Вінницький національний технічний університет
(повне найменування вищого навчального закладу)

Факультет інформаційних технологій та комп'ютерної інженерії
(повне найменування факультету)

Кафедра обчислювальної техніки
(повна назва кафедри)

Пояснювальна записка

до магістерської кваліфікаційної роботи

на тему **Методи розпізнавання зображень в багатоканальній комп'ютерній системі моніторингу цифрового телевізійного мовлення**

Виконав: студент 2 курсу, групи 1КІ-19м
спеціальності:
123 «Комп'ютерна інженерія»
(шифр і назва напряму підготовки, спеціальності)

Самко В. В.
(прізвище та ініціали)
Керівник к.т.н доц. Крупельницький Л. В.
(прізвище та ініціали)

м. Вінниця
2020 року

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Освітній рівень магістр
Спеціальність 123 Комп'ютерна інженерія

ЗАТВЕРДЖУЮ
Завідувач кафедри обчислювальної техніки
_____ О.Д. Азаров.
«____» _____2020 року

З А В Д А Н Н Я
НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТА

Самка Вадима Валерійовича
(прізвище, ім'я, по батькові)

1. Тема роботи: «Методи розпізнавання зображень в багатоканальній комп'ютерній системі моніторингу цифрового телевізійного мовлення.»
керівник роботи Крупельницький Л. В., к.т.н, доц. затверджена наказом Вінницького національного технічного університету від №214 від 25.09.2020 р.
2. Строк подання студентом роботи 23 листопада 2020 року
3. Вихідні дані до роботи Методи та алгоритми розпізнавання заданих зображень у відеофайлах, що враховують специфіку багатоканальних цифрових систем телевізійного мовлення.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
   1) розробка специфічних методів розпізнавання зображень у відеофайлах, які враховують специфіку багатоканальних систем цифрового телебачення;
   2) розробка та тестування алгоритму та програмного забезпечення для опрацювання багатоканальних відеосигналів.
5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Блок-схема алгоритму роботи програми, візуальне представлення роботи програми, графіки порівняння роботи алгоритмів пошуку та опису особливої точки.

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Розділ 1. Аналіз методів розпізнавання відеозображень в багатоканальній комп'ютерній системі	Крупельницький Л. В.		

моніторингу цифрового телебачення			
Розділ 2. Розробка методу розпізнавання зображень у відеофайлах для багатоканальної системи комп'ютерного моніторингу	Крупельницький Л. В.		
Розділ 3. Розробка та тестування алгоритму та програмного забезпечення	Крупельницький Л. В.		
Розділ 4. Розрахунок економічної доцільності створення програми ущільнення багатоканального відеозображення для систем комп'ютерного моніторингу	Руда Л.П.		

7. Дата видачі завдання_____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів випускної кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	*Опрацювання літератури з теми дослідження*	10.09.2020	
2	*Складання плану дипломної роботи*	19.09.2020	
3	*Написання першого та другого розділів дипломної роботи*	24.10.2020	
4	*Проведення експериментальної роботи*	15.11.2020	
5	*Написання третього та четвертого розділів дипломної роботи*	23.11.2020	

Магістрант _____ Самко В.В
Керівник роботи _____ Крупельницький Л.В.
Рецензент _____ Куперштейн Л.М.

АНАТОЦІЯ

Дана магістерська кваліфікаційна робота присвячена створенню алгоритма та програмного забезпечення для розпізнавання зображень в багатоканальній комп'ютерній системі моніторингу цифрового телебачення.

За основу було обрано алгоритм виявлення та опису особливих точок BRISK, та покращено його, опираючись на алгоритм FAST.

Програмне забезпечення надає можливість одночасно опрацьовувати включно до чотирьох каналів відеоданних. Окрім цього користувачеві надається зручний інтерфейс для можливості здійснення необхідних налаштувань для розпізнавання, а також додаткові функції, такі як: покадрове виведення відео, з можливістю зробити знімок кадру та використовувати його для розпізнавання, та інше.

У дипломній роботі розроблено: алгоритм та програму для паралельного багатоканального розпізнавання відеоданих.

ABSTRACT

This qualification work is devoted to the creation of an algorithm and software for image recognition in a multi-channel computer digital television monitoring system.

The algorithm for detecting and describing BRISK special points was chosen as a basis, and it was improved based on the FAST algorithm.

The software provides the ability to process up to four channels of video data. In addition, the user is provided with a user-friendly interface for the ability to make the necessary settings for recognition, as well as additional features such as: frame-by-frame video output, with the ability to take a picture of the frame and use it for recognition, and more.

In diploma work was developed: algorithm and program for parallel multichannel video data recognition.

ЗМІСТ

					08-23.МКР.012.00.000 ПЗ		
Змн.	*Лист*	*№ докум.*	*Підпис*	*Дата*			
Розроб.		*Самко В. В.*			*Методи розпізнавання зображень в багатоканальній комп'ютерній системі моніторингу цифрового телевізійного мовлення.» Пояснювальна записка.*	*Літ.*	*Арк.* 6 / *Акрушів* 107
Перевір.		*Крупельницький Л. В.*					
Реценз.		*Куперштейн Л. М.*				1КІ – 19м	
Н. Контр.		*Швець С. І*					
Затверд.		*Азаров О. Д.*					

ВСТУП

В даний час з розвитком обчислювальних систем та інформаційних технологій зростає популярність комп’ютерних систем моніторингу й архівування, як в промисловості і науці, так і в повсякденному житті. Як наслідок зростає потреба в ефективних методах обробки інформації, що надходить у вигляді відео даних.

Ефективна обробка і використання вхідної відео і аудіо інформації дозволяє значно підвищити продуктивність і розширити коло застосування даних систем. Підтвердженням вищесказаного є повсюдний розвиток і використання баз даних, систем розпізнання текстів, охоронних систем заснованих на розпізнанні зображень у відеофайлах, і т.д. Останнім часом все більша увага приділяється системам, що використовують машинний зір в якості основного джерела інформації — це призвело до виникнення потреби в нових алгоритмах обробки і розпізнавання зображень у відеофайлах.

Не дивлячись на вище вказане, завдання розпізнавання образів до сих пір не вирішена в повному обсязі особливо для багатоканальних моніторингових систем реального часу. Однак, в рамках істотних обмежень, є методи, що дозволяють наблизиться до її вирішення.

Актуальністю проблеми є необхідність розробки для систем моніторингу ефективного багатоканального методу розпізнавання зображень у відеофайлах, таких як рекламні блоки, заставки телерадіопередач, тощо.

Об’єктом дослідження є процеси моніторингу багатоканального телевізійного мовлення за допомогою спеціалізованих комп’ютерних систем.

Предметом дослідження є методи та алгоритми розпізнавання заданих зображень у відеофайлах, що враховують специфіку багатоканальних цифрових систем телевізійного мовлення.

Метою роботи є розробка методів і програмних засобів багатоканального розпізнавання зображень у відеофайлах.

Для досягнення цієї мети необхідно вирішити такі **завдання**:

— проаналізувати наявну інформацію щодо методів розпізнавання зображень у відеофайлах;

— розробити специфічні методи розпізнавання зображень у відеофайлах, які враховують специфіку багатоканальних систем цифрового телебачення;

— розробити та протестувати алгоритм та програмне забезпечення для опрацювання багатоканальних відеосигналів;

— обгрунтувати основні техніко-економічні показники і ефективність виконаної розробки.

Методами дослідження для досягнення поставленої в роботі мети є:

— системний аналіз, який застосовується для дослідження методів розпізнавання зображень у відеофайлах;

— об’єктно-орієнтовані методи програмування, які використовуванні при розробці програмного забезпечення;

— методи порівняльного комп’ютерного тестування з метою оцінки ефективності запропонованих алгоритмів.

Наукова новизна отриманих результатів магістерської роботи полягає у тому, що дістав подальшого розвитку метод виділення особливих точок зображення, який в системах паралельного потокового багатоканального опрацювання даних пришвидшує розпізнавання заданих об’єктів при збереженні якості їх розпізнавання.

Практичне значення одержаних результатів магістерської роботи:

— розроблено алгоритм та програму для паралельного багатоканального розпізнавання відеоданих, який має самостійне значення;

—розроблене програмне забезпечення інтегровано у систему моніторингу центрального та регіонального цифрового телерадіомовлення.

Апробація результатів відбулась на XLVIII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії.

Публікація за темою роботи Л. В. Крупельницький, В. В. Самко, М. В. Жилін Комп’ютерна система моніторингу телевізійного мовлення [Електронний ресурс] / Л. В. Крупельницький, В. В. Самко, М. В. Жилін. — 2019. — режим

доступу до ресурсу: https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki 2019/paper/view/6876 [1].

1 АНАЛІЗ МЕТОДІВ РОЗПІЗНАВАННЯ ВІДЕОЗОБРАЖЕНЬ В БАГАТОКАНАЛЬНІЙ КОМП'ЮТЕРНІЙ СИСТЕМІ МОНІТОРИНГУ ЦИФРОВОГО ТЕЛЕБАЧЕННЯ

1.1 Огляд та аналіз особливостей розпізнання відеозображень для типових колірних моделей

У комп'ютерному зорі кольори представлені за допомогою математичних моделей, що описують їх як упорядковані набори чисел, як правило, у вигляді трьох або чотирьох значень або кольору компоненти. Ці математичні подання кольорів відомі як колірні моделі. Колірна модель намагається моделювати те як сприймає зорова система людини колір предмета при різних умовах освітленності та з різним фоном.

У технічному плані колірна модель перетворює виміряні значення, наприклад значення зразка, отримані колориметром та оточуючі його кольори, співвідносячи їх з перцептивними атрибутами кольору. Це математична модель, яка використовується для прогнозування сприйняття кольору в різних умовах, наприклад на вулиці чи в приміщенні, на ЕПТ моніторі або з друкованого документа. Існують різні типи колірних моделей, наприклад RGB, що є найпоширенішою, а також спеціалізовані моделі, такі як колірний простір Манселла та колірна модель CIECAM02 [2].

Колірна модель часто корисна для налаштування передачі кольорів зображень, щоб ті виглядали однаково при перегляді в різних умовах освітлення. Ця здатність колірної моделі робить її корисною при розробці універсального методу зберігання зображень, та являється одним з найважливіших елементів їх обробки, що дозволяє ідентифікувати збережені цифрові зображення [2,3].

Колірні системи можуть бути двох видів. Деякі колірні системи базуються на технічних характеристиках, таких як довжина хвилі, і сформовані, щоб допомогти людям вибирати кольори, наприклад система Манселла. Інші колірні системи сформовані для полегшення обробки даних в ЕОМ, наприклад модель RGB.

Модель RGB — це колірна модель без будь-якої прив’язаності до технічно визначених колірних систем, її прийнято розглядати як довільну колірну систему.

Коли колірна модель пов’язана з точним описом того, як компоненти мають інтерпретуватися, отриманий набір кольорів називається колірним простором [3]. Технічне поєднання колірної моделі та еталонного колірного простору дає в результаті певний слід в межах еталонного колірного простору, що є відомим як гама. В цифрових колірних дисплеях, змінюючи яскравість основних кольорів системи RGB, змінюється і колір, що виводиться, повний набір кольорів, який можна відтворити на дисплеї, називається колірною гамою дисплея [4]. Гамма в поєднанні з колірною моделлю визначає новий колірний простір, наприклад Adobe RGB і sRGB — це два різні колірні простори, засновані на загальні моделі RGB.

У кольорознавстві існують колірні простори, в яких сприймається різниця між двома кольорами, що лінійно пов'язані з евклідовою, або лінійною, відстанню між ними. Вони відомі як однорідні колірні простори або колірні простори однорідної шкали кольоровості (UCS), наприклад колірні простори L * a * b * та L * u * v *. Модель UCS є математичною моделлю, яка відповідає чутливості людського ока з комп’ютерною обробкою. Колірні простори, в яких різниця кольорів сприйняття людиною не може бути прямо виражена евклідовою відстанню, відомі як колірні простори, що не належать до UCS, такі як RGB та колірні простори XYZ [5]. Кольори, які вибрані з однорідних колірних просторів є оптимізованими для візуального розпізнавання комп’ютерною обробкою.

В даний час існує ряд різних типів колірних просторів, які використовуються в полі комп'ютерного зору. Використання певного колірного простору залежить від програми, для якої він використовується.

1.1.1 Аналіз колірної компонентної моделі RGB

Колірна модель RGB є одним з основних колірних форматів і найпростіших систем моделювання кольору, вона базується на адитивних кольорах: червоному, зеленому та синьому. У цій моделі всі три кольори суміщаються разом, і їх світлові спектри перемішуються різними способами, відтворючи широкий спектр кольорів (рисунок 1.1) [4,5,8].

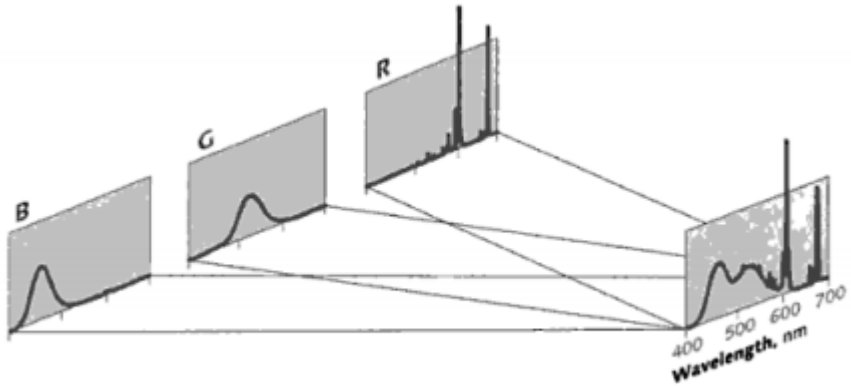


Рисунок 1.1 — Адитивне відтворення у відеосигналах

Назва моделі походить від ініціалів трьох основних кольорів, а саме: R — red (червоний), G — green (зелений), B — blue (синій). Основною метою розробки колірної моделі RGB було представлення і виведення зображень в електронних системах, хоча це і раніше використовувалось, наприклад у звичайній фотографії [4,8].

Модель RGB в основному використовується в апаратно-орієнтованих додатках з виведенням на кольорові монітори. До настання електронної ери колірна модель RGB вже мала грунтовну теорію, засновану на сприйнятті людьми кольорів. Вибір основних кольорів пов'язаний з фізіологією людського ока [8].

Колір у моделі RGB описується на значеннях кількості червоного, зеленого та синього кольорів, що поєднуються між собою. З цієї моделі можна отримати довільний колір шляхом зваженої суми з трьох компонент:
C = r R + g G + b B, де r, g і b є значеннями колориметра R, G і B компонент, а C є колірним стимулом [7,8].

Колірну модель RGB також можна трактувати як тривимірний одиничний куб (рисунок 1.2), в якому три первинні форми утворюють координатні осі. Значення RGB є позитивними і лежать в межах діапазону [0, Cmax]; для більшості цифрових зображень Cmax = 255. Значення RGB часто нормуються до інтервалу [0,1] для формування колірного простору, який може бути представлений одиничним кубом [4,8].

