув остаточно. Швидкість зміни кадрів вимірюється в кадрах за секунду (fps – frames per second). Чим більше значення швидкості, тим більше якісним і «живим» вийде відеозображення.

Для здобуття більш-менш якісного відеозображення в реальному часі досить переглядати відеофрагмент з швидкістю не менше 16 fps. Світовим стандартом в кінематографії є 24 fps. Для відеосистем найбільш поширеними є значення 25 fps для відеостандарту PAL (Phase Alternation Line – Західна Європа) і 30 fps для відеостандарту NTSC (National Television System Committee – США).

Розглянемо основні стандарти стискування цифрового відео [4].

MPEG – один з основних стандартів стискування. Абревіатура MPEG (Moving Pictures Expert Group) – це назва міжнародного комітету, даного стандарту стискування, що займається розробкою. Різновиди MPEG: MPEG-1, MPEG-2, MPEG-4.

MPEG-1 – формат стискування для компакт-дисків (CD-ROM). Якість відео – як в звичайного відеомагнітофона, дозвіл 352x240, диск з фільмом в такому форматі зазвичай позначається VCD (VideoCD).

MPEG-2 – формат для DVD-дисков, цифрового телебачення. У цьому форматі знімають відео DVD-, HDD-, Flash-камери.

MPEG-4 – це формат, що отримується за допомогою відомих кодеків DIVX, XVID, H.264 та інші Зменшує відеопотік ще сильніше, ніж MPEG-2, але картинка ще пристойної якості, тому цей формат підтримує більшість сучасних DVD – плеєров. Особливо потрібно відзначити високу якість відео, стислого кодеком останнього покоління H.264.

HD (High Definition) – формат високого дозволу, новий формат особливої чіткості зображення. Має два різновиди: HD1 з дозволом 1280x720 і HD2 - 1440x1080.

Розглянемо найбільш популярні та часто використані формати відео файлів [4].

AVI (Audio-Video Interleaved) – це розширення величезної кількості відеофайлів, але не є форматом або кодеком. Це контейнер, розроблений Microsoft, в якому можуть зберігатися потоки 3-х типів - відео, аудіо і текст. У цей контейнер може входити відео будь-якого формату від mpeg1 до mpeg-4, звуки різних форматів, також можливе будь-яке поєднання кодеків.

WMV (Windows Media Video) – це формат від Microsoft, у ньому зберігаються відеоролики, зроблені за допомогою програми Movie Maker.

MOV – формат Apple Macintosh QuickTime, може містити окрім відео також графіку, анімацію, 3D. Найчастіше для програвання цього формату потрібний QuickTime Player.

MKV (Matroska video) – теж контейнер, який може містити відео, аудіо, субтитри, меню і ін. Може містити в собі аудіодоріжки на різних мовах, чого не допускає AVI, також він може зберігати в собі інформацію про глави відео, меню і так далі – загалом, всіх функцій DVD. Окрім цього з файлу у форматі mkv дуже легко витягнути аудио- і відеодані, для цього не потрібні спеціальні редактори. Реалізований більш абсолютно, ніж AVI – при відтворенні великих файлів відсутні пригальмовування і підвисання, особливо помітні на не дуже швидких комп'ютерах.

Зображення – об'єкт, образ, явище, в тій або іншій мірі подібне, але не ідентичне змальовуваному або сам процес їх створення.

Під зображенням розуміється інформація, організована у вигляді деякої квадратної числової матриці, відтворююча властивості змальованого об'єкту (сцени) і деформації, які пов'язані із способом і процесом здобуття зображення.

Виходячи з особливостей інформаційної надмірності зображень, можна виділити наступні основні класи зображень [5].

Клас 1. Монохромні зображення (відскановані креслення і текстові документи). Містять великі області одного кольору, контекстні залежності між

сусідніми пікселями, часто кількість пікселів одного кольору сильно перевищує кількість пікселів іншого.

Клас 2. Кольорові зображення з невеликим числом кольорів з певної палітри (ділова графіка, деякі види мультиплікації). Містять великі області одного кольору, контекстні залежності між сусідніми пікселями, часто кількість пікселів одного кольору сильно перевищує кількість пікселів іншого.

Клас 3. Зображення в градаціях сірого (рентгенівські знімки, чорно-білі фотографії). Значення яскравості сусідніх пікселів зазвичай мало відрізняються. Характерна плавна зміна яскравості між різними ділянками зображення.

Клас 4. Повнокольорові зображення (кольорові фотографії, фотореалістичні тривимірні сцени). Значення колірних компонент сусідніх пікселів зазвичай мало відрізняються. Характерна плавна зміна значень колірних компонент між різними ділянками зображення. Крім того, є області, неоднорідні по яскравості, але що мають однаковий відтінок і насиченість кольорів.

Надалі у нашій роботі ми будемо розглядати зображення четвертого класу, тобто повнокольорові зображення.

2.2 Колірні моделі

Колірна модель – це метод для визначення кольорів. Колірні моделі розташовані в тривимірній системі координат, утворюючі колірний простір. Існують різні кольорі моделі. Розглянемо найбільш поширені в області обробки зображень [6].

CIE XYZ – лінійна 3-компонентна колірна модель, що ґрунтується на результатах виміру характеристик людського ока. Побудована на основі зорових можливостей «стандартного спостерігача», тобто гіпотетичного глядача, можливості якого були ретельно вивчені і зафіксовані в ході тривалих досліджень людського зору, проведених комітетом CIE (фр. Commission Internationale de l'Eclairage).

Комітет CIE провів безліч експериментів з величезною кількістю людей, пропонуючи їм порівнювати різні кольори, а потім за допомогою сукупних даних цих експериментів побудував так звані функції відповідності кольорів (color-matching functions) і універсальний колірний простір (universal color space), в якому був представлений діапазон видимих кольорів, характерний для середньостатистичної людини.

Функції відповідності кольорів – це значення кожної первинної складової світла – червоною, зеленою і синій, які мають бути присутніми, аби людина з середнім зором могла сприймати всі кольори видимого спектру. Цим трьом первинним складовим були поставлені у відповідність координати X, Y і Z. Для графічного зображення моделі CIE XYZ використовують X і Y, а Z рахують як 1 – X – Y (рисунок 2.1).

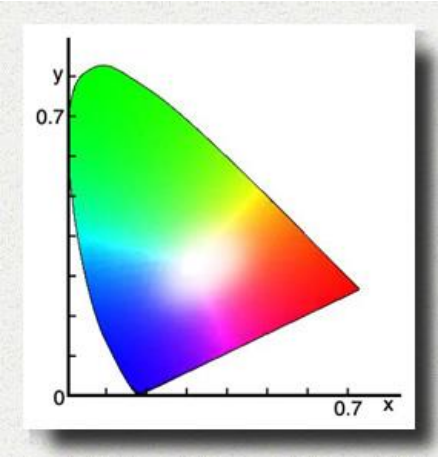


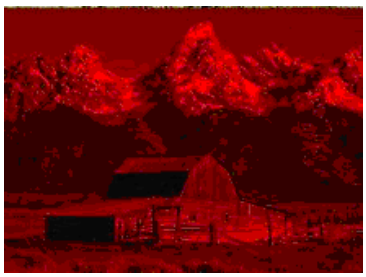
Рисунок 2.1 – Графічне представлення колірної моделі CIE XYZ

Основна властивість цій системи – позитивна визначеність, тобто будь-який фізично відчутний колір представляється в системі XYZ лише позитивними величинами. З іншого боку, не всім крапкам в просторі XYZ відповідають реальні кольори.

Зображення в колірній моделі RGB складається з трьох каналів (рисунок 2.2). Основними кольорами в цій моделі є червоний, зелений та синій.

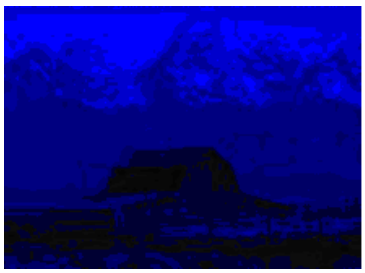
При змішуванні основних кольорів – наприклад, синього (B) і червоного (R), ми отримуємо пурпурний (M magenta), при змішуванні зеленого (G) і червоного (R) – жовтий (Y yellow), при змішуванні зеленого (G) і синього (B) – ціаністий (З cyan). При змішуванні всіх трьох колірних компонентів ми отримуємо білий колір (W).

а)



б)

в)



г)

Рисунок 2.2 – Зображення в моделі RGB: а) початкове зображення; б) початкове зображення в компоненті R; в) початкове зображення в компоненті G; в) початкове зображення в компоненті B

Матриці для перекладу кольорів між системами RGB і XYZ можна виразити співвідношеннями (2.1) та (2.2).

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.431 & 0.342 & 0.178 \\ 0.222 & 0.707 & 0.071 \\ 0.020 & 0.130 & 0.939 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}, \qquad (2.1)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 3.063 & -1.393 & -0.476 \\ -0.969 & 1.876 & 0.042 \\ 0.068 & -0.229 & 1.069 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \quad (2.2)$$

YUV – апаратно-орієнтована модель, використовувана в телебаченні і служить для скорочення передаваємої смуги частот за рахунок використання психофізіологічних особливостей зору. Перетворення між просторами RGB і YUV визначаються наступними співвідношеннями: пряме – формула (2.3), зворотне – формула (2.4).

$$\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}. \quad (2.3)$$

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.623 \\ 1 & -0.272 & -0.648 \\ 1 & -1..105 & 0.705 \end{bmatrix} \cdot \begin{bmatrix} Y \\ U \\ V \end{bmatrix}. \quad (2.4)$$

Модель YUV вже давно використовується в системах телемовлення (наприклад, NTSC) і заснована на розкладанні сигналу на три складові (рисунок 2.3) [7].

