

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»
МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ імені ІГОРЯ СІКОРСЬКОГО»

Кваліфікаційна наукова
праця на правах рукопису

СЕРГІЄНКО ПАВЛО АНАТОЛІЙОВИЧ

УДК 004.383 : 004.415.2

ДИСЕРТАЦІЯ

МЕТОДИ ТА ЗАСОБИ ПРОЕКТУВАННЯ ОБЧИСЛЮВАЧІВ ДЛЯ РОЗПІ-
ЗНАВАННЯ ОБРАЗІВ У ЗОБРАЖЕННЯХ

123 Комп'ютерна інженерія

12 Інформаційні технології

Подається на здобуття наукового ступеня доктора філософії

Дисертація містить результати власних досліджень. Використання ідей, ре-
зультатів і текстів інших авторів мають посилання на відповідне джерело

П. А. Сергієнко

Науковий керівник:

Романкевич Віталій Олексійович

доктор технічних наук, професор

Київ — 2023

АНОТАЦІЯ

Сергієнко П.А. Методи та засоби проектування обчислювачів для розпізнавання образів у зображеннях. Кваліфікаційна наукова праця на правах рукопису. Дисертація на здобуття наукового ступеня доктора філософії за спеціальністю 123 «Комп'ютерна інженерія». Національний технічний університет України «Київський політехнічний інститут імені Ігоря Сікорського», Київ, 2023.

Тема дисертації пов'язана з розробкою алгоритмів оброблення зображень та з проектуванням спеціалізованих обчислювальних засобів для їх реалізації на базі інтегральних схем надвисокої інтеграції (НВІС) та програмованих логічних інтегральних схем (ПЛІС).

Об'єктом дослідження є розпізнавання образів у зображеннях та проектування апаратних засобів для їх виконання.

Предметом дослідження є алгоритми пошуку характерних точок у зображеннях та проектування спеціалізованих обчислювальних систем для виконання цих алгоритмів.

Метою дисертації є підвищення ефективності розробки обчислювальних систем для розпізнавання образів у зображеннях на основі ПЛІС та НВІС шляхом створення нових методів проектування спеціалізованих конвеєрних структур, які дають змогу прискорити проектування обчислювальних систем і підвищити відношення продуктивності — апаратні витрати завдяки формалізації проектування та новим алгоритмам обробки зображень і пошуку характерних точок в них.

Для досягнення мети в дисертації виконуються завдання: проаналізувати задачі, алгоритми і пристрої розпізнавання образів у зображеннях, зокрема, системи розпізнавання образів на основі штучних нейронних мереж і сформулювати вимоги до елементної бази й засобів проектування обчислювальних систем для розпізнавання образів; теоретично обґрунтувати та розробити алгоритм пошуку характерних точок, тобто, локальних елементів зображення з

найбільш інформативними ознаками, які необхідні для класифікації зображень, який на відміну від існуючих алгоритмів має меншу складність та забезпечує пошук у складних умовах освітлення; створити метод побудови буферних схем для обробки одно- та двохвимірних сигналів, який забезпечує заданий порядок слідування вхідних та вихідних даних і мінімізовані апаратні витрати при його реалізації у ПЛІС; розробити способи побудови допоміжних блоків для систем розпізнавання образів, таких як обчислювач елементарних функцій, декомпресор даних; перевірити ефективність розробленого методу при проектуванні модулів спеціалізованої системи на базі ПЛІС для вирішення кола завдань розпізнавання образів.

Наукова новизна роботи. Запропоновано новий метод пошуку характерних точок у зображенні, який на відміну від існуючих методів пошуку характерних точок, таких як *scale-invariant feature transform (SIFT)* та похідних від нього, завдяки використанню нового алгоритму адаптивної фільтрації, виконує пошук характерних точок у несприятливих умовах освітленості та має обсяг обчислень зменшений до чотирьох разів.

Запропоновано новий алгоритм адаптивної фільтрації на основі блоку аналізу зображення, який детектує локальні градієнтні характеристики і формує з них зображення ознак за допомогою паралельної двовимірної фільтрації та селекції результатів фільтрації у логарифмічному масштабі, який на відміну від відомого алгоритму білатеральної фільтрації, має учетверо менше операцій множення, не потребує обчислень з підвищеною точністю і плаваючою комою та дає змогу обробляти зображення з динамічним діапазоном до 120 дБ і більше.

Створено метод синтезу буферних схем для обробки двовимірних потоків даних, який на відміну від існуючих методів дає змогу виконувати розробку буферних схем формалізовано з мінімізацією апаратних витрат, який, шляхом застосування методу просторового графа синхронних потоків даних,

направляє синтез на одержання буферів типу FIFO або пам'яті довільного доступу, забезпечуючи наперед заданий порядок та період вводу-виводу даних.

Запропоновано новий спосіб проектування буферних схем з конвеєрних регістрів у ПЛІС, який відрізняється від існуючих способів формальною побудовою функціональної схеми, в якій використовуються елементи SRL16, за рахунок чого p регістрів замінюються на k логічних таблиць і таким чином, основні апаратні витрати на регістри у ПЛІС зменшуються у $p/k \in [2..16]$ разів.

Удосконалено алгоритм та структура апаратного модуля обчислення квадратного кореня, який на відміну від відомого алгоритму зі зсувом та відніманням і його апаратної реалізації має меншу латентну затримку обчислення за рахунок застосування блоків постійної пам'яті для збереження результатів перших ітерацій алгоритму.

Практична цінність результатів дисертаційної роботи полягає в тому, що використання запропонованого методу пошуку характерних точок у зображенні у системах технічного зору дає змогу пришвидшити розпізнавання образів, зменшуючи латентну затримку між вводом зображення та виводом його ознак, покращити розпізнавання в несприятливих умовах освітлення, зменшити навантаження на лінії телекомунікацій за рахунок прорідження інформації.

Новий метод синтезу буферних схем для обробки двовимірних потоків даних та новий спосіб проектування буферів з конвеєрних регістрів у ПЛІС дають змогу пришвидшити проектування складних систем технічного зору. Інтелектуальна відеокамера, яка розроблена з їх застосуванням та в якій застосовано новий метод пошуку характерних точок, при своєму впровадженні здатна замінити камери відео-нагляду завдяки ефективному стисненню зображення з широким динамічним діапазоном та можливістю розпізнавання образів.

Розроблений вперше апаратно-програмний модуль для LZW-декомпресії має невеликі апаратні витрати, може бути впроваджений у ПЛІС різних

серій і завдяки цьому, у порівнянні з програмною реалізацією має вдвічі більшу пропускну здатність, може зменшити об'єм пам'яті та енергоспоживання і має можливість переналаштовуватись при відсутніх або невеликих додаткових апаратних витратах.

Удосконалений алгоритм та структура модуля обчислення квадратного кореня впроваджені у Web-застосунку, що генерує модулі обчислення цієї функції з заданими параметрами швидкодії, точності, апаратних витрат, які вільно поширюються і можуть бути вбудовані в довільні проєкти ПЛІС.

Результати роботи впроваджені у двох НДР, що проводяться Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського» на кафедрах обчислювальної техніки та системного програмування і спеціалізованих комп'ютерних систем, які присвячені проєктуванню високопродуктивних апаратних і програмних засобів.

Матеріали дисертації є корисними для викладачів і спеціалістів у галузях проєктування апаратних засобів обчислювальної техніки, систем телекомунікацій, зв'язку, вимірювання, штучного інтелекту, засобів мікроелектроніки, а також можуть бути застосовані у навчальному процесі у вищих навчальних закладах.

Ключові слова: Розпізнавання контурів, семантична сегментація, розпізнавання образів, стиснення даних, характерна точка, програмована логічна інтегральна схема, система на кристалі, класифікація, цифрова обробка зображень, структурний синтез, граф синхронних потоків даних.

ABSTRACT

Serhiienko P.A. Methods and tools of computer design for pattern recognition in images. Qualified scientific work on manuscript rights. Thesis for obtaining PhD scientific degree in specialty 123 "Computer Engineering". National Technical University of Ukraine "Ihor Sikorsky Kyiv Polytechnic Institute", Kyiv, 2023.

The topic of the thesis is related to the development of image processing algorithms and the design of application-specific computing tools for their implementation based on very large-scale integrated (VLSI) circuits and field programmable gate arrays (FPGA).

The object of research is the image recognition and the design of hardware for their implementation.

The subject of research is algorithms for feature extraction in images and application-specific computer systems design for these algorithms' implementation.

The purpose of the thesis is to increase the efficiency of the design of computational systems for pattern recognition in images based on FPGAs and VLSIs by developing new methods of application-specific pipeline structure design, which allows to speed up computational systems design and increases the performance-hardware cost ratio due to the formalization of design and new image processing and feature extraction algorithms.

To achieve the purpose of the thesis, the following tasks are performed: the analysis of the problems, algorithms and devices of pattern recognition in images, in particular, pattern recognition systems based on artificial neural networks and formulation of requirements for the elemental basis and design tools of the computational systems for pattern recognition; theoretical substantiation and development of the algorithm for the extraction of feature points, i.e., local image elements with the most informative features that are necessary for image classification, which, unlike existing algorithms, has less complexity and provides the search in difficult lighting conditions; development of a method of the design of buffer circuits for processing one- and two-dimensional signals, which ensures a specified sequence of input and output data and minimized hardware costs when it is implemented in FPGA; development of the approaches to building auxiliary blocks for pattern recognition systems, such as a calculator of elementary functions, a data decompressor; assessment of the effectiveness of the developed method when designing modules of

an application-specific system based on FPGA for solving a range of pattern recognition tasks.

Scientific novelty of the work. A new method for feature extraction in images is proposed, which, unlike existing feature extraction methods, such as scale-invariant feature transform (SIFT) and derived from it, by using new algorithm of adaptive filtering, performs the feature extraction in adverse lighting conditions and has a volume of calculations reduced up to four times.

A new adaptive filtering algorithm based on an image analysis block is proposed, which detects local gradient characteristics and forms an image of features using parallel two-dimensional filtering and selection of logarithmic-scale filtering results, which, unlike the well-known bilateral filtering algorithm, has four times fewer multiplication operations and does not require increased precision and floating point computations and capable to process images with a dynamic range of up to 120 dB and more.

A method for the synthesis of buffer circuits for processing two-dimensional data streams has been created, which, unlike existing methods, allows the buffer design in a formalized manner with the minimization of hardware costs, which, by applying the method of a spatial synchronous dataflows, directs the synthesis to obtaining the FIFO buffers or RAM, ensuring a predetermined order of data input and output.

A new method of designing buffers from pipeline registers in FPGAs is proposed, which differs from existing methods by the formal design of a functional network that uses SRL16 elements, due to which p registers are replaced by k logical tables and thus, the main hardware costs for registers in the FPGA are reduced by $p/k \in [2..16]$ times.

The algorithm and structure of the hardware square root calculation module have been improved, which, unlike the well-known shift-and-subtract algorithm and its hardware implementation, has a lower latency calculation delay due to the use of ROM blocks to store the results of the first iterations of the algorithm.

The practical value of the thesis results are that the use of the proposed method of feature extraction in the image in technical vision systems makes it possible to speed up pattern recognition, reducing the latent delay between the input of the image and the output

of its features, to improve recognition in adverse lighting conditions, and to reduce the load on telecommunication lines due to information thinning.

A new method of synthesizing buffers for processing two-dimensional data flows and a new way of designing buffers from pipeline registers in FPGAs make it possible to speed up the design of complex technical vision systems. An intelligent video camera, which is developed with their application and in which a new method of feature extraction is implemented, can replace video surveillance cameras due to effective image compression with a high dynamic range and the pattern recognition ability.

The first-developed hardware-software module for LZW decompression has low hardware costs, can be implemented in FPGAs of various series, and due to this, compared to the software implementation, it has twice the bandwidth, can reduce the amount of memory and power consumption, and has the reconfigurability with no or little additional hardware costs.

The improved algorithm and structure of the square root calculation module are implemented in a Web application that generates the modules of this function calculation with given parameters of performance, accuracy, hardware costs, which are freely distributed and can be embedded in arbitrary FPGA projects.

The results of the work have been implemented in two scientific research works held at the National Technical University of Ukraine "Ihor Sikorsky's Kyiv Polytechnic Institute" at the departments of computer engineering and system programming, and specialized computer systems, which are dedicated to the design of high-performance hardware and software. The thesis materials are useful for teachers and specialists in the fields of computer hardware design, telecommunication systems, communication, measurement, artificial intelligence, microelectronics, and can also be used in the educational process at higher educational institutions.

Keywords: Contour detection, edge computing, semantic segmentation, pattern recognition, data compression, feature point, field programmable gate array, system on chip, classification, digital image processing, structural synthesis, synchronous data flow.

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці в яких опубліковано основні наукові результати дисертації:

1. Serhiienko P. A., Sergiyenko A. M., Romankevich V. O., Molchanov O. A., Mozghovyi I. V., Zacharioudakis L. Decompressor for hardware applications. *Applied Aspects of Information Technology*. 2023. V. 6, No.1: pp.74-83. (фахове вид. кат. Б)
2. Serhiienko P. A., Romankevich V. O., Sergiyenko A. M. Image buffering in application specific processors. *Applied Aspects of Information Technology*. 2022. V.6. No. 5, pp 228–239.
<https://doi.org/10.15276/aait.05.2022.16> (фахове вид. кат. Б)
3. Serhiienko P. A., Romankevich V. O., Sergiyenko A. M. Local Feature Extraction in High Dynamic Range Images. *Elektronnoe modelirovanie*, 2022, v. 44. No. 4, pp. 125–139.
<https://doi.org/10.15407/emodel.44.04.041> (фахове вид. кат. Б)
4. Сергієнко П.А., Кліменко І.А., Сергієнко А.М. Реконфігурована багатопроцесорна обчислювальна система на ПЛІС. Вісник НТУУ «КПІ». Ін форматика, управління та обчислювальна техніка: Зб. наук. пр. К.: Век+, 2016, № 64, с. 47-50. (фахове вид. кат. Б)
5. Сергієнко А.М., Сергієнко П.А. Реалізація функції квадратного кореня у ПЛІС. Вісник НТУУ «КПІ»: “Інформатика, управління та обчислювальна техніка”: зб. наук. праць. 2014. Т.60, с. 40–45. ISSN 0135-1729. (фахове вид. кат. Б)

В яких засвідчено апробацію матеріалів дисертації

6. Serhiienko P., Orlova M., Sergiyenko A., Molchanov O. System for Video Pattern Recognition on FPGA. *2019 IEEE 2nd Ukraine Conference on Electrical and Computer Engineering*, Lviv, Ukraine, July 2-

- 6, 2019, pp. 1175–1178.
<https://doi.org/10.1109/UKRCON.2019.8879958> (реф Scopus).
7. Sergiyenko A., Serhienko P., Zorin Ju. High Dynamic Range Video Camera with Elements of the Pattern Recognition. *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*, Kyiv, Ukraine, 2018, pp. 435-438, doi: 10.1109/ELNANO.2018.8477556 (реф Scopus).
 8. Сергієнко П.А. Проблеми розвитку комп'ютерного зору. *Філософські засади креатосфери у контексті творчості: Матеріали 15 Міжнародної науково-практичної конференції*, Київ, Україна, 30 травня 2019, С. 178-181.
 9. Сергиєнко П. А. Сергиєнко А. М., Молчанов А. А., Мозговой И. В. Система динамической визуализации на базе аппаратного GIF-декомпрессора. *Збірник праць міжнародної конференції Simulation-2018*, 12-14 вересня 2018, Київ, - с. 221–223.
 10. Serhienko P. A., Hasan M. J., Sergiyenko A. M. Square root calculations in FPGA. *System analysis and information technology: 20-th International conference SAIT`2018*, Kyiv, Ukraine, May 21 – 24, 2018. *Proceedings. – ESC “IASA” NTUU “Igor Sikorsky Kyiv Polytechnic Institute”*, 2018.p. 163-164.
 11. Serhienko P., Sergiyenko A., Molchanov O. Reconfigurable Many-core System. *Праці 5-ї міжнародної конференції «High Performance Computing» HPC-UA`2018*. 2018, с. 127 – 130.
 12. Сергієнко А.М., Зорін Ю.М., Сергієнко П.А. Пошук характерних ознак у зображенні з широким динамічним діапазоном. *10 наукова конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг", ПМК-2018*. Київ: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка». — 2018, С. 65-69.

- 13.Сергієнко П.А., Зорін Ю. М. Детектування характерних точок у зображенні. *Прикладна математика та комп'ютинг. ПМК, 2017 : 9 наук. конф. магістрантів та аспірантів*, Київ, 19-21 квіт. 2017 р. : – К.: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка».
- 14.Сергієнко А.М., Сергієнко П.А. Багатопроцесорна система на ПЛІС для синхронної обробки потоків даних. *Прикладна математика та комп'ютинг. ПМК, 2016 : восьма наук. конф. магістрантів та аспірантів*, Київ, 20-22 квіт. 2016 р. : зб. тез доп. К. ПМК-2016. – К.: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка». 2016, с. 124-127.
- 15.Сергієнко П.А., Лепеха В.Л., Ядро 16-розрядного RISC–процесора». *Матеріали наук. конф. студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка – ІОТ-2016»*, 25 – 27 квітня, 2016. — К: НТУУ «КПІ», 2016. — С. 118 – 119.
- 16.Сергієнко П. А., Сергієнко А. М. Ядро RISC-процесора для реалізації у ПЛІС. *InfoCom '2015 : праці 1 міжнародної конференції*, Київ, 24–25 листопада 2015 р. К.: НТУУ “КПІ”, ВПІ “Політехніка”, 2015. С. 54–55.

Які додатково відображають наукові результати дисертації

- 17.Serhiienko P., Sergiyenko A., Mozgovyi I., Molchanova A. Design of Data Buffers in Field Programmable Gate Arrays. *Information, Computing and Intelligent systems*, 2022, No. 3, pp. 1–16.
<https://doi.org/10.20535/2708-4930.2.2021.244191>
- 18.Serhiienko P. A., Orlova M. M, Sergiyenko A. M. Local Feature Extraction in Images. *Information, Computing and Intelligent systems*, 2021, No. 2, pp. 1–13. <https://doi.org/10.20535/2708-4930.2.2021.244191>

ЗМІСТ

ЗМІСТ	12
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ	16
ВСТУП	17
РОЗДІЛ 1. МЕТОДИ ТА ЗАСОБИ РОЗПІЗНАВАННЯ ОБРАЗІВ У ЗОБРАЖЕННЯХ	26
1.1 Комп'ютерний зір	27
1.1.1 Види цифрової обробки зображень комп'ютерного зору	28
1.1.2 Методи розпізнавання образів.....	30
1.2 Розпізнавання образів у штучних нейронних мережах	31
1.2.1. Вступ	31
1.2.2. Багатошаровий перцептрон	32
1.2.3. Згорткова нейронна мережа.....	33
1.2.4. Функція втрат	36
1.2.5. Навчання нейронних мереж.....	37
1.2.6. Реалізація нейронної мережі у ПЛІС	39
1.2.7. Огляд поширених моделей нейронних мереж	40
1.2.8. Висновки щодо згортувальних нейронних мереж	41
1.3. Методи визначення характерних точок.....	43
1.3.1. Методи, основані на аналізі градієнта яскравості	44
1.3.2 Методи на основі аналізу яскравості	55
1.3.3. Методи на основі функцій перетворень	56
1.3.4. Методи, що ґрунтуються на аналізі форми	58
1.3.5. Методи з перетворенням простору	60
1.3.6. Глибоке навчання і пошук характерних точок	61
1.3.7. Розпізнавання образів за характерними точками	62

1.3.7. Порівняння методів пошуку характерних точок	63
1.3.8. Висновки щодо методів пошуку характерних точок.	64
1.4. Апаратна реалізація розпізнавання образів.....	65
1.4.1. Апаратна реалізація методів дескрипторів характерних точок	66
1.4.2. Реалізація буферних схем	69
1.5. Висновки до першого розділу	72
РОЗДІЛ 2. РОЗРОБЛЕННЯ МЕТОДУ ПОШУКУ ХАРАКТЕРНИХ ТОЧОК У ЗОБРАЖЕННІ	75
2.1. Передумови для створення методу	75
2.2 Оброблення зображення з широким динамічним діапазоном	77
2.2.2. Алгоритми перетворення тонів зображення	78
2.2.3. Висновки щодо алгоритмів перетворення тонів зображення.....	85
2.3. Алгоритм адаптивної фільтрації HDR-зображення	86
2.3.1. Основа алгоритму	86
2.3.2. Адаптивна фільтрація HDR-зображення.....	87
2.3.3. Алгоритм адаптивної фільтрації для обробки HDR-зображень	90
2.3.4. Алгоритм компресії HDR-зображення	93
2.3.5. Висновки щодо алгоритму компресії HDR-зображення	98
2.4. Метод пошуку характерних точок на основі зображення ознак.....	99
2.4.1. Особливості зображення ознак.....	99
2.4.2. Алгоритм фільтрації зображення ознак.....	103
2.4.3. Алгоритм побудови піраміди зображень ознак	107
2.4.4. Алгоритм знаходження характерних точок	110

2.4.5. Формування дескриптора характерної точки	117
2.4.6. Метод пошуку характерних точок у зображенні.....	119
2.4.7. Застосування метод пошуку характерних точок у зображенні.....	121
2.4.8. Висновки до розділу	123
РОЗДІЛ 3. СПОСОБИ АПАРАТНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМІВ ПОШУКУ ХАРАКТЕРНИХ ТОЧОК У ЗОБРАЖЕННІ	126
3.1. Методи і способи проєктування апаратних засобів для цифрової обробки сигналів та зображень.....	126
3.1.1. Графові моделі потокових алгоритмів.....	126
3.1.2 Просторовий ГСПД	129
3.1.3. Синтез конвеєрних обчислювачів на базі просторового ГСПД	131
3.2. Метод проєктування буферних схем для цифрової обробки зображень	137
3.2.1. Методи та способи проєктування буферних схем.....	137
3.2.2. Ресурси ПЛІС для проєктування буферних схем	144
3.2.3. Цілі та задачі дослідження проєктування буферних схем..	146
3.2.4. Проектування буферних схем для обробки сигналів.....	146
3.2.5. Спосіб проєктування буферів з конвеєрних регістрів	151
3.2.6. Приклад синтезу буфера з конвеєрних регістрів.....	152
3.2.7. Метод синтезу буферних схем для обробки двовимірних потоків даних	156
3.2.8. Приклад застосування методу	159
3.3. Проектування конвеєрних пристроїв для обчислення квадратного кореня	164
3.3.1. Функція квадратного кореня у системах обробки зображень	164

3.3.2. Особливості апаратних ресурсів ПЛІС та складність модулів	164
3.3.3. Алгоритми добування квадратного кореня.....	165
3.3.4 Модифікація алгоритму зі зсувом та відніманням	167
3.3.5. Реалізація модифікованого алгоритму	170
3.4. Засоби компресії даних в системах машинного зору	174
3.4.1. Використання компресії в системах оброблення даних	174
3.4.2. Алгоритми і засоби декомпресії.....	175
3.4.3. Декомпресія за алгоритмом LZW	177
3.4.4. Модуль для LZW-розпакування	180
3.4.5. Результати реалізації в ПЛІС	183
3.5. Висновки до розділу 3	185
ВИСНОВКИ.....	187
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	189
ДОДАТОК А.....	218
ДОДАТОК Б	221

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

RTL – register-transfer level

FIFO – first in, first out

LZW – Lempel–Ziv–Welch algorithm

CNN – convolutional neural network

HDR – high dynamic range

SDF – synchronous data flow

BRAM – block random access memory

ВСТУП

Актуальність теми

У сучасних засобах обчислювальної техніки, таких, як персональні комп'ютери, засоби мобільного зв'язку та інтернету речей (IoT) виконуються і різноманітні задачі розпізнавання образів у зображеннях. Наприклад, таке розпізнавання є основою машинного зору. Система машинного зору фіксує зображення за допомогою камери та аналізує їх для створення опису зображеного. Типовим застосуванням системи машинного зору є розпізнавання дефектів у обробній промисловості, або для автоматизованого візуального контролю, або для розпізнавання дорожніх знаків чи перешкод на транспорті [1].

Розпізнавання образів стає масовим завдяки наявності обчислювальних ресурсів достатньо високої потужності. Але можливості сучасних процесорів загального призначення є недостатніми для вирішення таких задач. Через нестачу власної обчислювальної потужності зараз обчислювальні засоби звертаються до хмарних ресурсів з метою вирішення складних задач розпізнавання образів. При цьому часте звертання до хмари приводить до перенавантаження телекомунікацій [2].

Щоб покращити вирішення цих задач необхідно застосовувати спеціалізовані процесори, які адаптовані до розпізнавання образів. Так, до смартфонів додаються спеціалізовані процесори штучного інтелекту [3]. На самих хмарних серверах для розпізнавання образів використовуються високопродуктивні графічні акселератори. Вони все частіше замінюються на програмовні логічні інтегральні схеми (ПЛІС) які виконують ті самі алгоритми зі значно меншими енерговитратами [4]. Відеокамери спостереження та інші системи збору інформації оснащують ПЛІС для розпізнавання образів та кардинального зменшення потоків інформації до хмарного сервера [5].