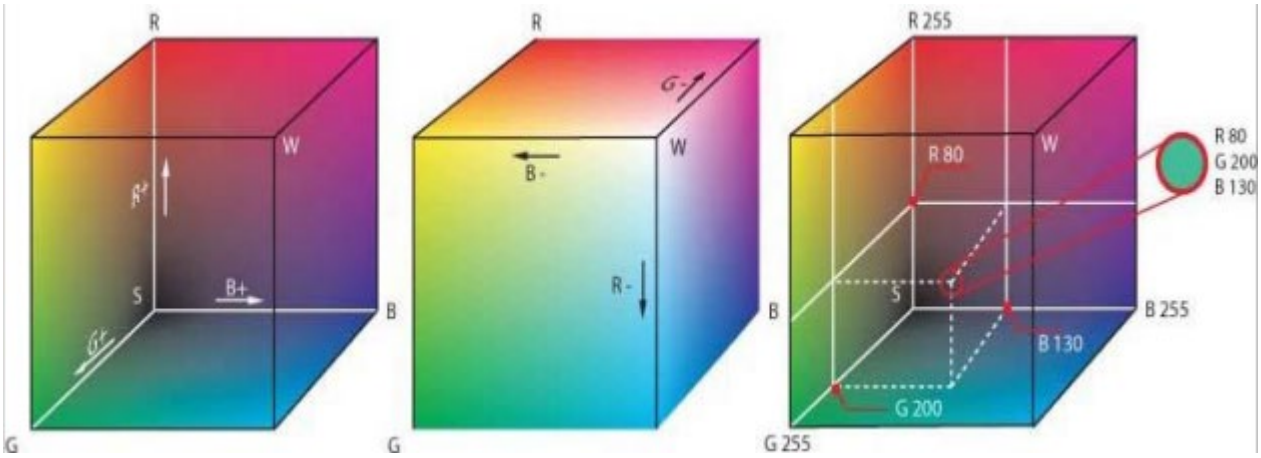


Рисунок 1.2 — модель RGB-куба

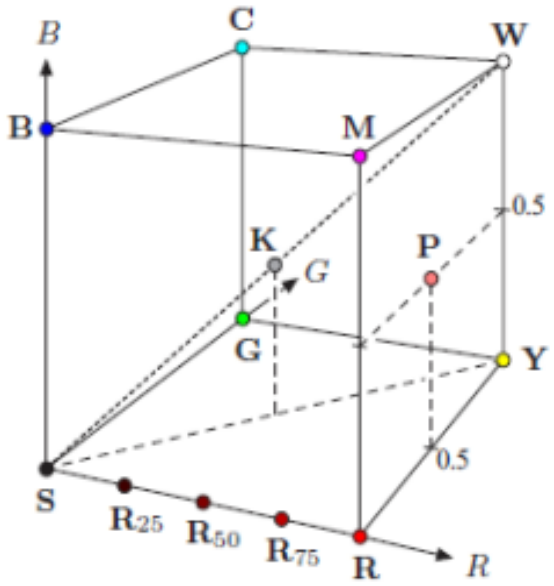
Основні кольори червоний (R), синій (B) та зелений (G) утворюють систему координат. “Чистий” червоний колір (R), зелений (G), синій (B), блакитний (C), пурпуровий (M) та жовтий (Y) лежать на вершинах куба.

1.1.2 Особливості колірного простору RGB та sRGB в задачах розпізнавання

Колірний простір RGB — це не колірний простір UCS, що використовує властивості трьох основних кольорів: червоного, зеленого та синього. Цей колірний простір базується на колірні моделі RGB. У цьому колірному просторі ці основні кольори поєднуються, використовуючи принципи додавання та змішування для утворення нових кольорів. Колірний простір RGB є найпоширенішим колірним представленням, його було прийнято у великій кількості пристроїв введення / виведення . Двома найпоширенішими поданнями кольорів RGB є NTSC - (RGB) та CIE - (RGB) [4,8]. Створення колірного простору є компромісом між наявністю якісних початкових даних, сигнального шуму та кількості цифрових рівнів, що підтримуються типом файлу. Математично модель RGB, пояснюється наступним чином: R, G і B утворюють вісь, а тривимірний колірний простір, представлений кубом (рисунок 1.3) [4,8].

У цьому загальному кубі колірного простору RGB чорний (S) розташований у декартових координатах (0, 0, 0) і білий (W) на (1, 1, 1). Інші кути куба мають три основні кольори червоного, синього та зеленого на (1, 0, 0), (0, 1, 0) та (0, 0, 1) відповідно. Пари зважених сум RGB, також відомі як вторинні кольори (блакитний,

пурпуровий і жовтий), розташовані в інших трьох кутах куба. Більшість програм, що використовують колірні простори RGB, обмежують значення діапазоном (0, 1) [4,5,8].



Point	Color	RGB Value R	G	B
S	Black	0.00	0.00	0.00
R	Red	1.00	0.00	0.00
Y	Yellow	1.00	1.00	0.00
G	Green	0.00	1.00	0.00
C	Cyan	0.00	1.00	1.00
B	Blue	0.00	0.00	1.00
M	Magenta	1.00	0.00	1.00
W	White	1.00	1.00	1.00
K	50% Gray	0.50	0.50	0.50
R_{75}	75% Red	0.75	0.00	0.00
R_{50}	50% Red	0.50	0.00	0.00
R_{25}	25% Red	0.25	0.00	0.00
P	Pink	1.00	0.50	0.50

Рисунок 1.3 — Метематичне представлення RGB

Стандартний колірний простір RGB (sRGB) — це стандартизована специфікація кольорів RGB зображення з точки зору віртуального дисплея. Це нелінійний колірний простір, який вже був стандартизований Міжнародною електротехнічною комісією (IEC) як IEC 61966-2-1. Спочатку колірний простір був створений спільно HP і Microsoft у 1996 році для використання на моніторах, принтерах та Інтернеті. Він приймається як загальне призначення колірний простір для споживчого використання, де вбудовування просторового профілю може не бути зручний для розміру файлу або через проблеми з сумісністю. Він широко використовується у Всесвітній павутині та враховує специфікацію sRGB зображення, що переглядаються в офісному середовищі. Нижче наведені координати кольоровості для простору sRGB щодо CIE XYZ (пояснено пізніше) та діаграма кольоровості sRGB (рисунок 1.4) [4,5,8].

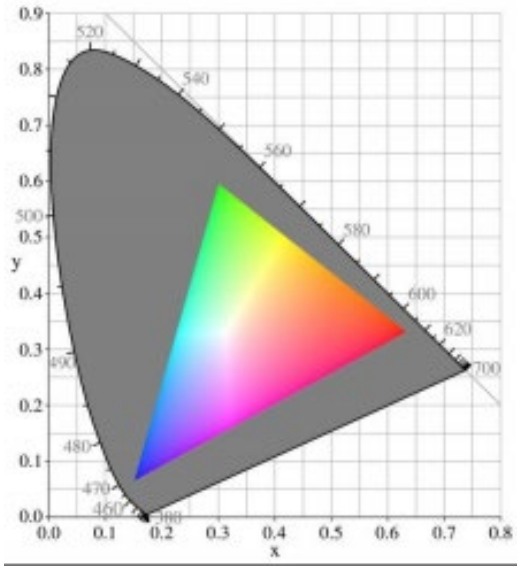


Рисунок 1.4 — Координати кольоровості sRGB

1.1.3 Можливості розпізнавання в колірному просторі NTSC RGB

Колірний простір NTSC був розроблений Комітетом національної телевізійної системи Північної Америки. Це був, мабуть, єдиний колірний простір, який реалізовував повноцінний поділ між яскравістю та колірними даними [15]. Цей колірний простір перетворив систему координат RGB на колірний простір YIQ (рисунок 1.5), який був ближче до людського сприйняття, де «Y» — це яскравість, «I» — фаза і «Q» означає квадратуру. Y-компонент представляє інформацію про яскравість; I і Q представляють колірну інформацію [4-6].

Перетворення RGB в YIQ відбувалось по таким формулам:

Y = 0,29YR + 0,587G + 0,114B

I = 0,596R - 0,275G - 0,321B

Q = 0,212R - 0,523G + 0,311B

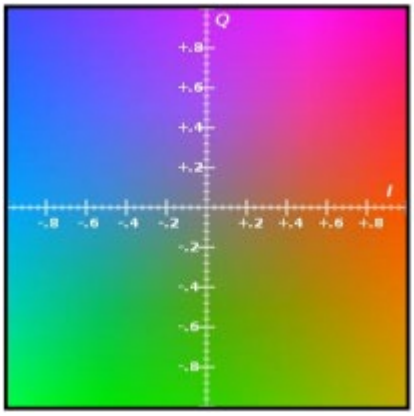


Рисунок 1.5 — Колірний простір YIQ при Y = 0,5

1.1.4 Розпізнавання в колірному простору CIE1931 / CIE RGB / CIE XYZ

У сприйнятті кольорів CIE 1931 XYZ є одним із перших математично визначених колірних просторів [46]. Його було отримано в результаті різних експериментів, які були проведені В.Девідом Райтом та Джоном Гільдією [48]. Їх результати експериментів були об'єднані в одну специфікацію, колірний простір CIE RGB. А пізніше від CIE RGB було виведено колірний простір CIE XYZ. У цьому колірному просторі набір виміряних функцій збігу кольорів трансформується, щоб створити набір кривих, які є зручніше у використанні. Ці криві називаються x, y і z як показано на рисунку 1.6. [4,7,8]

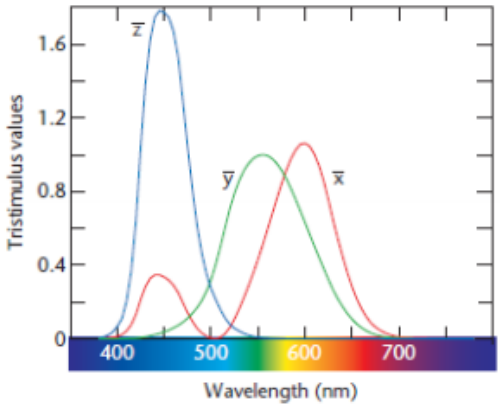


Рисунок 1.6 — Рекомендовані CIE функції узгодження кольорів

У всьому видимому спектрі цей набір кривих є додатним. Крива y може використовуватися для порівняння сприйнятої яскравості вимірюваного кольору. Функція z дорівнює нулю для більшості довжин хвиль. У цьому колірному просторі замість трьох тристимульних значень S, M та L ми маємо набір тристимульних значень x, y та z, які були обчислені через функції відповідності кольорів CIE для x, y, i, z [4,7,8].

1.2 Аналіз роботи алгоритмів розпізнавання відеозображень

Створюючи опис об'єкта зображення, центральне значення має виділення частин, які є характерними для зображення, таких як кути, краї, ділянки, що відповідають граничним значенням інтенсивності тощо, підлягають алгоритмам, що підкреслюють такі особливості.

Особлива точка m — є точкою образу, околиця o (m) якої можна виділити з околиці будь-якої іншої точки образу o (n) в деякій іншій околиці особливої точки o2 (m) [9,10].

Особливі точки повинні мати наступні особливості.

Повторюваність (repeatability) — особлива точка знаходиться в тому ж місці сцени або об'єкта зображення, незважаючи на зміни в освітленні і в точці огляду.

Відмінність / інформативність (distinctiveness / informativeness) — околиці особливих точок повинні мати великі відмінності один від одного, так, щоб можливо було виділити і зіставити їх.

Локальність (locality) — для зниження чутливості фотометричних і геометричних спотворень між двома зображеннями, особлива точка повинна займати невелику область зображення.

Кількість (quantity) - для виявлення невеликих об'єктів число виявлених особливих точок має бути досить великим. Кількість виявлених особливих точок має визначатися адаптивно. Щоб забезпечити компактне представлення особливих точок повинна показувати інформаційний вміст зображення.

Точність (accuracy) — певні особливі точки повинні точно обмежуватися, як в зміненому масштабі, так і в оригінальному.

Ефективність (efficiency) — час на вираховування особливих точок на зображенні має бути задовільним в критичних за часом додатках.

Детектор особливих точок (Feature detector) — це метод вилучення особливих точок з зображення. Детектор забезпечує інваріантність знаходження одних і тих же особливих точок щодо різних перетворень [9-11].

Детектори особливих точок поділяються на такі основні типи (рисунок 1.7).

Edge detector — основані на алгоритмах виділення границь.

Corner detector — основані на алгоритмах виділення кутів.

Blob detector — алгоритми споріднені з візуальними модулями, що спрямовані на виявлення ділянок на зображенні, які є яскравішими або темнішими ніж оточуючі.

Region detector — група детекторів визначає ROI (область) на зображенні, вибираючи ключові точки та потім створює патчі (окремі ділянки, що перекриваються) навколо них.

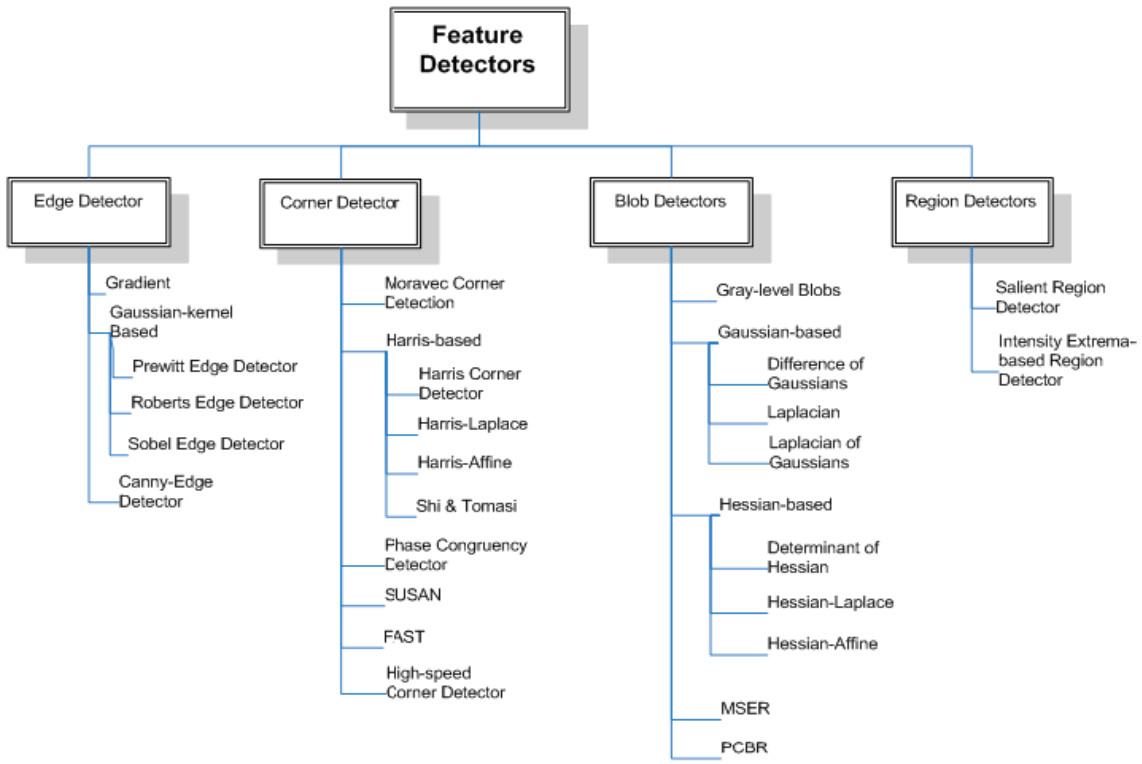


Рисунок 1.7 — Базові типи детекторів особливих точок

Дескриптор особливих точок (Feature descriptor) — ідентифікатор особливої точки, що виділяє її з решти множини особливих точок. У свою чергу, дескриптори повинні забезпечувати інваріантність знаходження відповідності між особливими точками щодо перетворень зображень. В результаті побудови дескрипторів для вихідного набору особливих точок виробляється безліч векторів ознак [9,11,13].

Існують такі основні типи дескрипторів особливих точок (рисунок 1.8).

Shape descriptors — описують форму об’єктів та, в свою чергу, поділяються на дескриптори контурів та дескриптори областей зображення. Різниця між ними в тому, що ознаки виділяються з контуру фігури, в першому випадку, або використовується вся область фігури — як в другому. А ті в свою чергу поділяються на структурні підходи, ті що представляють форму сегментами або перетинами та глобальні підходи, ті що представляють форму в цілому.

Color descriptors — опис базується на гістограмі об’єктів, виділивши розподіл кольорів в патчі чи області зображення.

Texture descriptors — описують текстуру патчів. Текстура, в цифровій обробці, — це кожне цифрове зображення, що складається з повторюваних елементів.

Motion descriptors — описують рух об’єктів та використовується для відстеження їх у відеопослідовностях або для опису загального руху об'єкта в сцені

Результатом роботи детектора є набір особливих точок, для яких повинен бути створений математичний опис. Вхідними даними дескриптора є зображення та ряд особливих точок, виділених на конкретному зображенні. Результатом роботи дескриптора є набір векторів ознак для вихідного набору точок. Також існують дескриптори, які одночасно вирішують дві задачі — знаходження особливих точок та їх побудова. А окремі детектори особливих точок можна поєднувати з кількома окремими типами відповідних дескрипторів [9-11].