а)



б) в) г)

Рисунок 2.3 – Зображення в моделі YUV: а) початкове зображення; б) початкове зображення в компоненті Y; в) початкове зображення в компоненті U; в) початкове зображення в компоненті V

Історично термін YUV використовувалися для конкретного аналогового кодування інформації про колір у телевізійних системах, тоді як YCbCr використовувався для цифрового кодування інформації про колір, що підходить для стиснення та передачі відео та нерухомих зображень, таких як MPEG та JPEG. Сьогодні термін YUV зазвичай використовується у комп'ютерній індустрії для опису файлових форматів, які кодуються за допомогою YCbCr.

Розглянемо призначення колірних складових. Y – сама значуща складова визначальна яскравість точки зображення. Давно було відмічено, що людина набагато сильніше реагує на зміну яскравості, чим на зміну колірних складових. U, V – дві колірні складові (U = G - R, V = G - B).

Процедура перекладу з RGB в YUV є лінійною і оборотною для реальних значень кольоровості, тобто при зворотному відновленні сигналу RGB з YUV втрати відсутні.

2.3 Модифікація алгоритму виявлення руху камери

Вважатимемо, що маємо відео, яку зняла цифрова відеокамера. Цифрове відео при обробці представлено набором кадрів. Кадрами є статичні зображення. Вважатимемо, що можна получити любий кадри з відео. Получений кадр це

зображення в моделі RGB. Кожний кадр перетворюється до моделі YUV і в подальшому огляді ми будемо використовувати для аналізу тільки складову Y.

Нехай є відеопослідовність, що складається з безлічі кадрів $F = F_1 \cup F_2 \cup \ldots \cup F_n$, де n – кількість кадрів, F_i- і - ий кадр. Кожний кадр має розмір $M \times N$. Розглянемо два сусідніх кадру F_i та F_{i+1}. Кадр F_i розіб'ємо на чотири не пересічні блоки $P^i = \{P_1^i, P_2^i, P_3^i, P_4^i\}$, які розташуються по периметру кадру (рисунок 2.4).

Тоді модифікація алгоритму виявлення руху відеокамери має наступні кроки:

1. Маємо відео V, яке складається з кадрів $F_1, F_2, \ldots, F_t$. Нехай F_i – це і – кадр, цифрове зображення розміром $n \times m$.

2. Для кожної пари кадрів F_i та F_{i-1}:

2.1 Кадр F_i розіб'ємо на чотири блоки $P^i = \{P_1^i, P_2^i, P_3^i, P_4^i\}$, які розташуються по периметру кадру. Нехай (x_1, y_1), (x_2, y_2), …, (x_4, y_4) – координати лівої верхньої вершини блоків P^i. Нехай f – зміщення для пошуку.

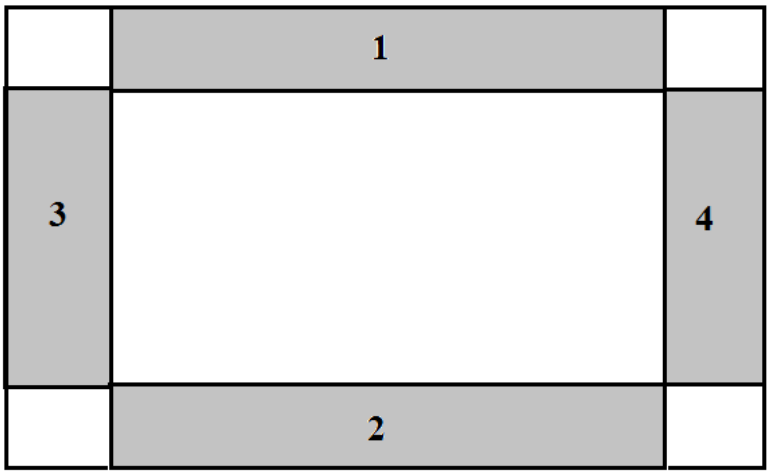


Рисунок 2.4 – Блоки периметру, що аналізуються

2.2 Розбити кадр F_{i+1} на множину пересічних блоків $S^{i+1} = \{s_1^{i+1}, \ldots, s_q^{i+1}\}$, для кожного з чотирьох блоків, починаючи с координати $(x_i - f, y_i - f)$ до

координати $(x_i + f,\ y_i + f)$, де $(x_i,\ y_i)$ – координати лівої верхньої вершини блоку P_j^i.

2.3 Для кожної пари блоків p_j^i та s_k^{i+1} обчислити коефіцієнт метрики між двома блоками. Нехай mk_w – значення коефіцієнту метрики подібності для пари блоків p_j^i та s_k^{i+1}. Нехай після розглядання всієї множини пересічних блоків маємо послідовність значень коефіцієнту метрики подібності $MK = \{mk_1,\ mk_2,\ \dots,\ mk_q\}$.

2.4 Серед значень $MK = \{mk_1,\ mk_2,\ \dots,\ mk_q\}$ знайдемо екстремум та координати лівої верхньої вершини блоку s_k^{i-1}, якому відповідає екстремум. Нехай ці координати є $(x_r,\ y_r)$, позначимо їх, як вектор руху між двома кадрами F_i та F_{i-1}, $\overline{v_z} = (-x_r,\ -y_r)$.

3. Після проходження усіх кадрів маємо вектор руху камери для відео: $\overline{v} = \{\bar{v}_1,\ \bar{v}_2,\ \dots,\ \bar{v}_t\}$.

2.4 Метрики подібності

Позначимо через Y_{ij} значення яскравості пікселя к-го кадру, що перебуває на перетині і строки та j стовпця. Через $Y_{tp}^{'}$ - значення яскравості пікселя стисненого зображення, що перебуває на перетині t строки та p стовпця.

Найбільш простою та поширеною метрикою для оцінки подоби є Mean Square Error – MSE (2.5):

$$MSE = \frac{1}{MN} \sum_{ij,tp} \left(Y_{ij} - Y_{tp}^{'}\right)^2 \qquad (2.5)$$

Розглянемо модифікації, основою яких лягла метрика MSE. Перша модифікація зветься Normalized Mean Square Error – NMSE (2.6):

$$NMSE = \frac{\sum_{ij,tp}\left(Y_{ij} - Y'_{tp}\right)^2}{\sum_{ij}\left(Y_{ij}\right)^2} \quad (2.6)$$

Відрізняється від MSE, тим що суму квадратів різниці між значеннями яскравості пікселів поточного і наступного кадру нормуємо не добутком розмірів зображення, а сумою квадратів значень яскравості пікселів поточного кадру.

Друга модифікація Laplacian Mean Square Error – LMSE (2.7):

$$LMSE = \frac{\sum_{ij,tp}\left(\nabla^2 Y_{ij} - \nabla^2 Y'_{tp}\right)^2}{\sum_{ij}\left(\nabla^2 Y_{ij}\right)^2} \quad (3.6)$$

где $\nabla^2 Y_{ij} = Y_{i+1\ j} + Y_{i-1\ j} + Y_{i\ j+1} + Y_{i\ j-1} - 4Y_{ij}$;

$$\nabla^2 Y'_{tp} = Y'_{t+1\ p} + Y'_{t-1\ p} + Y'_{t\ p+1} + Y'_{t\ p-1} - 4Y'_{tp}$$

Відрізняється від NMSE, тим що у обчисленнях беруть участь як поточні значення яскравості поточного і наступного кадру, а й значення яскравості пікселів, що становлять околиця поточної точки. Це робить цю метрику чутливішою до змін на околиці пікселів [8].

3. ОПИС ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Обґрунтування вибраної мови програмування

Для реалізації алгоритму виявлення руху камери була використана програма для розробки додатків Visual Studio, мова програмування C# та Chart.

Microsoft Visual Studio – лінійка продуктів компанії Microsoft, що включають інтегроване середовище розробки програмного забезпечення і ряд інших інструментальних засобів.

Дані продукти дозволяють розробляти як консольні додатки, так і ігри та програми з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms, а також веб-сайти, веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows, Windows Mobile, Windows CE, .NET Framework, Xbox, Windows Phone .NET Compact Framework і Silverlight.

Враховуючи всі переваги програмних мов для розробки подібних продуктів ми вибрали мову – C#. До цієї мови входить багато корисних особливостей – простота, об'єктна орієнтованість, типова захищеність, підтримка сумісності версій та багато іншого. Дані можливості дозволяють швидко та легко розробляти програми, особливо COM+ програми та Web сервіси. При створенні C# його автори враховували досягнення багатьох інших мов програмування: C++, C, Java, SmallTalk, Delphi, Visual Basic і т.д. Треба зауважити, що через те, що C# розроблявся з чистого листа, у його авторів була можливість (якою вони явно скористалися), залишити в минулому всі незручні та неприємні особливості (існуючі, як правило, для зворотної сумісності), будь-якої з попередніх мов. В результаті вийшов дійсно проста, зручна і сучасна мова, що за потужністю не поступається C++, але істотно підвищує продуктивність розробок.

Однією з найпоширеніших завдань при роботі з візуальними об'єктами є побудова діаграми. Елемент керування Chart у ASP.NET пропонує широкий набір типів діаграм та параметрів конфігурації. Елемент управління Chart був

доступний як завантажуваний додатково в .NET версії 3.5 SP1, але тепер входить до складу .NET 4.0.

Різноманітність функціональних засобів і параметрів цього елемента управління настільки велике, що їх просто неможливо висвітлити в одному розділі. Тому я спробую показати переваги, створення деяких доступних типів діаграм які я використав в написанні цієї роботи, прив'язувати діаграми до різних джерел даних і виконувати ряд корисних функцій створення діаграм.

Кожна діаграма має одну або більше областей, де можна відображати дані. У наведеному вище базовому оголошенні визначено одну діаграму, яка використовуватиметься для графічного відображення двох наборів даних.

Chart дозволяє будувати графіки наступних типів:

- Двовимірні графіки, що містять точки, лінії та багатокутники;
- Графіки з розривною горизонтальною віссю;
- Багатоденні графіки;
- Біржові свічкові графіки;
- Вертикальні та горизонтальні гістограми;
- Мережеві та деревоподібні схеми.

Chart має багаті можливості для роботи з кольорами, заливками, рамками, формою об'єктів тощо, дозволяючи створювати професійний дизайн графіків.