ПЛІС переважають універсальні процесори і надвеликі інтегральні схеми (НВІС) при реалізації паралельних алгоритмів завдяки можливості

перепрограмування апаратних функцій і більшому відношенню продуктивності — енергоспоживання [6].

Зараз великі проекти для НВІС або ПЛІС виконують за допомогою їх опису на рівні регістрових передач (RTL), що є трудовитратним, повільним і залежить від майстерності розробника [7, 8]. Щоби прискорити розробку, використовують САПР високорівневого синтезу, такі як Celoxica, Cadence C2S, Mathworks Simulink System Generator, Xilinx AccelDSP, Synopsys SynplifyDSP. Однак, вони дають ефективні рішення лише для деяких видів алгоритмів. Поширений підхід ґрунтується на тому, що шукана структура системи компілюється з макроблоків, які виконують стандартні процедури з, таких бібліотек як OpenCE, OpenCV, TensorFlow. Для цього макроблоки мають бути заздалегідь розроблені й описані мовою Verilog, VHDL чи SystemC. При цьому підготовка макроблоків вимагає трудомісткої кваліфікованої праці розробників [6, 9].

Наявна велика множина робіт у теорії відображення паралельних алгоритмів в обчислювальні системи (ОС), які орієнтовані на реалізацію на основі ПЛІС і НВІС, як, наприклад, [10, 11, 12]. Але лише деякі з цих методів набули широкого застосування, зокрема в комерційних САПР. Отже, щоб ефективно розробляти проекти систем розпізнавання образів на базі НВІС або ПЛІС зі складністю більш 200–500 тис. вентилів, необхідно розробляти нові методи й засоби відображення алгоритмів у обчислювальне середовище [13].

Отже, завдання проєктування ОС для ПЛІС і НВІС, які ставить сучасний етап розвитку систем штучного інтелекту, не можуть бути виконані без розроблення нових методів і засобів відображення алгоритмів розпізнавання образів. Таке розроблення можливе після аналізу наявних моделей і засобів завдання алгоритмів розпізнавання образів, вибору такої обчислювальної моделі, яка забезпечувала б простоту свого аналізу і високу ефективність відображення алгоритму в апаратні засоби ОС, а також розроблення методів оптимізації такого відображення.

Отже, дослідження й розроблення методів та засобів проектування обчислювачів для розпізнавання образів у зображеннях, а також їх практична реалізація й апробація є актуальними, мають теоретичний і практичний інтерес.

Об’єктом дослідження є розпізнавання образів у зображеннях та проектування апаратних засобів, для їх виконання.

Предметом дослідження є алгоритми пошуку характерних точок у зображеннях та проектування спеціалізованих обчислювальних систем для виконання цих алгоритмів.

Метою дисертації є підвищення ефективності проектування ОС для розпізнавання образів у зображеннях на основі ПЛІС завдяки використанню нових методів проектування спеціалізованих конвеєрних структур, які дають змогу прискорити проектування ОС і підвищити відношення продуктивність — апаратні витрати завдяки формалізації проектування і новим алгоритмам обробки зображень і пошуку характерних точок в них.

Для досягнення поставленої мети в дисертації вирішуються такі завдання:

1. Проаналізувати завдання, алгоритми і пристрої розпізнавання образів у зображеннях і сформулювати вимоги до елементної бази й засобів проектування ОС для розпізнавання образів.
2. Теоретично обґрунтувати та розробити алгоритм пошуку характерних точок, який на відміну від існуючих алгоритмів має меншу складність та забезпечує пошук у складних умовах освітлення.
3. Створити метод побудови буферних схем для обробки одно- та двовимірних сигналів, який забезпечує заданий порядок слідування вхідних та вихідних даних і мінімізовані апаратні витрати при його реалізації у ПЛІС.
4. Розробити способи побудови допоміжних блоків для систем розпізнавання образів, таких як обчислювач елементарних функцій, декомпресор даних.

5. Перевірити ефективність розробленого методу під час проєктування модулів спеціалізованої системи на базі ПЛІС для вирішення кола завдань розпізнавання образів.

На захист виноситься наступне.

1. Метод пошуку характерних точок у зображенні.
2. Метод синтезу буферних схем для обробки двовимірних потоків даних.
3. Спосіб проєктування буферів з конвеєрних регістрів у ПЛІС.
4. Алгоритм та структура модуля обчислення квадратного кореня.

Методи досліджень ґрунтуються на використанні теорії графів яка використовувалась при дослідженні алгоритмів, теорії алгоритмів, на основі якої обґрунтовувалось відображення алгоритмів у структури, теорії моделювання на основі якої доводилась коректність одержаних практичних результатів, методів комбінаторної оптимізації, які використовувались при синтезі конвеєрних структур, а також тверджень та висновків, які доведені в дисертації. Достовірність висновків і результатів використання запропонованих методів перевірялась за допомогою їх комп'ютерного моделювання, а також конфігурування одержаних проєктних рішень у ПЛІС з їх випробуванням у лабораторних умовах.

Наукова новизна роботи.

Запропоновано новий метод пошуку характерних точок у зображенні, який на відміну від існуючих методів пошуку характерних точок, таких як SIFT, завдяки використанню нового алгоритму адаптивної фільтрації виконує пошук характерних точок у несприятливих умовах освітленості та має обсяг обчислень зменшений до чотирьох разів.

Запропоновано новий алгоритм адаптивної фільтрації на основі блоку аналізу зображення, який детектує локальні градієнтні характеристики і формує з них зображення ознак, який на відміну від алгоритму білатеральної

фільтрації, має учетверо менше операцій множення і не потребує обчислень з підвищеною точністю.

Створено новий метод синтезу буферних схем для обробки двовимірних потоків даних, який на відміну від існуючих методів дає змогу виконувати розробку буферних схем формалізовано з мінімізацією апаратних витрат, який, завдяки застосуванню методу просторового графа синхронних поттоків даних, направляє синтез на одержання буферів типу FIFO або пам'яті довільного доступу, забезпечуючи наперед заданий порядок вводу-виводу даних.

Запропоновано новий спосіб проектування буферів з конвеєрних регістрів у ПЛІС, який відрізняється від існуючих способів формальною побудовою функціональної схеми, в якій використовуються елементи SRL16, за рахунок чого p регістрів замінюються на k логічних таблиць, де $p/k = 2 - 16$.

Удосконалено алгоритм та структура модуля обчислення квадратного кореня, який на відміну від відомого алгоритму зі зсувом та відніманням має меншу затримку обчислення за рахунок застосування блоків постійної пам'яті.

Практична цінність результатів дисертаційної роботи полягає в тому, що використання запропонованого методу пошуку характерних точок у зображенні у системах технічного зору дає змогу пришвидшити розпізнавання образів, покращити розпізнавання в несприятливих умовах освітлення, зменшити навантаження на лінії телекомунікацій за рахунок прорідження інформації.

Новий метод синтезу буферних схем для обробки двовимірних потоків даних та новий спосіб проектування буферів з конвеєрних регістрів у ПЛІС дають змогу пришвидшити проектування складних систем технічного зору. Інтелектуальна відеокамера, яка розроблена з їх застосуванням та в якій впроваджено новий метод пошуку характерних точок, при своєму впровадженні здатна замінити камери відеонагляду завдяки ефективному стисненню зображення з широким динамічним діапазоном та можливістю розпізнавання образів.

Удосконалений алгоритм та структура модуля обчислення квадратного кореня впроваджені у Web-застосунок, що генерує модулі обчислення цієї функції з заданими параметрами швидкодії, точності, апаратних витрат, які вільно поширюються і можуть бути вбудовані в довільні проєкти ПЛІС.

Розроблений вперше апаратно-програмний модуль для LZW-декомпресії має невеликі апаратні витрати, може бути впроваджений у ПЛІС різних серій і завдяки цьому, у порівнянні з програмною реалізацією має вдвічі більшу пропускну здатність, може зменшити об'єм пам'яті та енергоспоживання і має можливість переналашуватись при відсутніх або невеликих додаткових апаратних витратах.

Зв'язок дисертації з науково-дослідними роботами полягає у такому: Результати роботи впроваджені у Національному технічному університеті України «Київський політехнічний інститут імені Ігоря Сікорського»: 1. НДР ЗОТ/2027 «Методи і засоби відображення потокових алгоритмів у конфігуровані комп'ютери» № держреєстрації 0117U005087; 2. НДР «Методи, моделі та комп'ютерні засоби виявлення деструктивного впливу в медіапросторі», № держреєстрації 0121U110662;

Правдивість тверджень та висновків підтверджується строгими теоретичними доведеннями та результатами випробувань моделей і зразків обчислювальних схем, які побудовані на їхній основі. Основні положення й теоретичні оцінки підтверджені результатами імітаційного моделювання на комп'ютері та в VHDL-симуляторі. Експериментальна перевірка наукових положень, пропозицій і одержаних результатів виконувалася через проєктування експериментального зразка інтелектуальної відеокамери на базі HDR-датчика зображення та ПЛІС за запропонованими методами і способами з використанням мови VHDL з подальшим моделюванням у симуляторі, компілюванням у схему на рівні вентилів та конфігуруванням у ПЛІС фірм Lattice та AMD-Xilinx.

Апробація роботи. Основні результати дисертаційної роботи були представлені й обговорювалися на 13 міжнародних та регіональних наукових конференціях, у тому числі: науковій конференції студентів, магістрантів та аспірантів “Інформатика та обчислювальна техніка” (ІОТ-2016, Київ), 8-й, 9-й, 10-й наукових конференціях магістрантів та аспірантів “Прикладна математика та комп’ютинг” (ПМК-2016, Київ, 2016 р, ПМК-2017, Київ, 2017 р, ПМК-2018, Київ, 2018 р), 1-й, 3-й, 5-й міжнародних конференціях “Безпека, Відмовостійкість, Інтелект” (ICSFTI2018, Київ, 2018 р., ICSFTI`2020, Київ, 2020 р., ICSFTI`2022, Київ, 2022 р.), 5-й міжнародній конференції “High Performance Computing” (HPC-UA’18, Київ, 2018 р.), 20-й міжнародній конференції “System analysis and information technology” (SAIT`2018, Київ, 2018 р.), 6-й міжнародній конференції “Simulation-2018” (Київ, 2018 р.), 38-й міжнародній конференції “IEEE Electronics and Nanotechnology” (ELNANO’2018, Київ, 2018 р.), 2-й міжнародній конференції “IEEE Ukraine Conference on Electrical and Computer Engineering” (UKRCON`2019, Львів, 2019 р.).

Публікації. За матеріалами дисертації опубліковано у 21 наукових роботах. Основні результати викладені в 18 роботах, у тому числі в 5 статтях у виданнях зі списку МОН [212, 238, 139, 202, 244], а також у 2 статтях у збірниках праць, які входять у наукометричну базу Scopus [143, 162].

У роботах [13, 19, 22, 192, 193, 198] автором запропоновано основи методу пошуку характерних точок, алгоритм і структуру адаптивного фільтра для їх пошуку. У роботах [194, 195] автором зроблено огляд алгоритмів розпізнавання образів та висновки про перспективність методів пошуку характерних точок. У роботах [197, 199] автором запропоновано новий метод побудови буферів даних. У роботах [16, 191] автором запропоновано багатоядерну конфігуровану архітектуру для вирішення задач технічного зору. У роботах [14, 21, 190] автором запропоновані удосконалений алгоритм і структура для обчислення квадратного кореня. У роботах [20, 196] автором запропоновано використати алгоритми LZW і GIF для вирішення прикладних задач технічного

зору, а у роботі [200] розглянуто реалізацію апаратно-програмного декомпресора. Прикладні аспекти застосування результатів розглянуті в роботах [15, 16, 17, 18, 202].

Структура й обсяг роботи. Дисертаційна робота складається зі вступу, трьох розділів, висновків, які викладені на 168 сторінках тексту, вміщує 63 рисунків і 11 таблиць, список літературних джерел із 244 найменування.

У вступі обґрунтовується актуальність теми дисертаційної роботи, формулюється мета й завдання дослідження, основні положення, що виносяться на захист.

У першому розділі проаналізовані сучасні методи розпізнавання графічних образів і зроблено ряд висновків. Зокрема, це висновок про те, що нейронні мережі, що стали найпоширенішою технологією для розпізнавання образів, мають ряд недоліків, таких як висока складність навчання, негарантованість одержання ефективно навченої мережі, складність обчислень мережі, ненадійне розпізнавання об'єктів з різними масштабами та кутами повороту, а також довга латентна затримка. Ці недоліки мінімізуються у методах дескрипторів характерних точок, які дають змогу розпізнати об'єкт у різноманітних обставинах і дають можливість суміщати пошук характерних точок із нейронною мережею. В свою чергу, методи пошуку характерних точок мають той недолік, що вони розраховані на обробку зображення з невеликим динамічним діапазоном, який визначається розрядністю пікселів і втрачають ефективність при погіршенні умов освітленості. Тому варто шукати удосконалений метод пошуку характерних точок, такий як SIFT.

Також проаналізовані особливості апаратної реалізації методів пошуку характерних точок та формування їх дескрипторів і причини великих апаратних витрат на їх реалізацію. В результаті, сформульовані завдання на наукові дослідження, опис виконання яких приведений у інших розділах.

У другому розділі проаналізовано методи стиснення зображення з широким динамічним діапазоном і вибрано метод Retinex на основі білатеральної

фільтрації як найкращий за якістю стиснення. Запропоновано новий алгоритм адаптивної фільтрації на основі блоку аналізу зображення, який детектує локальні градієнтні характеристики і формує з них зображення ознак. Запропоновано удосконалений алгоритм компресії HDR-зображення, в якому білатеральну фільтрацію замінено на алгоритм адаптивної фільтрації, завдяки чому зменшена складність алгоритму компресії і покращується розпізнавання образів у складних умовах освітлення. Проаналізовані властивості зображення ознак, що генерується блоком аналізу зображення алгоритму адаптивної фільтрації і встановлено, що воно вміщує великий обсяг прорідженої інформації про форми і лінії об'єктів і може бути використане для побудови нових алгоритмів розпізнавання образів.

Створено новий алгоритм MHN-фільтрації, завдяки якому у зображенні ознак вилучаються шумові пікселі, які заважають подальшому знаходженню характерних точок. Створено алгоритм аналізу зображення, який обчислює піраміду зображень ознак на основі алгоритму адаптивної фільтрації, який для прорідження зображень не потребує додаткової фільтрації нижніх частот. Розроблено новий алгоритм знаходження характерних точок, який на відміну від відомих алгоритмів пошуку характерних точок забезпечує пошук у несприятливих умовах освітленості та має меншу складність. Запропонована процедура формування дескриптора характерної точки за результатами виконання нового алгоритму знаходження характерних точок, яка є менш складною, ніж у відомих алгоритмах пошуку характерних точок. І насамкінець, розроблено новий метод пошуку характерних точок у зображенні з етапами формування піраміди зображень ознак за допомогою запропонованого алгоритму адаптивної фільтрації, фільтрації зображень піраміди з застосуванням нового алгоритму MHN-фільтрації, пошуку координат характерних точок, вибору характерних точок і формування дескрипторів знайдених характерних точок.

У третьому розділі проаналізовані способи представлення алгоритмів, що обробляють потоки даних і вибрано метод проєктування конвеєрних

пристроїв для обробки сигналів на основі просторового графа синхронних потоків даних як метод для структурного синтезу спеціалізованих конвеєрних пристроїв для обробки зображень. Проаналізовані відомі методи проєктування буферних схем для пристроїв обробки зображень і встановлено їхні недоліки. Вперше запропоновано спосіб проєктування буферів з конвеєрних регістрів у ПЛІС, який відрізняється від існуючих способів формальною побудовою функціональної схеми, в якій використовуються елементи SRL16. Розроблено новий метод синтезу буферних схем для обробки двовимірних потоків даних, який дає змогу проєктувати буферні схеми формалізовано з мінімізацією апаратних витрат, направляючи синтез на одержання буферів з заданими властивостями. Удосконалено алгоритм та структура модуля обчислення квадратного кореня, який на відміну від відомого алгоритму зі зсувом та відніманням має меншу затримку обчислення. Проаналізовано алгоритми безвтратної компресії і вибрано алгоритм LZW як такий, що може забезпечити досить великий коефіцієнт компресії, високу пропускну спроможність та невеликі апаратні витрати при його реалізації в ПЛІС. Запропоновано апаратно-програмний безвтратний декомпресор за алгоритмом LZW та GIF на основі процесорного ядра зі стековою архітектурою для реалізації у ПЛІС, який у порівнянні з програмною реалізацією має вдвічі більшу пропускну здатність, а в порівнянні з апаратною реалізацією даного алгоритму забезпечує багатофункціональне використання процесорного ядра.

У висновку наведено основні теоретичні та практичні результати роботи, рекомендації з їхнього використання.

РОЗДІЛ 1. МЕТОДИ ТА ЗАСОБИ РОЗПІЗНАВАННЯ ОБРАЗІВ У ЗОБРАЖЕННЯХ

У сучасних засобах обчислювальної техніки, таких, як персональні комп'ютери, засоби мобільного зв'язку та інтернету речей (IoT) реалізовані методи та засоби комп'ютерного зору зіграли вирішальну роль у розробці

різноманітних застосунків на основі обробки зображень протягом останнього десятиліття. Це, наприклад, різноманітні служби розпізнавання образів, що надаються фірмами Google, Facebook, Microsoft. За цей час технологія, що базується на комп'ютерному зорі, перетворилася з простого способу обробки зображень на інтелектуальні обчислювальні системи, які можуть сприймати реальний світ.

Знання з комп'ютерного зору та розпізнавання образів є важливими в багатьох сучасних інноваційних проєктах і, ймовірно, стануть ще важливішими в найближчому майбутньому [14].

1.1 Комп'ютерний зір

Комп'ютерний зір – це галузь інформатики, яка має на меті надати комп'ютерам аналогічні можливості, які має людина. Так, дивлячись на зображення, людина може побачити на ньому об'єкт, класифікувати, локалізувати його. Так само вона може локалізувати та позначити всі об'єкти, які присутні на зображенні і сегментувати їх, включивши їх в певні множини. Отже, наука комп'ютерного зору займається розробкою методів, які здатні відтворити властивості людської зорової системи — визначення властивостей тривимірного реального світу виключно за допомогою світла, відбитого від різних об'єктів.

Методи комп'ютерного зору сьогодні використовуються у великій множині реальних програм, призначених для інтелектуальної взаємодії людини з комп'ютером, робототехніки та мультимедіа.

Застосуванню методів комп'ютерного зору передуює обробка зображення. Метою обробки зображень є виділення фундаментальних примітивів зображення, включаючи краї та кути, фільтрацію, тощо. Сукупність цих примітивів зазвичай представляє зображення. На відміну від обробки зображення, яка в основному зосереджена на опрацюванні інформації з датчиків зображень, комп'ютерний зір створює семантичний опис зображення [15].

1.1.1 Види цифрової обробки зображень комп'ютерного зору

Цифрова обробка зображення — це обробка двовимірного представлення просторового об'єкта чи сцени за допомогою послідовності математичних операцій з метою одержання бажаного результату з використанням необхідних комп'ютерних засобів.

Покращення зображення означає таку його обробку, коли підвищується його суб'єктивна якість або здатність розпізнавання об'єктів у зображенні. Це, наприклад, зменшення шуму, покращення контрасту, різкості, кольорова корекція.

Реставрація зображення — це таке його покращення, яке використовує інформацію про процес виникнення деградації зображення і ґрунтується на побудові зворотного, інверсного процесу.

Реконструкція зображення — це така обробка непрямих даних про зображення (але прямих даних про властивості об'єкта), щоб одержати якісне зображення. Наприклад, для комп'ютерної томографії зображення формується непрямим способом - на основі моделі взаєморозміщення і властивостей біологічних тканин, яка отримується шляхом математичного аналізу системи отриманих сигналів.

Аналіз зображення — це вилучення з нього певної інформації за допомогою комп'ютерних засобів. Результатом цього є деякі цифрові параметри зображення.

Розпізнавання образів — це ідентифікація об'єктів у зображенні на основі множини образів, які визначені заздалегідь.

Комп'ютерний зір — це обробка зображення на основі певної моделі створення зображення, яка приймає до уваги початкову тривимірну сцену та стереоскопічний ефект. Результатом такої обробки є певна інтерпретація вмісту сцени, що розглядається.

Результати обробки зображення використовуються у відповідних галузях, серед яких найчастіше застосовуються наступні.

Машинний зір — це використання обробки зображень як частини системи автоматизованого керування. При цьому зображення реєструються, аналізуються у реальному часі, а результати обробки використовуються для керування деяким пристроєм.

Віддалене спостереження — це використання аналізу зображення для одержання даних на відстані. Це, наприклад, обробка супутникових знімків, даних з веб-камер.

Обробка медичних зображень передбачає обробку зображень різної природи — рентгенівських, ультразвукових, інфрачервоних зображень та зображень, отриманих шляхом ядерного магнітного резонансу. При цьому використовується реконструкція зображення з потоків даних від різноманітних датчиків.

Ущільнення зображень та відеопотоків — це компресія одиничних або множинних зображень для зменшення обсягів пам'яті для їхнього зберігання чи для ущільнення каналів зв'язку при їхньому передаванні. При цьому, як правило, зображення перетворюється у кодовану форму, яка за допомогою реконструкції може бути перетворена у зображення необхідної якості.

Отже, методи розпізнавання зображень ґрунтуються на процедурах покращення, реставрації, реконструкції, аналізу зображення та розпізнавання образів та використовуються в системах машинного зору, віддаленого спостереження, обробки медичних зображень та глибокого ущільнення зображень [16].

У даному розділі зроблено огляд методів розпізнавання образів, зокрема, методів визначення характерних точок, який є розширенням огляду, виконаного автором у роботі [17] з метою вибору найбільш перспективного методу, який вартий удосконалення. Спочатку, слід дати визначення основним поняттям теорії розпізнавання образів.

1.1.2 Методи розпізнавання образів

Розпізнавання образів є складовою теорії машинного навчання (machine learning). Існує два підходи до навчання: індуктивне і дедуктивне.

Індуктивне навчання, або навчання за прецедентами, засноване на виявленні загальних властивостей об'єктів на підставі емпіричної інформації. Дедуктивне навчання передбачає формалізацію знань експертів у вигляді експертних систем.

Графічні об'єкти важко описати, використовуючи дедуктивне навчання. Тому в системах технічного зору застосовується, в основному, індуктивне навчання.

Прецедент — це об'єкт, приналежність якого до заданого класу визначена заздалегідь. Прецедентом можуть бути об'єкти, які розпізнані людиною, штучні зображення, тощо.

Кожний образ об'єкту являє собою набір чисел, що описують його властивості. Ці числа називаються ознаками (feature). Упорядкований набір ознак об'єкта називається вектором ознак (feature vector). Вектор ознак — це точка в просторі ознак (feature space).

Класифікатор, або вирішальне правило (decision rule) — це функція, яка ставить у відповідність вектору ознак образу клас, до якого він належить.

Задачу розпізнавання образів можна розділити на наступні підзадачі.

1. Генерування ознак (feature generation) — вимірювання або обчислення числових ознак, що характеризують об'єкт.

2. Вибір ознак (feature selection) — визначення найбільш інформативних ознак для класифікації.

3. Побудова класифікатора (classifier construction) — конструювання вирішального правила, на підставі якого здійснюється класифікація.

4. Оцінка якості класифікації (classifier estimation) — обчислення показників правильності класифікації (точність, чутливість).

Отже, алгоритм розпізнавання зображення має найбільш відповідальні етапи — генерування ознак, що характеризують об'єкт, та вибір ознак. Ці етапи є найбільш трудомісткими, тому що вони повинні проріджувати інформацію у зображенні від мегабайтів до сотень байтів. При цьому вилучені ознаки повинні ефективно характеризувати об'єкт, щоби забезпечити надійне розпізнавання у несприятливих умовах освітленості, шумів [18].

В даний час набули великого поширення методи розпізнавання графічних образів на основі штучних нейронних мереж та методи, які, основані на детектуванні характерних точок. Ці методи далі розглядаються детальніше.

1.2 Розпізнавання образів у штучних нейронних мережах

1.2.1. Вступ

Штучні нейронні мережі визнані потужним інструментом для розпізнавання образів у різних областях застосування. Нейромережа нагадує поведінку мозку і складається з одного або декількох шарів нейронів, зв'язаних між собою. Нейронні мережі мають можливість навчатись по вхідним даним. Навчальний процес досягається зміною архітектури (зв'язків) мереж та вагами з'єднань відповідно до даної інформації. Топологія мережі та пов'язані з нею ваги можуть бути отримані як з навчальних даних, так і від знань області застосування, або від комбінації цих двох способів [19].