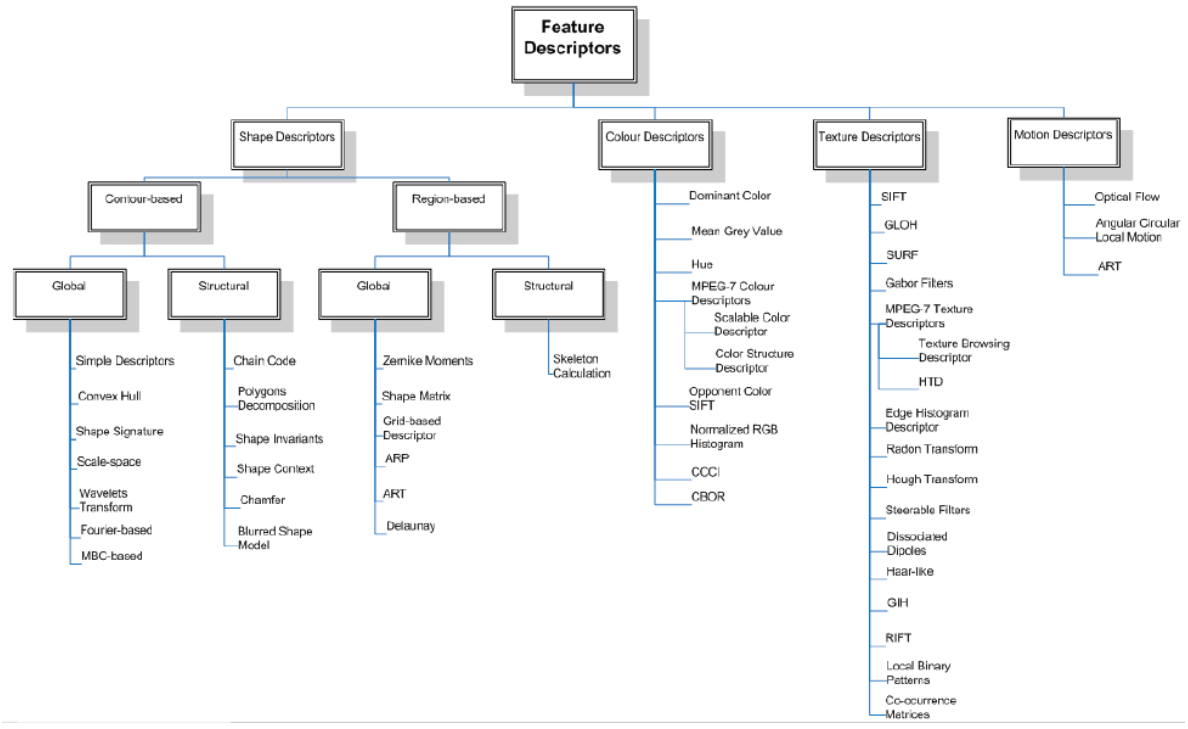


Рисунок 1.8 — Базові типи дескрипторів особливих точок

Розглянемо та порівняємо деякі основні алгоритми для розпізнавання.

Алгоритм SIFT (Speeded-Up Robust Features) виконує два основні етапи в частині детекції особливих точок:

— виділення екстремумів масштабованого простору, в цій операції застосовується різниця гаусіанів (DoG, Difference of Gaussian) для ідентифікації особливих точок інваріантних до масштабу, наступним виконується локальна перевірка екстремумів із сусідніми пікселями (рисунок 1.9);

— локалізація особливих точок, яка відкидає низькоконтрастні точки, а потім усуває некрайові точки на основі матриці Гессе [13].

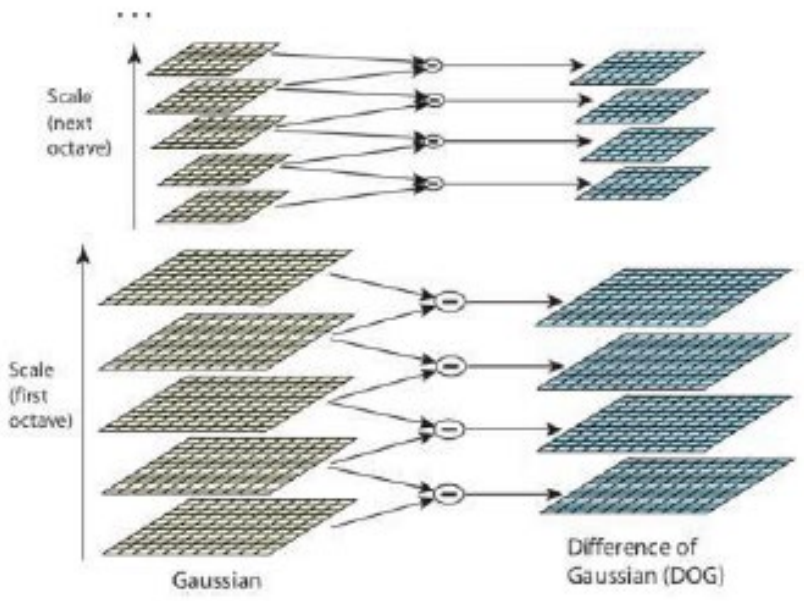


Рисунок 1.9 — Обчислення різниці гаусіанів

Для побудови дескриптора алгоритм виконує два подальші кроки:

— призначення напрямку, алгоритм формує гістограми напрямків з місцевих градієнтів для визначення домінуючого напрямку особливої точки;

— формування дескриптору особливої точки, в якому власний вектор будується на основі положень особливих точок та локальних областей навколо них, і нарешті дескриптори нормалізуються для покращення інваріантності освітленості [13].

Алгоритм SURF (Speeded-Up Robust Features) функціонально наслідує SIFT, розроблявся, як швидша та надійніша альтернатива попереднім детекторам. Він використовує цілісні зображення та спрощені ядра фільтрів в порівнянні з SIFT через швидкий детектор Гессе, що заснований на 2D вейвлетах Хаара. Дескриптор поєднує в собі локальну інформацію про градієнт, як SIFT, та 2D вейвлет Хаара в

локальній області та вікна навколо особливих точок для наближення градієнтів [14,15].

Алгоритм ORB — це орієнтований FAST і Rotated BRIEF — поліпшена версія і комбінація FAST-точкового детектора і двійкових дескрипторів BRIEF.

Мета створення BRIEF — дескриптора полягала в тому, щоб одні і ті ж частини зображення, взяті з різних точок зображення, були розпізнані. Завдання полягало в тому, щоб мінімізувати кількість виконаних обчислень. Алгоритм розпізнавання зводиться до побудови випадкового лісу (randomize classification trees) або наївного Байєсівського класифікатора на деякому навчальному наборі зображень і подальшою класифікацією тестових областей зображення. У спрощеному варіанті може використовуватися метод найближчого сусіда для пошуку найбільш схожої ділянки в навчальній вибірці. Невелика кількість операцій забезпечується за рахунок подання характеристичного вектора у вигляді двійкових рядків і використання відстані Хеммінга як міри схожості [16].

Алгоритм BRISK заснований на детекторі AGAST, він може отримувати особливості із зображення при різних його масштабах. Для дескриптора він використовує концентричну схему вибірки кілець для отримання сірих значень їх сусідів та обробки локальних градієнтів інтенсивності, щоб отримати напрямок особливої точки. Потім він формує двійковий дескриптор, що порівнює інтенсивність між парами за зразком [9,12].

Алгоритм AKAZE фокусується на багатомасштабному виявленні особливих точок, використовуючи нелінійні масштабовані простори. Пошук максимуму в просторовому розташуванні масштабованих зображень виконується детермінантом матриці Гессе.

Дескриптором виступає M-LDB (Modified-Local Difference Binary), який використовує градієнт та інтенсивність від стадії виділення особливої точки. Він базується на алгоритмі BRIEF, що виконує операції над середнім значенням площ замість пікселів. Він включає значення інтенсивності, а орієнтація особливих точок подібна до такої в алгоритмі KAZE [14-17].

1.3 Огляд і вибір засобів для програмного розпізнавання відеозображень

Emgu CV — кросплатформна «обгортка» для .NET бібліотеки обробки зображень OpenCV (основна бібліотека), Emgu CV також називають бібліотекою машинного зору. Вона використовується для вирішення різноманітних завдань пов'язаних з 2D графікою, розпізнаванням осіб і предметів на фото, розпізнання осіб і предметів на відео та ін. Emgu CV може бути використана на декількох різних мовах, включаючи C #, VB.NET, C++ і IronPython .

Emgu CV повністю написана на C#. ЇЇ перевага в тому, що вона може працювати на будь-якій платформі: Linux, Mac OS X, IOS і Android. Код є кросплатформним. [18]

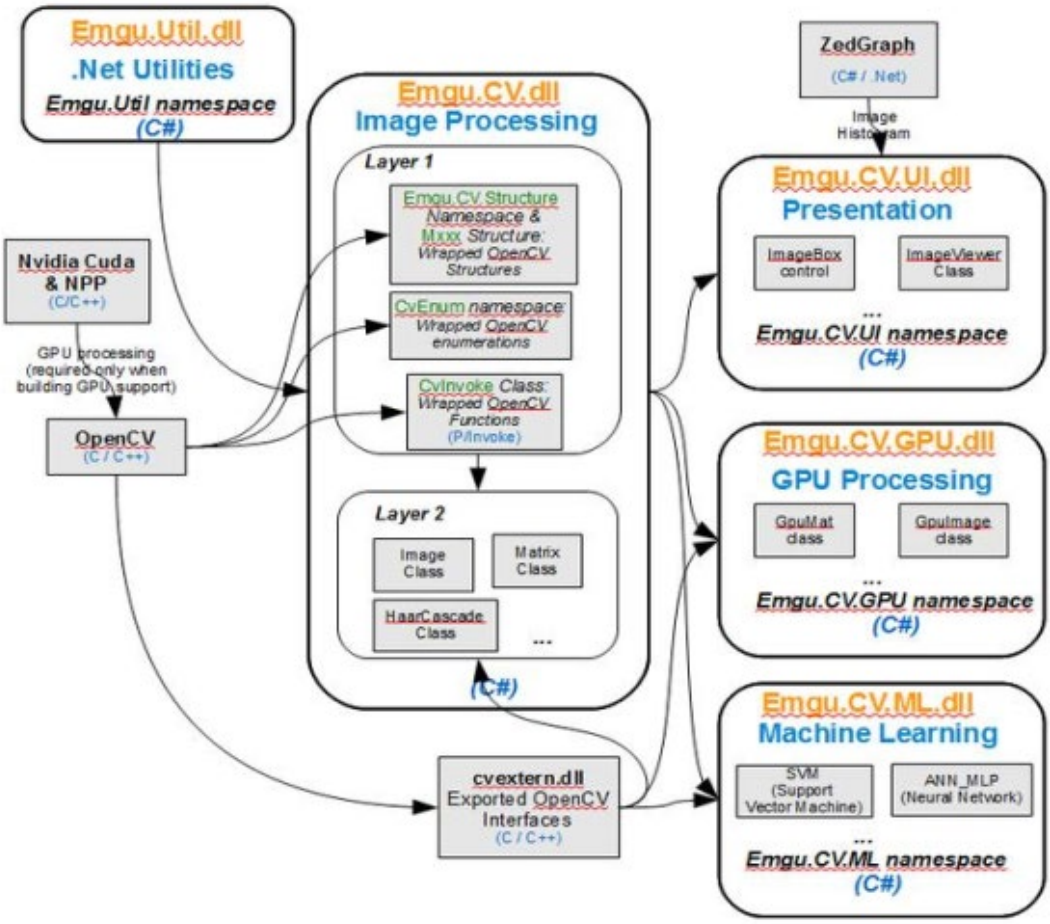


Рисунок 1.9 — Огляд функціонально набору бібліотек Emgu CV

AForge.NET є основою C# і призначена для розробників і дослідників в області комп'ютерного зору і штучного інтелекту — обробка зображень, нейронні мережі, машинне навчання, робототехніка і т.д. Бібліотека включає кілька

основних компонентів. AForge.Imaging — бібліотека підпрограм для обробки зображень і фільтрів. AForge.Vision — бібліотека комп'ютерного зору. AForge.Video — набір бібліотек для роботи з відео інформацією. AForge.Neuro — бібліотека для виконання різноманітних дій і операцій з нейронними мережами. AForge.Genetic — бібліотека підпрограм для використання генетичних алгоритмів для вирішення різних завдань. AForge.Fuzzy — бібліотека для роботи з нечіткою логікою. AForge.Robotics — бібліотека, забезпечує підтримку деяких методів, застосовуваних у сфері робототехніки. AForge.MachineLearning — бібліотека для роботи з елементами машинного навчання [19].

Accord.NET є основою для наукових обчислень .NET. Структура ґрунтується на AForge.NET, призначена для обробки зображень, використовуються нові інструменти і бібліотеки. Ці бібліотеки охоплюють широкий спектр наукових обчислювальних додатків, таких як статистична обробка даних, машинне навчання, розпізнавання образів. Структура пропонує велику кількість імовірнісних розподілів, перевірки гіпотез і підтримку для більшості популярних методів вимірювання продуктивності [19]

OpenCV бібліотека машинного зору з відкритим вихідним кодом. Бібліотека написана на C і C++, але поставляється також і для інших мов, таких як: Python, Java, Ruby, Matlab, Lua і т.д. та може виконуватись на великій кількості операційних систем таких як: Linux, Windows, Mac OS X, Android та інші. OpenCV була розроблена для ефективних обчислень і з орієнтуванням на додатки реального часу. OpenCV написана на оптимізованому C і використовує переваги багатоядерних процесорів [20].

Головною метою OpenCV є надання простої у використанні інфраструктури комп'ютерного зору, яка допомагає швидко будувати досить складні додатки. OpenCV містить більше 500 функцій, які охоплюють багато областей, в тому числі моніторинг конвеєрної продукції, медична візуалізація, безпека, призначені для користувача інтерфейси, калібрування камер, стерео зір і робототехніка. Оскільки комп'ютерний зір і машинне навчання, часто йдуть поряд, OpenCV також містить комплексну бібліотеку машинного навчання загального призначення Machine

Learning Library (MLL). Ця підсистема орієнтована на статистичне розпізнавання образів і кластеризації. Бібліотека MLL вельми корисна для завдань комп’ютерного зору, але при цьому носить досить загальний характер і може використовуватися для будь-яких завдань машинного навчання [20].

З моменту свого альфа-релізу в січні 1999 року бібліотека OpenCV використовувалась в багатьох додатках, продуктах і дослідженнях, наприклад: "зшивання" зображень в супутникових системах і веб-картах, вирівнювання зображень, зниження шуму в медичних зображеннях, аналіз об'єктів, системи виявлення вторгнень, автоматичний контроль, системи безпеки, виробництво систем контролю, калібрування камер, широке військове застосування, безпілотні літальні , наземні і підводні апарати [20].

Можливості OpenCV:

— людино-машинна взаємодія;

— ідентифікація об'єктів;

— сегментація і розпізнавання;

— розпізнавання осіб;

— розпізнавання жестів;

— трекінг руху;

— структура руху;

— 3D трекінг. [7, 8, 11]

Отже, в даному розділі проаналізовано алгоритми та програмні засоби для їх реалізації.

2 РОЗРОБКА МЕТОДУ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ У ВІДЕОФАЙЛАХ ДЛЯ БАГАТОКАНАЛЬНОЇ СИСТЕМИ КОМП'ЮТЕРНОГО МОНІТОРИНГУ

За базову основу для розробки методу розпізнавання зображень у відеофайлах для багатоканальної системи комп'ютерного моніторингу було взято алгоритм BRISK.

2.1 Методи детекції особливих точок на зображенні

Для детекції особливих точок будується пірамідальний масштабований простір шляхом багаторазового зменшення вибірки вхідного зображення на n октав *ci* і n інтра-октав *di*. *n* октави створюються шляхом багаторазової напіввибірки початкового зображення. Інтра-октави *di* генеруються аналогічним чином, за винятком того, що перша інтра-октава *d*0 створюється шляхом заниженої вибірки початкового зображення в 1,5 рази. Кандидати на роль особливої точки вибираються з пірамідального простору за допомогою алгоритма FAST [21].

На наступному етапі обробки ці кандидати на роль особливої точки відфільтровуються, виконуючи відкидання точок, що не є трьохмірними максимумами в піраміді масштабного простору. Решта особливих точок інтерполюється до позиції субпікселя. Перші двомірні максимуми оцінюються шляхом введення двомірної квадратичної функції в 3 × 3-піксельну область у шарі піраміди особливої точки, а також прикордонних шарів. Далі для трьох максимумів будується одинична парабола, для знаходження трьохмірного максимуму у масштабованому просторі. З міркувань продуктивності цей крок виконується дещо інакше для октави *c*0, де використовується для створення інтра-октави *d*−1 [21].