Способи виведення та інтерактивні можливості:

- Chart дозволяє виводити графіки у вікно прикладної програми, на принтер та файли графічних форматів BMP, EMF, PNG, JPEG, GIF.
- Графічні зображення можуть бути імпортовані до офісних документів через буфер обміну. Графік вставляється у векторному форматі EMF, що забезпечує високу якість зображення. Будь ласка, не вставляйте скріншоти - це складніше, і виглядають вони гірше!
- При виведенні графіка у вікно Chart дозволяє масштабувати та зрошувати зображення, виділяти точки та переглядати додаткову інформацію про них, вибирати підмножини точок та виводити по них окрему інформаційну таблицю.

- Декілька графіків можуть бути відображені на одному слайді. У цьому підтримується можливість синхронного масштабування осей.
- Декілька слайдів можуть бути зібрані у фільм (слайд-шоу). При виведенні графіка у вікно можна увімкнути режим перегляду фільму, коли слайди змінюватимуть один одного автоматично.

З допомогою елемента управління Chart можна не тільки візуалізувати, а й аналізувати різні дані. Зокрема можна прогнозувати зміни того чи іншого показника за допомогою методу поліноміальної регресії.

3.2 Огляд програмного забезпечення

Для демонстрації реалізації розробленої модифікації алгоритму виявлення руху камери було створено програмний продукт. Користувачу необхідно запустити файл «TrackingInVideo.exe» для початку роботи з ним, після чого на екрані з’явиться головне вікно інтерфейсу.

У вікні, що з'явилося натиснувши на пункт меню «File» та підпункт «Open file» відкриється вікно вибору файлу.

Для попереднього перегляду користувач може натиснути кнопку «Play Video». Після цього відкриється вікно із плеєром для попереднього перегляду.

Для того, щоб розпочати відстежування рухів камери, необхідно натиснути на пункт меню «Tracking» та обрати підпункт «Track camera». Це призведе до відкриття вікна для налаштування та запуску трекінгу (рисунок 3.1).

В цьому вікні можна обрати метод оцінки: Corralation, LMSE, NMSE.

Якщо у відео попадають зайві об’єкти по периметру кадру, які можуть спотворити обчислення трекінгу, то в програмі є інструменти для позначення області корисної для обробки (рисунок 3.2).

Після вибору методу оцінки та налаштування області аналізу можна вибрати спосіб виявлення руху камери. У вікні знаходяться 2 кнопки. Першу потрібно використовувати коли камера слідкує за об’єктом зовні або на відео зйомка пейзажу. Другу використовуємо коли камера знаходиться в середині або

на об’єкті, що рухається. Від вибору кнопки залежить як буде проводитися аналіз векторів з чотирьох блоків для остаточного прийняття рішення о векторі руху всього кадру.

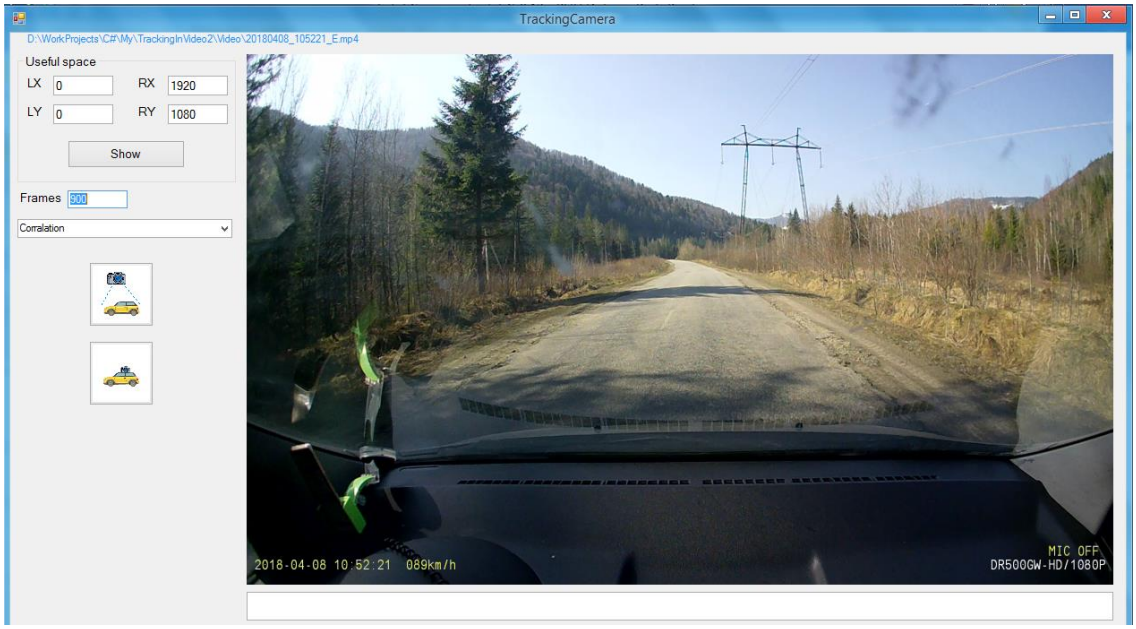


Рисунок 3.1 – Вікно для налаштування та запуску трекінгу

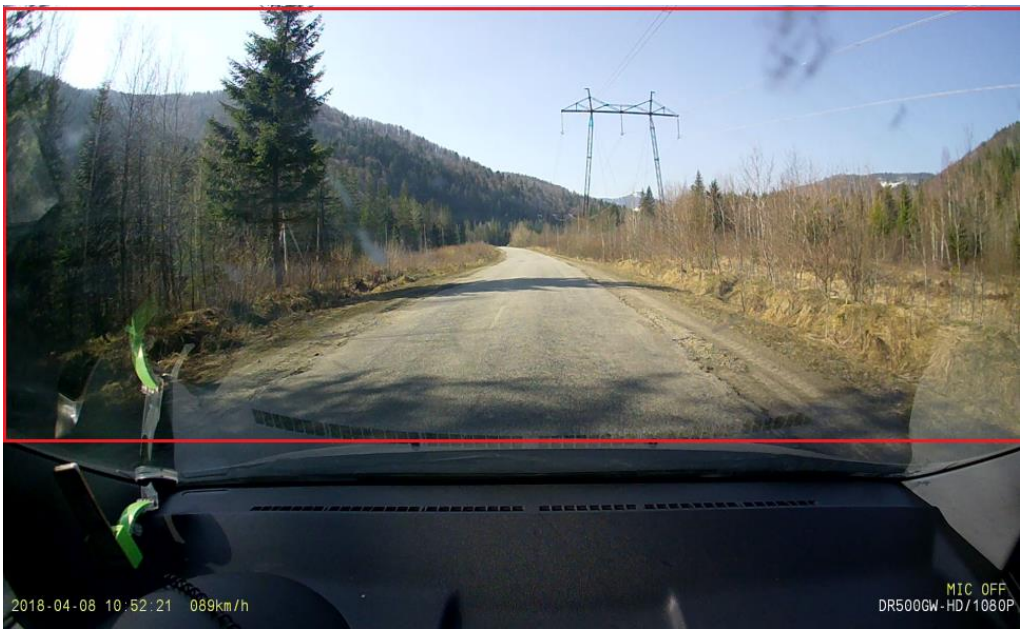


Рисунок 3.2 – Кадр відео та область корисна для обробки (позначена червоним прямокутником)

3.3 Результати експериментів

Для проведення експериментів, спрямованих на оцінку ефективності розробленого алгоритму та порівняння показника візуальної якості трекінгу використовувалось 50 відепослідовностей.

Під показником візуальної якості трекінгу розуміємо 2D-зображення вектору руху камери.

Для перевірки розробленої модифікації алгоритму виявлення руху відеокамери, здійсненої шляхом використання в алгоритмі коефіцієнта кореляції, LMSE та NMSE. Метою обчислювального експерименту є аналіз ефективності використання зазначених метрик.

Перший експеримент проводився над відео з різним розташуванням камери: всередині або поза об'єктом руху. Приклади результатів з використанням коефіцієнту кореляції зображені на рисунку 3.3.

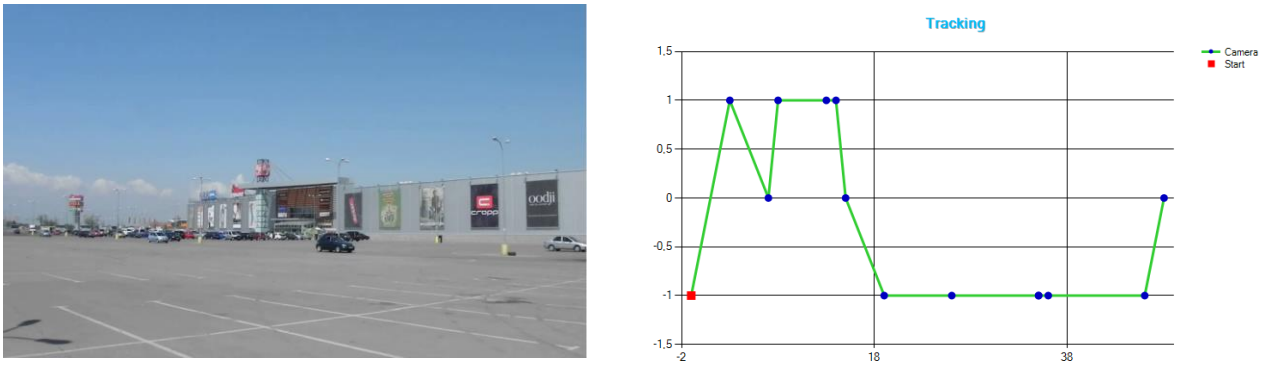
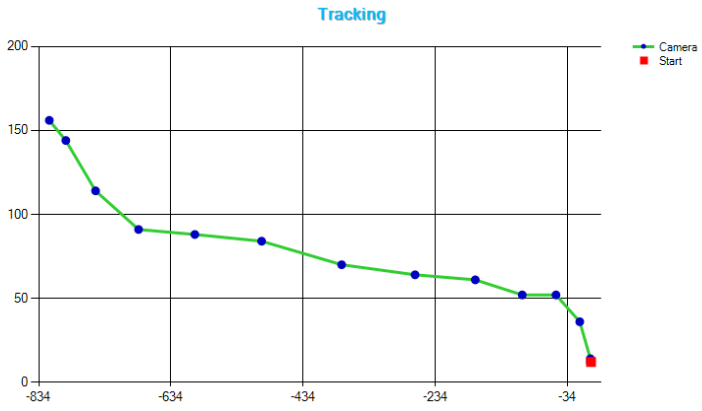