Функціональні мережі [18] є альтернативою нейронним мережам і можуть об'єднувати інформацію як з вхідних даних, так і з області застосування. Функціональні мережі вимагають знання області застосування для отримання функціональних рівнянь і робити припущення щодо форми функцій цих рівнянь. З'ясовано, що функціональні рівняння значно зменшують ступінь свободи множини параметрів нейронної мережі, що призводить до значних її спрощень.

Для розпізнавання графічних образів набули поширення згорткові нейронні мережі (convolutional neural net, CNN). За своєю суттю, вони належать до функціональних мереж. Вони мають шари первинної та вторинної обробки.

Перші шари виконують згортку зображення з певним набором ядер фільтрації, результатом чого є ряд зображень, які побудовані з певних локальних ознак, таких як краї, плями, кути. Другі шари виконані, власне, як нейронна мережа, що виконує остаточне розпізнавання образів.

Особливий тип згорткової нейронної мережі складають мережі для глибокого навчання (deep learning CNN). Така мережа складається з кількох шарів CNN, кожен наступний з яких виконує проріджування інформації та розпізнавання більш узагальнених образів [20].

1.2.2. Багатошаровий перцептрон

Багатошаровий перцептрон є першою моделлю нейронної мережі, що досконало досліджена. Кожен шар мережі складається з ряду вузлів обробки званих нейронами. Тому ця мережа називається багатошаровий перцептрон (multilayer perceptron, MLP). На перший шар подаються данні для розпізнавання, а з останнього шару зчитуються сигнали про те, що розпізнано. Процес автоматичного налаштування параметрів мережі називається навчанням.

Вузол i MLP на j -му входах має помножувач сигналу на вагу θ_{ij} . Причому трапляється кількість входів від одиниць до сотень. Під час навчання за узагальненим дельта-правилом (алгоритмом зворотного поширення) оновлюються ваги входів θ_{ij} вузлів враховуючи різниці між наявним та прогнозованим результатами (помилки розпізнавання), на основі якої розраховуються градієнти δ_i^l . Такий градієнт вказує, у якому напрямку і з якою швидкістю слід змінювати ваги θ_{ij} , щоб вони наприкінці досягли оптимального значення.

Кожен вузол обчислює свою активацію a_i як суму входів x_i , зважених вагами θ_{ij} , яка обробляється нелінійною функцією активації p_i . Ця функція має бути диференційованою, щоб параметри мережі можна було налаштувати за допомогою зворотного поширення помилок. Однією з часто застосованих функцій активації є сигмоїдальна функція $p_i = 1/(1 + \exp(-a_i))$.

Ваги θ_{ij} вузлів різних шарів налаштовуються за правилом ланцюга. При цьому градієнти δ_i^l , що підлаштовують ваги θ_{ij} у внутрішніх шарах MLP, обчислюються з урахуванням градієнтів δ_i^{l+k} , що поширюються з усіх наступних шарів. Процес навчання включає ряд ітерацій, і параметри θ_{ij} , δ_i^l постійно оновлюються, доки мережа не буде оптимізована (наприклад, коли ваги θ_{ij} перестають змінюватися).

Коли мережа є глибокою (кількість шарів більше трьох), процес навчання може страждати від проблем зникнення або вибуху градієнтів δ_i^l залежно від вибору функції активації. У результаті, ваги початкових шарів не можуть бути належним чином налаштовані [19].

1.2.3. Згорткова нейронна мережа

Згорткова нейронна мережа (convolutional neural net, CNN) є найпоширенішою категорією нейронних мереж для розпізнавання зображень і відео. Працюють CNN у спосіб, який дуже схожий на функціонування мережі MLP. Ключова відмінність полягає в тому, що кожен вузол на перших рівнях CNN є одно-, дво- або тривимірним фільтром, який виконує згортку вхідних даних цього рівня. При цьому фільтри виділяють ознаки у вхідних зображеннях або відеосигналах за певними шаблонами. Як результат, з кожним шаром значно зменшується кількість змінних, які можна вивчати.

Першою формою CNN була модель Neocognitron [21]. Автори цієї моделі взяли до уваги будову первинної зорової кори мозку, у якій нейрони організовані у формі шарів. Ці шари навчаються розпізнавати візуальні шаблони, спочатку аналізуючи локальні особливості, а потім комбінуючи їх для отримання представлень вищого рівня. Ця модель була удосконалена у моделі LeNet, в якій параметри моделі були отримані за допомогою зворотного поширення помилок [22].

Функціонування CNN подібне до роботи MLP. Але ключовою відмінністю є автоматичне навчання ієрархії представлень характерних ознак

зображення та інтеграція її етапів класифікації і виділення ознак в один конвеєр обробки.

Перший шар CNN виконує попередню обробку сигналу. В кадрі зображення, що обробляється, обчислюється середнє значення, яке віднімається від усіх відліків кадру. Далі ці відліки нормалізуються до одиничного значення з урахуванням їх стандартного відхилення. Для зменшення кореляції між пікселями виконується відбілювання за допомогою обчислення коваріаційної матриці, декомпозиції її на власні значення і декореляції зображення множенням на матрицю власних векторів. Завдяки цьому, прискорюється навчання CNN, але шум у зображенні також підсилюється. І нарешті, виконується локальна нормалізація контрасту зображення.

Згортковий рівень є найважливішим компонентом CNN. Він містить набір фільтрів (згорткових ядер), які фільтрують вхідний сигнал з одержанням карти ознак. Це такі ознаки, як пляма, кут лінії, перехрестя ліній тощо. При цьому, як правило, до зображення розміром 256×256 використовується ядро розміром від 3×3 до 9×9 . Розміри ядра співпадають з квадратною апертурою, через яку фільтр може змінювати зображення на кожному кроці згортки і яка називається *полем сприйняття* (receptive field) фільтра.

Після кількох етапів згортки поле сприйняття збільшується. Щоб створити глибокі моделі з відносно зменшеною кількістю параметрів (кількістю коефіцієнтів ядра фільтра) і з великим результуючим полем сприйняття, успішною стратегією є укладання багатьох шарів згортки з малими полями сприйняття [23]. Наприклад, два шари фільтрів 3×3 дають поле сприйняття 5×5 . Однак це обмежує просторовий контекст згорткових фільтрів, які масштабуються лише лінійно з кількістю шарів.

Метод розширеної згортки (dilated convolution) — це підхід, який розширює розмір поля сприйняття без збільшення кількості параметрів [24]. Операція об'єднання шарів фактично виконує децимацію вхідного кадру і є

корисною для отримання компактного представлення ознак, яке є інваріантним для помірних змін у масштабі і повороті об'єкта [20].

Згорткові шари в CNN часто супроводжуються нелінійною функцією активації, яка має невелику множину результуючих значень, наприклад, $(-1, 0, 1)$. Таку нелінійну функцію також можна розуміти як механізм перемикавання або вибору. Функції активації, які зазвичай використовуються в глибоких мережах, є диференційованими, щоб можна було задіяти зворотне розповсюдження помилок.

Шар *об'єднання регіонів інтересів* (RoI) є важливим компонентом згорткових нейронних мереж, який здебільшого використовується для виявлення у великих зображеннях об'єктів-кандидатів на розпізнавання [25,26]. У задачі виявлення об'єктів мета полягає в тому, щоб визначити місцезнаходження кожного об'єкта на зображенні за допомогою обмежувальної рамки та позначити його відповідною категорією об'єкта. Зазвичай, спочатку створюють великий набір пропозицій об'єктів-кандидатів, замкнених у рамки RoI. Для цих початкових пропозицій використовуються готові детектори, такі як детектор вибіркового пошуку (selective search) [27] або EdgeBox [28]. Далі CNN вибирає шукані об'єкти у множині RoI, розміри яких є значно меншими за розміри початкового кадру. Оскільки шари CNN, як правило, працюють лише з кадрами фіксованих розмірів, у шарі об'єднання RoI кожна область інтересу перетворюється у кадр фіксованого розміру.

У зображенні один і той самий об'єкт може мати різний масштаб. Якщо CNN налаштована на пошук об'єкта з конкретним масштабом, то варто його шукати у стосі зображень, які мають різний масштаб. Ця ідея використана у методі шару об'єднання просторових пірамід (spatial pyramid pooling, SPP) [29]. При цьому кожне зображення дає вектори ознак, які складаються у один дескриптор ознак з певними координатами у зображенні. Такі дескриптори дають змогу досягти стійкості до варіацій позиції об'єкта, масштабу та форми. Оскільки вихід шару SPP не залежить від довжини та ширини кадру, це

дозволяє CNN обробляти вхідні зображення будь-якого розміру. Ця сама ідея використана у шарі локально агрегованих дескрипторів (Vector of Locally Aggregated Descriptors, VLAD) [30]. При цьому для розпізнавання знайдені сусідні дескриптори збираються у кластери.

Щоб розпізнати повернутий об'єкт, зображення слід скоректувати афінним перетворенням. Для цього у CNN використовують шар просторового перетворювача, який навчається [31]. При цьому цей шар передбачає параметри афінних перетворень і застосовує ці перетворення до цікавих частин вхідних даних. Насамкінець, генеруються вектори регресії, які зберігають параметри шуканого об'єкта.

1.2.4. Функція втрат

У останньому шарі CNN при навчанні обчислюється *функція втрат* (loss function) для оцінки якості передбачень, для яких відомі фактичні об'єкти. Функція втрат кількісно визначає різницю між оціненим результатом моделі (прогноз) і правильним результатом [32].

Втрата перехресної ентропії (логарифмічна втрата, втрата м'якого максимуму) визначається таким чином:

$$L(\mathbf{p}, \mathbf{y}) = - \sum_n y_n \log(p_n), \quad (1.1)$$

де y_n позначає бажаний результат, а p_n — ймовірність для кожної вихідної категорії.

Петльова (feedback loss) втрата методу опорних векторів (SVM) максимізує різницю між істинним і негативним класом зразків:

$$L(\mathbf{p}, \mathbf{y}) = \sum_n \max(0, m - (2 y_n - 1) p_n), \quad (1.2)$$

де m — запас, p_n, y_n — прогнозовані та бажані результати відповідно.

Евклідова функція втрат (квадратична функція втрат, середньоквадратична помилка, помилка e^2) визначається як:

$$L(\mathbf{p}, \mathbf{y}) = \frac{1}{2N} \sum_n (p_n - y_n)^2, \quad n \in [1, N]. \quad (1.3)$$

Контрастна функція втрат використовується для пар подібних або несхожих вхідних об'єктів [33]:

$$L(\mathbf{p}, \mathbf{y}) = \frac{1}{2N} \sum_n y d^2 + (1 - y) \max(0, m - d)^2, \quad n \in [1, N]. \quad (1.4)$$

де m – запас, $y \in [0, 1]$ показує, чи є вхідні пари різними чи подібними, d — міра відстані між дескрипторами об'єктів.

Функція втрат очікування:

$$L(\mathbf{p}, \mathbf{y}) = \sum_n \left| y_n - \frac{\exp(p_n)}{\sum_k \exp(p_k)} \right|, \quad (1.5)$$

мінімізує очікувану ймовірність неправильної класифікації і забезпечує більшу надійність щодо викидів. Однак ця функція втрат рідко використовується в глибоких нейронних мережах, оскільки вона призводить до проблем з оптимізацією під час процесу навчання.

Задачі, для вирішення яких зазвичай використовуються CNN і пов'язані з ними функції втрат, розділяють на наступні категорії.

1. Бінарна класифікація (середньоквадратичні функції втрат).
2. Перевірка ідентичності (контрастна функція втрат).
3. Багатокласова класифікація (функція втрат очікування).
4. Регресія (евклідова функція втрат).

1.2.5. Навчання нейронних мереж

Перед навчанням виконується ініціалізація ваг $\theta_{i,j}$. Невідповідна ініціалізація може призвести до проблеми зникнення або вибуху градієнтів δ_i^l під час зворотного поширення помилки. Загальним підходом до ініціалізації ваги в CNN є метод випадкової ініціалізації з використанням випадкових матриць, елементи яких вибираються з розподілу Гауса з нульовим середнім і малим

стандартним відхиленням. У згорткових шарах матриці ваг часто заповнюють з рівномірним розподілом. Якщо випадкові коефіцієнти ортогоналізувати, то глибока нейронна мережа навчається краще [34]. Для уникнення проблем з градієнтом у глибокій нейронній мережі, виконують пошарове попереднє навчання без нагляду. Це, наприклад, встановлення коефіцієнтів згорткових фільтрів у перших шарах CNN [35].

Часто як початкові значення ваг беруть ваги готової CNN, яка налаштована на множину споріднених об'єктів і навчена на великому масиві кадрів. Цю CNN навчають на новій множині кадрів. Такий процес називають адаптацією домену [36]. Аналогічно виконується трансферне навчання [37] чи навчання з перенесенням [38].

Процес навчання CNN налаштовує параметри мережі θ_{ij} . таким чином, щоб множина вхідних образів правильно зіставлялась з вихідною множиною результатів. На кожному кроці навчання поточна оцінка розпізнавання узгоджується з бажаним результатом за допомогою функції втрат. При цьому виконується багаторазове оновлення параметрів CNN таким чином, щоб функція втрат поступово зменшувалася до мінімального значення. Така оптимізація є важким завданням, яке посилюється тим фактом, що ці моделі складаються з великої кількості настроюваних параметрів. Отже, ітеративно шукаються локально оптимальні рішення на кожному кроці. Часто це методи на основі градієнтного спуску. Кожна ітерація, яка оновлює параметри за допомогою повного навчального набору, називається **епохою навчання**.

Оптимізація з градієнтним спуском передбачає обчислення похідної у кожному шарі, що настроюється. При цьому виконується аналітичне, символічне диференціювання (при навчанні в пакеті Matlab), числове диференціювання як обчислення скінченної різниці.

Часто трапляється, що CNN добре розпізнає навчені образи і погано нові тестові образи. Для усунення цього явища виконують **регуляризацію** CNN. Регуляризація виконується за різними підходами. Наприклад, кількість

навчальних образів, які варто надійно розпізнавати, збільшують дублюванням образу з різними трансформаціями [39], або використовують синтетичні образи для навчання [40]. В інший спосіб застосовують випадковий запуск і вилучення (dropout) вузлів на навчання [41]. Так само випадковим чином підключаються чи дезактивуються ваги θ_{ij} [42].

Якщо об'єкти можна розділити на множини за певними ознаками (ансамблі), то використовують ансамблеве усереднення, при якому CNN налаштовується окремо для кожного ансамбля, а потім результати навчання об'єднуються. Часто трапляється, що при надмірному навчанні ефективність CNN деградує і варто ухопити момент зупинки навчання.

1.2.6. Реалізація нейронної мережі у ПЛІС

Добре натренована CNN може одержати поширення у вбудованих системах, таких, як система допомоги водію (ADAS), інтелектуальна відеокамера, система пошуку дефектів у виробництві та ін. При цьому немає необхідності у засобах навчання мережі та обчисленнях з високою точністю. Таким чином, впровадження CNN у вбудованій системі проходить два етапи. На першому етапі CNN тренується з використанням універсальних обчислювальних засобів високої продуктивності та точності, а також колекції образів для тренування, що відповідають призначенню застосунку.

На другому етапі, CNN реалізується на вибраній елементній базі: мікроконтролер, процесор штучного інтелекту чи ПЛІС. При цьому в CNN відкидаються незадіяні зв'язки у повністю з'єднаних шарах, коефіцієнти з плаваючою комою округлюються до цілих чисел, з вузлів мережі викидаються множення на нульовий коефіцієнт та елементи, що призначені для настроювання параметрів.

ПЛІС є гарним середовищем для реалізації CNN. На відміну від інших засобів реалізації навченої CNN, ПЛІС забезпечує обчислення вузлів з високим ступенем паралелізму. В ПЛІС є змога вибрати розрядність даних і параметрів, яка є оптимальною за апаратними витратами, енергоспоживанням та

швидкодією. Як результат, CNN, реалізована в ПЛІС, має у кілька разів менше енергоспоживання, ніж при реалізації в графічному акселераторі [43,44]. Завдяки цій властивості, ПЛІС також широко використовуються в датацентрах для реалізації CNN [45].

1.2.7. Огляд поширених моделей нейронних мереж

Архітектура AlexNet [39] була першою великомасштабною моделлю CNN. Вона відрізняється від попередників збільшеною глибиною мережі, що призводить до значно більшої кількості настроюваних параметрів, а також використанням способів регуляризації, такими як відключення активації і збільшення числа даних. CNN має п'ять початкових шарів згортки і три повністю зв'язані шари, останній з яких має тисячу виходів. Між цими шарами вставлені шари виділення відсівання за максимумом. Кількість параметрів мережі перевищує шістдесят мільйонів. Тому навчання мережі триває кілька діб.

Архітектура VGGnet [46] є однією з найпопулярніших моделей CNN. В ній застосовані згорткові ядра розміром 3x3, а також проміжні і кінцеві шари максимального об'єднання. Малі ядра призводять до відносно зменшеної кількості параметрів, а отже, до ефективного навчання та тестування а також до більшої кількості шарів, що призводить до кращої ефективності розпізнавання. У найефективнішій моделі VGGnet-16 задіяно 138 млн. параметрів.

Архітектура GoogleNet [47] є першою поширеною моделлю, яка використовує архітектуру з кількома гілками мережі. GoogleNet складається із 22 вагових шарів, хоча загальна кількість рівнів у мережі більше ста. Основним блоком мережі є Початковий модуль (Inception Module). Він представляє собою шар мережі, що складається з чотирьох гілок фільтрів. В кожній гілці зібрано від 16 до 208 фільтрів ознак, що виконують згортку 1x1, 3x3 чи 5x5. Гілка налаштована на специфічний вид ознаки. Гілки працюють паралельно і видають одночасно ознаки на входи шару, який об'єднує ці ознаки. Щоб уникнути проблеми зникаючих градієнтів, виходи з проміжних шарів подаються на класифікатори для навчання і цим глибина зворотного зв'язку зменшується. Для

регуляризації використовується відсівання ненадійних ознак перед останніми повністю пов'язаними шарами. Ця модель має значно зменшену кількість параметрів (~6 мільйонів), доволі великий вхідний кадр (50 тис. пікселів) і дає помилок розпізнавання менше 7%.

Як і в мережі GoogleNet, в архітектурі залишків (Residual Network, ResNet) та її удосконаленні ResNeXt від фірми Microsoft задіяно специфічні блоки виділення ознак – блоки залишків (Residual Block) [48]. Вона має помилкових розпізнавань менше 4%. Блок залишків має кілька шарів згортки, шар нормалізації та шар активації. Паралельно ним проходить гілка, що перемикається щоб пропускати вхідні дані без обробки. Перемикання гілки дає змогу навчати дуже глибокі мережі (200 вагових шарів).

1.2.8. Висновки щодо згортувальних нейронних мереж

На сьогоднішній день CNN стала найпоширенішою технологією для розпізнавання образів. Це досягнуто завдяки універсальному підходу до класифікації образів довільної природи, який притаманний CNN і тому, що рівень обчислювальної потужності сучасних комп'ютерів дає змогу виконувати їх навчання та розпізнавання за їх допомогою за прийнятні проміжки часу.

Кожен шар згортки CNN дає змогу виділяти у зображенні піксель або групу пікселів, що належать до частини зображення з характерною ознакою. Таких характерних ознак може бути від одиниць до сотень. Причому, та чи інша ознака є результатом навчання і може мати незрозумілий смисл. Кожен наступний шар мережі узагальнює таку ознаку. Для зменшення складності мережі та формування узагальнених ознак чи для аналізу об'єктів з різним масштабом вихідні дані шарів підлягають просторовій децимації.

Згортка – це фактично двовимірна фільтрація, яка виконується з ядром фільтра розміром $n \times n$, коефіцієнти якого є настроюваними параметрами мережі. Таких параметрів може бути до десятків мільйонів. Для мінімізації кількості параметрів при збереженні поля сприйняття фільтрів (куди попадає фрагмент зображення з шуканою ознакою) кількість шарів збільшують, а ядро

фільтра зменшують до 3×3 . Також для зменшення числа коефіцієнтів та підвищення якості розпізнавання вузли згорткових шарів мають специфічну структуру.

Перший шар CNN, як правило, виконує попередню обробку сигналу для підвищення локального контрасту і вилучення шумів. Буває CNN в якій для розпізнавання об'єктів у різному масштабі за методом просторових пірамід вхідний кадр перетворюється у стос кадрів, кожен з яких одержано децимацією попереднього за допомогою фільтрації у перших шарах. Останні шари є повнозв'язаними і класифікують об'єкти за множиною визначених параметрів на категорії, число яких досягає кількох сотень. Причому рішення приймається за однією з функцій втрат, що залежить від категорії задач, що вирішує CNN.

Для можливості навчання CNN з багатьма шарами вузли повинні обробляти сумарний сигнал від зважених входів нелінійною функцією активації, яка повинна бути диференційованою. При навчанні CNN коефіцієнти вузлів інкрементуються на величину, яка пропорційна похідній похибки передбачення (градієнту), що поширюється від останніх шарів до перших. Для зменшення явищ зникання та вибуху градієнту шари навчають поступово, починаючи з перших та аналізуючи виходи внутрішніх шарів.

Недоліки CNN наступні

- висока складність навчання, яке потребує високоточних тривалих обчислень з використанням високопродуктивних обчислювальних засобів і висока складність розпізнавання, яке вимагає десятків мільйонів множень на одну ідентифікацію об'єкта;

- негарантованість одержання ефективно навченої CNN; для навчання необхідна регуляризація, спосіб якої підбирається експериментально досвідченими розробниками;

— складність мережі більш ніж лінійно пропорційна площі кадру, що аналізується; важко розпізнається об'єкт, якщо він має інший масштаб чи повернутий відносно об'єкта, що приймав участь у навчанні;

— неможливо встановити причини, за якими CNN прийняла те чи інше рішення, бо інформація з проміжних шарів важко осмислюється.

Навчену CNN ефективно реалізувати в ПЛІС. При цьому можна одержати мінімізовані апаратні витрати і високу швидкодію за рахунок вибору оптимальної розрядності даних і параметрів, а також відмови від схем, що приймають участь у навчанні. Апаратна мережа CNN за потреби може бути оперативно переналаштована в ПЛІС.

Можна висунути гіпотезу про те, що параметри перших шарів CNN варто визначати не за допомогою навчання, а аналітично чи шляхом розрахунків, або експериментальним підбором, одержуючи детектування наперед відомих ознак. Тоді складність мережі та її навчання мають зменшитись, а ефективність розпізнавання збільшиться за рахунок того, що ознаки, які відфільтровує шар, є очікуваними і більш виразними. Розроблені таким чином шари можуть бути ще ефективніше реалізовані у ПЛІС.

1.3. Методи визначення характерних точок

Характерною точкою (feature point) вважається невелика частина зображення, яка за певних причин виділяється з оточуючого околу, і яку можна знайти в ряді схожих зображень. Дескриптор — це певний математичний об'єкт, як правило, вектор ознак, який описує окрему характерну точку та який можна використовувати для визначення того, чи є дві характерні точки тотожними в певному контексті.

Дескриптори характерних точок відіграють важливу роль у задачах комп'ютерного зору та розпізнавання образів, включаючи склеювання зображень [49], розпізнавання або кластеризація об'єктів [50], побудова зображень [51], стеження за об'єктами [52], розпізнавання облич [53], виявлення змін у

зображеннях [54], реєстрація зображень [55]. Потреба у швидкому виявленні характерних точок проявляється у застосунках комп'ютерного зору для отримання зображень на основі вмісту (content-based image retrieval, CBIR) у роботів та автономного водіння.

Сучасні підходи до виявлення характерних точок об'єкта дозволяють включити у розгляд масштаб: об'єкт можна розпізнати незалежно від його видимого розміру. Об'єкт також може характеризуватись набором характерних точок. Це дозволяє виконувати розпізнавання зміненого об'єкту (об'єкт, розглянутий під іншим кутом, у іншому масштабі). Використання взаємного розташування точок також дозволяє розпізнавати складні об'єкти в умовах зашумленості та неоднорідної освітленості.

Визначення характерних точок виконується у два етапи: детектування точки за допомогою відповідного алгоритму-детектора і кодування інформації про точку у вигляді дескриптора.

Якщо зображення переміщується, обертається, змінює масштаб по двох осях, а результат детектування залишається незмінним, це означає стабільність розпізнавання, а знайдені точки – характерними точками.

Дескриптори характерних точок повинні мати наступні властивості: вони описують ключові характеристики зображення з відмінністю, повторюваністю, компактністю, точністю та ефективним поданням, яке є незмінними та надійним щодо зміни масштабу, обертання, відображення, оклюзії та освітлення [56, 57]. Далі розглядаються відомі методи розпізнавання характерних точок та побудови їх дескрипторів.

1.3.1. Методи, основані на аналізі градієнта яскравості

У цих методах детектори характерних особливостей виділяють краї фігур у зображеннях переважно на основі аналізу градієнту яскравості. При цьому алгоритм детектування використовує ту чи іншу математичну модель яскравості.