Процес детекції схематично показаний на рисунку 2.1.

Опис особливої точки має значний вплив на подальшу ефективність алгоритму.

В нашому випадку опис особливої точки відбувається наступним чином.

Область навколо особливої точки розбивається на 60 ділянок p, це схематично продемонстровано на рисунку 2.2.

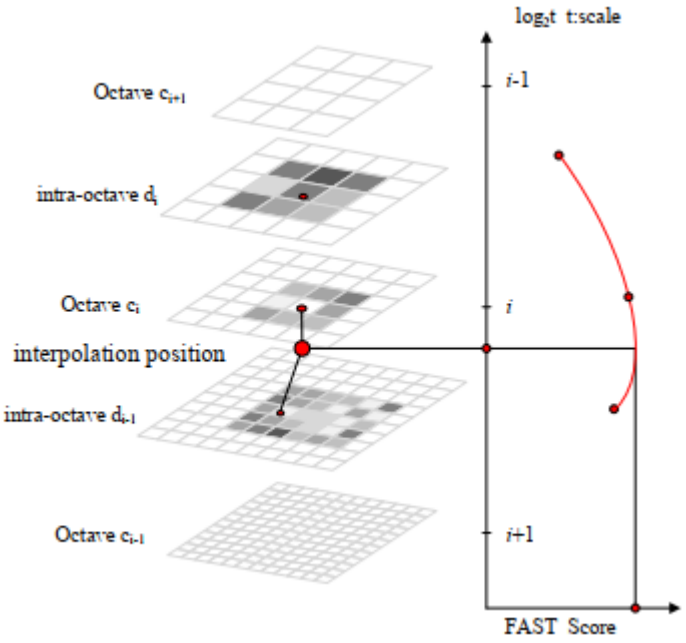


Рисунок 2.1 — Виявлення особливої точки в масштабованому просторі

При цьому визначений набір пар точок множини A формується з вибірки усіх пар точок, як показано у формулі 2.1.

$$\mathcal{A} = \{(p_i, p_j) \in R^2 \times R^2 | i < N \wedge j < i \wedge i, j \in \mathbb{N}\}, \quad (2.1)$$

де (p_i, p_j) пара точок множини A.

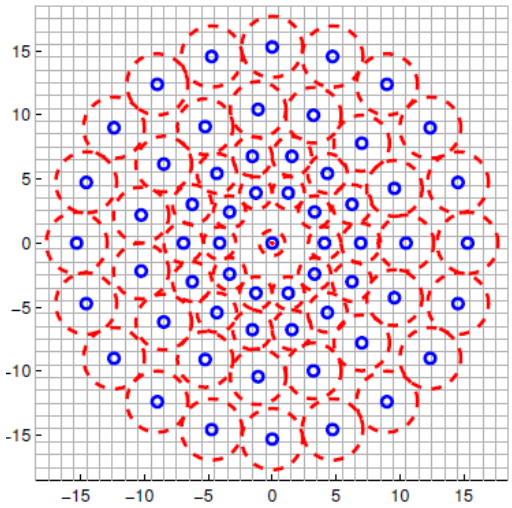


Рисунок 2.2 — Область обчислення дескриптора

Далі множина $\mathcal{A}$ розбивається на дві підмножини $\mathcal{S}$ та $\mathcal{L}$ по формулі 2.2 та формулі 2.3.

$$\mathcal{S} = \{(\mathrm{p_i}, \mathrm{p_j}) \in \mathcal{A} | \|\mathrm{p_j} - \mathrm{p_i}\| < \delta_{\max}\} \subseteq \mathcal{A} \ , \quad (2.2)$$

$$\mathcal{L} = \{(\mathrm{p_i}, \mathrm{p_j}) \in \mathcal{A} | \|\mathrm{p_j} - \mathrm{p_i}\| > \delta_{\min}\} \subseteq \mathcal{A} \ , \quad (2.3)$$

де $\delta_{\min} = 13.67\mathrm{t},\ \delta_{\max} = 9.75\mathrm{t},\ \mathrm{t} -$ розмір особливої точки.

Потім обчислюється середнє значення градієнта множини $\mathcal{L}$, по формулі 2.4.

$$g = \begin{pmatrix} g_x \\ g_y \end{pmatrix} = \frac{1}{|\mathcal{L}|} * \sum_{(p_i, p_j) \in \mathcal{L}} \left[(p_j - p_i) \frac{I(p_j, \sigma_j) - I(p_i, \sigma_i)}{\|p_j - p_i\|^2} \right] . \quad (2.4)$$

Дескриптор складається з бінарного рядка довжиною 512, заповненого результатами проведених тестів в множині $\mathcal{S}$, формула 2.5.

$$\mathrm{b} = \begin{cases} 1, \mathrm{I}\left(\mathrm{p_j^{\alpha}}, \sigma_\mathrm{j}\right) > I\left(\mathrm{p_i^{\alpha}}, \sigma_\mathrm{i}\right) \\ 0 \end{cases}; \ \forall \left(\mathrm{p_i^{\alpha}}, \mathrm{p_j^{\alpha}}\right) \in \mathcal{S}, \quad (2.5)$$

де $\mathrm{I}(\mathrm{p_i^{\alpha}}, \sigma_\mathrm{i})$ — інтенсивність околиці радіуса σ_i точки $\mathrm{p_i}$,

$\alpha = \mathrm{arctan2}(\mathrm{g_y}, \mathrm{g_x}) -$ кут напрямку градієнта g.

Значення відстані Хеммінга обчислюється для оцінки ступіня відповідності двох дескрипторів BRISK. Чим більше значення відстані Хеммінга, тим нижче ступінь відповідності дескрипторів [22].

2.2 Адаптація алгоритму BRISK для використання в багатоканальній системі

Алгоритм BRISK є досить точним у виявленні особливих точок, а також інваріантним до обертання та масштабу, при цьому їх обчислення вимагає досить помірних обчислювальних ресурсів, але займає досить багато часу, що не дуже підходить до використання в багатоканальній системі розпізнавання відеозображень. Тому є необхідність в використанні покращеного алгоритму.

Поєднання алгоритму виявлення та опису дає найкращий результат, оскільки кожен алгоритм має свої переваги та недоліки. Отже, коли обидва алгоритми поєднані, це дає найкращий результат із зменшенням часу на їх обчислення.

У алгоритмі BRISK виявлення особливих точок є досить повільним, але завдяки цьому зменшується затрата обчислювальних ресурсів, на приклад, на обертання та інваріантність масштабу [22,23]. Для вирішення проблеми часу в BRISK було віддано перевагу алгоритму FAST. Його назва говорить сама за себе (fast, з англійської, швидко), він є швидшим, ніж BRISK, обрахунки займають менше часу, але має недоліки, він не є інваріантним до масштабу і залежить від порогового значення. Також його принцип роботи оснований на виділенні кутових точках і не підходить для опису особливих точок [22,24].

Отже, беручи ці два алгоритми, пропонується комбінований підхід для отримання як виявлення, так і опису особливих точок об’єкта в значно коротший час. Перевагою такого підходу є не лише скорочення часу обрахунку, але й зменшення затрачуваних обчислювальних ресурсів [22,24].

У гібридних детекторі та дескрипторі, спочатку будується кутова точка, використовуючи область (круг) з 16 пікселів, які вилучені з набору зображень. Набір з усіх 16 пікселів розглядається як частина із трьох наборів для результату. Перший працює з більш яскравою частиною, другий — з темною частиною, третій — вибірково.

Для більш яскравої області інтенсивність значення «і» додається до порогового значення «t». Для більш темної області значення «і» береться зі значенням «t. Також при обчисленнях даних областей використовується діагональний підхід з усіх 16 пікселів. При діагональному підході, оцінку центральної точки можна показати схематично двома лініями (рисунок 2.6).

Похила лінія починається з 3-го пікселя по 11-й номер піксель, а інша похила лінія починається з 15 пікселів до 7 пікселів.

Один з нахилів йде справа наліво, кількість пікселів від 3 до 11, а інший — нахил зліва направо, тобто кількість пікселів (від 15 до 7). Де обидві діагоналі перетинаються, там і буде центральний кутовий піксель.

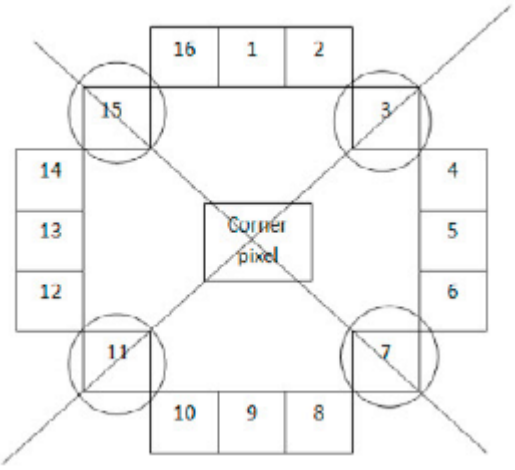


Рисунок 2.6 — Оцінка центральної точки

Отже, замість того, щоб знаходити всі пікселі, оцінюється лише чотири пікселі, тож результат обрахунків повинен з’явитися швидше. В кінці застосовується локальний градієнт на результуючому центральному пікселі кута, щоб знати чи відстань до пікселя нижче або вище порога значення.

2.3 Уточнення розташування особливих точок

Для кращого визначення місця розташування особливої точки, використовуємо три патчі 3 x 3, які включає алгоритм FAST для обрахунку особливої точки та 26 пікселів, що оточують її в масштабованій піраміді. Три патчі розміром 3 x 3 описуються трьома рівнями: перший патч на рівні октав нижче особливої точки, другий патч на рівні особливої точки та третій патч на рівні — вище [22, 24] . У патчі значення кожного пікселя дорівнює результату обрахованому по алгоритму FAST для цього пікселя та позначається sc (i, j), де i та j означають положення обрахованого пікселя щодо особливої точки у просторі. Наприклад, якщо особлива точка розташована на (1, 1, 0), результат sc (-1,0) буде розташована на (0, 1, 0). На рисунку 2.7 показаний приклад обрахованого патча.

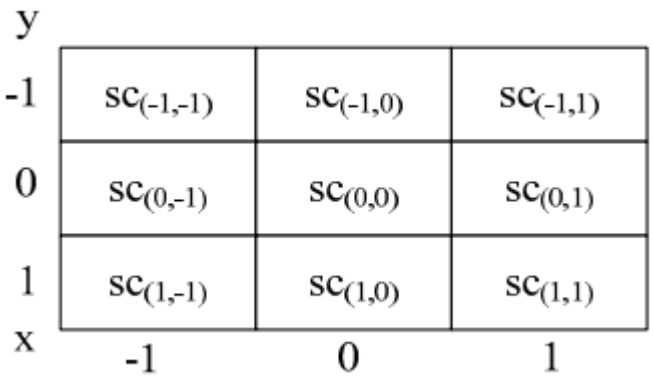


Рисунок 2.7 — Обрахований патч 3 х 3

Максимуми результатів алгоритму FAST з трьох рівнів октав позначимо через sc' (-1), sc' (0) та sc' (1), де індекс -1 та індекс 1 представляють відповідно рівень нижче та вище особливої точки. Позиції цих трьох максимумів позначаються $\Delta x'$ (-1), $\Delta x'(0)$, $\Delta x'(1)$, де $\Delta x'(0)= (\Delta x'(0), \Delta y'(0))$.

Для знаходження максимуму sc' та його положення x' з трьох розрахованих патчей, підбираємо параметри двомірної квадратичної функції, по формулі (2.6), щоб отримати місце розташування та результат квадратичного локального максимуму [23 - 25].

$$sc = ai^2 + bij + cj^2 + di + ej + f \quad . \tag{2.6}$$

Параметр φ двомірної квадратичної функції обраховується за допомогою методу найменших квадратів. Двомірна квадратична функція представлена формулою 2.7

$$sc = \varphi^T w(i,j) \quad , \tag{2.7}$$

де $\varphi = [a, b, c, d, e, f]^T$,

$w(i,j) = [i^2, ij, j^2, i, j, 1]^T$.

Методом найменших квадратів, розв’язується функція $E(\varphi)$, як показано в формулі 2.8.

$$E(\varphi) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \left(\varphi^T w(i,j) - sc_{i,j}\right)^2. \tag{2.8}$$

Значення похідної для φ з рівнянням 2.8, приймаємо рівною нулю, значення φ обчислюється за формулою 2.9.

$$\varphi = P^{-1} W sc, \tag{2.9}$$

де $P = \sum_{i=-1}^{1} \sum_{j=-1}^{1} w(i,j) w(i,j)^T$,

$W = [w(-1,-1)w(0,-1) \cdots w(1,1)]$

Тест другої похідної через детермінант матриці Гессе H (sc) відбувається по формулі 2.10 та формулі 2.11:

$$H(sc) = \begin{bmatrix} \frac{\partial^2 BC}{\partial^2} & \frac{\partial^2 sc}{\partial i\, \partial j} \\ \frac{\partial^2 sc}{\partial i\, \partial j} & \frac{\partial^2 sc}{\partial j^2} \end{bmatrix}, \tag{2.10}$$

$$\det\left(H(sc)\right) = \frac{\partial^2 sc}{\partial^2} \frac{\partial^2 sc}{\partial y^2} - \frac{\partial^2 sc}{\partial t\, \partial j} \frac{\partial^2 sc}{\partial j\, \partial t} < 0. \tag{2.11}$$

Локальний максимум для рівняння 2.6 знаходиться в місці $\Delta x'$, де часткові похідні відносно і та j дорівнюють 0. Часткові похідні визначаються по формулі 2.12 та формулі 2.13:

$$\frac{\partial sc}{\partial i} = 2ai + bj + d, \tag{2.12}$$

$$\frac{\partial sc}{\partial j} = 2cj + bi + e. \tag{2.13}$$

Значення часткових похідних приймаємо рівними нулю та вираховуємо розташування $(\Delta x', \Delta y')$, представлених (i, j) з рівнянь 2.12 та 2.13 відповідно, по формулі 2.14, формулі 2.15 та формулі 2.16.

$$\Delta \mathrm{x}' = (\Delta x', \Delta y')^T, \tag{2.14}$$

$$\Delta x' = \frac{be - 2cd}{4ac - b^2}, \tag{2.15}$$

$$\Delta y' = \frac{bd - 2ae}{4ac - b^2}. \tag{2.16}$$

Останнім етапом є обрахування максимуму по формулі 2.17.

$$\begin{gathered} \mathrm{sc}' = \varphi^T w(\Delta x', \Delta y') = \\ = a(\Delta x')^2 + b(\Delta x')(\Delta y') + c(\Delta y')^2 + d(\Delta x') + e(\Delta y') + f. \end{gathered} \tag{2.17}$$

2.3. Обчислення коефіцієнта масштабу та положення особливих точок

Алгоритм BRISK виявляє особливі точки у масштабованій моделі піриміди, що робить дескриптори інваріантними до масштабу. Цей метод є повільним і вимагає досить багато пам'яті [22,24].

Алгоритм FAST, який використовується для виявлення особливих точок об'єкта, не має основного напрямку. Для того, щоб дескриптор мав стійку інваріантність до масштабу, необхідно використовувати інформацію про глибину зображення RGB-D для обчислення коефіцієнта масштабу [35]. Коефіцієнт масштабу s вираховується по формулі 2.18.

$$s = max\left(0.2\ \frac{3.8 - 0.4max(2, d)}{3}\right), \tag{2.18}$$

де d — глибина точки пікселя, max (2, d) позначає фільтруючі пікселі з глибиною менше 2.

Локальна орієнтація особливої точки в алгоритмі BRISK базується на парі точок, між якими досить велика відстань. Згідно з [22], це показує, що інваріантність точки до обертання є досить слабкою.