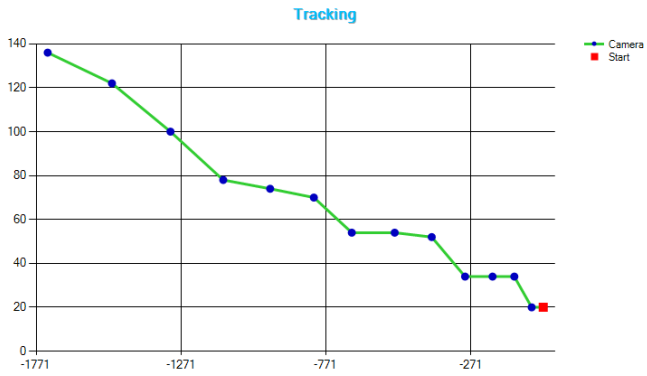
Рисунок 3.3 – Результати трекінку

Другий експеримент проводився над відео з різним розташуванням камери: всередині або поза об'єктом руху та для оцінки подібності блоків використовувались декілька метрик.

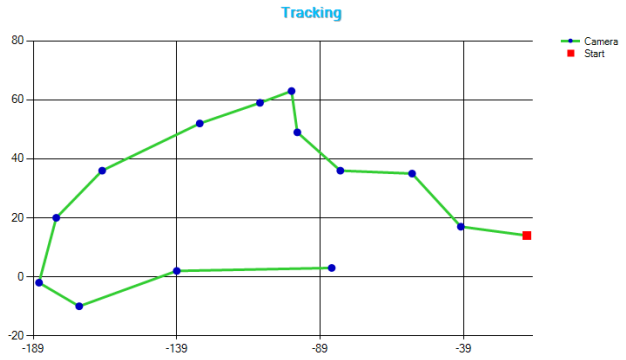
В попередній роботі результати експериментів показали, що коефіцієнт кореляції є найліпшою метрикою для оцінки подібності із розглянутих в роботі. За мету другого експерименту було використання та оцінка придатності інших метрик для оцінки подібності блоків. Приклади результатів з використанням коефіцієнту кореляції, LMSE та NMSE зображені на рисунку 3.4. Коефіцієнт кореляції розглядався в якості еталону для оцінки подібності блоків.



а)



б)



в)

Рисунок 3.4 – Результати трекінку одного відео: а) коефіцієнт кореляції; б) LMSE; в) NMSE

4 ОХОРОНА ПРАЦІ І БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ

Аналіз умов праці і вибір заходів і засобів захисту від небезпечних і шкідливих виробничих факторів

Аналіз умов праці виконуємо для робочого місця із персональним комп'ютером. Фахівці різних напрямків прийшли до висновку, що причиною відхилень у стані здоров'я користувачів є не стільки самі комп'ютери, скільки недостатнє дотримання принципів ергономіки в організації робочого місця. Для того, що б активне застосування комп'ютерних технологій не стало додатковим чинником погіршення здоров'я, вкрай необхідно, щоб робоче місце відповідало гігієнічним вимогам безпеки.

Трудова діяльність працівників в галузі кібербезпеки постійно пов'язана із використанням комп'ютера, тому фахівці піддаються впливу факторів виробничої небезпеки [51].

Основні фізичні фактори:

– підвищений рівень напруги в електричному ланцюзі, замикання якого може пройти через тіло працюючого;

– можливість ураження статичною електрикою;

– запиленість повітря робочого приміщення;

– нерівномірний розподіл яскравості в полі зору;

– підвищений рівень пульсації світлового потоку.

Основний хімічний фактор: підвищений вміст у повітрі вуглекислого газу, озону, аміаку, фенолу, формальдегіду та інших шкідливих речовин.

Основні психофізіологічні фактори:

– напруження зорового аналізатору;

– тривале статичне навантаження опорно-рухового апарату

– динамічне перевантаження кистьових суглобів;

– нераціональна організація робочого місця.

Фахівець з кібербезпеки може бути допущений до роботи за умови виконання наступних вимог: проходження медичного огляду, вступного

інструктажу з охорони праці у відділі охорони праці та первинного інструктажу з охорони праці на робочому місці. В подальшому він повинний проходити повторні інструктажі з охорони праці на робочому місці один раз на півріччя, періодичні медичні огляди один раз на два роки.

Для запобігання ураженню електричним струмом необхідно дотримуватись вимог електробезпеки, зокрема, перед початком робочого дгя слід перевіряти загальний стан апаратури, справність електропроводки, з'єднувальних шнурів, штепсельних вилок, розеток, заземлення захисного екрана і персонального комп'ютера в цілому. Не дозволяється торкатися задньої панелі системного блоку при включеному живленні, перемикати роз'єми інтерфейсних кабелів периферійних пристроїв при включенему живленні. Заборонено самостійно розкривати корпуси системного блоку і монітору і самостійно робити ремонт обладнання. З метою уникнення короткого замикання заборонено допускати потрапляння вологи на поверхні комп'ютера і периферійних пристроїв, особливо на робочу поверхню клавіатури і «миші».

Для нейтралізації зарядів статичної електрики в приміщенні, де розміщені робочі місця фахівців з кібербезпеки, в тому числі обладнані лазерними принтерами, рекомендується збільшувати вологість повітря. Для цього слід використовувати зволожувачі повітря і регулярно провітрювати приміщення. Також для запобігання ураженню розрядами статичної електрики не рекомендується одягати під час роботи одяг з синтетичних матеріалів.

Робочі місця фахівців з кібербезпеки, обладнані персональними комп'ютерами, в мають бути розташовані на відстані не менше 1,5 м від стіни з вікнами, від інших стін на відстані не менше 1 м, між собою – на відстані не менше 1,5 м. Відносно вікон робоче місце доцільно розташовувати таким чином, щоб природне світло падало на нього збоку, переважно зліва, уникаючи потрапляння в очі прямого світла. Джерела світла рекомендується розташовувати з обох боків екрану паралельно напрямку погляду. Для уникнення світлових відблисків екрану і клавіатури в напрямку очей користувача, від світильників загального освітлення або сонячних променів, необхідно використовувати

антиполискові сітки, спеціальні фільтри для екранів, захисні козирки, на вікнах можна закріплювати жалюзі [52]. Фільтри з металевої або нейлонової сітки використовувати не рекомендується, тому що сітка спотворює зображення через інтерференцію світла. Найкращу якість зображення забезпечують скляні поляризаційні фільтри. Вони усувають практично всі відблиски, роблять зображення чітким і контрастним.

Під час роботи з текстовою інформацією (в режимі введення даних, редагування тексту програм, читання інформації з екрану) найбільш фізіологічно вірним є зображення чорних знаків на білому фоні. Монітор слід розташовувати на робочому столі таким чином, щоб поверхня екрана знаходилася в центрі поля зору на відстані 400 – 700 мм від очей користувача. Рекомендується розміщувати елементи робочого місця, витримуючи однакову відстань від очей до екрана і документів на робочій поверхні [52].

Ергономічна робоча поза досягається регулюванням висоти робочого столу, крісла та підставки для ніг. Ергономічною вважають робочу позу, в якій ступні працівника розташовані горизонтально на підлозі або підставці для ніг, стегна зорієнтовані у горизонтальній площині, верхні частини рук зорієнтовані в вертикальній площині. Кут локтєвого суглоба повинний бути в межах 70 – 90 градусів, зап’ястя слід згинати під кутом не більше ніж 20 градусів, нахил голови не повинний перебільшувати 15 – 20 градусів. Монітор слід встановлювати таким чином, щоб було зручно дивитися на екран: під прямим кутом відносно центральної лінії симетрії обличчя користувача і трохи зверху вниз, при цьому екран має бути трохи нахиленим, нижній його край має бути ближче до користувача. Екран має бути розміщений на такій вишині, щоб рівень очей був спрямований на центр або на 2/3 висоти екрана. Лінія погляду повинна бути перпендикулярна центру екрана. Її оптимальне відхилення від перпендикуляра, що проходить через центр екрана у вертикальній площині, не повинне перевищувати ± 5°, припустиме відхилення не повинне перевищувати ± 10°. Рекомендована відстань від обличчя до екрана обирається з урахуванням розмірів

алфавітно-цифрових знаків і символів, і має триматися в межах 60 – 70 см, але обличчя має бути не ближче 50 см від екрану.

Клавіатуру слід стійко розташовувати на робочому столі на відстані 100 – 300 мм від краю, не допускати її руху і хитання. Для зниження несприятливого впливу на користувача маніпулятору типу «миша», необхідно залишати вільною поверхню столу, достатню для переміщення «миші» і зручного упору ліктьового суглоба.

Зниження психофізіологічних навантажень, зняття втоми досягається виконанням вправ аутогенного тренування після закінчення робочого дня. Також рекомендується періодично проходити сеанси психофізіологічного розвантаження в спеціально обладнаному приміщенні під керівництвом спеціаліста – психолога.

Виконання правил та вимог, регулярні навчання і контроль знань з охорони праці і інших заходів щодо їх виконання сприяє підвищенню продуктивності праці, якості виконання робіт та збереженню здоров’я працівників.

Аналіз техногенних небезпек і вибір заходів і засобів забезпечення безпеки у надзвичайних ситуаціях.

При розробці алгоритмів програмного забезпечення особливу небезпеку становлять пожежі під час роботи за комп’ютером, оскільки вони пов’язані із загрозою життю і здоров’ю фахівців високого класу, а також з великими матеріальними втратами.