Детектори характерних особливостей низького рівня

Характерні особливості низького рівня — це такі особливості зображення, які можна визначити в ньому без будь-якої інформації про форму (інформації про просторові зв'язки пікселів) [58]. Це може бути порогова обробка, функції виявлення краю, кута.

Для **детектування краю** використовуються детектори Собеля (перша похідна), Канні (похідна з фільтром Гауса), з обмеженням (відкидання слабого результату), з гістерезисом. Детектор Лапласа обчислює похідну другого порядку і там де вона пересікає нульовий рівень — там край форми (див. рис. 1.1). Оскільки функція похідної чутлива до шуму, часто для фільтрації використовують медіанний фільтр, фільтр нижніх частот.

Недоліки детекторів краю: неповні контури замкнених фігур, необхідність селективного порогового значення, чутливість до шуму, вибір порогу слід виконувати вручну окремо для регіонів з різним локальним освітленням.

Детектор з фазовою конгруентністю має переваги: виявлення характеристик, які є незмінні при різних локальній освітленості та контрасті. Детектор функціонує на тому принципі, що в точці краю синусоїдальні складові сигналу, одержані перетворенням Фур'є, мають одну і ту саму фазу. Але він також чутливий до шуму і має низьку точність локалізації [59, 60].

У детекторах кута та кривизни **кривизна** визначається як швидкість зміни напрямку краю чи плоскої кривої, які описують фігуру. Точки, де напрямок краю швидко змінюється, є кутами, тоді як точки, де цей напрямок мало змінюється, відповідають прямим лініям.

Очевидним підходом є розрахунок кривизни шляхом обчислення різниці між кутовим напрямком послідовних пікселів на кривій. Інший підхід полягає у отриманні міри кривизни на основі змін інтенсивності зображення. Також міру кривизни можна отримати за допомогою кореляції [61]. В усіх випадках пошук замкненої кривої та її аналітичний опис є складною задачею.

Те, що міру кривизни можна отримати, розглядаючи зміни яскравості вздовж альтернативних напрямків на зображенні, покладено в основу

оператора виявлення кутів Моравека. Цей оператор обчислює середню зміну інтенсивності зображення $E_{u,v}(x, y)$ у напрямку (u, v) , коли вікно аналізу зміщується в декількох напрямках. Тоді, якщо піксель знаходиться на прямій лінії, значення $E_{u,v}(x, y)$ є малим для зсуву вздовж лінії та великим для зсуву, перпендикулярного до лінії, якщо край визначає кут, то по всіх напрямках вихід детектора дає велике значення. Недоліком детектора є те, що він враховує лише невеликий набір можливих напрямків.

Ця проблема вирішується в детекторі кутів Гарріса [62]. В ньому обчислюється автокореляція градієнта яскравості по горизонталі $A(x,y)$, вертикалі $B(x,y)$ та взаємна кореляція по вертикалі і горизонталі $C(x,y)$. Тоді нормована міра належності до лінії у напрямку (u,v) з урахуванням власного вектора (α, β) обчислюється як

$$F_{u,v}(x, y) = \alpha(x,y)^2 u^2 + \beta(x,y)^2 v^2. \quad (1.6)$$

Причому, якщо точка визначає границю прямої лінії, то одне з значень α, β є великим, а друге — малим. Якщо точка визначає границю з високою кривизною, обидва значення великі. Тоді міра кривизни визначається як

$$K(x,y) = \alpha \beta - k(\alpha + \beta)^2, \quad (1.7)$$

де k — параметр керування чутливістю детектора.

Метод перетворення ознак яке інваріантне до масштабу — SIFT Масштабний простір (scale space) — це простір, в якому характерні точки представлені у різних масштабах, але зі збереженням їх взаємного розташування. Масштабний простір найчастіше будують із зображення, яке ітераційно згладжується Гаусовим фільтром, децимується, а потім формується піраміда зображень в різних масштабах [56]. На цьому ґрунтується **метод перетворення ознак яке інваріантне до масштабу** (scale invariant feature transform, SIFT) [63].

Метод SIFT включає наступні чотири етапи.

1. Виявлення екстремуму. На цьому етапі будується піраміда різномасштабних зображень за допомогою Гаусової фільтрації та децимації, знаходяться диференційні зображення як різниці сусідніх шарів піраміди.

Для виявлення характерних точок кадр зображення фільтрують фільтром з Гаусовим ядром $G(x, y, \sigma)$:

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y), \quad (1.8)$$

де $*$ — оператор згортки, σ — масштаб зображення, $I(x, y)$ — яскравість світла у пікселі з координатами (x, y) . На рис. 1.1, а, б, в показані графіки яскравості рядка зображення вертикальної лінії та її фільтрації фільтром з $\sigma = 1$ та 1,41. У результаті такої фільтрації зображення стає розмитим, причому ступінь розмиття пропорційний масштабу σ . Одержують стос з чотирьох зображень з $\sigma = 1, k, k^2$ і k^3 , де $k = \sqrt{2}$ — множник масштабу. Крім того, зображення проріджують, зменшуючи його удвічі та знову фільтрують, після чого знову проріджують і т.д. При цьому одержується, так звана, піраміда зображень, яка складається з октав — стосів зображень, що відрізняються за масштабом удвічі один від одного.

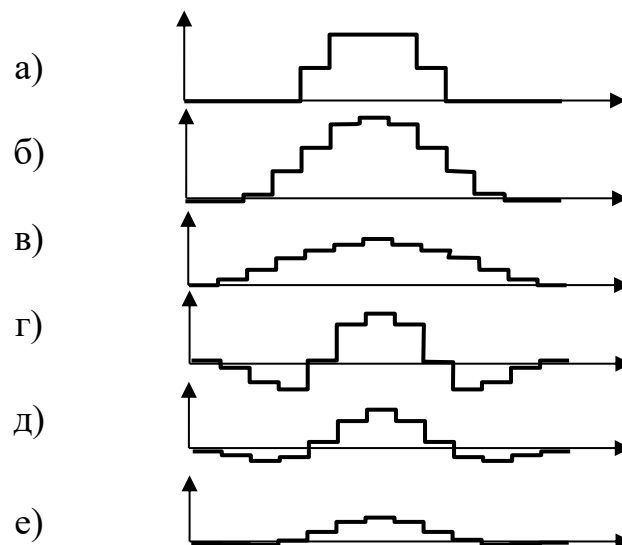


Рисунок 1.1 — Рядок яскравості кадру (а), його обробка фільтрами Гауса (б), (в) та результат віднімання результатів фільтрації (г), перша (д) та друга (е) похідні яскравості

Сусідні шари — кадри піраміди зображень віднімають один від одного, одержуючи в результаті по три зображення у октаві, які є профільтрованими диференційним Гаусовим фільтром:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) = \\ &= L(x, y, k\sigma) - L(x, y, \sigma). \end{aligned} \quad (1.8)$$

Цей фільтр є апроксимацією Лапласового фільтра, який обчислює двовимірну другу похідну зображення, тобто, побудова піраміди зображень та віднімання сусідніх з них значно спрощує обчислення похідних від $D(x, y, \sigma)$. На рис. 1.1, г показані результати такого віднімання при формуванні похідних зображень у одній октаві. Можна бачити, що різниця зображень на рис. 1.1, г є вдалою апроксимацією похідних від яскравості на рис. 1.1, д, е. Доведено, що функція $D(x, y, \sigma)$ дає найбільш стабільні характерні точки у порівнянні з іншими функціями, такими як градієнт, функція Гессіана або Гарріса при відносно невеликій складності обчислень [64].

Таке зображення є наближенням до зображення після фільтрації фільтром Лапласа (рис. 1.1, д). Тобто, у ньому максимум чи мінімум означає різкий градієнт освітленості. Далі в цих диференційних зображеннях шукаються локальні максимуми/мінімуми як центри характерних точок. При цьому в піраміді зображень, яка ілюструється рис. 1.2, шукають локальні екстремуми.

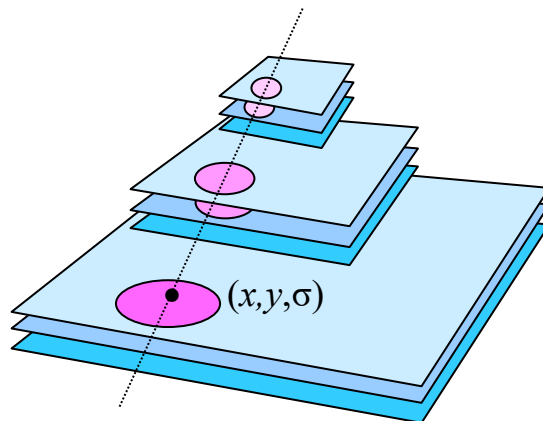


Рисунок 1.2 — Піраміда оброблених зображень

Такий екстремум знаходиться при порівнянні даного пікселя у точці (x, y, σ) з пікселями у $9+9+8$ сусідніх точках $(x-1, y-1, \sigma/k), \dots, (x-1, y-1, \sigma), \dots, (x+1, y+1, \sigma k)$. Точку максимуму (x, y, σ) вибирають як максимальну серед усіх октав з координатами (x, y) . При цьому піки, які є занадто малими, відкидаються, тому що вони сформовані сигналом з шумами.

2. Локалізація характерних точок. Відфільтровуються нестабільні характерні точки, тобто точки низької контрастності та ті, які не є кутами, кінцями ліній, перехрестями чи плямами. Наприклад, для розпізнавання лінії не потрібно визначати усі точки, які їй належать. Для цього формують матрицю Гессе H у точці, що розглядається

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}, \quad (1.9)$$

де D_{xx} , D_{yy} , D_{xy} — похідні у цій точці. Далі шукають власні значення α , β цієї матриці, які характеризують кривизну функції $D(x, y)$, виходячи з (1.1):

$$\text{Tr}(H) = D_{xx} + D_{yy} = \alpha + \beta,$$

$$\text{Det}(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta. \quad (1.10)$$

Для спрощення аналізу оцінюють не α , β , а відношення $(\alpha + \beta)^2/(\alpha\beta)$, яке має бути більше за певний поріг (наприклад, десять), інакше точка відкидається.

3. Пошук орієнтації. Будується кругова гістограма градієнтів яскравостей навколо знайденої точки. Для такої точки вибирається шар зображення L , у якого масштаб σ є найближчим до масштабу точки (x, y) . Тоді у найпростішому алгоритмі для цієї точки та точок у її околі у шарі L , тобто, до точки $L(x, y)$, розраховується амплітуда $m(x, y)$ та напрямок (кут) $\theta(x, y)$ градієнту, використовуючи різниці яскравостей сусідніх пікселів:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}; \quad (1.11)$$

$$\theta(x, y) = \arctg (L(x, y+1) - L(x, y-1)) / (L(x+1, y) - L(x-1, y)).$$

У оригінальному алгоритмі SIFT для точки $\mathbf{x} = (x, y, \sigma)$ розраховується матриця градієнтів та кутів у її околі. Наприклад, у околі 8×8 знаходяться 64 градієнти та кути, як на рис. 1.3. Далі розраховується гістограма розподілення напрямків по кутах для матриці градієнтів, яка має 36 стовпчиків, що відповідають секторам по 10° , як на рис. 1.4. Визначається нормалізуючий кут (локальний напрямок характерної точки), на який повертається локальне зображення навколо центру. Можливо знаходження 2 або 3 альтернативних кутів, тоді для цієї точки обчислюються 2 або 3 дескриптори.

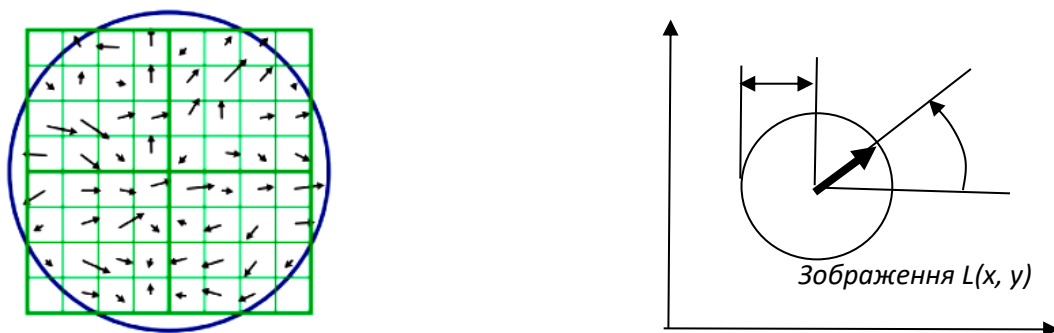


Рисунок 1.3 — Матриця градієнтів та кутів у околі характерної точки



Рисунок 1.4 — Гістограма розподілення напрямків по кутах

При цьому до стовпчика гістограми, який відповідає певній множині кутів (сектору) додається величина градієнту, яка помножена на зважуючий коефіцієнт. Цей коефіцієнт тим менший, чим далі піксель від характерної точки.

На рис.1.5 показано результат знаходження характерних точок у тестовому зображенні за методом SIFT.



Рисунок 1.5 — Приклад знайдених характерних точок за методом SIFT

4. Обчислення дескриптора характерної точки. Спочатку вікно 16×16 навколо кожної ключової точки розбивається на шістнадцять апертур 4×4 з подальшим обчисленням величин градієнта та орієнтацій для кожної апертури (рис. 1.6, а). Потім створюється набір гістограм орієнтації для кожної апертури з 8 бінами в кожному, в результаті чого, виходить вектор дескриптора із 128 елементів. Цей вектор нормалізується до одиниці довжини для досягнення інваріантності освітленості. Такий дескриптор є інваріантний щодо масштабу, освітлення та обертання об'єкта.

Метод SIFT вважається найбільш надійним серед багатьох методів визначення характерних точок, але й дуже складним. На його основі розроблена велика множина подібних методів, які є менш складними і які розглянуті нижче.

Методи, що є похідними від метода SIFT

Для спрощення обчислень у методі прискорених надійних функцій (speeded up robust features, SURF) [65], на відміну від SIFT, використовуються прості (коефіцієнти лише $0, \pm 1, -2$), але ефективні фільтри Гауса, які

обчислюють виявлення краю в різних масштабах. Але зображення з крупним масштабом не децимується. Це забезпечує більш швидке виконання і формування пакету зображень з різним масштабом.

Центральний піксель характерної точки виявляється як максимум детермінанту матриці Гессе (1.9), елементами якої є відгуки різних фільтрів краю з вейвлетом Хаара. Максимум, як і в методі SIFT, шукається у кубі розміром $3 \times 3 \times 3$. Потім ці максимуми інтерполюються в масштабному просторі та в просторі зображень і описуються разом з орієнтаціями.

Далі будується квадратне вікно з центром навколо ключової точки та відповідно повернуте. Воно ділиться на 16 субвікон, в яких обчислюються фільтри з вейвлетом Хаара. Кожний субрегіон дає чотири значення, в результаті чого виходить 64-мірний вектор дескриптора, який є інваріантним до обертання, масштабу, контрасту та частково інваріантним до інших перетворень.

Завдяки простоті обчислення фільтрів, забезпечується висока швидкість пошуку характерних точок. Але метод не використовує нормалізацію, таку як у SIFT, через що часто створюються некоректні дескриптори і кількість виявлених точок є меншою, ніж у методі SIFT.

Дескриптор гістограми розташування та орієнтації градієнта (gradient location and orientation histogram GLOH) є удосконаленням дескриптора SIFT завдяки зміні розташування сітки навколо характерної точки та використання аналізу основних компонентів (principal components analysis, PCA) для зменшення розміру дескриптора та підвищення його стійкості (рис. 1.6, б) [66].

У дескрипторі виразних ефективних надійних характерних ознак (distinctive efficient robust features, DERF) використовується модель з просторовим розподілом пікселів та їх обробкою, яка моделює властивості ганглієвих клітин та сітківки приматів [57].

Алгоритм побудови дескриптора SIFT, який є інваріантний до афінних перетворень (ASIFT) є удосконаленням оригінального алгоритму SIFT за

рахунок його ускладнення [67]. Для спрощення алгоритму запропоновано зкомбінувати алгоритми SURF та ASIFT [68].

Для дескриптора DAISY обчислюються вісім карт орієнтації Go , по одній для кожного квантованого напрямку, де $Go(u, v)$ дорівнює величині позитивного градієнта зображення в пікселі (u, v) для напрямку o . Потім формуються карти орієнтації з різними масштабами за допомогою фільтрації фільтрами Гауса різного розміру як і у SIFT (рис. 1.6, в). Для кожного пікселя у кожному масштабі формуються 25 векторів напрямку. Дескриптор DAISY для розташування (u, v) визначається як об'єднання нормованих векторів усіх масштабів і містить 200 елементів. Дескриптор DAISY виражає характерну точку навіть на 5% краще, ніж дескриптор SIFT [69].

В роботі [70] запропоновано всерну функцію Fan-SIFT, яка є інваріантною до масштабу, перетворень, кута, яка базується на детектуванні кута характерної точки за допомогою всерної функції Лапласіана Гауса (FLOG). Тому покриття дескриптором характерної точки має форму всера з визначеним кутом розкриття. Дескриптор доповнює інші дескриптори для підвищення ефективності.

В роботі [71] запропоновано локальний дескриптор характерних точок, який отримав назву адаптивного біннінгового SIFT-перетворення (AB-SIFT), який використовує стратегію адаптивного біннінгу для опису вмісту зображення навколо локального об'єкта та застосовує адаптивну стратегію квантування гістограми як для розташування, так і орієнтації градієнта (рис.1.5,г). Це значно покращує розпізнавання та надійність дескриптора.

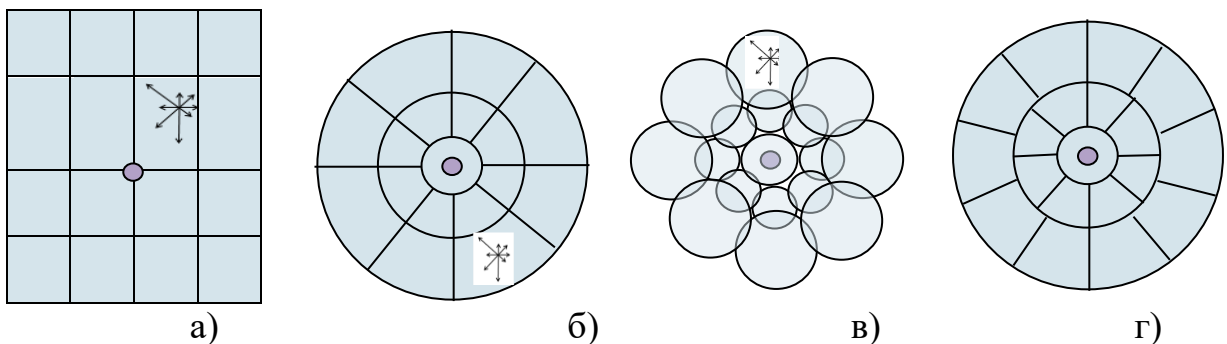


Рисунок 1.6 — Вікна для визначення гістограм дескриптора у методах SIFT (а), GLOH (б), DAISY (в), AB-SIFT (г)

У методі LDAHash [72] запропоновано підхід до створення двійкового вектора з дескриптора SIFT через вирівнювання його координат відповідно до коваріаційних властивостей зображення і навчального пошуку порогів. Це дало змогу вдесятеро зменшити розміри дескриптора SIFT та вчетверо — дескриптора DAISY.

У [73] запропоновано простий метод двійкового квантування дескриптора BIG-ОН на основі гістограми орієнтації градієнта шляхом обчислення бітового вектора, що представляє відносні величини локальних градієнтів, які пов'язані з сусідніми кавдратами орієнтації. При кодуванні дескриптора SIFT методом BIG-ОН одержують лише 16 байтовий вектор [73].

Інші методи засновані на аналізі градієнта яскравості

Алгоритм BOLD спочатку знаходить сегменти ліній. Потім генерує геометричні примітиви над результуючими парами сусідніх сегментів. Таким примітивом є відносна орієнтація між парами сегментів і він забезпечує незмінність щодо обертання, відображення, масштабу та шуму при високій інформативності. Орієнтація задається як пара кутів розташування двох сегментів. Дескриптор будується як множина пар кутів для десяти найближчих сусідніх сегментів лінії [74].

У [75] запропоновано дескриптор гістограм орієнтованого градієнта (Histogram of Oriented Gradient, HOG), який враховує вагу кожного біна гістограми орієнтації градієнта відповідно до значущості інформації про градієнт. При цьому об'єкт знаходиться завдяки методу вилучення контурної різниці переднього плану (Conour-Difference foreground extraction, CDFE). У

HOG-дескрипторі розмір комірки 10×10 , розмір блоку детектора 20×20 та розмір вікна 40×40 . Для масштабної інваріантності використовуються вікна з 11 різними масштабами.

1.3.2 Методи на основі аналізу яскравості

Методи, що ґрунтуються на аналізі яскравості зображення, застосовують порівняння інтенсивності пікселів, відібраних у різних місцях. Ці методи часто мають уточнюючу назву **двійкові дескриптори**, оскільки двійковий вектор є результатом послідовності порівнянь результату обчислення оператора визначення характерної точки з певним порогом.

Підхід визначення характерних точок з прискореного тесту сегментів (Features from Accelerated Segment Test, FAST) [76] базується на простих тестах 16 точок навколо пікселя з координатами (x, y) . І якщо яскравість пікселів у визначеному шаблоні з точок перевищує яскравість $I(x, y)$ самого пікселя, то піксел позначається як характерна точка. Завдяки тому, що в алгоритмі є лише порівняння, метод є доволі швидким. Але залежність від порогу та не прийняття до уваги орієнтації знижує його надійність. Для його покращення використовується машинне навчання.

У методі швидкого визначення характерних точок за властивостями сітківки (Fast Retina Keypoint, FREAK) [77] була використана аналогія з роботою сітківки в зоровій системі людини. Зображення згладжується за допомогою фільтра Гауса. Для точки, що розглядається, створюється вікно з ближніми та дальніми полями. Двійковий дескриптор формується як множина порогів різниць яскравостей пар полів. Серед великої множини варіантів порівняння полів автори вибрали 512 найефективніших, які використовуються для побудови дескриптора. Ці пари дають високоструктурований шаблон, що імітує стрибаючий пошук людських очей при розгляданні об'єкта. Орієнтація характерної точки обчислюється за допомогою локальних градієнтів серед вибраних пар.

Аналогічний підхід, який заснований на дескрипторі множини бінарних незалежних надійних характеристик (Binary Robust Independent Elementary Features, BRIEF), використовує детектор порівняння яскравостей і має високу швидкодію [78].

Дескриптор ORB (Oriented FAST and rotated BRIEF) виявляє характерні точки на основі міри Гарріса як у алгоритмі FAST. Потім створюється піраміда зображень у різних масштабах. Навколо характерної точки аналізується вікно 9×9 в якому центр і орієнтація обчислюються з використанням моментів першого порядку, які є мірами форми. З їх використанням обчислюються центр і орієнтація. Обчислюється дескриптор на основі вимірної орієнтації вікна. Потім використовується метод навчання для збільшення незалежності елементів дескриптора відносно обертання.

Пари сусідніх характерних точок досліджуються для побудови остаточного двійкового дескриптора, який є результатом послідовності їх порівнянь. Виконання алгоритму ORB є на порядок швидшим ніж алгоритму SIFT при аналізі зображень розміром 640×480 при майже такій самій ефективності розпізнавання [79].

1.3.3. Методи на основі функцій перетворень

Методи на основі функціональних перетворень дають дескриптори для реєстрації зображень, в основному включаючи аналіз форми, напрямку, керовані фільтри, інваріанти афінних перетворень, перетворення Фур'є або вейвлет-перетворення.

Форма сприймається як клас еквівалентності в групі перетворень, що дає змогу свідчити, що дві форми однакові [50]. Метод полягає в ітераційному визначенні дистанції між формами та пошуку відповідного апроксимованого перетворення простору. Знаходження відповідності між двома формами еквівалентно знаходженню точки в кожному об'єкті з подібним контекстом форми [80]. Коррентропія (correntropy) вимірює подібність двох точкових наборів у околиці спільного простору. Шукається матриця перетворень двома

алгоритмами: surprise з мінімізацією коррентропії методом модельованого відпалювання та алгоритмом дивергенції Коші-Шварца (CS-Div). Обчислювальна складність алгоритму, $O(N^2)$.

В роботі [81] запропоновано параметричні спектральні дескриптори, які враховують статистику форми і клас перетворень, до яких він стає нечутливим. Дескриптор розраховується як згортка спектру зображення з власною функцією. Причому спектр формується з відгуків фільтрів форми, ядра яких тренуються.

Дескриптор форми для ідентифікації форми можна будувати шляхом вилучення різномасштабних ознак форми з використанням ознак висоти арки замість кривизни для опису форми. Це дає змогу пришвидшити розпізнавання у сотні разів[82].

В роботі [83] стверджують, що дескриптор форми забезпечує бажані властивості інваріантності, які включають обертання, масштабування, відображення. Для конкретної форми обчислюється її *сигнатура* — функціонал, який її характеризує і відрізняє від інших. Як функціонал, використовується інтеграл над областю з формою, який нормалізований для інваріантності відносно масштабу.