Особливі точки, виявлені алгоритмом FAST, не мають основного локального напрямку, для того щоб дескриптор мав сильну стійку обертальну інваріантність, необхідно використовувати інтенсивність центроїди, для орієнтації основного напрямку точки об'єкта. Використання інтенсивності центроїди припускає, що інтенсивність кута зміщена від його центру, і цей вектор може бути використаний для обчислення орієнтації. Визначення моменту патча відбувається по формулі 2.19 [24].

$$m_{pq} = \sum_{x,y} x^p y^q I(x,y), \tag{2.19}$$

де x та y є відносно положення особливої точки;

x, y ϵ [-r, r];

r — радіус околу FAST особливої точки,

q та p дорівнюють "1" або "0";

I (x, y) — інтенсивність сірого в точці (x, y).

Отже, центроїд визначається m00, m01 і m10 по формулі 2.20.

$$C = \left(\frac{m_{10}}{m_{00}}, \frac{m_{01}}{m_{00}}\right). \tag{2.20}$$

Обрахування кута між векторами проведених через центроїд, що вказують на розташування особливої точки в околі по формулі 2.21.

$$\theta = \arctan\left(\frac{m_{01}}{m_{10}}\right) = \arctan\left(\frac{\sum_{x,y}\ yI(x,y)}{\sum_{x,y}\ xI(x,y)}\right). \tag{2.21}$$

2.4 Вибір методу порівняння дескрипторів

Для порівняння дескрипторів зразка та відеокадру можна використовувати два методи: Brute Force (метод «грубої сили») та FLANN (Fast Library for Approximate Nearest Neighbors, швидка бібліотека для приблизних найближчих сусідів) [24].

Алгоритм Brute Force виконує вичерпний пошук по всьому набору особливих точок, щоб знайти двох їх найближчих сусідів. Оскільки це вичерпний алгоритм пошуку, він гарантовано знайде правильне рішення, але обчислення проходять досить повільно.

В дескрипторах, що працюють з рухомими точками для знаходження відстані між особливими точками використовується Евклідова відстань, а в бінарних, як BRISK, — відстань Хеммінга [24,25].

Евклідова відстань розраховується по формулі 2.22.

$$d(x,y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2}, \tag{2.22}$$

де d — відстань між об'єктами x і y;

xi — значення і -властивості об'єкта x;

yi — значення і-властивості об'єкта y;

i — (1, 2, ... n).

Для відстані Хеммінга є два визначення [24,25].

Перше — відстанню Хеммінга називається міра відмінності об'єктів, що задаються дихотомічними ознаками. Визначається за допомогою формули 2.23.

$$d_H(X_i X_j) = \sum_{s=1}^{p} \left| x_i^{(s)} - x_j^{(s)} \right| \tag{2.23}$$

І, отже, дорівнює числу невідповідностей між значеннями відповідних ознак в розглянутих і-тому і j-тому об'єктах.

Друге — відстанню Хеммінга називається число біт, які відрізняються в двох бінарних векторах.

Відстань Хеммінга вже досить широко використовується для різних завдань, таких як пошук близьких дублікатів, розпізнавання образів, класифікація документів, виправлення помилок, виявлення вірусів і т.д. У більш загальному випадку відстань Хеммінга застосовується до рядків однакової довжини будь-яких k-ічних алфавітів і служить метрикою різниці (функцією, що визначає відстань в метричному просторі) об'єктів тієї ж розмірності. Таку метрику можна представити у вигляді формули 2.24.

$$\text{hamming}(x, y) = \sum_{x_i \neq y_i} 1, i = 1, \dots, n \qquad (2.24)$$

Таким чином, метод Brute Force швидший для двійкових дескрипторів, ніж для тих, що працюють з рухомими точками, оскільки відстань між двома двійковими дескрипторами зменшується до кількості одиниць через виключне АБО (XOR) між двома дескрипторами.

Алгоритм FLANN також можна застосовувати для вичерпного пошуку особливих точок. Він підтримує як дескриптори рухомих точок, так і двійкові дескриптори і дозволяє здійснювати пошук k-найближчого сусіда за різними значеннями k. Для двійкових дескрипторів алгоритм, оснований на використанні хешування чутливих місць (LSH, locality sensitivity hashing). Для дескрипторів, що працюють з рухомими точками використовується алгоритм, заснований на методі kd-tree^2 і використовує набір рандомізованих «дерев», запущених паралельно [24,25].

Порівняємо ефективність методів порівняння дескрипторів.

На першому графіку (рисунок 2.8) видно, що метод FLANN видає більше хороших збігів (Good Matches) на всіх вісьми зразках.

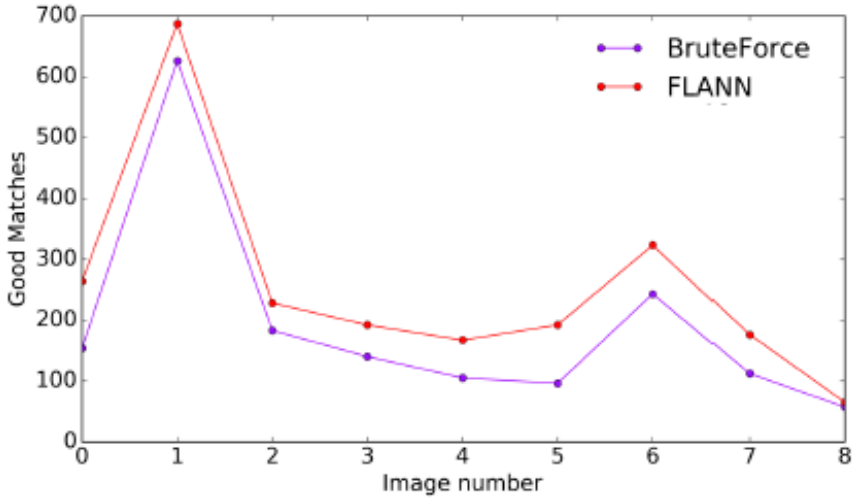


Рисунок 2.8 — Порівняння методів Brute Force та FLANN по кількості хороших збігів

На другому графіці (рисунок 2.9) порівнюється час (Total Matching Time), який був затрачений на знаходження всіх відповідностей дескрипторів особливих точок. Відповідно чим менше часу було затрачено, тим краще. І знову метод FLANN виявився кращим Brute Force.

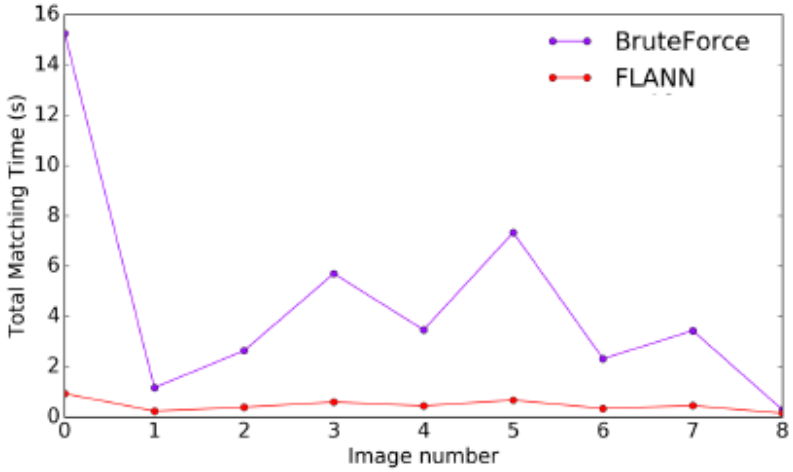


Рисунок 2.9 — Порівняння методів Brute Force та FLANN по часу затраченому на знаходження всіх відповідностей

На третьому графіці (рисунок 2.10) — порівнюється час (Time Per Keypoint), який був затрачений на обробку одного дескриптора особливих точок. Відповідно чим менше часу було затрачено, тим краще. І метод FLANN виявився кращим Brute Force.

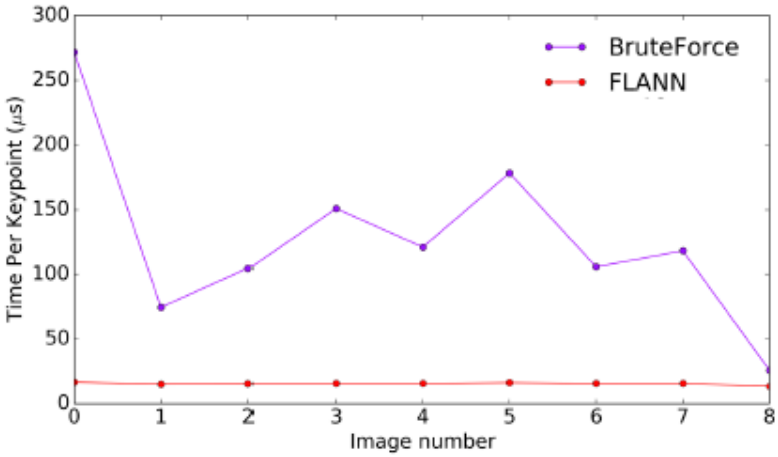


Рисунок 2.8 — Порівняння методів Brute Force та FLANN по часу затраченому на обробку одного дескриптора особливих точок

Отже, в даному розділі роботи було розроблено та обгрунтовано метод багатоканального розпізнавання відеозображень, за основу було взято алгоритм BRISK та покращено його опираючись на алгоритм FAST.

3 РОЗРОБКА ТА ТЕСТУВАННЯ АЛГОРИТМУ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Вибрані технології розробки

3.1.1 Вибір мови програмування

C# — це об'єктно-орієнтована мова програмування, розроблена в компанії Microsoft під керівництвом Андерса Хейльсберга в 1998-2001 роках як мова розробки програм для Microsoft .NET framework. C# розроблялася як мова програмування прикладного рівня для CLR (Common Language Runtime) і, як така, залежить, перш за все, від можливостей самої CLR [26, 27].

Синтаксис C# подібний синтаксису C / C ++, Java, тому ця мова називається C-подібною мовою. C# є мультипарадигменною мовою програмування, так як в ній підтримуються об'єктно-орієнтоване і процедурне програмування. Однак, на відміну від C++, мова не підтримує множинне успадкування класів [26, 27].

Об'єктно-орієнтоване програмування (ООП) — основна концепція C#. Технологія ООП невіддільна від C#, тому всі програми на C# є об'єктно-орієнтованими хоча б в найменшій мірі. Концепція ООП вносить в арсенал розробника новий засіб — класи. Класи поряд з об'єктами складають основу багатьох сучасних мов високого рівня. Під класом розуміється деяка сутність, яка задає деяку загальну поведінку для об'єктів. Таким чином, будь-який об'єкт може належати або не належати певному класу, тобто мати або не мати поведінку, яку має даний клас. Крім цього ООП дає підхід до наступних важливих властивостей класів.

Абстрагування, коли об'єкти представляють собою спрощений, ідеалізований опис реальних сутностей предметної області. Якщо відповідні моделі відповідають розв'язуваній задачі, то працювати з ними виявляється набагато зручніше, ніж з низькорівневим описом всіх можливих властивостей і реакцій об'єкта.

Інкапсуляція (приховування інформації) — це принцип, згідно з яким, будь-який клас повинен розглядатися як «чорний ящик» — користувач класу повинен бачити і використовувати тільки інтерфейсну частину класу (список декларованих

властивостей і методів класу) і не вникати в його внутрішню реалізацію. Тому дані прийнято інкапсулювати в класі таким чином, щоб доступ до них на читання або запис здійснювався не безпосередньо, а за допомогою методів. Принцип інкапсуляції (теоретично) дозволяє мінімізувати число зв'язків між класами і, відповідно, спростити незалежну реалізацію і модифікацію класів. Приховування даних — невіддільна частина ООП, керуюча областями видимості. Метою приховування є неможливість для користувача дізнатися або зіпсувати внутрішній стан об'єкта.

Успадкування — це можливість породжувати один клас від іншого зі збереженням усіх властивостей і методів класу-предка (іноді його називають суперкласом) і додаючи, при необхідності, нові властивості і методи. Набір класів, пов'язаних відношенням по спадкувості, називають ієрархією. Спадкування має на меті відобразити таку властивість реального світу, як ієрархічність.

Поліморфізм — явище, при якому функції (методу) з одним і тим же ім'ям відповідає різний програмний код (поліморфний код), в залежності від того, об'єкт якого класу використовується при виклику даного методу. Поліморфізм забезпечується тим, що в класі-нащадку змінюють реалізацію методу класу-предка з обов'язковим збереженням сигнатури методу. Це забезпечує збереження незмінним інтерфейсам класу-предка і дозволяє здійснити пов'язування імені методу в коді з різними класами, тобто, з об'єкта якого класу здійснюється виклик, з того класу і береться метод з такою назвою. Такий механізм називається динамічним (або пізнім) зв'язуванням — на відміну від статичного (раннього) зв'язування, здійснюваного на етапі компіляції.

Завдяки такій гнучкості C# можна використовувати практично для будь-якого, на сьогоднішній день, завдання, неважливо чи це системне програмування, проектування БД, написання математичних пакетів [26-28].

Призначення .NET Framework — служити середовищем для підтримки і виконання сильно розподілених додатків. Вона забезпечує спільне використання різних мов програмування, а так само безпеку, переміщення програм і загальну модель програмування для платформи Windows [26-28].

Середовище CLR управляє виконанням коду .NET Framework. Діє воно за наступним принципом: результатом компіляції програми на C# є не виконуваний код, а файл, що містить особливого роду псевдокод, що називається Microsoft Intermediate Language (MSIL). MSIL визначає набір переносяться інструкцій, що не залежить від конкретного процесора [26-28].

По суті, MSIL визначає мову асемблера. Будь-яка програма, скомпільована в псевдокод MSIL, може виконуватися на будь-якому комп'ютері, де є реалізація CLR (Common Language Runtime) [26-28].

3.1.2 Вибір програмних засобів реалізації

В якості платформи для розробки програми була обрана операційна платформа WIN64, яка трапляється практично на всіх сучасних настільних системах. Цей вибір обумовлений тим, що операційні системи Microsoft - сімейство найбільш популярних систем, що володіє простим зовнішнім оформленням, яке спрощує взаємодію з кінцевим користувачем.

Компілятор Microsoft Visual Studio 2019 — продукт компанії Microsoft, що включає інтегроване середовище розробки програмного забезпечення і широкий ряд різних інструментальних засобів [28].

Даний продукт дозволяє розробляти як консольні додатки, так і додатки з графічним інтерфейсом, в тому числі з підтримкою технології Windows Forms та Windows Presentation Foundation, а також веб-додатки, веб-служби як в рідному, так і в керованому кодах для всіх платформ, підтримуваних Windows [28].

Лінійка продуктів Visual Studio включає в себе редактор вихідного коду з підтримкою технології IntelliSense і можливістю найпростішого рефакторінга коду. Вбудований відладчик може працювати як відладчик рівня вихідного коду, так і як відладчик машинного рівня. Решта вбудованих інструментів включають в себе редактор форм для спрощення створення графічного інтерфейсу додатку, веб-редактор, дизайнери класів і схем баз даних і т.д. [27,28].

Visual Studio дозволяє створювати і підключати сторонні додатки (плагіни) для розширення функціональності практично на кожному рівні, включаючи

додавання підтримки систем контролю версій вихідного коду додавання нових наборів інструментів (наприклад, для редагування і візуального проектування коду на предметно-орієнтованих мовах програмування або інструментів для інших аспектів.

В якості цільової програмної технологій було обрано Windows Presentation Foundation (WPF).

Windows Presentation Foundation — це презентаційний функціональний шар для системи Windows. У той час, як підсистема Windows Forms для рендеринга використовує растрові механізми GDI / GDI +, підсистема WPF має свій власний векторний механізм рендеринга. З цієї причини вона не створює вікна і елементи управління стандартним способом і в стандартному вигляді, що притаманні системі Windows. Технологія WPF радикально відрізняється від технології Windows Forms. У графічної підсистеми Windows Forms розробник зазвичай визначав користувацький інтерфейс за допомогою візуального конструктора, автоматично створюючи код (на мові проекту) в файлі з розширенням .designer. Таким чином, по суті, призначений для користувача інтерфейс визначався і управлявся кодом на мові C# або VB [28].

У той же час призначений для користувача інтерфейс, створений за технологією WPF, фактично визначається мовою розмітки Extensible Application Markup Language (XAML). Ця мова заснована на мові XML і спеціально розроблена компанією Microsoft для використання в системі WPF [28].