Приміщення офісів програмістів слід оздоблювати системами вентиляції і кондиціонування повітря, які призначені не тільки для нормалізації параметрів мікроклімату, але і для відведення надлишкового тепла від комп’ютерів. Однак такі системи також представляють додаткову пожежну небезпеку для самого офісу та інших приміщень, оскільки, з одного боку, повітроводи забезпечують подачу у всі приміщення кисню, що є окиснювачем, а з іншого – при виникненні пожежі швидко поширюють вогонь і продукти горіння по всіх приміщеннях і пристроях, з якими вони пов’язані.

Експлуатація комп’ютерів пов’язана з необхідністю проведення обслугувовальних робіт. При цьому прокладають тимчасові електропроводи,

контроль за нагріванням яких може бути знижений внаслідок людського фактору, ведуть пайку та чищення окремих вузлів і деталей мережі. Пайка належить до вогневих робіт, для чищення використовують легкозаймисті рідини – спирт або керосин. Таким чином виникає додаткова пожежна небезпека, яка потребує відповідних заходів пожежної профілактики.

Усунення причин займання в електричному обладнанні проводиться в різних напрямках, а саме [53]:

– попередження короткого замикання здійснюється правильним вибором елементів, монтажем і експлуатацією електромереж;

– попередження короткого замикання здійснюється застосуванням захисту схем у вигляді швидкодіючих реле, а також вимикачів, плавких запобіжників, автоматичних вимикачів;

– всі види кабелів слід прокладати в металевих газонаповнених трубах; у машинних залах кабельні лінії слід прокладати під технологічними знімними підлогами, які виконують з негорючих або важкогорючих матеріалів з межею вогнестійкості не менше 30 хвилин.

Також забороняється:

– самостійно ремонтувати електроапаратуру у випадку відмовлень. Ремонт електроапаратури повинні виконувати спеціалістами з технічного обслуговування. З метою запобігання перегріванню електричних схем слід проводити чищення системних блоків від пилу: один раз на півроку слід відкривати корпус і видаляти пилососом пил і бруд, що накопичилися;

– класти будь-які предмети на оргтехніку і системні блоки комп’ютерів, оскільки це ускладнює тепловідвід с корпусів і сприяє перегріву;

– закривати будь-чим вентиляційні отвори оргтехніки і системних блоків комп’ютерів, оскільки це може призвести до їх перегрівання і виходу з ладу.

Вимоги безпеки в аварійних ситуаціях:

– кожен працівник при виявленні несправності в роботі приладу, що може спричинити небезпеку для працівників, повинен доповісти про це своєму безпосередньому керівнику. У тих випадках, коли несправність може бути

усунена працівником, він має це зробити, а потім повідомити керівнику. Усунення несправності виконується при дотриманні визначених вимог безпеки;

– при ураженні електричним струмом необхідно якомога скоріше звільнити потерпілого від дії струму шляхом вимкнення приладу, відключення обладнання від електромережі, або перерубати провід живлення інструментом з ізольованими ручками. Якщо вимкнути обладнання достатньо швидко неможливо, тоді необхідно застосувати інші міри щодо звільнення потерпілого від дії струму, наприклад, скористатися дошкою чи іншим сухим предметом, що не проводить електричний струм. Обов'язково потрібно викликати швидку допомогу чи рятувальну бригаду МЧС. До приїзду лікарів чи бригади МЧС потрібно надати потерпілому першу домедичну допомогу;

– при виникненні пожежі необхідно негайно повідомити в найближчу пожежну частину, відключити устаткування від джерела напруги і за можливості приступити до ліквідації пожежі засобами гасіння пожежі, що знаходяться в приміщенні.

Уваги слід приділяти пожежній безпеці організації, її будівлі в цілому та окремих її приміщень. Слід розробляти комплекс організаційних і технічних заходів пожежної профілактики, призначений запобігти пожежі, а в разі її виникнення забезпечити безпеку людей, обмежити поширення вогню, а також створити умови для успішного гасіння пожежі.

В будинках із офісами програмістів в коридорах, на майданчиках сходових клітин, біля входів слід встановлювати пожежні крани. В самих приміщеннях слід встановлювати ручні вуглекислотні вогнегасники із розрахунку один вогнегасник на 40 – 50 $м^2$ площі приміщення [53]. Вогнегасники повинні бути в робочому стані і перевірятися згідно з нормами.

Також будівлі слід оздоблювати системами світлової та звукової пожежної сигналізації і газовими установками автоматичного пожежогасіння. У будівлях на випадок виникнення пожежі слід облаштовувати щонайменше два евакуаційні виходи. Шляхи евакуації слід прокладати через коридори, кожне приміщення повинно мати евакуаційні виходи. Забороняється прокладати шляхи евакуації

через приміщення, в яких знаходяться співробітники, що працюють в інших підрозділах. В приміщеннях допускається проектувати один евакуаційний вихід за умови, що відстань від найбільш віддаленого робочого місця до виходу не перевищує 25 м, а кількість працюючих у приміщенні не перебільшує 25 осіб.

На евакуаційних шляхах слід встановлювати штучне аварійне і евакуаційне освітлення. Штучне евакуаційне освітлення може бути доповнене природним, якщо цього дозволяє конструкція будівлі.

Діагностика властивостей нервової системи за психомоторними показниками

Під час роботи фахівців з кібербезпки слід приділяти уваги не тільки ергономіці робочого місця, але й психофізіологічним показникам самого працівника. Це дозволяє оптимально обрати режими праці і відпочинку для конкретної особи. Визначимо психомоторні показники для діагностики властивостей нервової системи. Для цього виконуємо теппінг-тест (табл. 4.1) і обробимо його результати.

Час виконання теппінг-тесту – середа, 10.00. Такий час обраний тому, що відносно тижневих циклів працездатності середа – це середина робочого тижня, день стійкої працездатності; відносно добових циклів працездатності середньостатистичної людини астрономічний час 10.00 – це також період початку першої, найбільш ефективної фази стійкої працездатності.

Таблиця 4.1 – Результати теппінг-тесту

Номер поля	Проміжок часу, с	Кількість точок правою рукою, N	Темп руху руки T, точок/с	Кількість точок лівою рукою, N	Темп руху руки T, точок/с
1	0 – 5	31	6,2	20	4
2	6 – 10	28	5,6	15	3
3	11 – 15	24	4,8	18	3,6
4	16 – 20	26	5,2	23	4,6
5	21 – 25	24	4,8	21	4,2
6	26 – 30	26	5,2	21	4,2

Для кожної руки визначено коефіцієнт сили нервової системи:

– для правої руки $К_{СНС}$ = - 87 %

– для лівої руки $К_{СНС}$ = 40 %

Коефіцієнт функціональної асиметрії працездатності лівої і правої рук:

$$К_{ФА} = \frac{159 - 118}{159 + 118} \cdot 100\,\% = 14{,}8\,\%$$

За таблицею 5.1 визначимо для кожної руки середній темп руху:

– середній темп руху правої руки $Т_{ср\,п}$ = 5,3;

– середній темп руху лівої руки $Т_{ср\,л}$ = 3,93.

Побудуємо графікі працездатності правої і лівої рук (рисунок 4.1)

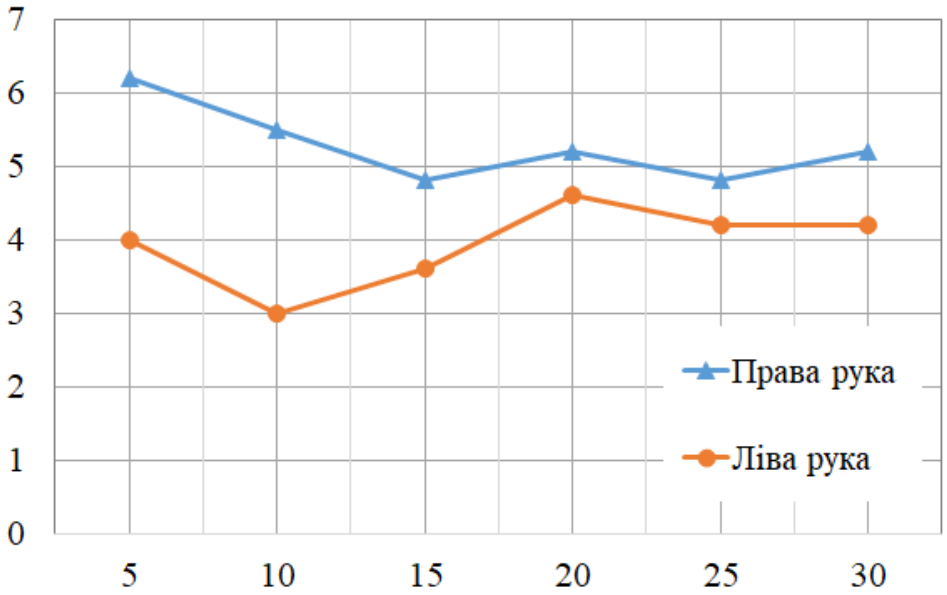


Рисунок 4.1 – Графіки працездатності правої і лівої рук

За результатом теппінг-тесту, можна визначити наступне: працездатність правої руки вище, ніж працездатність лівої; коефіцієнт сили нервової системи показує високу вираженість сильної нервової системи.

Графік працездатності правої руки має спадний тип кривої, максимальний темп знижується вже з другого секундного відрізка й залишається на зниженому рівні протягом усієї роботи. Графік працездатності лівої руки має увігнутий тип

кривої, початкове зниження максимального темпу змінюється потім короткочасним як проміжний між середньою і слабкою силою нервової системи – середньо-сильна нервова система.

ВИСНОВКИ

Проведено аналіз стану сучасних розробок в області трекінгу. Розглядалися підходи до оцінки руху.

В роботі розроблено модифікацію алгоритму виявлення руху відеокамери шляхом аналізу чотирьох блоків у кожному кадрі відео. У розробленій модифікації алгоритму виявлення руху відеокамери розглядались чотири блоку, що в сукупності складають периметр кадру, крім кутових блоків, що дозволяє обчислити рух камери, коли вона розташована в середині або на об’єкті, що рухається.