У методі оператора помітності (saliency) [84] регіони вважаються помітними, якщо вони є рідкісними одночасно як за деякими ознаками, так і у масштабних просторах. Метод ґрунтується на визначенні ентропії в межах областей на різномасштабних зображеннях, а помітність обчислюється як зважений максимум ентропії. Помітність пікселя визначається за його різницею у кольорі з іншими пікселями в зображенні.

Порівняльні дослідження ефективності (точності та швидкості) операторів помітності [85] відзначають, що спеціально розроблені оператори помітності перевершували ті, що були отримані в інших областях. Існують спроби використання глибокого навчання, яке спрямоване на кращу продуктивність і швидкість аналізу помітності [86].

1.3.4. Методи, що ґрунтуються на аналізі форми

Дескриптори форми використовують для виявлення подібних фігур у базі даних, навіть якщо вони є афінно змінені [87]. Дескриптор форми повинен мати можливість пошуку зображень для великої множини різних форм. Однією з важливих властивостей дескриптора форми є низька складність його розрахунку, чіткість і повторюваність результатів.

Дескриптор форми повинен мати наступні важливі особливості [88]. Властивість ідентифікації: подібні за сприйняттям людини форми повинні мати однакову особливість, яка відрізняється від інших. Інваріантність до обертання: дескриптор форми повинен генерувати однакову міру для даної фігури та для подібної фігури, яка повернута. Інваріантність до перетворення: дескриптор повинен генерувати подібну міру для фігури незалежно від її розташування в координатній площині. Інваріантність шкали: масштаб об'єкта не повинен впливати на міру, яку дає дескриптор форми. Афінна незмінність: лінійна проекція двовимірних координат у інші двовимірні координати повинна давати такі самі дескриптори форми. Стійкість до шуму. Інваріантність до перекриття: коли певні частини фігури перекриваються іншими фігурами, характеристики залишкової частини не повинні змінюватися у порівнянні з такою самою частиною цієї фігури. Статистична незалежність: дві характеристики фігури не повинні статистично залежати одна від одної. Надійність: обчислена функція форми повинна давати подібні результати для усіх подібних фігур.

Дескриптор форми, як правило, є вектором і повинен відповідати наступним вимогам: повинен бути достатньо повним, щоб правильно охарактеризувати форму, і його представлення має бути ефективним щоб виконувати порівняння, пошук та зберігання.

Було створено декілька методів пояснення та зображення фігур для програм пошуку фігури заданої форми [89]. Існують підходи на основі контурів та підходи на основі локальних ознак.

Методи подання форми та опису на основі контуру засновані на вилученні інформації про межу об'єкту або його контур [90]. Недоліки дескрипторів на основі контурів полягають в тому, що вони чутливі до шуму та змін через перекриття частин фігур. Вони не набули поширення, бо у багатьох застосунках більше значення має вміст об'єкту, а не його контур.

Підходи на основі локальних ознак більш надійні і можуть бути корисними в загальних застосунках. Вони можуть впоратися з перекриттям однієї фігури іншою. При цьому не тільки контур, а й уся площа фігури приймає участь у формуванні дескриптора [91].

Також підходи поділяються на структурний підхід та глобальний підхід. Глобальні методи не розбивають фігуру на частини, і вся інформація використовується для формування вектору ознак та процедури збігу. Структурні методи розділяють інформацію про фігуру на сегменти або підчастини (примітиви) і тому вони вважаються дискретними методами. Зазвичай кінцевим поданням структурного методу є рядок або граф (дерево).

Розглянемо декілька прикладів дескрипторів форми.

Дескриптор з ланцюговим кодом (Chain Code CC) — це метод представлення контуру об'єкта за допомогою послідовності прямолінійних відрізків, причому напрямки і довжина кодуються цілими числами [92]. Він є простим і інформативним, але не є інваріантним до обертання, масштабу.

Дескриптор гістограми ланцюгового коду дає ймовірності напрямків у межах контуру. Тому він є інваріантним до масштабу та обертання, хоча різні форми можуть мати спільний дескриптор [93].

Дескриптор контексту форми (Shape Context, SC) формується у два етапи. Спочатку в фігурі вибирається центр і вона описується множиною векторів, які напрямлені з центра до пікселів країв фігури. Потім дескриптор формується як гістограма розподілу векторів за напрямком, яка і називається контекстом [94]. Він є принципово незмінним до відображення, його нормування дає інваріантність до масштабу та повороту, змін форми.

Дескриптор кістяка фігури (SG) є описом графа кістяка, який складається з медіальних осей фігури [95]. Медіальна вісь – це геометричне місце центрів кіл, які вписані у фігуру. Граф кістяка представляє собою дерево, яке відповідає ієрархічному об'єднанню примітивів і тому зберігає топологічну інформацію про структуру фігури. Дескриптор є інваріантним до масштабу, поворотів, відображення та часткового перекриття форм. Але його складно обчислювати.

Локальний дескриптор зображення для використання гістограм градієнтів другого порядку (Histograms of Second Order Gradients HSOG) для фіксації пов'язаних із кривизною локальних геометричних властивостей запропоновано в [96].

Отже, для одержання властивості інваріантності щодо масштабу, відображення, повороту дескриптори форми використовують той чи інший статистичний алгоритм (розподіл кутів променів, гістограма ланцюгового коду, розподіл хорд і контексту фігури), представлення деревом або синтаксичний аналіз примітивів. Ця структура опису не тільки стискає інформацію про форму, але також забезпечує виразну та компактну структуру даних для подальших операцій розпізнавання.

1.3.5. Методи з перетворенням простору

При двовимірному перетворенні простору, де представлена фігура, можна одержати її ортогональні параметри, які формують дескриптор.

Дескриптор Фур'є – це набір коефіцієнтів двовимірного перетворення Фур'є області з формою. Узагальнений дескриптор Фур'є (generic Fourier descriptor, GFD) одержують попередньо нормалізувавши зображення та перетворивши його у полярні координати [97].

Узагальнений дескриптор Фур'є є інваріантним до обертання, масштабу та відображення. Але він не має інформації про просторове положення деталей форми.

Дескриптор вейвлет-перетворення є результатом розкладання об'єкта на піраміду зображень з різними масштабами та низькочастотними і високочастотними складовими [98]. Цей дескриптор має декілька необхідних функцій, таких як нескладність обчислення, представлення з різним масштабом, сталість, просторова локалізація та стабільність, можливість відтворення зображення.

Отже, дескриптор з перетворенням простору представляє не тільки контур, а й фігуру в цілому, правда, без завдання її структурних особливостей. При цьому необхідна точність представлення досягається вибором кількості коефіцієнтів перетворення, яка може бути досить великою. Недоліком є чутливість до зсуву, до повороту, відсутність фазової інформації.

1.3.6. Глибоке навчання і пошук характерних точок

Основними завданнями комп'ютерного зору є виявлення, локалізація та класифікація характерних точок, які можуть бути досягнуті глибоким навчанням. В методі Overfeat [99] для пошуку характерних точок запропоновано використовувати багатомасштабний підхід з вікнами регульованого розміру в нейронній мережі. При цьому, оператори розпізнавання низького рівня, які, власне, знаходять характерні точки, виконуються в перших згорткових шарах.

Обчислення операторів пошуку характерних точок дає змогу суттєво покращити зображення перед його обробкою у нейронній мережі. Дослідження в [100] пропонує метод очищення зображення від шумів для покращення глибокого навчання. Аналогічна мета досягнута в роботі [101], в якій модифікована мережа VGG.

Моделі машинного навчання та штучного інтелекту зазвичай застосовуються як чорна скриня, тобто вони не надають інформацію про те, що саме змушує їх прийти до своїх прогнозів. Однією з переваг методів розпізнавання образів на основі пошуку характерних точок є те, що можна легко знайти докази того, як і чому методи працюють успішно. При машинному навчанні це складне завдання, оскільки не можна обґрунтовано відповісти на запитання:

чому саме навчена система? Це може бути критичним у деяких галузях, таких як аналіз фінансів, медичних зображень, системи безпеки чи зброя з інтелектом. Для достовірних доказів необхідно знати ймовірну помилку оцінки значущості будь-якого результату [102].

Також машинне навчання використовується для оптимізації методів формування дескрипторів характерних точок. Наприклад, у дескрипторі BRISK випадковим чином вибираються 512 порівнянь пікселів навколо центру точки. Координати цих пікселів у ефективних конфігураціях варто шукати за допомогою машинного навчання [103].

1.3.7. Розпізнавання образів за характерними точками

Розпізнавання образів у зображенні, яке оброблене за алгоритмами пошуку характерних точок полягає у розгляданні кожної такої точки та порівнянні її з аналогічними точками з бази даних. База даних складається з дескрипторів точок, які були вилучені з навчальних зображень. Багато з таких порівнянь буде некоректним, бо властивості точки є неоднозначними при наявності шумів у зображенні. Тому у зображенні виділяють кластери принаймні з 3-х точок і тоді порівнюють такі кластери, бо вони у своєму взаємному розположенні мають більшу ймовірність мати відношення до об'єкту ніж окремі точки.

Порівняння характерних точок. Найближчою вважається точка, яка має мінімальну евклідову відстань до точки, яка розглядається, якщо порівнюють вектори дескрипторів цих точок. Але для прийняття рішення про розпізнавання важливим є взаємне розположення сусідніх точок. Отже, співпадіння образів можна зарахувати, якщо співпадуть принаймні дві сусідні характерні точки. У практиці авторів методу SIFT, якщо відстань між векторами дескрипторів більша за 0,8, то вважається, що точки не співпадають і це вірно у 90 % випадків, а у 5 % випадків ці точки насправді співпадають.

Індексація ефективних сусідніх точок. Алгоритм повного перебору забезпечує точну ідентифікацію сусідніх точок. Є алгоритми пошуку, такі як

k-d-дерева Фрідмана, але останній забезпечує прискорення пошуку лише коли розмірність вектора менше за 10. У випадку багатовимірних дескрипторів використовують алгоритм наближеного пошуку, згідно з яким краща координата вибирається першою (Best-Bin-First (BBF)) [104]. Це наближений алгоритм у тому сенсі, що він знаходить найближчого сусіда з великою ймовірністю. Алгоритм BBF використовує модифіковане упорядкування для алгоритму k-d-дерева, так що координати у просторі ознак шукаються у порядку, збільшення відстані від точки, з якою виконується порівняння [105]. Для бази даних з 10^5 характерних точок це забезпечує пришвидшення на два порядки у порівнянні з точним методом при втраті менше 5 % кандидатів, які були б знайдені точним методом.

1.3.7. Порівняння методів пошуку характерних точок

В табл. 1.1 показане порівняння методів пошуку дескрипторів [106]. Проаналізувавши цю таблицю, можна зробити висновок, що серед великого різноманіття методів пошуку характерних точок і побудови їх дескрипторів заслуговує увагу метод SIFT. Також ефективні похідні від SIFT – методи SURF, GLOH, ASIFT та ін. як методи, що дають найкращі результати виявлення цих точок. Недоліком методу є те, що надійність пошуку характерних точок залежить від освітленості, що ілюструє приклад на рис. 1.5.

Таблиця 1.1 – Методи пошуку дескрипторів характерних точок

Метод	Розмір дескриптора	% знайдених характерних точок	Ефективність визначення характерної точки в умовах			
			повороту зображення	шумів	поганої освітленості	різних масштабів зображення
SIFT	128	90	дуже добре	дуже добре	дуже добре	дуже добре
SURF	64	55	добре	дуже добре	дуже добре	добре
ORB	32	54	найкраще	дуже добре	дуже добре	добре
DAISY	200	56	погано	добре	дуже добре	погано
BRIEF	32	45	погано	дуже добре	дуже добре	погано
FREAK	64	50	середнє	добре	добре	погано

HoG	81	–	погано	погано	добре	дуже добре
BRISK	64	60	середнє	добре	дуже добре	добре
BOLD	144	18	погано	середня	дуже добре	погано

Композиція методів також дає гарні результати. Так, SIFT із використанням детектора SURF показує найкращі результати.

1.3.8. Висновки щодо методів пошуку характерних точок.

На відміну від нейронних мереж, методи дескрипторів характерних точок направлені на пошук і вилучення інформації з зображення, яка має заздалегідь відомі і очікувані властивості. Тому за дескрипторами можна визначити причину того чи іншого акту розпізнавання.

Дескриптори характерних точок дають змогу розпізнати об'єкт у різноманітних обставинах: при різних освітленні, масштабі об'єкта, його обертанні, а також коли об'єкт частково покривається іншими об'єктами.

Усі розглянуті методи виконуються у два етапи. На етапі пошуку характерної точки зображення сканується певною фільтруючою системою, яка за заданим порогом виділяє область з центром характерної точки та визначає множину різноманітних параметрів цієї області. На другому етапі знайдені параметри нормуються і з них формується вектор дескриптора. Методи відрізняються між собою за принципом дії і алгоритмом фільтруючої системи та особливостями формування вектору дескриптора.

Є сенс суміщати пошук характерних точок зі згортувальною нейронною мережею, в якій перші шари замінюються шаром визначення характерних точок, який видає на наступні шари знайдені ознаки зображення, як це виконано у методі Overfeat. При цьому нейронна мережа одержує здатність охоплювати кадри зображення більшої площі за рахунок того, що у новому шарі будується піраміда зображень та можливість аналізу причин прийняття того чи іншого рішення.

Варто шукати удосконалений метод типу SIFT, який мав би меншу складність та здатність знаходити характерні точки у несприятливих умовах освітлення.

У деяких методах задіяні ідеї, що походять з досліджень зорової системи організмів, як наприклад, у методі FREAK.

Методи пошуку характерних точок, в загальному, чутливі до умов освітленості зображення. Багато з них вимагають бінаризації зображення, яке досягається встановленням порогу яскравості, в результаті чого втрачається інформація в погано освітлених ділянках. У методах, що використовують піраміду зображень, яскравість кадру дещо нормалізується через те, що шукається різниця яскравостей пікселів. Але при цьому зростає рівень шумів зображення. Крім того, усі методи розраховані на обробку зображення з невеликим динамічним діапазоном, який визначається розрядністю пікселів, що дорівнює 8.

Як правило, методи пошуку характерних точок реалізуються програмно. Для їх прискорення використовуються графічні акселератори. Але апаратна реалізація цих методів трапляється рідко, як наприклад, [107]. Це пояснюється тим, що алгоритми такого пошуку є складними і відсутні методи їх занурення у апаратні засоби. Отже, у наступному розділі варто розглянути особливості апаратної реалізації методів пошуку характерних точок.

1.4. Апаратна реалізація розпізнавання образів

Як було вказано у підрозділі 1.2, штучні нейронні мережі при їх навчанні реалізують у апаратних прискорювачах — графічних акселераторах, а при використанні навченої мережі — застосовують або спеціалізовані процесори штучного інтелекту, або реалізацію мережі у ПЛІС. Причому CNN, яка використовується для різних цілей та галузей застосування, як правило, при апаратній реалізації має одну і ту саму структуру. Відміни є лише в вибраній розрядності даних, представленні даних з плаваючою чи фіксованою комою, структурі вузла-нейрона, яка визначається кількістю входів та функцією активації і можливо, мережею зворотного поширення помилки. Також CNN

відрізняються кількістю шарів, конфігурацією міжвузлових з'єднань, кількістю вузлів у кожному шарі та порядком їх виконання (послідовним, паралельним чи послідовно-паралельним).

Загалом, є досить багато робіт, які присвячені апаратній реалізації CNN. На противагу CNN, методи дескрипторів характерних точок суттєво різняться між собою. Відповідно, є різними їх апаратні реалізації. Причому, через те, що ці методи мають менше розповсюдження, таких апаратних реалізацій мало, хоча вони різноманітні. Тому варто розглянути реалізацію методів дескрипторів характерних точок в апаратурі.

1.4.1. Апаратна реалізація методів дескрипторів характерних точок

При реалізації алгоритму SIFT та похідних від нього алгоритмів кожний з чотирьох його етапів (див. підрозділ 1.3.1) виконується в окремому блоці, а саме, блоках знаходження простору масштабів зображення, детектування ключових точок, обчислення орієнтації і амплітуди градієнтів та формування дескриптора.

При виконанні перших двох етапів кадр зображення сканується апертурами фільтрів. Результати фільтрації обробляються інтелектуальною схемою детектора характерної точки. При цьому обчислення виконуються в реальному часі у темпі прийому пікселів з потоку зображення. Отже, при обробці кадру зображення розміром $n \times m$ алгоритм детектування характерної точки повторюється $n \times m$ разів у конвеєрному режимі. Тому тактова частота модулів f_C для пошуку характерних точок найчастіше співпадає з частотою слідування пікселів f_P у кадрі. Відповідно, модулі виконання першого і другого етапів виконують обчислення у паралельних схемах, структури яких співпадають з графами потоків даних алгоритмів, які в них виконуються. Це є причиною того, що потребуються великі апаратні витрати на реалізацію детектування характерних точок. У різних проектах модулів для пошуку характерних точок використовуються ті чи інші способи мінімізації апаратних витрат.

На першому етапі обчислюються фільтри Гауса з різними масштабами σ . Для зменшення кількості помножувачів у p разів у пристрої [108] використовується частота синхросигналу, яка є кратною частоті слідування $f_C = p f_P$. Тоді фільтри складаються з помножувачів, на виході яких стоять акумулятори результатів. Недоліком такої схеми є те, що вона може застосовуватись для невеликих кадрів або при невеликій пропускній здатності кадрів за секунду.

Для мінімізації апаратних витрат за рахунок зменшення кількості множень використовують декомпозицію двовимірного фільтра Гауса на фільтр, що фільтрує рядки і фільтр фільтрації колонок [109–111]. Недоліком такого способу є необхідність використання складної буферної схеми між ступенями кожного фільтра, яка затримує $k-1$ рядків зображення при апертурі розміром $k \times k$.

У багатьох пристроях, як наприклад, [111,112] використовується ланцюг Гаусових фільтрів. При цьому за рахунок того, що амплітудо-частотна характеристика (АЧХ) кожного наступного ступеня маскує АЧХ попередніх ступенів, складність такого фільтра (кількість коефіцієнтів) є суттєво нижчою. Крім того, при переході до нижніх октав виконується децимація зображення, яка кожен раз зменшує складність обчислень вчетверо. Але для подальшої обробки результати фільтрації слід затримати у додаткових буферних схемах.

Суть фільтрації Гаусовими фільтрами – виявлення градієнту (просторової похідної) у зображенні за допомогою подальшого обчислення різниці результатів двох сусідніх фільтрів. Наявність характерної точки виявляється за максимумом абсолютної величини цього градієнту. Тому важливе визначення приблизної амплітуди градієнту, точність обчислення якої достатня для виконання вибору характерної точки. І відповідно, одержання величини градієнту, точність якої є достатньою для прийняття рішення, є важливішою за дотримання АЧХ саме Гаусового фільтра. Як результат, у проєктах мінімізується складність Гаусових фільтрів за рахунок зменшення розрядності коефіцієнтів чи заміни множення на коефіцієнт операцією зсуву [113]. Крім того, у

обчисленнях використовується властивість симетрії імпульсної характеристики фільтра [111]

На другому етапі у стосах кадрів з градієнтами у різних масштабах шукаються характерні точки. Першим критерієм характерної точки є максимум абсолютного значення градієнту. При цьому у більшості проєктів паралельна деревовидна схема виконує пошук максимуму і мінімуму серед 27 сусідніх пікселів [114]. Другим критерієм є наявність в точці форми у вигляді кута, кінця лінії або плями. Для цього замість детектора Гарріса, як у оригінальному алгоритмі, використовують формування матриці Гессе (1.2) та обчислення відношення $(\alpha + \beta)^2 / (\alpha\beta)$ з (1.3), яке порівнюють з порогом. Для формування матриці Гессе беруться результати вимірювання градієнтів по вертикалі та горизонталі за допомогою фільтрів Превітта. Крім того, матриці Гессе для сусідніх пікселів згладжуються фільтром Гауса розміром 3×3 . Усі обчислення виконуються паралельною схемою, в якій коефіцієнти фільтрів приймають значення 0, 1 чи 2 [113].

Результуючий градієнт і його напрямок у знайденій точці (x,y) у кадрі G обчислюється за формулами [112]:

$$\begin{aligned} m(x,y) &= \sqrt{(G(x+1,y) - G(x-1,y))^2 + (G(x,y+1) - G(x,y-1))^2}, \\ \theta(x,y) &= \arctg(G(x,y+1) - G(x,y-1)) / (G(x+1,y) - G(x-1,y)). \end{aligned} \quad (1.12)$$

Вирази (1.6) обчислюються окремими блоками або з використанням алгоритму CORDIC [114], або з використанням табличного методу [108].

Після того, як чергова характерна точка розпізнана, її параметри, включаючи амплітуду $m(x,y)$ і напрямок максимуму градієнту $\theta(x,y)$, координати (x,y) передаються до обробки на третьому і четвертому етапах.

Вхідні дані на 3 і 4 етапи обчислення орієнтації і амплітуди градієнтів та формування дескриптора приходять у випадкові моменти часу. Тому вони, як правило, буферизуються в FIFO. Хоча, в середньому, одна характерна точка припадає на не менше ніж 100 пікселів, обсяг обчислень на цих етапах є великим. Наприклад, у системі [111] на реалізацію перших двох етапів витрачено

7708 ЛТ і 131 помножувач, то на решту етапів витрачено 100590 ЛТ і 171 помножувач. Тобто, на 3 і 4 етапи потрібно апаратних витрат у чотири рази більше, приймаючи до уваги складність помножувача. Це пояснюється тим, що обчислення є не такими регулярними, як на перших двох етапах і потребують швидкого обчислення елементарних функцій, таких як квадратний корінь, арктангенс, виконання повороту вектору та ін.

Спочатку, у вікні розміром до 23×23 навколо точки обертається квадрат 16×16 , на кут, який є близьким до $\theta(x, y)$, потім квадрат розбивається на 16 апертур. У кожній апертурі розраховуються гістограми, розраховується корінь із суми квадратів амплітуд для нормалізації діленням. В роботі [110] експериментально визначено, що 16 різних кутів повороту є достатньою кількістю для реалізації ефективного дескриптора. Повороти вікна у багатьох роботах виконують за допомогою конвеєрної схеми CORDIC, яка має 5-7 ступенів [111, 113, 114].

Після цього обчислюються градієнти зображення у кожному пікселі квадрата 16×16 з вимірюванням кута градієнта з розділенням 45° . Потім у 16 апертурах у 8 бінах збираються діаграми напрямленості градієнту, які нормуються діленням на квадратний корінь з суми квадратів амплітуд.

Через те, що обчислення на 3 і 4 етапах є нерегулярними та вимагають великих обсягів обчислень, у більшості проєктів ці етапи перекладені на виконання у універсальний процесор [112]. Таким процесором є вбудоване у ПЛІС ядро процесора ARM або зовнішній процесор. При цьому те, що обчислення є нерегулярними, стає неважливим в порівнянні з апаратною реалізацією. Недоліком такого підходу є мала швидкодія, яка обмежена періодом доступу до пам'яті кадра.

1.4.2. Реалізація буферних схем

При пошуку характерних точок і формуванні їх дескрипторів виконується обробка даних, які з одного боку, представляють двохвимірні масиви, а з іншого боку, поступають на конвеєрну обробку кількома потоками даних.

Причому ці потоки відповідають деякій апертурі, яка рухається по масиву, скануючи зображення. На кожному етапі обробки і на вході кожного фільтра використовуються різні буферні схеми.

У багатьох випадках для організації сканування по рядках необхідно вставляти буферні схеми, кожна з яких зберігають цілий рядок кадру. Розглянемо обчислення функції фільтра від дев'яти значень пікселів у апертурі. За алгоритмом, дев'ять пікселів потрібно зчитувати з пам'яті кадру для кожної позиції апертури у кожному такті синхросерії та кожен піксель має зчитуватися дев'ять разів, коли апертура сканує зображення. Пікселі, що розташовані поруч по горизонталі, потрібні в послідовних тактах, тому можуть бути буферизовані та затримані в регістрах. Це зменшує кількість читань до трьох пікселів за кожен такт. Буфер рядків зберігає значення пікселів попередніх рядків, щоб уникнути повторного читання значень пікселів (Рис. 1.7).