Саме мова XAML, що лежить в основі технології WPF, забезпечує її міць і гнучкість, дозволяючи розробляти набагато багатші і особливі користувацькі інтерфейси в порівнянні з технологією Windows Forms. Визначення користувацького інтерфейсу за допомогою мови XAML є однаковим для будь-якої мови проекту. З цієї причини, поряд з новими можливостями управління користувацьким інтерфейсом, з'явилася велика кількість нових допоміжних концепцій, що впливають на код. Наприклад, виникли властивості залежностей, значеннями яких є вираження, що часто потрібні в багатьох сценаріях зв'язування для підтримки складних можливостей зв'язування, що надаються мовою XAML.

Проте код в WPF-додатку майже не відрізняється від коду стандартного додатка, створеного за технологією Windows Forms.

Завдяки гнучкості мови XAML система WPF дозволяє розробляти унікальні інтерфейси користувача і способи інтерактивної взаємодії. В їх основі лежать стилі і шаблонні функціональні можливості системи WPF, яка відокремлює зовнішній вигляд елементів управління від їх поведінки.

Це дозволяє легко змінювати зовнішній вигляд елементів управління без зміни самого елемента. Але ця гнучкість і міць мови XAML вимагає значного часу для навчання навіть досвідчених розробників [27,28].

3.2 Опис реалізації програми

У даній магістерській кваліфікаційній роботі розробляється програма, що виконує розпізнавання зображень у відеофайлах. Дана програма використовує спеціально створений алгоритм для багатоканального розпізнавання.

ПЗ ViRec забезпечує інтерфейс користувача для завдання необхідних настроювань, таких як: вибір відеоматеріалів та зразка для розпізнавання, з можливістю вибору області розпізнавання, покадрове відтворення відео та можливість зробити знімок кадра.

Функціональні можливості програми побудовані на основі бібліотеки Emgu Cv [18].

Основним метод в програмі є ProcessImage:

```
private static VectorOfPoint ProcessImage(Image<Gray, byte> template,
Image<Gray, byte> scene_frame)
        {
            try
            {
                VectorOfPoint finalPoint = null;
                Mat homography = null;
                VectorOfKeyPoint templateOfKeyPoint = new VectorOfKeyPoint();
                VectorOfKeyPoint sceneOfKeyPoint = new VectorOfKeyPoint();
```

```
                    Mat templatedecriptor = new Mat();
                    Mat scenedecriptor = new Mat();
                    Mat mask;
                    int k = 2;
                    double uniqe_threshold = 0.80;
                    VectorOfVectorOfDMatch match = new
VectorOfVectorOfDMatch();
                FastFeatureDetector fastFeature = new FastFeatureDetector();
            Brisk brisk = new Brisk();
            fastFeature.Detect(template,null);
            fastFeature.Detect(scene_frame, null);
            brisk.Compute(template, templateOfKeyPoint, templatedecriptor);
            brisk.Compute(scene_frame, sceneOfKeyPoint, scenedecriptor);
                    // Matching
                    var ip = new AutotunedIndexParams();
                SearchParams sp = new SearchParams();
                FlannBasedMatcher flannBasedMatcher = new
FlannBasedMatcher(ip,sp);
                flannBasedMatcher.Add(templatedecriptor);
                flannBasedMatcher.KnnMatch(scenedecriptor, match, k);
                    mask = new Mat(match.Size, 1, DepthType.Cv8U, 1); //0 or 1
                    mask.SetTo(new MCvScalar(255));

                    Features2DToolbox.VoteForUniqueness(match, uniqe_threshold,
mask);
                    int count =
Features2DToolbox.VoteForSizeAndOrientation(templateOfKeyPoint,
sceneOfKeyPoint,
                        match, mask, 1.5, 20);
                    if (count >= 4)
```

```
                {
                    homography =
Features2DToolbox.GetHomographyMatrixFromMatchedFeatures(templateOfKeyPoint
, sceneOfKeyPoint, match, mask, 5);
                }
                if (homography != null)
                {
                    Rectangle rect = new Rectangle(Point.Empty, template.Size);
                    PointF[] pts = new PointF[]
                    {
                      new PointF (rect.Left, rect.Bottom),
                      new PointF (rect.Right, rect.Bottom),
                      new PointF (rect.Right, rect.Top),
                      new PointF (rect.Left, rect.Top)
                    };
                    CvInvoke.PerspectiveTransform(pts, homography);
                    Point[] points = Array.ConvertAll<PointF, Point>(pts,
Point.Round);
                    finalPoint = new VectorOfPoint(points);
                }
                return finalPoint;
            }
            catch (Exception ex)
            {
                throw new Exception(ex.Message);
            }
        }
```

В якості параметрів на вхід даний метод отримує зображення зразка та відеокадру в градаціях сірого Image<Gray, byte> template та Image<Gray, byte> scene_frame відповідно. Метод містить такі ключові поля.

Mat homography — матриця гомографії.

VectorOfKeyPoint templateOfKeyPoint = new VectorOfKeyPoint() — створення контейнера для точок взятих зі зразка.

VectorOfKeyPoint sceneOfKeyPoint = new VectorOfKeyPoint() — створення контейнера для точок взятих зі зразка.

Mat templatedecriptor = new Mat() — матриця, що описує особливі точки зразка.

Mat scenedecriptor = new Mat() — матриця, що описує особливі точки відеокадра.

Double uniqe_threshold = 0.80 — заданий поріг чутливості унікальності збігу.

FastFeature.Detect(template,null) — виділення особливих точок зі зразка з використанням алгоритму FAST.

FastFeature.Detect(scene_frame, null) — виділення особливих точок з відеокадру, по алгоритму FAST.

Brisk.Compute(template, templateOfKeyPoint, templatedecriptor) — створення дескрипторів особливих точок зразка.

Brisk.Compute(scene_frame, sceneOfKeyPoint, scenedecriptor — створення дескрипторів особливих точок відеокадру.

FlannBasedMatcher flannBasedMatcher = new FlannBasedMatcher(ip,sp) — створення FLANN співставника зображень, для пошуку збігів.

FlannBasedMatcher.Add(templatedecriptor) — вибір сформованих дескрипторів.

flannBasedMatcher.KnnMatch(scenedecriptor, match, k) — пошук збігів на відеозображенні методом співставлення к-найближчих сусідів.

Features2DToolbox.VoteForUniqueness(match, uniqe_threshold, mask) — фільтрація збігів, якщо вони не унікальні, то відкидаються.

Features2DToolbox.VoteForSizeAndOrientation(templateOfKeyPoint, sceneOfKeyPoint, match, mask, 1.5, 20) — фільтрація збігів по коефіцієнту масштабування та нахилу, якщо коефіцієнти більше вказаних, то такі збіги відкидаються.

Features2DToolbox.GetHomographyMatrixFromMatchedFeatures(templateOfKeyPoint, sceneOfKeyPoint, match, mask, 5) — отримання матриці гомографії, в програмі параметр, при якому буде формуватись матриця, дорівнює чотирьом, він напряму залежить від фільтрації збігів по коефіцієнту масштабування та нахилу.

На останньому кроці виділяється квадратом розпізнанана область на відеокадрі:

```
Rectangle rect = new Rectangle(Point.Empty, template.Size);
                PointF[] pts = new PointF[]
                {
                  new PointF (rect.Left, rect.Bottom),
                  new PointF (rect.Right, rect.Bottom),
                  new PointF (rect.Right, rect.Top),
                  new PointF (rect.Left, rect.Top)
                }.
```

А також якщо змінений коефіцієнт масштаба чи положення, то на виділену область накладається матриця гомографії.

```
                CvInvoke.PerspectiveTransform(pts, homography);
                Point[] points = Array.ConvertAll<PointF, Point>(pts,
Point.Round).
```

Завантажувальним модулем програми ViRec є файл ViRec.exe.

Програма працює під 64-х розрядними версіями ОС Microsoft Windows (7,8,10). Мінімальні вимоги до комп'ютера при запуску одного екземпляра ViRec: Intel Core i7 5th gen. і вище, 4096 MB RAM.

Лістинг програми наведено в додатках Б та В.

3.3 Налаштування і робота програми ViRec

Після запуску програми користувачеві необхідно здійснити налаштування.

Першим кроком необхідно перейти з головного меню у вікно розпізнавання, натиснувши кнопку «Recognize». У головному меню користувачеві також доступні такі опції, як: покадрове відтворення відео, з можливістю покадрової прокрутки,

можливість зробити знімок кадра та розбити відео чи вибраний відеофрагмент на кадри в оперативну пам'ять, з метою їх подальшого використання, як зразків для розпізнавання, чи на жорсткий диск зі збереженням у форматі «.jpeg».

Другим кроком необхідно вибрати кількість каналів, поставивши «галочку» у відповідному вікні, по замовчуванню налаштовано один канал.

На третьому кроці необхідно вибрати та завантажити в програму відеофайли для відповідної кількості каналів, для яких буде здійснено розпізнавання. Також слід вибрати зображення з накопичувача чи зробити знімок кадра для використання в якості зразка для розпізнавання.

Вибране зображення відкриється в меню, де необхідно буде вибрати область розпізнавання, робиться це наступним чином: натиснути в меню пункт «Обробка», далі в випадаючому списку навести мишку на підпункт «Вибрати область», де будуть знаходитись підпункти «Вибрати область» та «Виділити область», клацнути мишкою на перший підпункт, далі, націлитись мишкою в те місце зображення, з якого необхідно виділити область, далі затиснути ліву клавішу миші та протянути її вниз та вправо, регулюючи розмір області виділення, відпустити клавішу миші.

Далі натиснути на пункт «Виділити область», виділена область повинна з’явитись у вікні. Далі натиснути кнопку «ОК».

Потім натиснути кнопки «Play» та «Start» для початку виведення відео та розпізнавання.

Для скидання налаштувань слід натиснути кнопку «Refresh».

Алгоритм роботи програми представлений у вигляді блок - схеми в додатку Г.

3.4 Тестування роботи програми

Після запуску програми, відкривається головне меню, де знаходяться плеєр зі скролбаром для перемотки, функціональні кнопки та віконця з виведенням інформації про поточне відео (рисунок 3.6).

Рисунок 3.6 — Головне меню програми

Для переходу в меню розпізнавання слід натиснути кнопку «Recognize». Після чого відкривається відповідне вікно (рисунок 3.7).

Рисунок 3.7 — Меню розпізнавання

В лівій частині меню розпізнавання знаходяться елементи для налаштування модуля розпізнавання (рисунок 3.8), а в правій — відбувається виведення відео,

максимально в чотирьох каналах, та розпізнавання заданого зразка по кожному каналу.

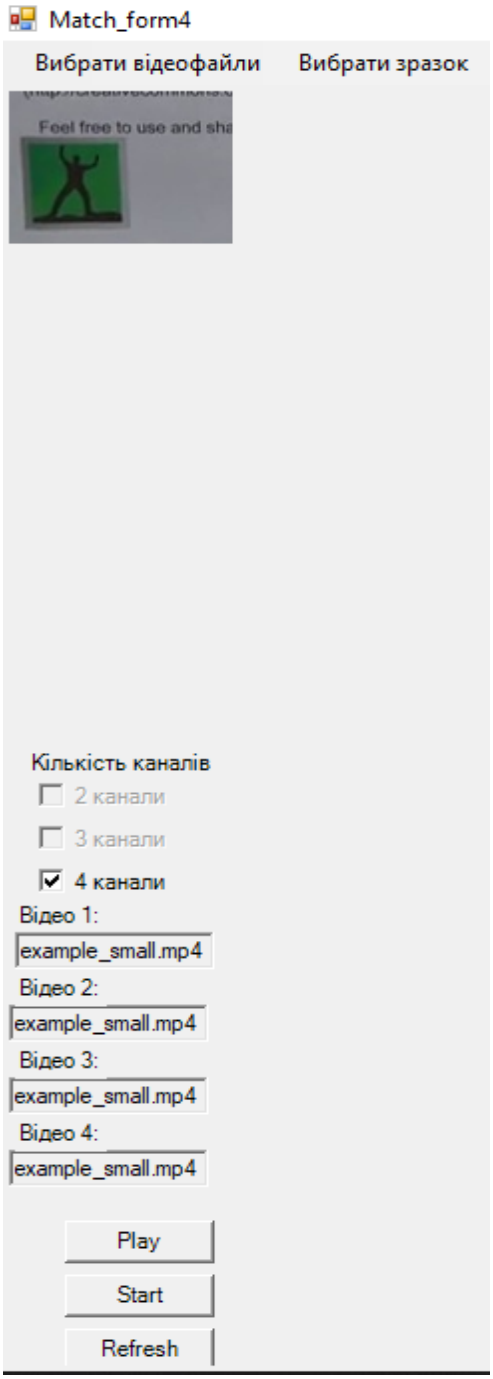


Рисунок 3.8 — Елементи для налаштування модуля розпізнавання

При натисненні на кнопку «Вибрати зразок», відкривається спеціальне вікно (рисунок 3.9), де необхідно вибрати зображення та область на ньому для розпізнавання.

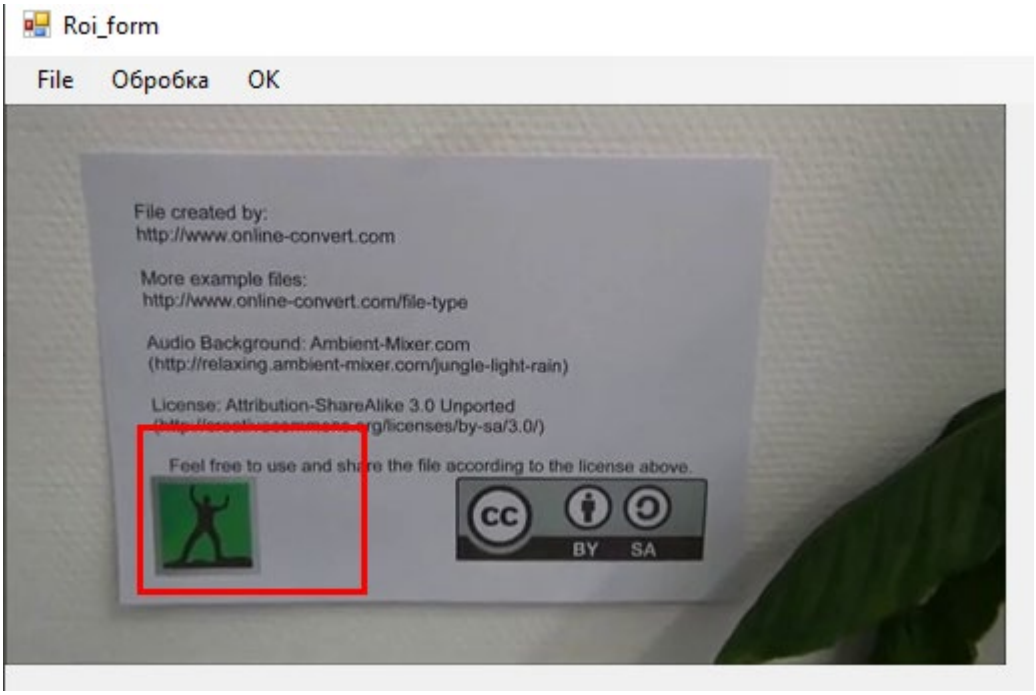


Рисунок 3.9 — Вікно вибору зразка для розпізнавання

Після вибору області зображення та натиснення кнопки «Виділити область», вибрана частина з’являється на місці попереднього зображення (рисунок 3.10).

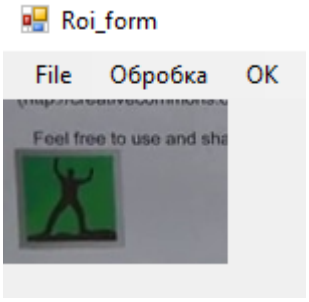


Рисунок 3.10 — Виділена область зображення

Після натиснення кнопки «ОК», вікно вибору зразка автоматично закривається, а виділена область зображення (зразок) відображається в меню розпізнавання.

Здійснивши всі потрібні налаштування можна починати процедуру розпізнавання заданого зразка на відеофайлах. При детекції область на відеофайлі помічається червоним квадратом. Функціонування та результат роботи програми показано на рисунку 3.11.

Рисунок 3.11 — Функціонування програми

Ефективність комбінованого алгоритму (BRISK_F) продемонстровано на рисунку 3.12, де його було порівняно зі стандартним алгоритмом BRISK та досить ефективним — SURF. Як видно з графіка комбінація алгоритмів BRISK та FAST виявилась значно точнішою, ніж BRISK та, навіть, кращою, ніж SURF.