Експерименти показали, що використання метрики LMSE, NMSE та кореляції Пірсона під час аналізу периметра дають різні результати. Що дає підставу для рекомендації о недоцільному використанні метрик LMSE, NMSE в задачах подібного типу.

ПЕРЕЛІК ПОСИЛАНЬ

1. ДСТУ 13699. Запис і відтворення інформації. Терміни та визначення. URL: https://docs.cntd.ru/document/1200004667

2. Audio Video Interleave. URL: http://ru.wikipedia.org/wiki/ Audio_Video_Interleave.

3. Что такое цифровое видео URL: http://video-practic.ru/digital-video.

4. Видеоформаты и видеостандарты URL: http://video-sam.ru/format.html.

5. Прэтт У. Цифровая обработка изображений. Москва: Мир, 1982. 280 с.

6. Фотокинотехника: Энциклопедия / Главный редактор Е. А. Иофис. М.: Советская энциклопедия, 1981. 447 с.

7. Цветовая система YUV. URL: https://www.hisour.com/ru/yuv-color-system-25916/

8. Голубничий В. О., Прогонов Д. О., Кущ С. М. Використання метрик якості цифрових зображень для виявлення стеганограм, створених на основі комплексних методів приховування повідомлень. *Матеріали XIII Всеукраїнської науково-практичної конференції студентів, аспірантів та молодих вчених «Теоретичні і прикладні проблеми фізики, математики та інформатики»*. Київ : НТУУ КПІ, 2015. С. 154-156

9. Motion detection and objects tracking algorithm implementation. URL: https://developex.com/blog/motion-detection-and-objects-tracking-algorithm-implementation/

10. Алгоритм стабилизации видеопоследовательности. URL: https://cyberleninka.ru/article/n/algoritm-stabilizatsii-videoposledovatelnosti-osnovannyy-na-postroenii-nechetkoy-modeli-dvizheniya

11. Сравнительный анализ метрик оценки качества восприятия потокового видео. URL: https://cyberleninka.ru/article/n/sravnitelnyy-analiz-metrik-otsenki-kachestva-vospriyatiya-potokovogo-video

12. Анализ реализации средств защиты потокового видео. URL: https://cyberleninka.ru/article/n/analiz-realizatsii-sredstv-zaschity-potokovogo-video-v-formate-jpeg2000-ot-oshibok-v-kanale-peredachi-dannyh-dlya-bortovyh-sistem

13. Оценка эффективности составляющих цветовых моделей. URL: https://cyberleninka.ru/article/n/otsenka-effektivnosti-sostavlyayuschih-tsvetovyh-modeley-pri-segmentatsii-izobrazheniy-po-tsvetovym-priznakam

14. Метод предварительного кодирования изображений в корреляционно-экстремальных системах. URL: https://cyberleninka.ru/article/n/metod-predvaritelnogo-kodirovaniya-izobrazheniy-v-korrelyatsionno-ekstremalnyh-sistemah

15. KuoW., Lin C. Two-stage road sign detection and recognition. *IEEE Int. Conference on Multimedia and Expo*. 2007. P.1427–1430. URL: https://ieeexplore.ieee.org/abstract/document/4426395

16. Escaleraet S. Background on traffic sign detection and recognition. *Springer Briefs Comput. Sci.,* 2011. P.5–13. URL: https://link.springer.com/chapter/10.1007/978-1-4471-2245-6_2

17. Benallal M., Meunier J. Real-time color segmentation of road signs. *IEEE Canadian Conf. on Electrical and Computer Engineering*. 2003. P.1823 –1826. URL: https://ieeexplore.ieee.org/abstract/document/1226265

18. Varunet S. A road traffic signal recognition system based on template matching employing tree classifier. *Proc. of the Int. Conf. on Computational Intelligence and Multimedia Applications*. 2007. P.360–365. URL: https://ieeexplore.ieee.org/abstract/document/4426395

19. Kuo W., Lin C. Two-stage road sign detection and recognition. *IEEE Int. Conference on Multimedia and Expo*. 2007. P.1427–1430 URL: https://ieeexplore.ieee.org/abstract/document/4284928

20. Broggiet A. Real time road signs recognition. *IEEE Intelligent Vehicles Symp*., 2007. P. 981 –986 URL: https://ieeexplore.ieee.org/abstract/document/4290244

21. Ruta A., Li Y., Liu X. Detection, tracking and recognition of traffic signs from video input. *Int. IEEE Conf. on Intelligent Transportation Systems*. 2008. P.55 – 60 URL: https://ieeexplore.ieee.org/abstract/document/4732535

22. Escalera A., Armingol J., Mata M. Traffic sign recognition and analysis for intelligent vehicles. *Image Vis. Comput*. 2003. №21(3), 247–258 URL: https://www.sciencedirect.com/science/article/abs/pii/S0262885602001567

23. Loy G. Fast shape-based road sign detection for a driver assistance system. *IEEE/RSJ Int. Conf. on Intelligent Robots and Systems*. 2004. P.70 –75.

24. Mobileye: adaptive headlight control. 2008. URL: http://www.mobileye-vision.com

25. Takabaet S. Measurement of traffic flow using real time processing of moving pictures. *Proc. 32nd IEEE Vehicular Technology Conf.,* 1982. P.488–494. URL: https://ieeexplore.ieee.org/abstract/document/1623062/

26. Hashimoto N. Development of an image processing traffic flow measurement system. *Sumitomo Electr. Tech. Rev.* 1990. №25. P.133–138. URL: https://ieeexplore.ieee.org/abstract/document/1304014

27. Dickinson K., Waterfall R. Image processing applied to traffic: a general review. *Traffic Eng. Contr.* 1984. №25(1). P.6–13. URL: https://pascal-francis.inist.fr/vibad/index.php?action=getRecordDetail&idt=9421080

28. Dickinson K., Wan C. Road traffic monitoring using the TRIP II system, *Proc. 2nd Int. Conf. on Road Traffic Monitoring*. 1989. P.56–60. URL: https://ieeexplore.ieee.org/abstract/document/19289

29. Ashworthet R. Applications of video image processing for traffic control. *2nd Int. Conf. on Road Traffic Control*. 1985. P.119–122. URL: https://trid.trb.org/view/237542

30. Inigo R. Traffic monitoring and control using machine vision: a survey. *IEEE Trans. Ind. Electron.* 1985. IE-32(3), P.177–185. URL: https://ieeexplore.ieee.org/abstract/document/4158617

31. Hoose N., Computer Image Processing in Traffic Engineering. Taunton, UK: Research Studies Press, 1991. URL: https://research-information.bris.ac.uk/en/publications/computer-vision-aided-road-traffic-monitoring

32. Крамаренко Т. А. К вопросу автоматизации процесса анализа данных научного исследования. *72-й науч.-практ. конф. преподавателей*. Краснодар: КубГАУ, 2017. С. 429-430.

33. Мисюра В.В., Кондратьева Т.Н., Бенгус Б.В. Сравнительный анализ методов прогнозирования тенденции развития. *РГУПС*. 2014. №2. С. 124-128.

34. Осовский, С. Нейронные сети для обработки информации. М. : Финансы и статистика, 2002. С. 210-214.

35. Троелсен Э. C# и платформа .NET. Библиотека программиста. СПб.: Питер, 2002. 800 с.

36. Ширяев, А. Основы стохастической финансовой математики. Факты модели. М.: Фазис, 1998. С. 489.

37. Hansen A.T. Complete market pricing in the Wiener filtration without existence of a martingale measure. Preprint. Aarbus University, 1996.

38. Kalman R.E. A new approach to linear filtering and prediction problems. *J. Basic Engineering, Trans. ASMA.* 1960. Vol. 82. Ser. D. P. 35-45

39. Rodriguez A. RESTful Web services: The basics. URL: http://www.ibm.com/developerworks/libra ry/ws-restful /

40. Richardson L., Amundsen M. RESTful Web APIs. Sam Ruby: O'Reilly Media, 2013. 406 p.

41. Kolleret D. Towards robust automatic traffic scene analysis in real-time. *Int. Conf. on Pattern Recognition.* 1994. P. 126 – 131 URL: https://ieeexplore.ieee.org/abstract/document/576243

42. KollerD. Weber J. Malik J. Towards real-time visual based tracking in cluttered traffic scenes. *Intelligent Vehicles Symp*. 1994. P.201–206. URL: https://ieeexplore.ieee.org/abstract/document/639503

43. Carlson B. Autoscope clearing the congestion: vision makes traffic control intelligent. *Adv. Imaging.* 1997. №12 (2), P.54 –56. URL: https://ieeexplore.ieee.org/abstract/document/1414453

44. Yu M., Jiang G., Bokang Y. An integrative method for video-based traffic parameter extraction in ITS. *IEEE Asia Pacific Conf. on Circuits and Systems*. 2000. P.136 –139 URL: https://ieeexplore.ieee.org/abstract/document/913425

45. Baş E., Tekalp M., Salman S. Automatic vehicle counting from video for traffic flow analysis. *Proc. 2007 IEEE Intelligent Vehicles Symp*., 2007. P. 392 –397. URL: https://ieeexplore.ieee.org/abstract/document/4290146

46. Versavel J. Road safety through video detection. *Int. Conf. on Intelligent Transportation Systems,* P. 753 – 757 (1999) URL: https://ieeexplore.ieee.org/abstract/document/821155

47. Kimachi M., Kanayama K., Teramoto K. Incident prediction by fuzzy image sequence analysis. *Proc. of the IEEE Int. Conf. on Vehicle Navigation and Information Systems*, 1994. P.51 –57. URL: https://ieeexplore.ieee.org/abstract/document/396867

48. Zhang X., Busch F., Blosseville J. M., Guidelines for implementation of automatic incident detection systems. 1995. URL: https://scholar.google.com/scholar_lookup?title=Guidelines+for+implementation+of+automatic+incident+detection+systems&publication_year=1995

49. Wells T., Toffin E., Video-based automatic incident detection on San Mateo Bridge in the San Francisco Bay Area. *12th World Congress on Intelligent Traffic Systems*. 2005. URL: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.548.9021&rep=rep1&type=pdf

50. Ekern D., Tolling facilities report. URL: https://scholar.google.com/scholar_lookup?title=Tolling+facilities+report&publication_year=2008

51. ДСТУ 2293-99. Охорона праці. Терміни та визначення основних понять. – Київ: Держстандарт України, 1999. – 172 с.