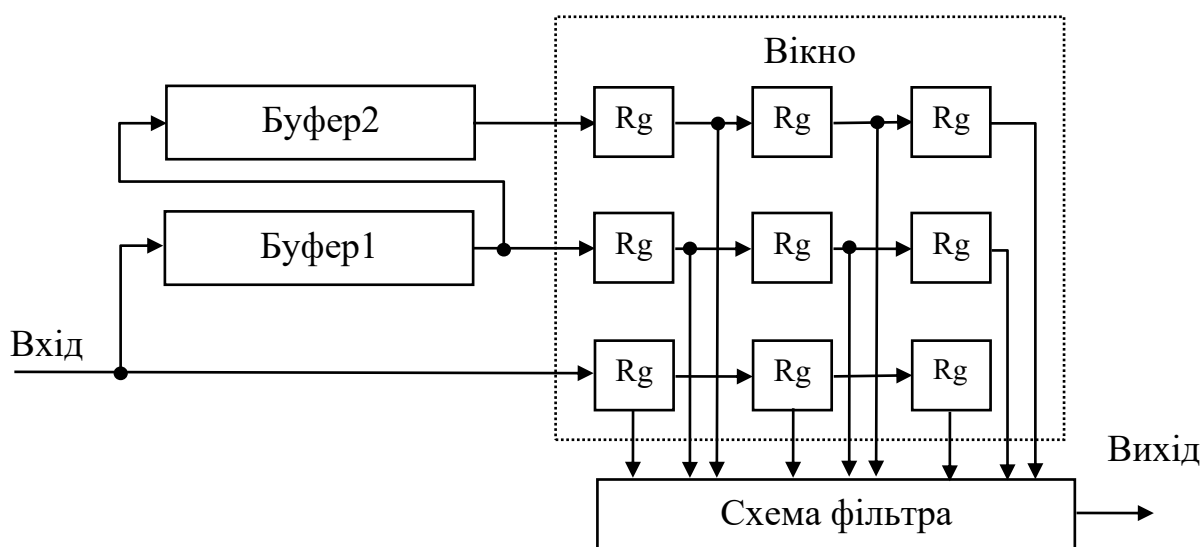


Рисунок 1.7 – Типова схема обробки кадру зображення

Кожен буфер рядка фактично затримує введення пікселів на один рядок. Очевидною реалізацією такої цифрової затримки є використання $N-3$ -ступеневого регістра зсуву, де N — довжина рядка зображення. Блок ОЗП (BRAM) у ПЛІС налаштовують як буфер FIFO за схемою циклічного буфера. Крім того, декілька паралельних буферів рядків реалізують як один буфер, але з більшою розрядністю даних.

За схемою, як на рис.1.7 будують буфери даних у кожному Гаусовому фільтрі в проєктах [115, 116]. Причому чим більше ядро фільтра – тим більше число затримок використовується [113, 117].

Такі ж буферні схеми використовуються для детектування характерної точки [108, 110], а також при обчисленні векторів градієнтів дескриптора [115]. Таким чином, через те, що на буфери FIFO припадають великі обсяги пам'яті та оскільки число блоків пам'яті BRAM у ПЛІС є обмеженим, то обмеженими є об'єм кадрів, що обробляються. У табл. 1.2 показані параметри деяких проєктів, що реалізують алгоритм SIFT, з аналізу якої видно вказану залежність. Апаратні витрати визначені у кількості логічних таблиць (ЛТ), апаратних помножувачів (DSP48) та блоків пам'яті (BRAM). Слід зауважити, що у ПЛІС, в середньому, на один помножувач DSP48 припадає один блок BRAM і 160 ЛТ. Вартість ПЛІС можна оцінити так, що ПЛІС з N блоками DSP коштує приблизно N USD.

Таблиця 1.2 – Апаратні реалізації методу SIFT

Джерело	Розміри кадру	Пропускна спроможність, кадрів за сек.	Апаратні витрати,		
			ЛТ	DSP48	BRAM, 18kB
Bonato [111]	320 x 240	30	43 366	64	75
Zhong [112]	320 x 256	100	18 195	56	156
Yao [108]	640 x 480	32	35 889	87	178
Chiu [118]	640 x 480	30	57 598	8	67
Li [114]	640 x 480	30	65 560	642	18
Vourvoulakis [113]	640 x 480	70	125 644	77	23
Kerowsky [110]	1920 x 1080	84	108 298	302	940

Для зменшення об'єму буферних схем використовують не послідовне, а паралельне з'єднання фільтрів. У цьому випадку виходи з регістрів, що формують дані з апертури (рис.1.7) підводяться паралельно до кількох схем

фільтрів. При цьому необхідно збільшити розміри апертури до 7×7 та більше і відповідно, складність фільтрів через те, що у крайнього фільтра збільшується коефіцієнт масштабу σ , який вимагає збільшення порядку фільтра [108, 112, 118].

Об'єм буферів FIFO суттєво зменшується, якщо зменшуються розміри кадра. У пристрої [109] характерні точки шукаються у вікні — регіоні інтересу розміром 80×60 . При цьому аналіз всього кадру виконується при послідовному пересуванні регіону інтересу, що зменшує швидкодію такого методу.

Якщо швидкодія метода не є актуальною, то найпростіше рішення — це схема з двома блоками пам'яті кадрів та проміжних результатів, а обробка виконується за алгоритмом пінг-понг, коли дані перекачуються через конвеєрну схему з одного блоку в інший і назад [119]. Так само в роботі [116] описано пристрій, в якому SIFT алгоритм реалізований за допомогою FIFO-буферів, в кожному зберігаються кадри зображення до і після обробки Гаусовим фільтром.

Іншою складною задачею є буферизація обробки паралельних потоків даних та підтримка їх синхронізації. Для цього використовують буфери FIFO з різною глибиною та розрядністю. Причому синхронізуючі сигнали, як наприклад, початок кадру, прапорець наявності характерної точки також затримуються в таких буферах [117].

В загалом, буферизація даних виконується за інтуїтивними правилами конвеєризації оброблення даних, використовуючи загальну схему, таку як на рис. 1.7. При цьому нерідко обчислювальна апаратура виявляється недовантаженою, особливо коли тактова частота у кілька разів вища за частоту слідування пікселів у зображенні, що обробляється.

1.5. Висновки до першого розділу

В даний час набули великого поширення методи розпізнавання графічних образів на основі штучних нейронних мереж та методи, які, ґрунтуючись на детектуванні характерних точок.

На сьогоднішній день такі нейронні мережі, як CNN, стали найпоширенішою технологією для розпізнавання образів. Але CNN має такі недоліки, як висока складність навчання, негарантованість одержання ефективно навченої CNN, складність обчислень мережі, ненадійне розпізнавання об'єктів з різними масштабами та кутами повороту.

На відміну від нейронних мереж, методи дескрипторів характерних точок направлені на пошук і вилучення інформації з зображення, яка має заздалегідь відомі і очікувані властивості. Дескриптори характерних точок дають змогу розпізнати об'єкт у різноманітних обставинах: при різних освітленні, масштабі об'єкта, його обертанні, а також коли об'єкт частково покривається іншими об'єктами.

Можливо суміщати пошук характерних точок із CNN, в якій перші шари замінюються шаром визначення характерних точок, який видає на наступні шари знайдені ознаки зображення. При цьому нейронна мережа одержує здатність охоплювати кадри зображення більшої площі, розпізнавати об'єкти у різному масштабі та при їх обертанні та можливість аналізу причин прийняття того чи іншого рішення.

Варто шукати удосконалений метод пошуку характерних точок, такий як SIFT, але такий, який мав би меншу складність та здатність знаходити характерні точки у несприятливих умовах освітлення, коли, наприклад частина об'єкту знаходиться в тіні.

Методи пошуку характерних точок розраховані на обробку зображення з невеликим динамічним діапазоном, який визначається розрядністю пікселів, що дорівнює вісім і втрачають ефективність при погіршенні умов освітленості.

Спеціалізовані комп'ютери, які виконують пошук характерних точок та формування їх дескрипторів, дають змогу зменшити латентну затримку розпізнавання образів, зменшити навантаження на телекомунікації при їх встановленні на боці відеодатчика та зменшити енергоспоживання. Але вони є ще

високовартісними через те, що потребують великих апаратних витрат при їх реалізації в сучасних ПЛІС.

Великі апаратні витрати у пристроях для пошуку характерних точок зумовлені виконанням в них складного алгоритму, складних схем цифрових фільтрів, схем обчислення параметрів характерних точок, включаючи модулі обчислення елементарних функцій. Складність багатьох схем обумовлена недо-завантаженістю апаратних блоків та використанням малоефективних буферів даних.

Завданнями для наукових досліджень мають бути наступні.

Розробка алгоритму пошуку характерних точок, який на відміну від існуючих алгоритмів мав би меншу складність та забезпечував би пошук у складних умовах освітлення.

Створення методу побудови буферних схем для обробки одно- та двох-вимірних сигналів, який забезпечує заданий порядок слідування вхідних та вихідних даних і мінімізовані апаратні витрати при його реалізації у ПЛІС.

Удосконалення методу побудови обчислювачів елементарних функцій, що реалізуються в ПЛІС та побудова за ним зразків таких обчислювачів.

Розробка і експериментальна перевірка системи обробки зображень, яка здатна виконувати пошук характерних точок за допомогою нових методів.

Вирішення цих завдань показане у наступних розділах.

РОЗДІЛ 2. РОЗРОБЛЕННЯ МЕТОДУ ПОШУКУ ХАРАКТЕРНИХ ТОЧОК У ЗОБРАЖЕННІ

У попередньому розділі було встановлено, що актуальним предметом дослідження є удосконалення методу пошуку характерних точок, такого як метод SIFT, з метою зменшення складності та покращення знаходження характерних точок у несприятливих умовах освітлення. Даний розділ присвячений досягненню цієї мети.

2.1. Передумови для створення методу

У першому розділі було встановлено, що методи пошуку характерних точок є ефективними для розпізнавання образів та вирішення інших задач машинного зору. Але вони розраховані на обробку зображення з невеликим динамічним діапазоном, який визначається розрядністю пікселів, що найчастіше дорівнює вісім. Крім того, ці методи дуже чутливі до умов освітленості об'єктів, що обробляються.

Зокрема, багато методів чутливі до границь діапазону яскравості робочої області, які нерідко встановлюються вручну. На рис. 1.5. показано приклад пошуку характерних точок методом SIFT, який показує, що навіть незначні зміни яскравості об'єкта призводять до суттєво різних результатів знаходження таких точок (кількість точок на лівій і правій частинах зображення симетричного об'єкта та їх положення суттєво відмінні).

При знаходженні шуканої точки в тіні на результати пошуку суттєво впливають шуми зображення, які підсилюються операцією диференціювання, яка виконується для детектування характерної точки. Доведено, що фільтрація цих шумів покращує надійність пошуку характерних точок за рахунок зменшення їх числа [118].

Ще гірші умови для розпізнавання стають тоді, коли окіл характерної точки припадає на границю між освітленою ділянкою та ділянкою в тіні. Тоді обчислений дескриптор цієї точки стає кардинально відмінним від дескриптора точки, яка знаходиться у нормальних умовах освітленості.

Отже, можна припустити, що якщо обробити зображення перед пошуком характерних точок таким чином, що вирівнюються яскравості у ділянках тіней, надмірно освітлених ділянках з відповідною фільтрацією шумів та збереженням локальних контрастів (зображень плям, кутів, ліній), то умови пошуку характерних точок при цьому суттєво покращаться.

Одним з перспективних методів такої обробки є стиснення зображення з широким динамічним діапазоном (high dynamic range, HDR). Стисненням HDR-зображення називається таке стиснення його динамічного діапазону, результатом якого є зображення з якістю, що дає змогу його відтворити на екрані дисплея або шляхом друку без втрат значимих деталей [120]. Наприклад, якщо динамічний діапазон HDR-зображення складає $10^6:1$, тобто, 120 дБ, то після стиснення одержується діапазон яскравості $255:1$, тобто, 48 дБ. Крім того, обробка HDR-зображення дає змогу знаходити характерні точки у місцях з дуже поганими умовами освітлення, в яких розглянуті у першому розділі методи неспроможні знайти будь-що. Ця можливість відкривається завдяки тому, що динамічний діапазон HDR-зображення може у тисячі разів перевищувати динамічний діапазон зображень, які обробляються відомими методами.

HDR-зображення часто використовується у фотографії, тому що воно має максимум інформації про сфотографовану сцену і його можна обробляти за певним художнім задумом. Також це зображення необхідне для робототехніки, коли об'єкти, що розпізнаються, знаходяться у несприятливих умовах освітлення. Особливо це стосується систем сприяння керуванню транспортом (ADAS). Також зберігання та обробка рентгенівських знімків в медицині відбувається при їх HDR-представленні. Для реалізації ефектів доповненої реальності (augmented reality) сцени, що відтворюються, також повинні мати великий динамічний діапазон, щоби можна було їх коректувати в залежності від умов. Тому пошук характерних точок при обробленні HDR-зображень розкриває нові перспективи впровадження комп'ютерного зору.

Вказане стиснення HDR-зображення не може бути виконане за допомогою простих алгоритмів, наприклад, за допомогою корекції яскравостей тіней та освітлених ділянок через те, що при цьому відбуваються втрати деталей та збільшення шумів. Як результат, умови пошуку характерних точок не покращуються, а погіршуються. Тому слід вибирати алгоритми стиснення з інтелектуальною обробкою. Така інтелектуальна обробка, можливо, дає змогу вилучити певні додаткові ознаки характерних точок, які підвищують їх достовірність та покращують їх дескриптори.

На основі вище приведеного слід висунути гіпотезу, що новий метод пошуку характерних точок має ґрунтуватись на аналізі HDR-зображень під час та після їх стиснення. Тому далі слід проаналізувати алгоритми оброблення HDR-зображень з метою пошуку такого алгоритму, який забезпечує ефективний пошук характерних точок.

2.2 Оброблення зображення з широким динамічним діапазоном

2.2.1. Створення та оброблення HDR-зображень

Дотепер HDR-зображення формується з кількох зображень однієї сцени але з різними експозиціями. Потім ці зображення з'єднуються одне з одним за допомогою одного з відомих методів, наприклад, таким, який описано в [121]. Зараз HDR-зображення у багатьох оптико-електронних системах фото- або відеокамер формується безпосередньо у датчику зображення з використанням різних експозицій і суміщення одержаних кадрів.

Далі HDR-зображення обробляється алгоритмами цифрової обробки зображення та відтворюється на дисплеї або запам'ятовується у пам'яті великого розміру. Як правило, HDR-зображення займає у чотири рази більше місця у пам'яті, ніж звичайне зображення, так як кожен колір пікселю зберігається як число з плаваючою комою. Через це є труднощі не тільки зберігання HDR-зображення, але й його обробки.

Візуалізація HDR-зображення виконується, як правило, за допомогою звичайних дисплеїв, які мають динамічний діапазон близько 200:1. Для цього необхідно виконати стиснення динамічного діапазону, яке називають перетворенням, відображенням тонів (tone mapping). Наприклад, на рис. 2.1 показані зображення сцени з великим динамічним діапазоном без перетворення та з перетворенням тонів.



а)



б)

Рисунок 1.2 – Зображення сцени з великим динамічним діапазоном без перетворення (а) та з перетворенням тонів (б)

Порівнюючи ці зображення, можна помітити, що на другому з них краще видно деталі як у тінях, так і у світлих місцях та краще відтворені кольори. Тобто, операція перетворення тонів є, свого роду, операцією компресії зображення (у даному прикладі – компресія у 3 рази), яка є втратною, але зі збереженням деталей.

2.2.2. Алгоритми перетворення тонів зображення

Операція перетворення тонів у загальному випадку може бути виражена як функція f відображення кадру I з множиною пікселів з великим діапазоном $\mathbb{R}^{w \cdot h \cdot 3}$ у множину пікселів з діапазоном $[0, 255]$

$$\mathbb{Z}^{w \cdot h \cdot 3} : f(I) : \mathbb{R}^{w \cdot h \cdot 3} \rightarrow \mathbb{Z}^{w \cdot h \cdot 3}, \quad (2.1)$$

де w і h — ширина та висота кадру. У більшості випадків відображення виконується для складової яскравості L_w зображення, в той час як інтенсивність кольорів пікселя, представленого показниками яскравості кольорів R_w, G_w, B_w , не перетворюється, а лише масштабується, тобто:

$$f(I) = \begin{cases} L_d = f_L(L_w) : \mathbb{R}^{w \cdot h} \rightarrow [0, 255]_{w \cdot h}, \\ R_d = L_d (R_w / L_w)^s, \\ G_d = L_d (G_w / L_w)^s, \\ B_d = L_d (B_w / L_w)^s, \end{cases} \quad (2.2)$$

де $s \in (0, 1)$ — коефіцієнт обрізання.

Різнорічні алгоритми обчислення оператора $f(I)$ можна розділити на наступні 4 групи:

- глобальні, коли одне й те саме перетворення виконується з кожним пікселем;
- локальні, коли відображення даного пікселя залежить від сусідніх пікселів;
- операції сегментації, коли зображення ділиться на великі регіони, для кожного з яких виконується окремий оператор відображення;
- частотно-градієнтні алгоритми, у яких зображення розділяється на низькочастотну та високочастотну складові і трансформація виконується над низькочастотною складовою.

Далі розглядаються різні типи алгоритмів метою вибору оптимального для реалізації високопродуктивних процесорів систем технічного зору.

Перед тим, як виконати однорічні обчислення над кожним пікселем зображення, у глобальному алгоритмі спочатку аналізуються його статистичні властивості. Це такі властивості, як, наприклад, максимальна, мінімальна та середня яскравості. Для збільшення надійності оцінок, на які впливають шуми зображення, будується таблиця розподілу яскравості, звідки визначаються максимальна та мінімальна яскравості. Найпростішими операторами є такі

поширені перетворення, як лінійне масштабування, логарифмічна та експоненціальна функції.

Основними недоліками глобального оператора є те, що він не зберігає локальний контраст зображення та його невеликі деталі, а також порівняно невеликий динамічний діапазон вхідних даних [122, 123].

Локальні оператори, на відміну від глобальних, покращують зображення, тому що вони обробляють як глобальний, так і локальний контраст зображення. Для цього у оператор відображення f додається вплив від сусідніх пікселів.

У роботі [124] запропоновано алгоритм неоднорідного масштабування, у якому вперше дається спроба збереження локального контрасту. Для цього використовується формула: $L_d(\mathbf{x}) = L_w(\mathbf{x})/s(\mathbf{x})$, де $s(\mathbf{x})$ — локальне середнє значення яскравості оточуючих пікселів, яке обчислюється як згортка з ядром Гаусового фільтра з параметром σ . Параметр розмиття σ вибирається з урахуванням, що при малих значеннях σ результуюче зображення стає низько-контрастним, а при великих — над контурами деталей стає видно перекручення у вигляді гало.

Для мінімізації гало, використовують алгоритм з різномасштабною обробкою. У роботі [124] запропоновано представлення HDR-зображення як складові з довгим, середнім та коротким періодом хвиль. Таке представлення будується як піраміда зображень у методі SIFT. Результуюче зображення формується як комбінація оброблених складових зображень. Недоліками алгоритму є виникнення гало, якщо діапазони частот в складових зображеннях вибрані невдало, а також значна складність обчислень.

Аналогічні результати дають алгоритми, запропоновані у [125] на основі теорії Retinex [126]. Згідно з цією теорією, яскравість пікселя вхідного зображення з координатами $\mathbf{x} = (x, y)$ є добутком

$$I(\mathbf{x}) = L(\mathbf{x}) \cdot R(\mathbf{x}) \quad (2.3)$$

освітлення $L(x)$ і віддзеркаленої яскравості $R(x)$ об'єкта. Згідно з методом Retinex, зображення $I(x)$ певним чином розкладають на складові $L(x)$ та $R(x)$, причому $L(x)$ обробляють зі стисненням динамічного діапазону, а $R(x)$ — з покращенням контрасту. Оброблені складові зображення перемножують, одержуючи результуюче зображення $I'(x)$ згідно з (2.3).

Компоненти освітлення добуваються за допомогою функції визначення освітленості $F(I)$, так що

$$\begin{aligned} L(x) &= F(I(x)), & R(x) &= I(x) / F(I(x)), \\ L'(x) &= \Gamma(L(x)), & R'(x,y) &= \beta(R(x)), \\ I'(x) &= L'(x) \cdot R'(x). \end{aligned} \quad (2.4)$$

Потоковий граф алгоритму Retinex показано на рис. 2.2.

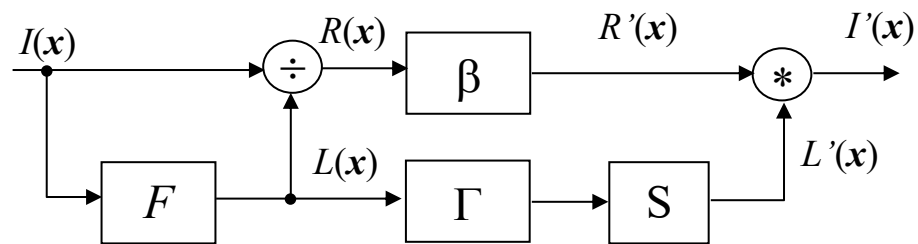


Рис. 2.2 – Потоковий граф алгоритму Retinex

У даному поточковому графі вершини (прямокутники, кола) позначають виконувані операції, а напрямлені дуги, лінії зв'язку — потоки даних. Таким чином, поточковий граф представляє собою модель, на якій задано поточковий алгоритм обробки зображень. Дані $I(x)$, $L(x)$, $R(x)$ та ін. представляють собою відповідні потоки даних, які обробляються у вершинах послідовно піксель за пікселем. Причому граф вважається синхронним, якщо дані в потоках мають взаємно однозначну відповідність. У даному графі ця відповідність визначена однойменними індексами даних $x = (x, y)$. Граф може бути ієрархічним. Тоді вершина графа представляється деяким графом нижчого рівня. Наприклад, якщо це вершина двовимірного фільтра, то алгоритм цього фільтра

представляється графом алгоритму згортки з відповідними затримками в дугах, що представляють буферні схеми [127].

В графі на рис. 2.2. вершина нелінійного низькочастотного фільтра F повинна зберігати краї зображення. Цей фільтр використовується для оцінки освітленості L . Тоді інформація про віддзеркалене освітлення R одержується шляхом ділення вхідного сигналу на L . Далі L і R обробляються у різних потоках за алгоритмами корекції яскравості Γ і покращення віддзеркаленого світла β . Після перетворення Γ компонент освітлення лінійно масштабується, щоб повністю покрити весь діапазон представлення вхідних даних ($0 - 2^{20}$ у разі 20-бітної вхідного пікселя), який відображається у діапазон результатів ($0 - 255$ у разі 8-бітної вихідного пікселя). Зрештою, ці два компоненти об'єднують шляхом множення для одержання результуючого зображення.

Як функцію Γ , яка залежить від параметра форми γ , часто використовують степеневу функцію (для діапазону представлення $0 - 255$). Після неї застосовується нормуюча функція S , яка віднімає рівень чорного кольору (близьке до нуля значення), підсилює яскравість сигналу та усикає його до рівня максимальної яскравості.

Досвід багатьох авторів, наприклад, в роботі [128], показав, що ефективними функціями для стиснення та покращення контрасту є

$$\Gamma(y) = K_{Mo}(y / K_M)^{\gamma(1+y/KM)}; \quad \beta(y) = (1 + e^{-b \cdot \log y})^{-1} + 0,5, \quad (2.5)$$

де K_M , K_{Mo} — динамічні діапазони представлення вхідного та вихідного сигналів, відповідно, що визначаються розрядністю пікселів, γ , b — коефіцієнти, що підбираються вручну, причому γ визначає коефіцієнт стиснення динамічного діапазону.

Функція $\Gamma(y)$ має велику крутизну для темних ділянок зображення та послаблює градієнт яскравості яскравих ділянок. Тому вона є основною для стиснення зображення. Функція $\beta(y)$ має велику крутизну для аргументу y в околі точки 1,0 та дуже похила для значень близьких до нуля та максимуму

аргументу. Тим самим вона підсилює різкі переходи яскравості у зображенні. Функція освітленості $F(I)$ у простому випадку є функцією двовимірного фільтра нижніх частот (ФНЧ), наприклад, Гаусового фільтра.

Алгоритм Retinex застосовується до складової яскравості кольорового зображення. Для обробки кольорових кадрів використовується окрема обробка у каналах кольорів (2.1). Але у результуючому зображенні виникає гало, якщо розкладання на складові R і L виконане неретельно. Для прецизійного виділення складової L необхідно використовувати не просто Гаусів фільтр, а фільтр, який зберігає краї об'єктів зображення [129].

Таким фільтром є білатеральний фільтр. Завдяки йому, у результуючому зображенні відсутній ефект гало. Білатеральний фільтр має ФНЧ-ядро яке адаптивно змінюється в залежності від характеру зображення. Цей фільтр обчислюється за формулою:

$$F(I(x_0, y_0)) = \frac{\sum_{\Omega} W_{\Omega} \cdot I(x, y) \cdot f(I(x, y) - I(x_0, y_0))}{\sum_{\Omega} f(I(x, y) - I(x_0, y_0))}, \quad (2.6)$$

де Ω — окіл пікселя з координатами (x_0, y_0) , апертура фільтра, $(x, y) \in \Omega$, W_{Ω} — ядро двовимірного ФНЧ, f — функція, яка досягає максимуму, якщо різниця яскравостей пікселів в околі і його центрального пікселя є максимальною, вираз у знаменнику є нормуючою функцією. Найчастіше W_{Ω} і f — результати фільтрації Гаусовим фільтром, а окіл має діаметр п'ять і більше пікселів [130].