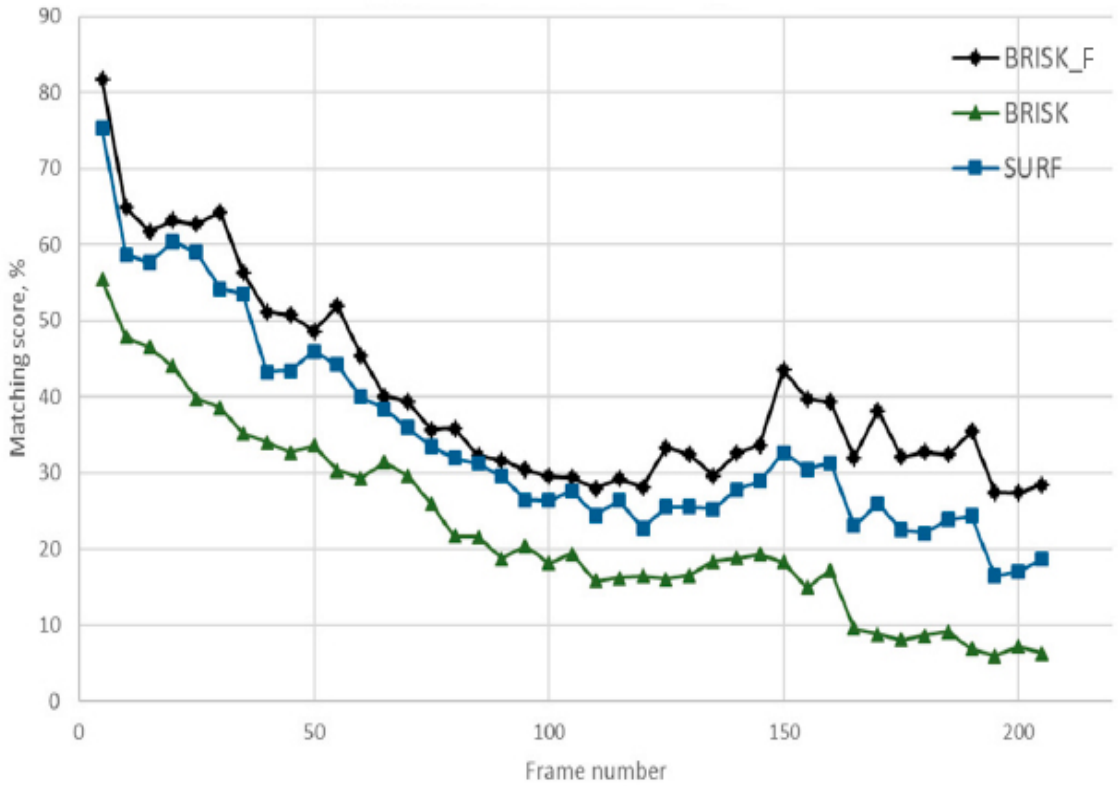


Рисунок 3.12 — Порівняння алгоритмів у точності розпізнавання

Отже, в даному розділі було описано реалізацію програмного забезпечення та алгоритму для багатоканального розпізнавання відеозображень, а також проведено тестування програмної частини та реалізованого в ній алгоритму.

4 РОЗРАХУНОК ЕКОНОМІЧНОЇ ДОЦІЛЬНОСТІ СТВОРЕННЯ ПРОГРАМИ РОЗПІЗНАВАННЯ ЗОБРАЖЕНЬ В БАГАТОКАНАЛЬНІЙ КОМП'ЮТЕРНІЙ СИСТЕМІ МОНІТОРИНГУ ЦИФРОВОГО ТЕЛЕВІЗІЙНОГО МОВЛЕННЯ

Створюючи будь-який, а особливо інноваційний, програмний продукт, необхідно перед безпосередньою розробкою розрахувати її економічну доцільнісь. Та на підставі чого зробити висновок про те чи варто інвестувати та фінансувати створення продукту, чи буде проект прибутковим і інвестиційно привабливим.

Тому даний розділ магістерської кваліфікаційної роботи і присвячений вирішенню цього важливого завдання та передбачає виконання таких етапів робіт (рисунку 4.1).

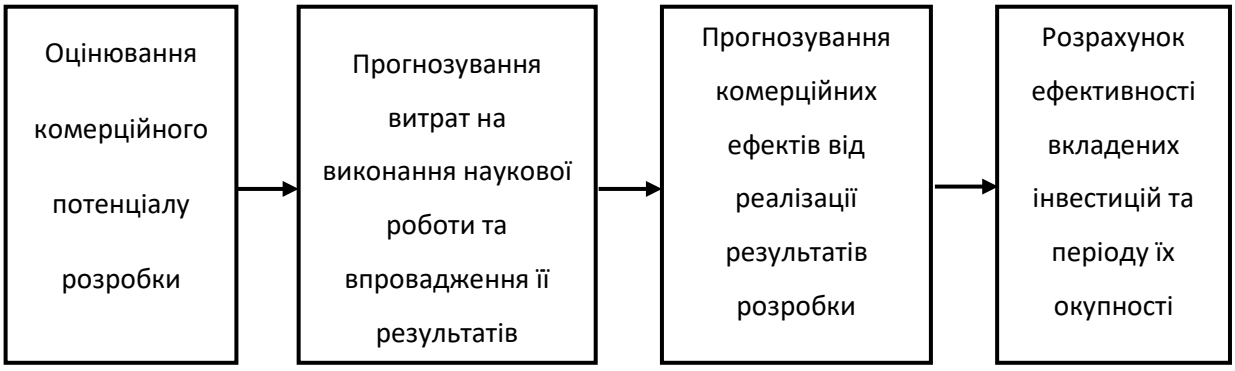


Рисунок 4.1 — Алгоритм виконання та складові економічної частини магістерської кваліфікаційної роботи

На такі складові буде поділено економічну частину даної магістерської роботи. Усі подальші економічні розрахунки, будуть висвітлені у згаданих підрозділах економічної частини.

4.1 Технологічний аудит розробки

В даному підрозділі буде проведено оцінювання комерційного потенціалу розробки, створеної в результаті науково-технічного дослідження. За результатами оцінювання робиться висновок щодо напрямів організації в майбутньому та її впровадження з врахуванням встановленого рейтингу.

Оцінювання комерційного потенціалу розробки будемо здійснювати за дванадцятьма критеріями, наведеними у таблиці 4.1.

Таблиця 4.1 — Оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-бальною шкалою)					
Критерій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Багато аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогів	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів
4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні.	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування

Кінець таблиці 4.1

Критерії оцінювання та бали (за 5-бальною шкалою)					
Критерій	0	1	2	3	4
Практична здійсненність					
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві
11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідно регламентні документи та велика кількість дозвільних документів на виробництво продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки повідомлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

На основі складеної таблиці ряд незалежних експертів, у нашому випадку керівник магістерської роботи та викладачі випускової кафедри поставили різні бали. Результати цього оцінювання комерційного потенціалу занесено до таблиці 4.2.

Таблиця 4.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище, ініціали, посада експерта		
	1 Крупельницький Л. В., к.т.н., доц. кафедри ОТ	2 Ткаченко О. М.., к.т.н., доц. кафедри ОТ	3 Снігур А. В., к.т.н., доц. кафедри ОТ
	Бали, виставлені експертами:		
1	3	3	3
2	4	3	3
3	4	4	4
4	3	4	2
5	3	4	3
6	3	2	3
7	4	3	3
8	4	3	3
9	3	3	3

Кінець таблиці 4.3

Критерії	Прізвище, ініціали, посада експерта		
	1 Крупельницький Л. В., к.т.н., доц. кафедри ОТ	2 Ткаченко О. М.., к.т.н., доц. кафедри ОТ	3 Снігур А. В., к.т.н., доц. кафедри ОТ
	Бали, виставлені експертами:		
10	3	3	3
11	4	4	3
12	3	3	4
Сума балів	$СБ_1 = 41$	$СБ_1 = 39$	$СБ_1 = 37$
Середньо-арифметична сума балів $\overline{СБ}$	$\overline{СБ} = \frac{\sum_1^3 СБ_i}{3} = \frac{41 + 39 + 37}{3} = \frac{117}{3} = 39$		

За даними таблиці 4.2, а також згідно із рекомендаціями, що наведені в таблиці 4.3, можна зробити висновок, щодо рівня комерційного потенціалу розробки.

Таблиця 4.3 — Рівні комерційного потенціалу розробки

Середньоарифметична сума балів $\overline{СБ}$, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 - 10	Низький
11 - 20	Нижче середнього
21 - 30	Середній
31 - 40	Вище середнього
41 - 48	Високий

Взявши до уваги, середньоарифметичну суму балів, $\overline{СБ} = 39$, що були виставлені експертами, можна стверджувати, що рівень комерційного потенціалу даної розробки - вище середнього.

Для порівняння властивостей було взято Video Recognizer. Дана програма має більш широкий спектр використання. В свою чергу нова розробка є вузько спеціалізованою, через що більш ефективною. Аналог розповсюджується по платній підписці у той час, як плата за користування програмою, яка є продуктом розробки, є одноразовою

Таблиця 4.4 — Порівняння характеристик розробки із аналогом

Показники	**Розробка**	**Аналог**
Функціонал	7	9
Швидкодія	9	6
Надійність	9	6
Метод розповсюдження	8	7
Інтерфейс, простота використання	8	7

Просування продукту здійснюватиметься засобами цифрової дистрибуції. За допомогою аналітики можна буде спрямувати на цільову аудиторію — телерадіокомпанії цифрового телевізійного мовлення.

Новизною розробки є вдосконалення методу розпізнавання зображення у відеофайлах зі збільшенням потоків обробки, що дає змогу більш швидко обробляти інформацію. Дана характеристика позитивно впливатиме на конкурентоспроможність продукту.

Виходячи з результатів даного порівняння можна зробити висновок, що нова розробка є конкурентоспроможною, оскільки по деяким параметрам переважає один з найкращих аналогів на ринку.

4.2 Прогнозування витрат на виконання та впровадження результатів наукової роботи

У магістерській кваліфікаційній роботі розглядається програмне забезпечення для багатоканального розпізнавання зображень у відеофайлах, тому значна частина витрат — це витрати на розробку, а не на виробництво і відтворення. Звідси, й певна специфіка розрахунків.

Основна заробітна плата розробників, які працюють над проектом визначається за формулою 4.1.

$$Зо = \frac{М}{Тр} \cdot t \text{ [грн]}, \qquad (4.1)$$

де М — місячний посадовий оклад розробника;

Тр — число робочих днів в місяці (T_p= 22дні);

t — число днів роботи розробника.

Над створенням розробки працювали керівник проекту та інженер-програміст , отже, виконаємо для них всі необхідні розрахунки:

$$Зо = \frac{14\,000}{22} \cdot 14 = 8909,09\ (грн).$$

$$Зо = \frac{10\,000}{22} \cdot 60 = 27272,73\ (грн).$$

Таблиця 4.5 — Заробітна плата

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
1 Керівник	14 000	636,36	14	8909,09
2 Старший інженер-програміст	10 000	454,55	60	27272,73
Всього				$\sum Зо = 36181,82$

Додаткова заробітна плата (Зд) всіх розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховується як (10…12)% від суми основної заробітної плати всіх розробників та робітників, тобто:

$$Зд = (10 \ldots 12\%) * Зо\ [грн], \qquad (4.2)$$

де Зо - основана заробітна плата.

$$Зд = \frac{10 \cdot 36181,82}{100} = 3618,18(грн).$$

Нарахування на заробітну плату (Нзп) розробників та робітників, які брали участь у виконанні даного етапу роботи, розраховуються за формулою:

$$\text{Нзп} = 22\% \cdot (\text{Зо} + \text{Зд})[\text{грн}], \quad (4.3)$$

$$\text{Нзп} = \frac{22 \cdot (36181,82 + 3618,18)}{100} = 8756\ (\text{грн}).$$

Амортизація обладнання, комп'ютерів та приміщень (А), які використовувались під час виконання даного етапу роботи.

Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

У спрощеному вигляді амортизаційні відрахування (А) в цілому бути розраховані за формулою 4.4.

$$\text{А} = \frac{\text{Ц} \cdot \text{На}}{100} \cdot \frac{\text{Т}}{12}[\text{грн}], \quad (4.4)$$

де Ц - загальна балансова вартість всього обладнання, комп'ютерів, приміщень тощо, що використовувались для виконання даного етапу роботи, грн;

На - річна норма амортизаційних відрахувань, для нашого випадку можна прийняти, що На = (10...25)%;

Т— термін, використання обладнання, приміщень тощо, місяці.

Всі розрахунки зводимо до таблиці 4.6.

У магістерській кваліфікаційній роботі розробляється програмний продукт, який для споживача буде поширюватися через інтернет, а саме через веб-сторінку. При розробці сторінки до послуг виробничого характеру сторонніх підприємств можна віднести надавання послуги «Хостинг», а також направлення обраного доменного імені на сервери хосту, тому на протязі всього існування проекту за ці послуги, щорічно потрібно сплачувати абонентську плату, дані заносимо в таблицю 4.7.

Таблиця 4.6 — Амортизація обладнання та приміщень

Найменування обладнання, приміщень	Балансова вартість, грн.	Норма амортизації, %	Термін використання, міс.	Величина амортизаційних відрахувань, грн.
ЕОМ	18 000	20%	3	900
Приміщення	80 000	15%	3	3000
Всього				3900

Таблиця 4.7 - Послуги, що використовуються при виготовленні програми

Найменування комплектуючих (робіт, послуг)	Кількість, шт.	Ціна за одиницю, грн.	Сума, грн.
1. Послуга «Хостинг», шт.	1	430	430
2. Послуга «Доменне ім’я», шт.	1	170	170
Всього	600 грн.		

Витрати на силову електроенергію $В_е$, якщо ця стаття має суттєве значення для виконання даного етапу роботи, розраховуються за формулою 4.6:

$$В_е = В \cdot П \cdot Ф \cdot К_п [грн], \quad (4.6)$$

де В — вартість 1 кВт електроенергії, грн.;

П — установлена потужність обладнання, кВт/год;

Ф — фактична кількість годин роботи обладнання, яке задіяне на виготовлення одного виробу, годин;

$К_п$ —коефіцієнт використання потужності, $К_п \leq 1$.

$$В_е = 3{,}35 \cdot 0{,}09 \cdot 230 \cdot 0{,}7 = 48{,}5 \text{ (грн)}.$$

Інші витрати охоплюють: загально виробничі витрати (витрати управління організацією, ремонт та експлуатація основних засобів, витрати на опалення, освітлення тощо), адміністративні витрати (проведення зборів, оплата юридичних та аудиторських послуг, тощо), витрати на збут (витрати на рекламу, перепідготовка кадрів) на інші операційні витрати (штрафи, пені, матеріальні допомоги, втрати від знецінення запасів тощо).

Інші витрати $I_в$ можна прийняти як (100…300)% від суми основної заробітної плати розробників та робітників, які були виконували дану роботу, тобто:

$$I_в=(1..3)\cdot(З_о+З_р) \tag{4.7}$$

Отже, розрахуємо інші витрати:

$$I_в = 1 \cdot (36181,82 + 3618,18) = 39800 \text{ грн.}$$

Сума всіх попередніх статей витрат дає витрати на виконання даної частини (розділу, етапу) роботи — В.

$$В = 36181,82 + 3618,18 + 8756 + 3900 + 600 + 48,5 + 39800 = 92904{,}5(\text{грн}).$$

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної наукової роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}[\text{грн}], \tag{4.8}$$

де β—коефіцієнт, який характеризує етап (стадію) виконання даної роботи, оскільки, розробка знаходиться на стадії впровадження, то $\beta \approx 0{,}9$;

$В_{заг}$—загальна вартість всієї наукової роботи, у даному випадку $В_{заг} = В$.

$$ЗВ = \frac{92\,904{,}5}{0{,}9} = 103227,2(\text{грн}).$$

Отже, розрахований кошторис витрат на розробку програмного забезпечення для розпізнавання зображень в відеофайлах складає $103227,2$ грн.

4.3 Прогнозування комерційних ефектів від реалізації результатів розробки

У даному підрозділі здійснено прогнозування, яку вигоду можна отримати у майбутньому від впровадження результатів даної наукової роботи.