52. Жидецький В.Ц. Основи охорони праці. Підручник. – Львів: Афіша, 2005. – 319 с.

53. Правила пожежної безпеки в Україні. – Київ: Держстандарт України, 2004. – 126 с.

ДОДАТОК А. Лістинг програмного забезпечення

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Drawing;
using Accord.Video.FFMPEG;
using System.Diagnostics;

namespace TrackingInVideo
{
    public partial class TrackingCamera : Form
    {
        private string FileName;
        private Bitmap firstFrame;
        private VideoFileReader reader;
        private VideoFileWriter writer;

        long frameCount = 0L;

        private Rectangle rect;
        private static Point begin = new Point(-1, -1);
        private Point end = new Point(-1, -1);
        bool IsMouseDown = false;

        private Bitmap original;

        private int frameFrom = 0, frameTo = 0;

        private int width, height;
        private Bitmap area;
        private Point blockPoint;
        private double originalSize = 1;

        struct vectorMotion
        {
            public int num;
            public int[] x;
            public int[] y;
        }

        struct offsets
        {
            public int leftX, leftY;
            public int rightX, rightY;
            public int topX, topY;
```

```
            public int bottomX, bottomY;
        }

        public Bitmap Area { get; set; }
        public Point BlockPoint { get; set; }

        public TrackingCamera(string name)
        {
            InitializeComponent();

            FileName = name;
            label5.Text = "" + FileName;
            textBoxX1.Text = "" + 0;
            textBoxY1.Text = "" + 0;

            comboBox1.Items.AddRange(new string[] { "Corralation",
"LMSE", "NMSE" });
            comboBox1.SelectedIndex = 0;

            if (FileName == "")
            {
                MessageBox.Show("Open video file", "Message",
MessageBoxButtons.OK, MessageBoxIcon.Error);
                return;
            }

            reader = new VideoFileReader();
            reader.Open(FileName);

            frameCount = reader.FrameCount;
            frameTo = (int)frameCount;
            firstFrame = reader.ReadVideoFrame(frameFrom);
            original = firstFrame;

            textBoxX2.Text = "" + original.Width;//1920;
            textBoxY2.Text = "" + original.Height;//803;
            textBoxFrames.Text = "" + frameCount;

            ShowVideoFrame(firstFrame);
        }

        private void ShowVideoFrame(Bitmap first)
        {
            Image img = first;

            panel1.AutoScroll = true;

            pb = new PictureBox();
            pb.Image = img;
            pb.Width = panel1.Width;
            pb.Height = panel1.Height;
            pb.SizeMode = PictureBoxSizeMode.StretchImage;
            pb.MouseDown += Pb_MouseDown;
```

```
            pb.MouseUp += Pb_MouseUp;
            pb.MouseMove += Pb_MouseMove;
            pb.Paint += Pb_Paint;

            panel1.Controls.Add(pb);
        }

        private void CreateBlock(out ImageReader blk, int blk_sizeW,
int blk_sizeH)
        {
            Bitmap bmpBlock = new Bitmap(blk_sizeW, blk_sizeH);
            using (var g = Graphics.FromImage(bmpBlock))
                g.Clear(Color.Black);

            blk = new ImageReader(bmpBlock);
            blk.CreateArrayYUV();

            bmpBlock = null;
        }

        private void button1_Click(object sender, EventArgs e)
        {
            int x1 = Convert.ToInt32(textBoxX1.Text);
            int y1 = Convert.ToInt32(textBoxY1.Text);
            int x2 = Convert.ToInt32(textBoxX2.Text);
            int y2 = Convert.ToInt32(textBoxY2.Text);

            ShowImage form = new ShowImage(firstFrame, x1, y1, x2,
y2);
            form.Show();
        }

        private void ComputeStatistics(string nameStatistics, int i,
int j, byte[] block, byte[] part, int imWidth, int imHeight)
        {
            if (nameStatistics == "Corralation")
            {
                double corr = Statistics.Corralation(block, part,
imWidth, imHeight);
                if (corr > DetermineStatistics.stat)
                {
                    DetermineStatistics.stat = corr;
                    DetermineStatistics.XY = new Point(j, i);
                }
            }
            else if (nameStatistics == "LMSE")
            {
                double lmse = Statistics.LMSE(block, part, imWidth,
imHeight);
                if (lmse < DetermineStatistics.stat)
                {
                    DetermineStatistics.stat = lmse;
                    DetermineStatistics.XY = new Point(j, i);
```

```
                }
            }
            else if (nameStatistics == "NMSE")
            {
                double nmse = Statistics.NMSE(block, part, imWidth,
imHeight);
                if (nmse < DetermineStatistics.stat)
                {
                    DetermineStatistics.stat = nmse;
                    DetermineStatistics.XY = new Point(j, i);
                }
            }
        }

        private void InitStatistics(string nameStatistics)
        {
            if (nameStatistics == "Corralation")
            {
                DetermineStatistics.stat = -1;
                DetermineStatistics.XY = new Point(-1, -1);
            }
            else if (nameStatistics == "LMSE")
            {
                DetermineStatistics.stat = Double.MaxValue;
                DetermineStatistics.XY = new Point(-1, -1);
            }
            else if (nameStatistics == "NMSE")
            {
                DetermineStatistics.stat = Double.MaxValue;
                DetermineStatistics.XY = new Point(-1, -1);
            }
        }

        private void buttonOutside_Click(object sender, EventArgs e)
        {
            int x1 = Convert.ToInt32(textBoxX1.Text);
            int y1 = Convert.ToInt32(textBoxY1.Text);
            int x2 = Convert.ToInt32(textBoxX2.Text);
            int y2 = Convert.ToInt32(textBoxY2.Text);
            string stat = comboBox1.SelectedItem.ToString();

            int n = Convert.ToInt32(textBoxFrames.Text);

            Stopwatch watch = new Stopwatch();
            watch.Start();

            AroundPerimeter4Bars(reader, n, x1, y1, x2, y2, stat);

            watch.Stop();
            TimeSpan ts = watch.Elapsed;
            string elapsedTime =
String.Format("{0:00}:{1:00}:{2:00}:{3:00}", ts.Hours, ts.Minutes,
ts.Seconds, ts.Milliseconds / 10);
```

```
            MessageBox.Show("Done! " + elapsedTime, "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void buttonInside_Click(object sender, EventArgs e)
        {
            int x1 = Convert.ToInt32(textBoxX1.Text);
            int y1 = Convert.ToInt32(textBoxY1.Text);
            int x2 = Convert.ToInt32(textBoxX2.Text);
            int y2 = Convert.ToInt32(textBoxY2.Text);
            string stat = comboBox1.SelectedItem.ToString();

            int n = Convert.ToInt32(textBoxFrames.Text);

            Stopwatch watch = new Stopwatch();
            watch.Start();

            AroundPerimeter4BarsInside(reader, n, x1, y1, x2, y2,
stat);

            watch.Stop();
            TimeSpan ts = watch.Elapsed;
            string elapsedTime =
String.Format("{0:00}:{1:00}:{2:00}:{3:00}", ts.Hours, ts.Minutes,
ts.Seconds, ts.Milliseconds / 10);
            MessageBox.Show("Done! " + elapsedTime, "Message",
MessageBoxButtons.OK, MessageBoxIcon.Information);
        }

        private void AroundPerimeter4Bars(VideoFileReader reader,
int numFrames, int x1, int y1, int x2, int y2, string stat)
        {
            Bitmap videoFrame1;
            Bitmap videoFrame2;
            vectorMotion v;
            v.num = numFrames - 1;
            v.x = new int[numFrames - 1];
            v.y = new int[numFrames - 1];

            int block_size_x = 16, block_size_y = 16;
            int range = 20;

            int x = 0, y = 0;

            videoFrame1 = reader.ReadVideoFrame(0);
            Bitmap area1 = videoFrame1.Clone(new Rectangle(x1, y1,
x2, y2), videoFrame1.PixelFormat);
            ImageReader im1 = new ImageReader(area1);
            im1.CreateArrayYUV();
            ColorModels.RGBToYUV(im1.R, im1.G, im1.B, im1.Y, im1.U,
im1.V, im1.ImWidth, im1.ImHeight);

            for (int f = 1; f < numFrames; f++) //numFrames
```

```
            {
                InitStatistics(stat);

                videoFrame2 = reader.ReadVideoFrame(f);
                Bitmap area2 = videoFrame2.Clone(new Rectangle(x1,
y1, x2, y2), videoFrame2.PixelFormat);

                ImageReader im2 = new ImageReader(area2);
                im2.CreateArrayYUV();
                ColorModels.RGBToYUV(im2.R, im2.G, im2.B, im2.Y,
im2.U, im2.V, im2.ImWidth, im2.ImHeight);

                if (f == 1)
                    PerimetrRangeSearchVector4Bars(im1.Y, im2.Y,
im1.ImWidth, im1.ImHeight, block_size_x, block_size_y, range, out x,
out y, 0, 0, stat);
                else
                    PerimetrRangeSearchVector4Bars(im1.Y, im2.Y,
im1.ImWidth, im1.ImHeight, block_size_x, block_size_y, range, out x,
out y, v.x[f - 2], v.y[f - 2], stat);

                v.x[f - 1] = x;
                v.y[f - 1] = y;

                textBox1.Text = textBox1.Text + " (" + x + "; " + y
+ ") ";

                //area1 = (Bitmap)area2.Clone();
                im1 = im2;
                //im2.Dispose();
            }