Отже, для кожного пікселя у білатеральному фільтрі динамічно розраховується ядро фільтра. За основу береться ядро Гаусового фільтра W_{Ω} , коефіцієнти якого крім центрального зменшуються в залежності від локального контрасту. Як результат, дія білатерального фільтра є така, що, якщо зображення незначно змінюється в околі, то воно згладжується, а якщо є різкі переходи яскравості, то для них згладжування не виконується, бо відповідні числа ядра помножуються на близькі до нуля значення функції f .

Білатеральний фільтр (2.6) у методі Retinex є важким у обчисленнях, особливо якщо обробляється зображення з великим динамічним діапазоном. При цьому яскравість кожного пікселя слід нормувати діленням на нормуючий коефіцієнт, складність обчислення якого наближається до складності обчислення фільтра Гауса. Таким чином, для обчислення одного пікселя результату цього фільтра у апертурі Ω розміром $n \times n$ необхідно

$$Q_M = 4n^2, \quad Q_S = 4n^2 \quad (2.7)$$

операцій множення та додавання, відповідно, а також одна операція ділення. Оскільки вхідні дані є багаторозрядними (до 20 і більше біт) та для одержання одного результату слід виконати послідовно три множення, одне ділення і кілька додавань, обчислення слід виконувати з підвищеною точністю, бажано з плаваючою комою. Якщо обчислення виконувати з цілими числами, то через потрібне множення та результуюче ділення проміжні результати мають потрібну розрядність (до 60 і більше).

При використанні частотно-градієнтних операторів відображенню підлягає зображення з низькочастотних складових, а високочастотні складові залишаються без змін. У методі, описаному в [131], спочатку зображення розкладається на складові яскравості та кольорів, причому яскравість перетворюється за логарифмічною функцією. Далі яскравість фільтрується білатеральним фільтром подібним (2.4), але з операціями у логарифмічному масштабі. При цьому зображення, яке вміщує деталі одержується як віднімання вхідної яскравості від фільтрованої яскравості у логарифмічному представленні. Нарешті, до перетвореного зображення додається зображення з деталями у логарифмічному представленні. Далі відновлюються канали кольорів так, як у рівняннях (2.2).

Основна ідея побудови градієнтного алгоритму полягає в тому, що різкі переходи у яскравості зображення виконані як великі градієнти амплітуди з певною шкалою. З іншого боку, невеликі деталі формуються градієнтами з малою амплітудою. Тому стиснення HDR-зображення можна виконати,

перетворивши градієнти з великою амплітудою — у градієнти з малою, не зачіпаючи градієнти з малою амплітудою. При цьому формують піраміду зображень, за допомогою якої виділяються контури, які представляють собою оцінку градієнтів [132].

Ідея методу сегментації полягає у розділенні зображення на однорідні сегменти, які мають невеликий градієнт, обробити кожен сегмент окремо і відновити зображення об'єднанням сегментів. В роботі [133] зображення ділиться на регіони, в межах яких адаптується яскравість. Поділ на регіони виконується на основі гістограм розподілу яскравості у логарифмічному масштабі. Однорідні пікселі, що належать до однієї категорії, групуються за допомогою підходу заливки. Нарешті, маленькі групи пікселів захоплюються великими, формуючи шар. Сегментація виконується у межах шарів, які відрізняються рівнем освітленості. Для сформованого шару адаптована яскравість розраховується як середня логарифмічна яскравість пікселів у регіоні $L_a(x)$. Функція $L_a(x)$ допомагає зберегти краї зображення без виникнення гало. Але у алгоритмі необхідно завести велику кількість шарів, щоби не виникали артефакти, що суттєво збільшує його складність. Як результат — великі обсяги обчислень.

У методі [134] використовується властивість зору бути нечутливим до темних ділянок біля яскравих плям зображення. Тому воно розділяється на області, в межах яких є домінуючі яскравості — якорі. При цьому зображення розмивається за допомогою білатерального фільтра (2.4), який відфільтровує локальні шуми.

2.2.3. Висновки щодо алгоритмів перетворення тонів зображення

Серед алгоритмів обробки зображення високоякісне стиснення HDR-зображення відрізняється складністю та великим об'ємом обчислень. Причому у застосунках, таких як інтелектуальна відеокамера, ці обчислення необхідно виконувати у реальному часі.

Як локальні, так і частотно-градієнтні алгоритми основані на теорії Retinex, в основі якої є обробка зображення функцією локальної яскравості. Серед алгоритмів стиснення найкращі результати стиснення HDR-зображення одержуються за методом Retinex, у якому замість Гаусового фільтра на рис. 2.2 використовується фільтр, який є адаптивним до локального контрасту зображення і таким фільтром є білатеральний фільтр.

Білатеральний фільтр є важким у обчисленнях (2.6), (2.7), тому що ядро фільтра формується динамічно і результат фільтрації слід ділити на нормуючий коефіцієнт, який також розраховується динамічно для кожного пікселя.

Для розробки методу виявлення характерних точок та його подальшої апаратної реалізації необхідно створити модифікацію методу стиснення Retinex, яка б мала меншу складність.

2.3. Алгоритм адаптивної фільтрації HDR-зображення

2.3.1. Основа алгоритму

Як визначено вище, для ефективного стиснення HDR-зображення необхідне використання білатерального фільтра (2.4), який є складним у реалізації (2.5). Причому мета застосування цього фільтра – підкреслювати локальні ділянки зображення, які мають контрастні деталі та згладжувати зображення, відфільтровувати шуми у інших ділянках.

Для досягнення такої самої мети пропонується адаптивний фільтр з поточним графом на рис. 2.3.

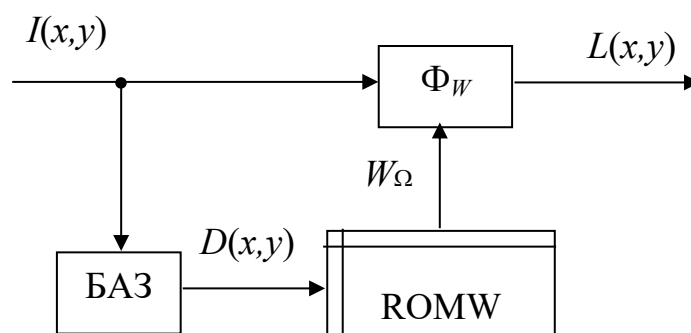


Рисунок 2.3 – Поточковий граф алгоритму адаптивного фільтра HDR-зображення

В цьому фільтрі вершина блоку Φ_W виконує згортку сигналу $I(x,y)$ з ядром фільтра W_Ω , коефіцієнти якого поступають з таблиці коефіцієнтів ROMW. Блок аналізу зображення (БАЗ) аналізує окіл зображення Ω навколо пікселя $I(x,y)$ і видає код характеристики цього околу $D(x,y)$, який служить адресою вибору ядра W_Ω з таблиці ROMW. Таким чином, даний адаптивний фільтр виконує обчислення

$$F(I(x_0, y_0)) = \sum_{\Omega} W_{\Omega}(D(x, y)) \cdot I(x, y) \quad (2.8)$$

Не приймаючи до уваги складність блоку БАЗ, алгоритм, що пропонується, вчетверо простіший за складністю за алгоритм білатерального фільтра (2.4).

Для завершення формування адаптивного алгоритму слід визначитись зі множиною примітивних образів, які виявляє блок БАЗ, алгоритмом детектування та множиною ядер фільтрів W_Ω .

2.3.2. Адаптивна фільтрація HDR-зображення

У підрозділі 1.3.1 описані детектори характерних особливостей низького рівня. Серед них набув поширення за свою надійність детектор Гарріса. Але він відзначається високою складністю свого обчислення, яка збільшується через те, що пікселі $I(x,y)$ є багаторозрядними. Такий детектор можна замінити на детектор Гарріса-Лапласа [135]. Цей детектор виконаний як двовимірний

фільтр з ядром оператора Лапласа, який є чутливим до того чи іншого примітивного образу.

Як свідчать роботи, присвячені CNN, множина примітивних образів може складати до кількох десятків, а діаметр апертури варіюється від трьох до дев'яти [23, 24]. Для алгоритму, що пропонується, вибрано такі примітивні образи, як кругла пляма, вертикальна, горизонтальна та дві діагональні лінії.

Як і в білатеральному фільтрі, вихід фільтра Лапласа слід нормувати з урахуванням середнього рівня освітленості, який обчислюється фільтром Гауса з ядром W_G .

Отже, детектор примітивних образів повинен мати p двовимірних фільтрів, які паралельно обробляють HDR-зображення. З метою спрощення цих фільтрів коефіцієнти їхніх ядер слід вибрати примітивними, тобто, такими, що належать до множини $(0, 1, 2, 3, 4, 5, 6)$. Тоді сумарна складність адаптивного фільтра буде малою.

Результатом аналізу БАЗ має бути малорозрядний код. Отже, цей блок має виконувати стиснення даних у кілька разів. Для такого стиснення підходящим є логарифмування даних, яке використовується, наприклад, в роботі [131]. Тоді виходом i -го детектора є

$$D_i = \log \left| \sum_{\Omega} W_i(x,y) \cdot I(x,y) \right|, i \in [0, p-1]. \quad (2.9)$$

А оцінкою локальної яскравості є

$$D_G(x,y) = \log \sum_{\Omega} W_G(x,y) \cdot I(x,y). \quad (2.10)$$

Операція абсолютної величини введена в (2.7) для того, щоб була можливість логарифмування від'ємних чисел. Слід відмітити, що у алгоритмі SIFT пошук характерних точок також виконується як максимум абсолютної величини значення виходу детектора.

Отже, кожен детектор видає число D_i , яке є значенням логарифму. Серед цих чисел вибирається максимальне значення, від якого з метою нормування віднімається логарифм оцінки локальної яскравості:

$$D(x,y) = \max_i (D_i) - D_G(x,y) + D_0, \quad (2.11)$$

де D_0 — нормуючий доданок, який приводить результати у діапазон $D(x,y) \in [0, k-1]$, у якому k — кількість ядер W_Ω у таблиці ROMW для i -го образу.

Таким чином, виходами БАЗ є номер визначеного образу P_N та логарифмічна оцінка його яскравості $P_S = \lfloor D(x,y) \rfloor$, де “ \lfloor ” — операція усікання до цілого. Параметри P_N та P_S подаються на адресні входи ROMW для вибору відповідної таблиці коефіцієнтів ядра фільтра $W_\Omega(P_N, P_S)$. На рис. 2.4 показаний умовний вигляд ядер фільтра для образу діагональної лінії при $P_S = 1, \dots, 5$. При цьому інтенсивність кольору є пропорційною значенню коефіцієнта.

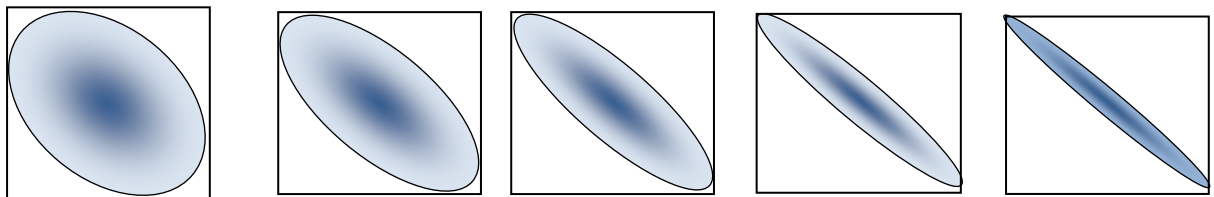


Рисунок 2.4 – Ядра фільтра $W_\Omega(P_N, P_S)$ для різних значень P_S

Таким чином, якщо піксель $I(x,y)$ знаходиться в околі з рівномірною освітленістю, то його яскравість осереднюється з пікселями оточення з результуючою фільтрацією шумів. Якщо БАЗ визначає, що цей піксель належить, наприклад, до діагональної лінії, то його яскравість осередниться лише з пікселями, що належать цій лінії, причому чим інтенсивніше проявляється ця лінія — чим більше число P_S — тим менше оточуючих пікселів приймають участь у осередненні і тим менше буде розмиття зображення, включаючи дану лінію.

2.3.3. Алгоритм адаптивної фільтрації для обробки HDR-зображень

Для перевірки алгоритму детектування примітивних образів та адаптивної фільтрації HDR-зображення в цілому було розроблено програму адаптивного фільтру. Програма написана мовою Java і використовує типи даних, які відповідають алгоритму. Потоковий граф алгоритму адаптивного фільтру показаний на рис. 2.5.

Для адаптивного фільтру вибрано такі примітивні образи, як кругла пляма, вертикальна, горизонтальна та дві діагональні лінії, тобто, $p = 5$. Ці образи виявляються за допомогою ядер згортки W_* , W_{\downarrow} , W_{\uparrow} , W_{\backslash} , $W_{/}$, відповідно. Розрядність вхідних даних дорівнює 20. Діаметр апертури вибрано 5, хоча для спрощення можливо вибрати і 3. Коефіцієнти фільтрів детекторів адаптивного фільтра представлені у таблиці 2.1.

Отже, для множення на ці коефіцієнти у переважній кількості випадків слід множене зсунути на 1, 2 чи 3 розряди, а іноді – один раз скласти зсунуте множене при множенні на 3, 5, 6, 12 чи 24.

Обчислення логарифму виконується за формулою

$$D_i = F_{\log}(2^p \cdot |I_{Li}(x, y)|) + p, \quad (2.12)$$

де $I_{Li}(x, y)$ — результат фільтрації i -м фільтром Лапласа, p — кількість нульових розрядів перед старшою значущою одиницею абсолютної величини $|I_{Li}(x, y)|$, F_{\log} — таблична функція логарифму за основою два від старших розрядів нормалізованого числа $|I_{Li}(x, y)|$.

Вершина графу Мах виконує вибір максимального значення логарифму D_M та супроводжує його номером N_M детектора, якому належить результат. За формулою (2.9) від D_M віднімається логарифм оцінки локальної яскравості $D_G(x, y)$ та додається нормуючий доданок D_0 . Результируючий код D_I приводиться до діапазону $[0, 7]$ за допомогою зсуву і операції насичення.

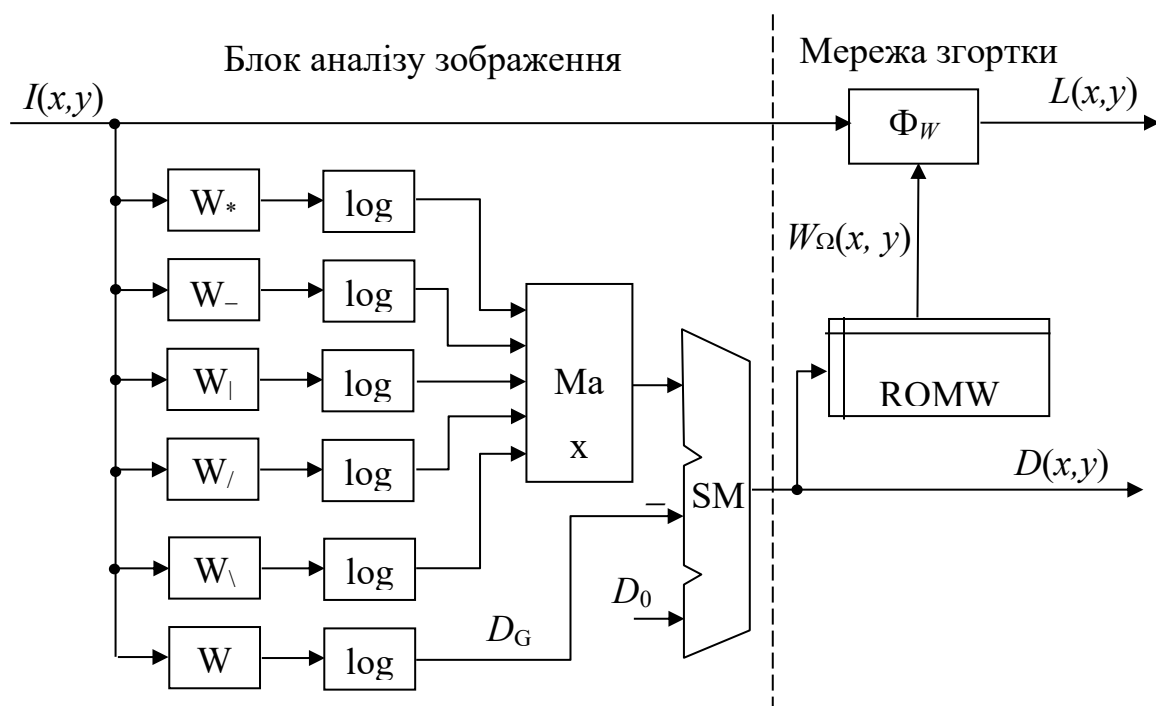


Рисунок 2.5 – Поточковий граф алгоритму адаптивного фільтру

Таблиця 2.1 – Коефіцієнти фільтрів детекторів адаптивного фільтру

Ядро фільтра	Коефіцієнт підсилення S	Коефіцієнти фільтра	Кількість операцій, Q_s
Гауса з $\sigma = 1,6$	61	1, 2, 2, 2, 1, 2, 3, 4, 3, 2, 2, 4, 5, 4, 2, 2, 3, 4, 3, 2, 1, 2, 2, 2, 1	26
Лапласа для пліями	96	- 3, -3, -3, -3, -3 - 3, 0, 6, 0, -3, - 3, 6, 24, 6, -3 - 3, 0, 6, 0, -3, - 3, -3, -3, -3, -3	21
Лапласа для го- ризонтальної лінії	96	1, 4, 6, 4, 1, 2, 8, 12, 8, 2, 0, 0, 0, 0, 0, -2, -8, -12, -8, -2, -1, -4, -6, -4, -1	21
Лапласа для похилої лінії	96	0, 2, 4, 4, 4, -2, 0, 8, 8, 4, -4, -8, 0, 8, 4, -4, -8, -8, 0, 2,	19

		-4,-4, -4, -2, 0	
--	--	------------------	--

Отже, піксель представляє собою вектор

$$D(x,y) = (D_I(x,y), N_M(x,y)) . \quad (2.13)$$

Коди $D_I(x,y)$ та $N_M(x,y)$ подаються на адресні входи таблиці ядер фільтрів ROMW і вибирають в ній одне з 40 ядер, які поділені на 5 груп. Ці коди формують специфічне зображення ознак $D(x,y)$ (рис.2.5). Приклади ядер, що використовуються, показані у таблиці 2.2.

Таблиця 2.2. Коефіцієнти фільтрів, що зберігаються в ROMW

Фільтр для об- разу	Коефіцієнти для розмиття		
	найбільшого, $D(x,y) = 0$	середнього $D(x,y) = 4$	найменшого $D(x,y) = 7$
плями	8, 15, 18, 15, 8, 15, 27, 33, 27, 15, 18, 33, 40, 33, 18, 15, 27, 33, 27, 15, 8, 15, 18, 15, 8,	0, 4, 7, 4, 0, 4, 29, 57, 29, 4, 7, 57, 108, 57, 7, 4, 29, 57, 29, 4, 0, 4, 7, 4, 0,	0, 0, 0, 0, 0, 0, 0, 3, 0, 0, 0, 3, 456, 3, 0, 0, 0, 3, 0, 0, 0, 0, 0, 0, 0
горизонтальної лінії	8, 15, 18, 15, 8, 15, 27, 33, 27, 15, 18, 33, 40, 33, 18, 15, 27, 33, 27, 15, 8, 15, 18, 15, 8,	1, 3, 3, 3, 1, 15, 27, 33, 27, 15, 33, 59, 72, 59, 33, 15, 27, 33, 27, 15, 1, 3, 3, 3, 1,	0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 61, 108, 130, 108, 61, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
діагональної лінії	18, 21, 10, 4, 0, 21, 43, 35, 13, 4, 10, 35, 58, 35, 10, 4, 13, 35, 43, 21, 0, 4, 10, 21, 18	27, 12, 0, 0, 0, 12, 92, 23, 0, 0, 0, 23, 134, 23, 0, 0, 0, 23, 92, 12, 0, 0, 0, 0, 27	39, 0, 0, 0, 0, 0, 103, 0, 0, 0, 0, 0, 184, 0, 0, 0, 0, 0, 103, 0, 0, 0, 0, 0, 39

Таким чином, якщо зображення у околі пікселя $I(x, y)$ не має великого градієнту яскравості, то воно згладжується у результуючому зображенні $L(x, y)$ сильніше, ніж зображення з великим градієнтом. Причому градієнтне зображення сильно згладжується поперек градієнта, наприклад, вздовж лінії та майже не згладжується вздовж вектору градієнта. Завдяки цьому, відфільтроване зображення має розмиття крім областей з краями.

Слід оцінити складність алгоритму. При апертурі Ω розміром $n \times n = 5 \times 5$

$$Q_M = 25 = n^2, \quad Q_S = 153 = 6,12n^2. \quad (2.14)$$

Порівнюючи з (2.5), можна зробити висновок, що у запропонованому алгоритмі адаптивної фільтрації учетверо менше операцій множення при збільшенні числа додавань у півтора рази. Крім того, додатковими апаратними витратами є шість ПЗП табличної функції логарифму на 16 слів та ПЗП ROMK об'ємом $40 \times 25 = 1000$ слів. Також необхідно виконати 6 операцій нормалізації числа. Слід нагадати, що операції у білатеральному фільтрі виконуються з плаваючою комою і потребують додаткову операцію ділення багаторозрядних чисел. А у запропонованому фільтрі розрядність проміжних і остаточних результатів з фіксованою комою не набагато перевищує розрядність вхідних даних.

Вказані характеристики фільтра та особливості його алгоритму свідчать про те, що його доцільно реалізувати у ПЛІС.

2.3.4. Алгоритм компресії HDR-зображення

Згідно з графом алгоритму HDR-компресії (рис. 2.3) крім адаптивного фільтру алгоритм має виконувати операції ділення, обчислення функцій корекції яскравості Γ і покращення віддзеркаленого світла β , а також перемноження їх результатів.

Блок ділення виконує операцію

$$R(x,y) = \frac{I(x,y)}{L(x,y)}, \quad (2.15)$$

де $L(x,y)$ — значення оцінки яскравості з виходу адаптивного фільтру. Причому, операнди масштабуються і коректуються таким чином, щоби результат належав діапазону $(0 - 8)$ і було відсутнє ділення на нуль.

Блок обчислення бета-функції виконує обчислення за формулою (2.6). Вони реалізовані за допомогою кусково-лінійної інтерполяції

$$Y = a(x) * x + b(x), \quad (2.16)$$

Тут коефіцієнти $a(x)$, $b(x)$ для прийнятної точності досить задати на десяти відрізках аргументу x . Вхідний сигнал $x = R(x,y)$ є оцінкою відносної яскравості світла, віддзеркаленого від об'єкта. Тобто, при $R(x,y) \approx 1$ сигнал

представляє особливості фактури об'єкта. Як видно з графіка функції на рис. 3.9, бета-функція підсилює сигнал $R(x,y)$ у діапазоні $(0,5 - 1,5)$ та придушує його у інших діапазонах. Таким чином, бета-функція підсилює локальний контраст і дає вихідний сигнал, що змінюється у межах $(0,5 - 1,5)$.

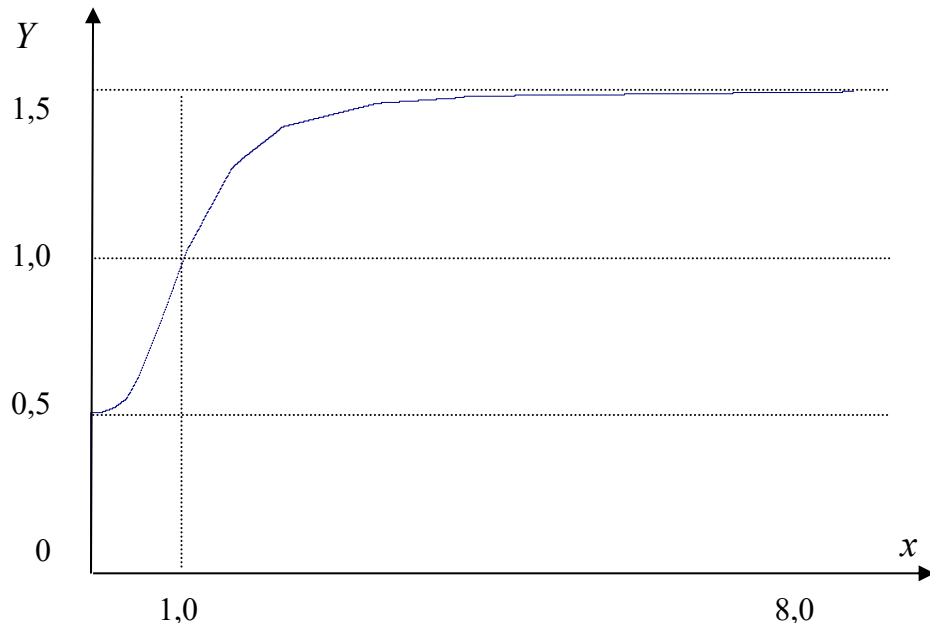


Рисунок 2.6 – Графік бета-функції

Блок обчислення гама-функції виконує обчислення за формулою (2.4). Він виконує, як і блок обчислення бета-функції, алгоритм кусково-лінійної інтерполяції. Графік цієї функції показано на рис. 2.7.