Передбачається, що виконання наукової роботи та впровадження результатів по розробці програмного забезпечення для розпізнавання зображень в відеофайлах займе 1 рік.

Основні позитивні результати від впровадження розробки очікуються протягом 3 років після її впровадження. Саме зростання чистого прибутку забезпечить підприємству (організації) надходження додаткових коштів, які дозволять покращити фінансові результати діяльності.

$$\Delta П_{\text{і}} = \sum_{1}^{n} (\Delta П_{я} \cdot N + П_{я}\Delta N)_i \, , [\text{грн}], \tag{4.9}$$

де $\Delta П_{я}$ — покращення основного якісного показника від впровадження результатів розробки у даному році;

N — основний кількісний показник, який визначає діяльність підприємства у даному році до впровадження результатів наукової розробки;

ΔN — покращення основного кількісного показника діяльності підприємства від впровадження результатів розробки;

$П_{я}$ — основний якісний показник, який визначає діяльність підприємства у даному році після впровадження результатів наукової розробки;

n — кількість років, протягом яких очікується отримання позитивних результатів від впровадження розробки.

В результаті впровадження результатів наукової розробки покращується якість програмного продукту, що дозволяє підвищити ціну його реалізації на 500 грн., а кількість потенційних користувачів ресурсу збільшиться: протягом першого року - на 130 шт., протягом другого року - ще на 115 шт., протягом третього року - ще на 70 шт.

Орієнтовно реалізація продукції до впровадження результатів наукової розробки складала 1 шт., а прибуток, що його отримувало підприємство на одиницю продукції до впровадження результатів наукової розробки — 800 грн.

Спрогнозуємо збільшення чистого прибутку підприємства від впровадження результатів наукової розробки у кожному році відносно базового.

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом першого року складе:

$$\Delta П_1 = 800 + (800 + 500) \cdot 130 = 169800(\text{грн}).$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом другого року (відносно базового року, тобто року до впровадження результатів наукової розробки) складе:

$$\Delta П_2 = 800 + (800 + 500) \cdot 115 = 150300(\text{грн}).$$

Збільшення чистого прибутку підприємства $\Delta П_i$ протягом третього року складе:

$$\Delta П_3 = 800 + (800 + 500) \cdot 70 = 91800(\text{грн}).$$

4.4 Розрахунок ефективності вкладених інвестицій та періоду їх окупності

Розрахований комерційний ефект можливого впровадження розробки однак не є гарантією, що розробка буде впроваджена. Якщо збільшення прогнозованого прибутку від впровадження результатів наукової розробки є вигідним для

підприємства, то це ще не означає, що вона зацікавить інвесторів. Основні показники, що визначають рентабельність фінансування розробки певним інвестором, є відносна і абсолютна ефективність вкладених інвестицій та термін їх окупності.

Розрахунок ефективності вкладених інвестицій передбачає проведення таких робіт:

На першому кроці розраховуємо теперішню вартість інвестицій PV, що вкладаються в наукову розробку. Такою вартістю, можна вважати прогнозовану величину загальних витрат ЗВ на виконання та впровадження результатів НДДКР, розраховану нами раніше за формулою 4.8, тобто будемо вважати, що ЗВ = PV=$103227,2$.

На другому кроці розрахуємо очікуване збільшення прибутку $\Delta П_i$, що його отримає підприємство (організація) від впровадження результатів наукової розробки, для кожного із років, починаючи з першого року впровадження.

На третьому кроці, для спрощення подальших розрахунків побудуємо вісь часу, на яку нанесемо всі платежі (інвестиції та прибутки), що мають місце під час виконання науково-дослідної роботи та впровадження її результатів.

Платежі показуються у ті терміни, коли вони здійснюються. Рисунок, що характеризує рух платежів (інвестицій та додаткових прибутків) буде мати вигляд, наведений на рисунку 4.2.

На четвертому кроці розрахуємр абсолютну ефективність вкладених інвестицій $Е_{абс}$.

Для цього користуються формулою 4.10:

$$Е_{абс} = (ПП - PV), \quad (4.10)$$

де ПП - приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.

Тис. грн Початок отримання прибутків

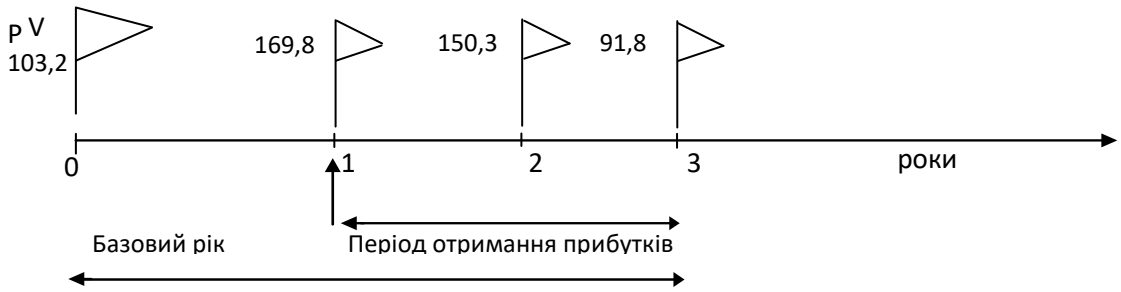


Рисунок 4.2 — Вісь часу з фіксацією платежів, що мають місце під час розробки та впровадження результатів НДДКР

PV — теперішня вартість інвестицій PV = ЗВ = 103227, 2(грн).

У свою чергу, приведена вартість всіх чистих прибутків ПП розраховується за формулою:

$$ПП = \sum_{1}^{T} \frac{\Delta П_i}{(1+\tau)^t}, [грн], \quad (4.11)$$

де $\Delta П_i$ — збільшення чистого прибутку у кожному із років, протягом яких виявляються результати виконаної та впровадженої НДДКР, грн;

Т — період часу, протягом якого виявляються результати впровадженої НДДКР, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні; для України цей показник знаходиться на рівні 0,1;

t — період часу (в роках) від моменту отримання чистого прибутку до точки «0».

Отримаємо:

$$ПП = \frac{169800}{(1+0{,}1)^1} + \frac{150300}{(1+0{,}1)^2} + \frac{91800}{(1+0{,}1)^3} = 154363,64 + 124214,88 + 68970,70 = 347549{,}22$$

Тоді $Е_{абс} = (347549{,}22 - 103\,227, 2) = 244322, 02$(грн).

Оскільки $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження може принести прибуток, але це також ще не гарантує те, що інвестор зацікавиться у фінансуванні даної роботи.

5-й крок. Розраховують відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $E_в$. Для цього користуються формулою:

$$E_в = \sqrt[Т_ж]{1 + \frac{E_{абс}}{PV}} - 1 \qquad (4.12)$$

де $E_{абс}$ - абсолютна ефективність вкладених інвестицій, грн;

PV - теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ - життєвий цикл наукової розробки, роки.

$$E_в = \sqrt[3]{1 + \frac{244322,02}{103\ 227,2}} - 1 = 1,44 - 1 = 44\ \%$$

Далі, розрахована величина $E_в$ порівнюється з мінімальною (бар'єрною) ставкою дисконтування $\tau_{мін}$, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатись не будуть. У загальному вигляді мінімальна (бар'єрна) ставка дисконтування $\tau_{мін}$ визначається за формулою:

$$t = d + f\ [\%], \qquad (4.13)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках, в 2019 році в Україні d = (0,19...0,22);

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = (0,1), але може бути і значно більше.

$$t = d + f = 0,20 + 0,1 = 0,30 = 30\%$$

Величина $E_в > \tau_{мін}$, інвестор може бути зацікавлений у фінансуванні даної наукової розробки.

Розрахуємо термін окупності вкладених коштів у реалізацію наукового проекту за формулою:

$$T_{ок} = \frac{1}{E_в} [років], \quad (4.14)$$

$$T_{ок} = \frac{1}{0{,}44} = 2,27 \text{ роки.}$$

Оскільки $T_{ок} = 2, 27$ роки, то розробка являється доцільною.

4.5 Висновки економічного ґрунтування

У даному розділі магістерської кваліфікаційної роботи здійснено розрахунки, які доводять прибутковість та ефективність впровадження нового продукту.

Проведено оцінювання комерційного потенціалу розробки. На основі компетентної думки експертів було сформовано систему критеріїв та за 5-ти бальною шкалою, виставлено бали по кожному з них. Виставлені бали, показують, що рівень комерційного потенціалу є вище середнього.

Розраховано витрати на розробку. Розрахований кошторис витрат на розробку склав $103227,2$ грн.

Були спрогнозовані комерційні ефекти від реалізації розробки, тобто який дохід, можна отримати у майбутньому від впровадження виконаної наукової роботи. Доведено, що розробка отримає вигоду від впровадження.

Розраховано основі показники, які визначають доцільність фінансування наукової розробки інвестором. Такими показниками є абсолютна та відносна ефективність вкладених інвестицій, а також термін їх окупності.

Обрахована абсолютна ефективність становить $244322,02$ грн, що свідчить про те, що інвестор буде зацікавлений у фінансуванні даної розробки. Відносна

(щорічна) ефективність становить 44%, що більше мінімальної ставки дисконтування, що ще раз підтверджує зацікавленість інвестора.

Термін окупності вкладених коштів у реалізацію наукового проекту становить 2, 27 роки, що означає, що вкладені кошти повернуться, приблизно, через 28 місяців.

Отже, можна стверджувати, що фінансування даної розробки є доцільним.

ВИСНОВКИ

Основним результатом даної магістерської кваліфікаційної роботи є розробка методів і програмних засобів багатоканального розпізнавання зображень у відеофайлах.

Для досягнення цієї мети було вирішено наступні завдання:

— проаналізовано наявну інформацію щодо методів розпізнавання зображень у відеофайлах, та вибрано за основу алгоритм розпізнавання особливих точок BRISK;

— розроблено специфічні методи розпізнавання зображень у відеофайлах, які враховують специфіку багатоканальних систем цифрового телебачення;

— розроблено та протестовано алгоритм та програмне забезпечення для опрацювання багатоканальних відеосигналів;

— обгрунтовано основні техніко-економічні показники і ефективність виконаної розробки.

Для досягнення поставленої в роботі мети використовувались такі методи дослідження:

— системний аналіз, який застосовувався для дослідження методів розпізнавання зображень у відеофайлах;

— об’єктно-орієнтовані методи програмування, які використані при розробці програмного забезпечення;

— методи порівняльного комп’ютерного тестування з метою оцінки ефективності запропонованих алгоритмів.

Науковою новизною отриманих результатів даної магістерської роботи є подальший розвиток метод виділення особливих точок зображення, який в системах паралельного потокового багатоканального опрацювання даних пришвидшує розпізнавання заданих об’єктів при збереженні якості їх розпізнавання.

Практичними значеннями одержаних результатів даної магістерської роботи є :

— алгоритм та програма для паралельного багатоканального розпізнавання відеоданих, який має самостійне значення;

— розроблене програмне забезпечення інтегровано у систему моніторингу центрального та регіонального цифрового телерадіомовлення.

Апробація результатів відбулась на XLVIII Науково-технічній конференції факультету інформаційних технологій та комп'ютерної інженерії.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. В. Крупельницький, В. В. Самко, М. В. Жилін Комп’ютерна система моніторингу телевізійного мовлення [Електронний ресурс] / Л. В. Крупельницький, В. В. Самко, М. В. Жилін. — 2019. — Режим доступу до ресурсу:https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki 2019/paper/view/6876

2. S. J. Sangwine, "Colour in image processing," Electronics & Communication Engineering Journal, vol. 12, pp. 211-219, 2000.

3. K. P. Ngoi and J. C. Jia, "A colour contrast model for edge and contour detection," in TENCON '94. IEEE Region 10's Ninth Annual International Conference. Theme: Frontiers of Computer Technology. Proceedings of 1994, 1994, pp. 309-313 vol.1.

4. D. W. Ball, "The Baseline: Color," Spectroscopy, vol. 24, p. 16, 2009.

5. B. A. Harrison and D. L. B. Jupp, Introduction to image processing. Canberra: Commonwealth Scientific and Industrial Research Organization, 1990.

6. M. D. Fairchild, Color Appearance Models. Sussex: John Wiley & Sons, 2005.

7. M. D. Fairchild, "Color Appearance Models: CIECAM02 and Beyond," in IS&T/SID 12th Color Imaging Conference, 2004.

8. M. C. Stone, A field guide to digital color. Natick, Mass: AK Peters, 2003.

9. Boulkenafet, Z.; Komulainen, J.; Hadid, A. Face Antispoofing Using Speeded-Up Robust Features and Fisher Vector Encoding. IEEE Signal Proc. Lett. 2017, 24, 141—145.

10. Calonder, M.; Lepetit, V.; Strecha, C.; Fua, P. BRIEF: Binary robust independent elementary features. In Proceedings of the 11th European Conference on Computer Vision: Part IV, Heraklion, Greece, 5—11 September 2010; pp. 778—792.

11. Mohammad, S.; Morris, T. Binary Robust Independent Elementary Feature Features for Texture Segmentation. Adv. Sci. Lett. 2017, 23, 5178—5182.

12. Leutenegger, S.; Chli, M.; Siegwart, R.Y. BRISK: Binary Robust invariant scalable keypoints. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6—13 November 2011; pp. 2548—2555.

13. Qu, X.; Soheilian, B.; Habets, E.; Paparoditis, N. Evaluation of SIFT and SURF for Vision Based Localization. ISPRS Int. Arch. Photogramm. Remote Sens. Spat. Inf. Sci. 2016, XLI-B3, 685—692.

14. Lowe, D.G. Object Recognition from Local Scale-Invariant Features. In Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 20—27 September 1999; Volume 2, pp. 1150—1157.

15. Bay, H.; Tuytelaars, T.; Van Gool, L. SURF: Speeded Up Robust Features. In Computer Vision—ECCV 2006; Leonardis, A., Bischof, H., Pinz, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2006; pp. 404—417.

16. Rublee, E.; Rabaud, V.; Konolige, K.; Bradski, G. ORB: An Efficient Alternative to SIFT or SURF. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6—13 November 2011; pp. 2564—2571.

17. Дж. Стокман, Л. Шапиро. Компьютерное зрение. Пер. с англ. — М.: БИНОМ. Лаборатория знаний, 2006.

18. Бібліотека EmguCV [Електронний ресурс] — Режим доступу до ресурсу: http://www.emgu.com.

19. Бібліотека AForge.NET [Електронний ресурс] — Режим доступу до ресурсу: http://www.aforgenet.com.

20. Бібліотека OpenCV [Електронний ресурс] — Режим доступу до ресурсу: https://opencv.org.

21. Alcantarilla, P.F.; Solutions, T. Fast Explicit Diffusion for Accelerated Features in Nonlinear Scale Spaces. IEEE Trans. Pattern. Match. Intell. (TPAMI) 2011, 34, 1281—1298.

22. Leutenegger, S.; Chli, M.; Siegwart, R.Y. BRISK: Binary Robust Invariant Scalable Keypoints. In Proceedings of the 2011 International Conference on Computer Vision, Barcelona, Spain, 6—13 November 2011; pp. 2548—2555.

23. Khan, N.Y.; McCane, B.; Wyvill, G. SIFT and SURF Performance Evaluation against Various Image Deformations on Benchmark Dataset. In Proceedings of the 2011 International Conference on Digital Image Computing: Techniques and Applications, Noosa, QLD, Australia, 6—8 December 2011; pp. 501—506.

24. Lowe, D.G. Distinctive Image Features from Scale-Invariant Keypoints. Int. J. Comput. Vis. 2004, 60, 91—110.

25. Viola, P.; Jones, M. Rapid Object Detection Using a Boosted Cascade of Simple Features. In Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR 2001), Kauai, HI, USA, 8—14 December 2001; Volume 1, pp. I-511—I-518.

26. Тюкачев М. C#. Основы программирования / М. Тюкачев, В. Хлєбостроєв. — М.: Лань, 2016. — 272 с.

27. Разработка Windows-приложений на Microsoft Visual Basic .NET и Microsoft Visual C# .NET. Учебный курс MCAD MCSD: официальное пособие Microsoft. — М:, 2011. — 478 с.

28. Мак-Дональд М. WPF: Windows Presentation Foundation в .NET 4.5 с примерами на C#5.0 для профессионалов / Мэтью Мак-Дональд. — СПб.: Вильямс, 2013. — 1024 с. — (4-е).