            ChartForm form = new ChartForm(v.x, v.y, v.num);
            form.Show();
        }

        private void PerimetrRangeSearchVector4Bars(byte[] predY,
byte[] Y, int width, int height, int block_size_x, int block_size_y,
                                                    int range, out int
h, out int v, int refX, int refY, string stat)
        {
            int y, b_x, b_y, x, j, p = 1;
            int beg_x, beg_y, tmp_width, tmp_height;
            //float m = INT_MAX;

            byte[] tmp_Y, tmp_predY;

            tmp_Y = new byte[width * height];
            tmp_predY = new byte[width * height];

            for (y = -range - refY; y <= range - refY; y++)
            {
                for (x = -range - refX; x <= range - refX; x++)
```

```
                {
                    if (x > 0) beg_x = x;
                    else beg_x = 0;

                    if (y > 0) beg_y = y;
                    else beg_y = 0;

                    tmp_width = width - Math.Abs(x);
                    tmp_height = height - Math.Abs(y);

                    Array.Clear(tmp_Y, 0, tmp_Y.Length);
                    Array.Clear(tmp_predY, 0, tmp_Y.Length);

                    for (j = 0; j < p * block_size_y; j++)
                    {
                        Array.Copy(Y, (j + beg_y) * width + beg_x,
tmp_Y, j * tmp_width, tmp_width);
                        Array.Copy(Y, (height - 1 - j - beg_y + y) *
width + beg_x, tmp_Y, (tmp_height - 1 - j) * tmp_width, tmp_width);
                        Array.Copy(predY, (j + beg_y - y) * width +
beg_x - x, tmp_predY, j * tmp_width, tmp_width);
                        Array.Copy(predY, (height - 1 - j - beg_y) *
width + beg_x - x, tmp_predY, (tmp_height - 1 - j) * tmp_width,
tmp_width);
                    }
                    for (j = p * block_size_y; j < tmp_height - p *
block_size_y; j++)
                    {
                        Array.Copy(Y, (j + beg_y) * width + beg_x,
tmp_Y, j * tmp_width, block_size_x);
                        Array.Copy(Y, (j + beg_y) * width + beg_x +
tmp_width - p * block_size_x, tmp_Y, j * tmp_width + tmp_width - p *
block_size_x, block_size_x);
                        Array.Copy(predY, (j + beg_y - y) * width +
beg_x - x, tmp_predY, j * tmp_width, block_size_x);
                        Array.Copy(predY, (j + beg_y - y) * width +
beg_x - x + tmp_width - p * block_size_x, tmp_predY, j * tmp_width +
tmp_width - p * block_size_x, block_size_x);
                    }

                    ComputeStatistics(stat, y, x, tmp_Y, tmp_predY,
tmp_width, tmp_height);
                }
            }
            h = -DetermineStatistics.XY.X;
            v = -DetermineStatistics.XY.Y;
        }

        private void AroundPerimeter4BarsInside(VideoFileReader
reader, int numFrames, int x1, int y1, int x2, int y2, string stat)
        {
            Bitmap videoFrame1;
            Bitmap videoFrame2;
```

```
            vectorMotion v;
            v.num = numFrames - 1;
            v.x = new int[numFrames - 1];
            v.y = new int[numFrames - 1];

            int block_size_x = 16, block_size_y = 16;
            int range = 20;

            int x = 0, y = 0;

            videoFrame1 = reader.ReadVideoFrame(0);

            Bitmap area1 = videoFrame1.Clone(new Rectangle(x1, y1,
x2, y2), videoFrame1.PixelFormat);
            ImageReader im1 = new ImageReader(area1);
            im1.CreateArrayYUV();
            ColorModels.RGBToYUV(im1.R, im1.G, im1.B, im1.Y, im1.U,
im1.V, im1.ImWidth, im1.ImHeight);

            for (int f = 1; f < numFrames; f++) //numFrames
            {
                InitStatistics(stat);

                videoFrame2 = reader.ReadVideoFrame(f);
                Bitmap area2 = videoFrame2.Clone(new Rectangle(x1,
y1, x2, y2), videoFrame2.PixelFormat);
                ImageReader im2 = new ImageReader(area2);
                im2.CreateArrayYUV();
                ColorModels.RGBToYUV(im2.R, im2.G, im2.B, im2.Y,
im2.U, im2.V, im2.ImWidth, im2.ImHeight);

                if (f == 1)
                    PerimetrRangeSearch(im1.Y, im2.Y, im1.ImWidth,
im1.ImHeight, block_size_x, block_size_y, range, out x, out y, 0, 0,
stat);
                else
                    PerimetrRangeSearch(im1.Y, im2.Y, im1.ImWidth,
im1.ImHeight, block_size_x, block_size_y, range, out x, out y, v.x[f
- 2], v.y[f - 2], stat);

                v.x[f - 1] = x;
                v.y[f - 1] = y;
                textBox1.Text = textBox1.Text + " (" + x + "; " + y
+ ") ";
                im1 = im2;
            }
            ChartForm form = new ChartForm(v.x, v.y, v.num, 1);
            form.Show();
        }

        private void PerimetrRangeSearch(byte[] predY, byte[] Y, int
width, int height, int block_size_x, int block_size_y,
```

```
                                                    int range, out int
h, out int v, int refX, int refY, string stat)
        {
            int y, b_x, b_y, x, j, p = 1;
            int beg_x, beg_y, tmp_width, tmp_height;

            offsets of = new offsets();
            byte[] blk_Y, blk_predY;
            h = 0; v = 0;
            of.leftX = 0; of.leftY = 0;
            of.rightX = 0; of.rightY = 0;
            of.topX = 0; of.topY = 0;
            of.bottomX = 0; of.bottomY = 0;

            for (j = 0; j < 4; j++)
            {
                for (y = -range - refY; y <= range - refY; y++)
                {
                    for (x = -range - refX; x <= range - refX; x++)
                    {
                        if (x > 0) beg_x = x;
                        else beg_x = 0;

                        if (y > 0) beg_y = y;
                        else beg_y = 0;

                        tmp_width = width - Math.Abs(x);
                        tmp_height = height - Math.Abs(y);

                        if (j == 0 || j == 1) //up - down
                        {
                            blk_Y = new byte[tmp_width * p *
block_size_y];
                            blk_predY = new byte[tmp_width * p *
block_size_y];

                            Array.Clear(blk_Y, 0, blk_Y.Length);
                            Array.Clear(blk_predY, 0,
blk_predY.Length);
                        }
                        else //left - right
                        {
                            blk_Y = new byte[tmp_width * p *
block_size_y];
                            blk_predY = new byte[p * block_size_x *
tmp_height];

                            Array.Clear(blk_Y, 0, blk_Y.Length);
                            Array.Clear(blk_predY, 0,
blk_predY.Length);
                        }

                        if (j == 0) //up
```

```
                        {
                            Blocks.GetBlockFromImage(Y, blk_Y,
width, height, beg_x, beg_y, tmp_width, p * block_size_y);
                            Blocks.GetBlockFromImage(predY,
blk_predY, width, height, beg_x, beg_y, tmp_width, p *
block_size_y);

                            ComputeStatistics(stat, y, x, blk_Y,
blk_predY, tmp_width, p * block_size_y);
                        }
                        else if (j == 1) //down
                        {
                            Blocks.GetBlockFromImage(Y, blk_Y,
width, height, beg_x, height - beg_y + y - p * block_size_y,
tmp_width, p * block_size_y);
                            Blocks.GetBlockFromImage(predY,
blk_predY, width, height, beg_x, height - beg_y + y - p *
block_size_y, tmp_width, p * block_size_y);

                            ComputeStatistics(stat, y, x, blk_Y,
blk_predY, tmp_width, p * block_size_y);
                        }
                        else if (j == 2) //left
                        {
                            Blocks.GetBlockFromImage(Y, blk_Y,
width, height, beg_x, p * block_size_y + beg_y, p * block_size_x,
tmp_height);
                            Blocks.GetBlockFromImage(predY,
blk_predY, width, height, beg_x, p * block_size_y + beg_y, p *
block_size_x, tmp_height);

                            ComputeStatistics(stat, y, x, blk_Y,
blk_predY, p * block_size_x, tmp_height);
                        }
                        else //right
                        {
                            Blocks.GetBlockFromImage(Y, blk_Y,
width, height, beg_x + tmp_width - p * block_size_x, p *
block_size_y + beg_y, p * block_size_x, tmp_height);
                            Blocks.GetBlockFromImage(predY,
blk_predY, width, height, beg_x + tmp_width - p * block_size_x, p *
block_size_y + beg_y, p * block_size_x, tmp_height);

                            ComputeStatistics(stat, y, x, blk_Y,
blk_predY, p * block_size_x, tmp_height);
                        }
                    }
                }
                if (j == 0) //up
                {
                    of.topX = -DetermineStatistics.XY.X;
                    of.topY = -DetermineStatistics.XY.Y;
                }
```

```
                else if (j == 1) //down
                {
                    of.bottomX = -DetermineStatistics.XY.X;
                    of.bottomY = -DetermineStatistics.XY.Y;
                }
                else if (j == 2) //left
                {
                    of.leftX = -DetermineStatistics.XY.X;
                    of.leftY = -DetermineStatistics.XY.Y;
                }
                else //right
                {
                    of.rightX = -DetermineStatistics.XY.X;
                    of.rightY = -DetermineStatistics.XY.Y;
                }
            }
            if (!SameSign(of.leftX, of.rightX))
            {
                h = 0;
            }
            else
            {
                h = (Math.Abs(of.leftX) < Math.Abs(of.rightX)) ?
of.leftX : of.rightX;
            }

            if (!SameSign(of.topY, of.bottomY))
            {
                if (of.topY <= 0 && of.bottomY >= 0)
                {
                    int t = (Math.Abs(of.topY) <
Math.Abs(of.bottomY)) ? Math.Abs(of.topY) : Math.Abs(of.bottomY);
                    v = t;
                }
                else
                {
                    int t = (Math.Abs(of.topY) <
Math.Abs(of.bottomY)) ? Math.Abs(of.topY) : Math.Abs(of.bottomY);
                    v = (-1) * t;
                }
            }
            else
            {
                v = (Math.Abs(of.topY) < Math.Abs(of.bottomY)) ?
of.topY : of.bottomY;
            }
        }

        private bool SameSign(int a, int b)
        {
            if (a * b >= 0) return true;
            else return false;
        }}}
```