Графік гама-функції представлено у логарифмічному масштабі. Ця функція стискає сигнал оцінки яскравості $R(x,y)$, який представлено 18-розрядними числами до вихідного сигналу, представленого 8-розрядними даними.

Після помноження сигналів $R'(x,y)$ і $L'(x,y)$ з виходів блоків бета- та гама-корекції одержується стиснутий чорно-білий 8-розрядний сигнал

$$I'(x,y) = R'(x,y) \cdot L'(x,y). \quad (2.17)$$

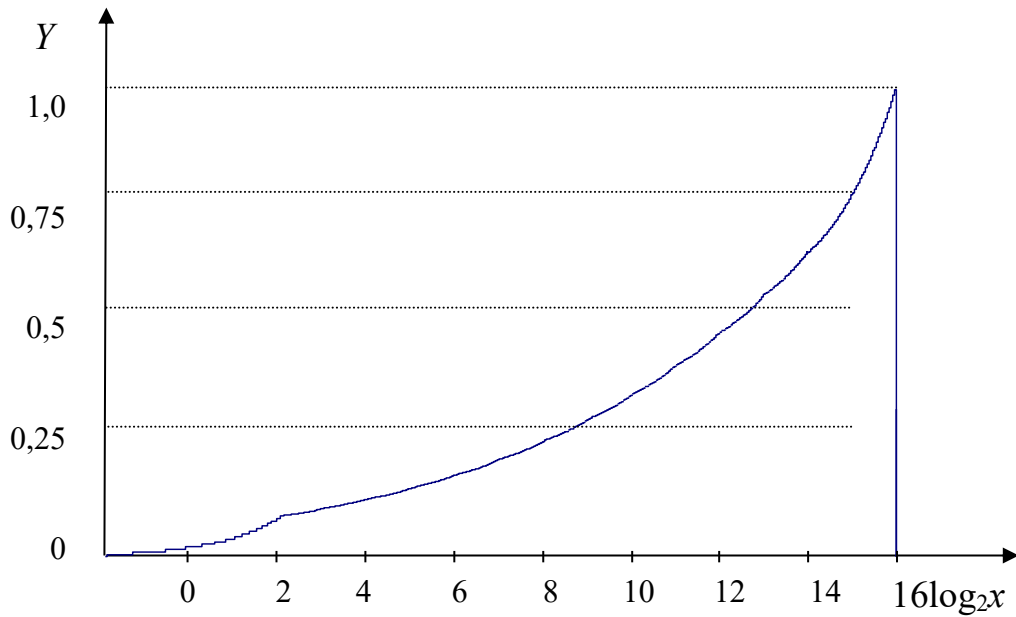


Рисунок 2.7 – Графік гамма-функції

Даний алгоритм було випробувано за допомогою розробленого програмного комплексу для обробки та дослідження зображень, який запрограмовано мовою Java. Вхідні дані приймалися з відеодатчика MT9M024, який забезпечує одержання HDR-зображення з динамічним діапазоном 120 дб, тобто з 20-розрядним сигналом $I(x,y)$. Відеодатчик входить у склад плати Helion-60 фірми Helionvision, на якій встановлена ПЛІС Lattice ECP-3 LFE3-70EA [136]. Ця плата виконує прийом зображення з відеодатчика, виділення сигналу яскравості $I(x,y)$, передачу його на дисплей і у комп'ютер.

На рис. 2.8 показана фотографія сцени з широким динамічним діапазоном, що обробляється. На плафоні лампи прикріплене слово “HDR”, яке може бути прочитане лише після HDR-стиснення. На рис. 2.9 показане зображення сигналу яскравості $I(x,y)$. Причому видно, що воно не здатне відтворити високий рівень яскравості. Як результат обробки сигналу яскравості $I(x,y)$ блоком аналізу зображення виступає зображення ознак $D(x,y)$ (рис. 2.10).

Зображення сигналу яскравості $I(x,y)$ після обробки адаптивним фільтром та перетворення гамма-функцією показане на рис. 2.11.



Рисунок 2.8 – Зображення сцени для дослідів з HDR-компресією

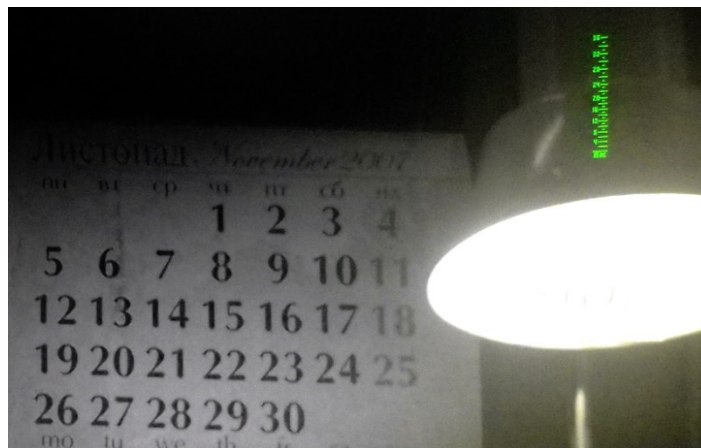


Рисунок 2.9 – Складова яскравості зображення $I(x,y)$

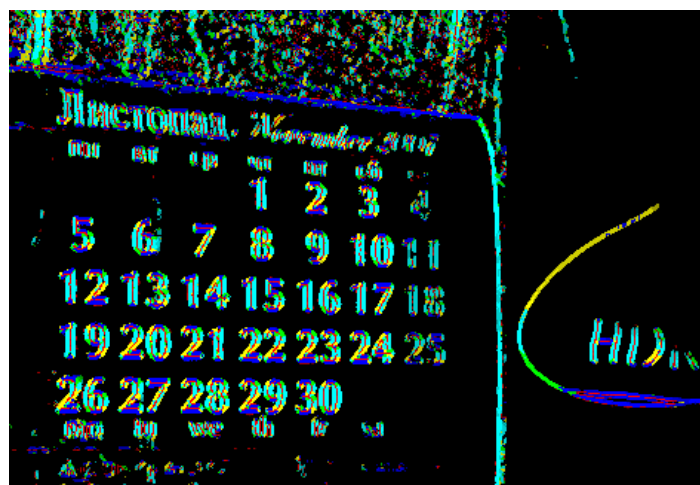


Рисунок 2.10 – Зображення ознак $D(x,y)$

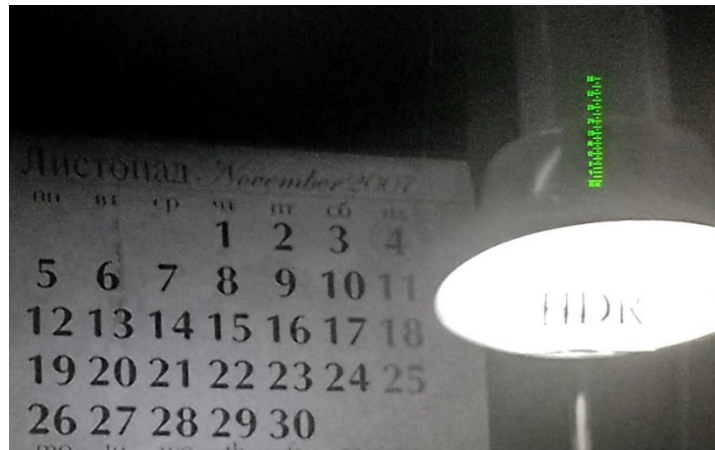


Рисунок 2.11 – Зображення $L(x,y)$ після обробки гама-функцією

Вхідне чорно-біле зображення $I(x,y)$ попіксельно ділиться на оцінку локальної яскравості $L(x,y)$ і результат $R(x,y)$ обробляється бета-функцією. Результат такої обробки показано на рис. 2.12. Як видно з рис. 2.12, результатом такої обробки є виділені деталі зображення як у темних, так і у світлих місцях. Причому ці деталі мають приблизно однаковий контраст та чіткі контури.

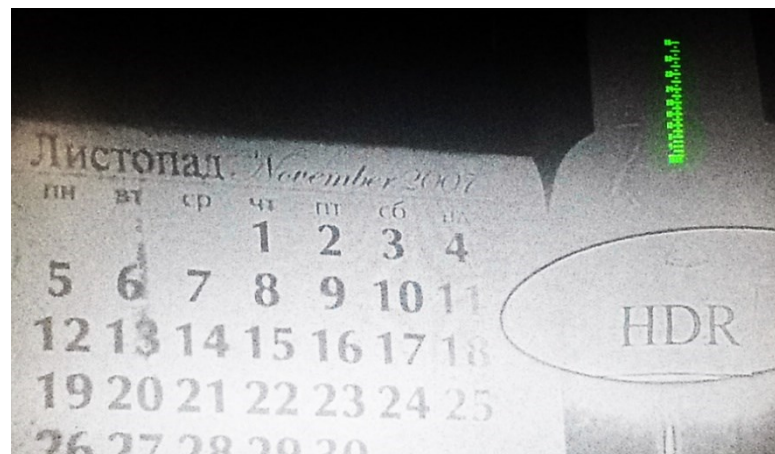


Рисунок 2.12 – Зображення після обробки бета-функцією

Зображення після бета- та гама-обробки перемножуються, даючи результуюче стиснуте зображення, таке, як на рис. 2.13. При цьому, завдяки множенню на зображення після бета-обробки, покращується чіткість деталей, особливо на яскравих ділянках. Крім того, ці деталі (літери, цифри) мають

рівномірно освітлений окіл, тобто, вони не мають ефекту гало.



Рисунок 2.13 – Стиснуте чорно-біле зображення

Таким чином, результуюче стиснуте HDR-зображення має чіткі деталі як у темних, так і надмірно освітлених місцях. Отже, пошук характерних точок у такому зображенні та розпізнавання образів в цілому значно покращується.

Крім того, для пошуку характерних точок підходящими є зображення ознак $D(x,y)$ (рис. 2.10) та зображення $R(x,y)$ після обробки бета-функцією (рис. 2.11).

2.3.5. Висновки щодо алгоритму компресії HDR-зображення

Модернізовано алгоритм компресії HDR-зображення методом Retinex, у якому білатеральний фільтр замінено на вперше створений адаптивний фільтр оцінки яскравості зображення. Запропонований адаптивний фільтр має учетверо менше операцій множення, менші вимоги до точності представлення проміжних даних і придатний для стиснення зображення з динамічним діапазоном 120 дБ і більше.

Модернізований алгоритм дає змогу покращити пошук характерних точок у зображенні з несприятливими умовами освітленості. Причому для такого пошуку придатне як HDR-зображення після стиснення, так і проміжні результати обчислення, такі як зображення ознак $D(x,y)$ чи зображення $R(x,y)$ після

обробки бета-функцією.

Слід висунути гіпотезу, що на основі зображення ознак, яке представляється малорозрядними кодами $D(x,y)$ (близько 6 розрядів) можна побудувати новий метод пошуку характерних точок у зображенні. Доведення цієї гіпотези викладено у наступному підрозділі.

2.4. Метод пошуку характерних точок на основі зображення ознак

2.4.1. Особливості зображення ознак

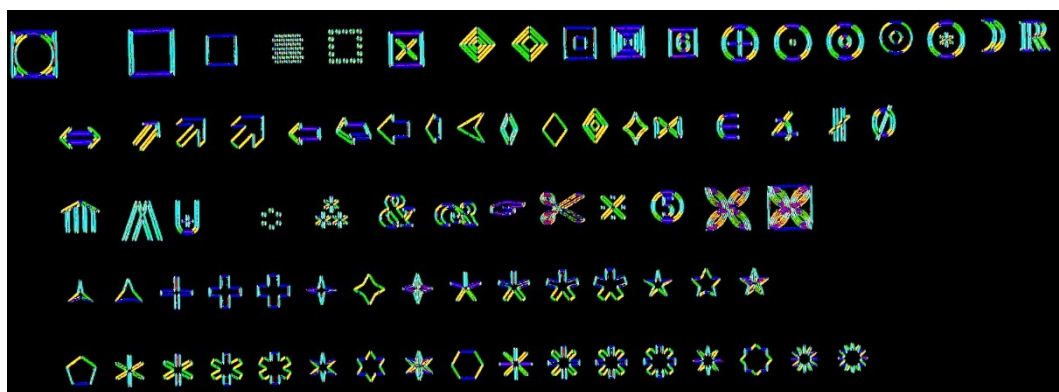
Результатами обробки блоком аналізу зображення адаптивного фільтра $D(x,y)$ є зображення, пікселі якого кодують розпізнану локальну ознаку та її інтенсивність у логарифмічному масштабі. Для візуалізації такого зображення використовується наступне кодування локальних ознак:

- горизонтальний край — синій (позначається як В),
- вертикальний край — блакитний (С),
- похилий край — жовтий (Y) чи зелений (G),
- крапка, пляма — червоний (R),
- рівномірний фон (відсутність локальних ознак) — чорний (K).

На рис. 2.14 показане тестове зображення і результат його обробки $D(x,y)$. Далі, на рис. 2.15 — 2.25 показані типові фрагменти зображення після обробки аналізатором. Причому, на рис. 2.16 — 2.23, а також на рис. 2.14, 2.15, де показані фрагменти лінії і краю з великою кривизною показані елементи, які мають бути розпізнані як характерні точки зображення.



a)



б)

Рисунок 2.14 – Тестове зображення (а) і його результат обробки (б)

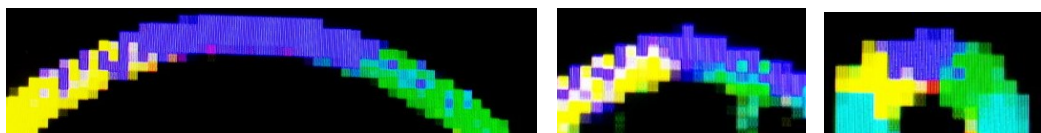


Рисунок 2.15 – Краї форми

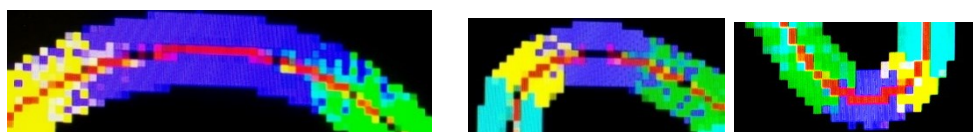


Рисунок 2.16 – Тонкі лінії

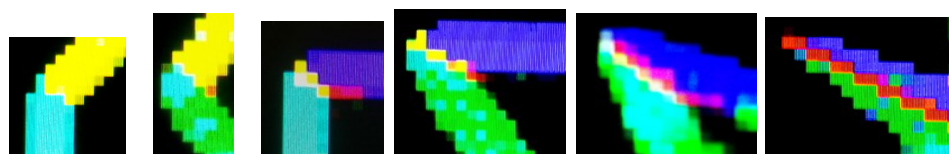


Рисунок 2.17 – Кути форми

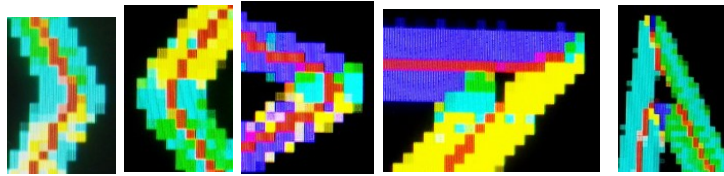


Рисунок 2.18 – Кути тонких ліній

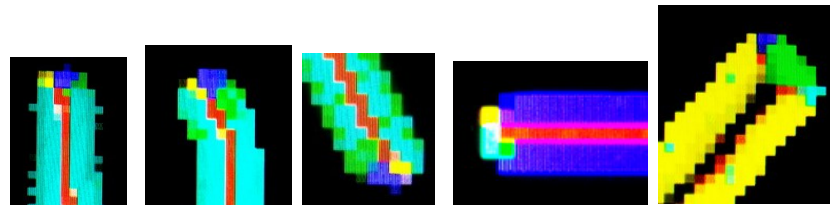


Рисунок 2.19 – Кінці тонких ліній

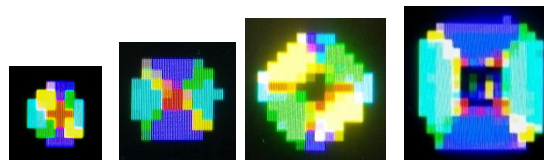


Рисунок 2.20 – Крапки і плями

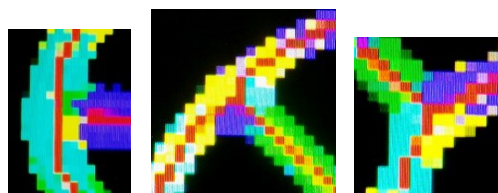


Рисунок 2.21 – Т-подібний перетин тонких ліній

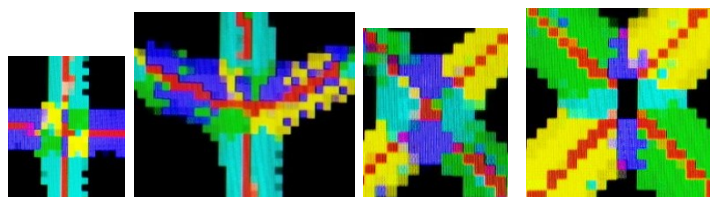


Рисунок 2.22 – Х-подібний перетин тонких ліній



Рисунок 2.23 – Торкання тонкої лінії і края форми

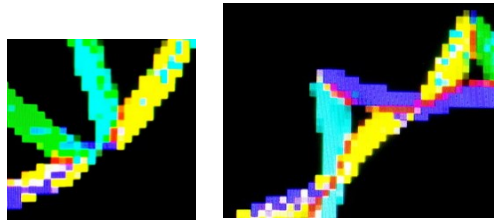


Рисунок 2.24 – Торкання двох країв форм

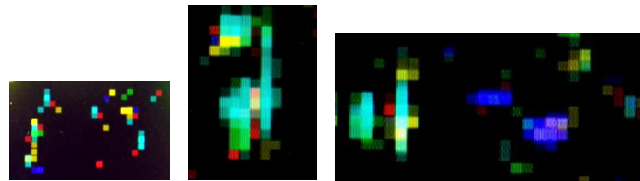


Рис. 2.25 – Шум

Аналіз зображень на рис. 2.15 — 2.25 дозволяє зробити наступні висновки.

В околі характерної точки трапляються пікселі двох, трьох і більше кольорів великої амплітуди. Пікселі червоного кольору (R) знаходяться в центрі характерної точки, вони найчастіше формують центр плями або позначають середину тонкої лінії. Пікселі інших кольорів формують ділянки з однорідним заповненням.

В околі характерних точок, таких як кут, лінія з великою кривизною зустрічаються пікселі споріднені кольорів. Це такі пари, як (Y, B), (B, G), (G, C), (C, Y), (Y, G) та (C, B). Причому кут нахилу краю знаходиться у діапазоні $45 \pm 10^\circ$ до горизонталі чи вертикалі. Якщо край чи лінія знаходяться під кутом нахилу у діапазоні $22,5 \pm 10^\circ$, то в ній трапляється суміш пікселів спорідненого кольору, як на рис. 2.15.

На відміну від тонкої лінії, край форми не має пікселів кольору R. А товста лінія розглядається як форма.

Пікселі шуму сильно впливають на визначення характерної точки. Вони мають суттєво меншу амплітуду ніж пікселі, що належать характерним точкам. Трапляються одиночні пікселі шуму великої амплітуди.

Пікселі шуму з великою амплітудою не формують ділянки одного кольору. А пікселі інших кольорів формують ділянки з однорідним заповненням. Ці властивості можна використати у морфологічному фільтрі для фільтрації шуму у зображенні, який розглядається далі.

2.4.2. Алгоритм фільтрації зображення ознак

Результатом обробки блоком аналізу зображення є зображення $D(x,y)$, пікселі якого кодують розпізнану локальну ознаку та її інтенсивність. Але у ньому часто трапляються пікселі шуму завдяки тому, що оператори Лапласа підсилюють рівень шуму через те, що вони обчислюють похідну двовимірного сигналу. Отже, для покращення розпізнавання характерних точок у зображенні $D(x,y)$ слід відфільтрувати шуми.

Метою фільтрації зображення $D(x,y)$ є його покращення задля спрощення подальшого розпізнавання характерних елементів. Вимоги для такої фільтрації:

- мінімізація шумів, таких як на рис.2.25;
- зменшення числа пікселів з локальними ознаками (утончення країв ліній);
- збереження взаємного положення пікселів локальних ознак у місцях, характерних елементів зображення.

Оскільки пікселі зображення $D(x,y)$ мають логарифмічний масштаб, поширені лінійні методи фільтрації шумів, наприклад, фільтрації верхніх частот фільтром Гауса, не підходять.

Аналіз рис. 2.15 —2.24 показує наступне:

- після фільтрації пікселі кольору R повинні зберігатись, якщо у їхній локальній області є інші пікселі R приблизно такої самої інтенсивності;
- пікселі кольорів G, B, C, Y повинні зберігатись, якщо вони належать до поля зображення такого самого кольору і приблизно такої самої інтенсивності;

— те саме стосується випадку, коли у цьому полі – пікселі спорідненого кольору (пари (Y,B), (B,G), (G,C), (C,Y), (Y,G) та (C, B));

— пікселі K повинні зберігатись і після фільтрації їх повинно ставати більше; якщо у околі якогось піксель переважно пікселі K чи пікселі різних кольорів, чи пікселі з дуже відмінною інтенсивністю, то такий піксель слід замінити на K.

Для фільтрації слід використати метод максимуму однорідності зображення у околі (maximum homogeneity neighbor, MHN-filter) [137]. Ідея такої фільтрації полягає в тому, що піксель зображення не є шумом, якщо він належить до локальної області, в якій знаходяться пікселі такого самого кольору і приблизно такої самої яскравості. Те, що піксель належить до такої області, визначається за тим, що він належить до певного шаблону, який покриває пікселі такого самого кольору.

Отже, фільтр має бути адаптивним. Його модифікація повинна вибиратись у залежності від кольору пікселя, що обробляється. Далі розглядаються ці модифікації.

Алгоритм MHN-фільтрації для пікселів кольору R.

1. Піксель залишається, коли знайдеться будь-який шаблон з показаних на рис. 2.26, а також йому аналогічний шаблон, який одержаний поворотом на 90, 180 і 270° і віддзеркалюванням відносно вертикалі. При цьому усі три пікселі — кольору R.

2. Знаходяться різниці інтенсивності

$$\Delta_1 = |I_1 - I_2|; \Delta_2 = |I_1 - I_3|; \Delta_3 = |I_3 - I_2|,$$

де I_i — складова яскравості пікселя $D(x,y)$. Піксель залишається, якщо $\Delta_i < \Delta$, $i = 1, 2, 3$, де поріг $\Delta \approx 1$ — підбирається при настроюванні. При застосуванні логарифму за основою 2, $\Delta = 1$ означає, що інтенсивності яскравості ознаки у пікселях відрізняються удвічі.

3. Інакше піксель замінюється на піксель кольору К.

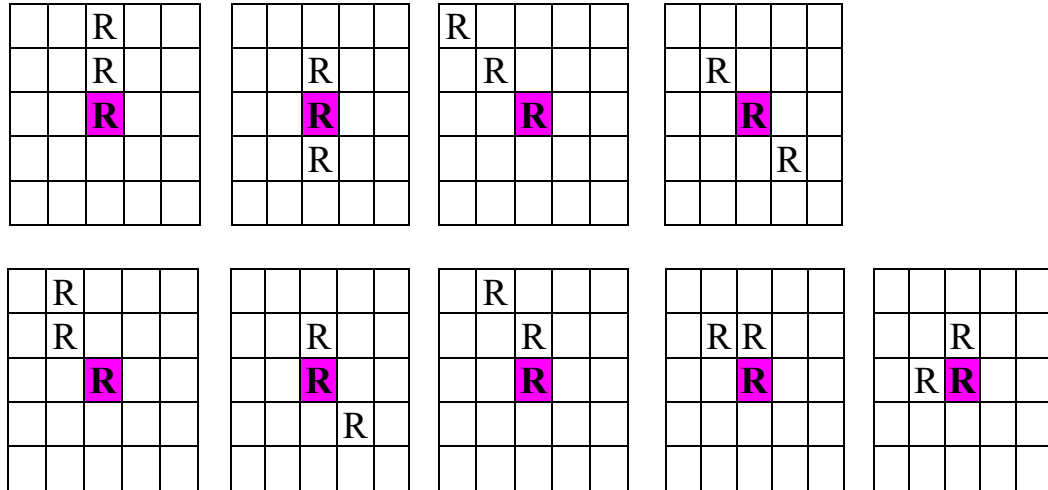


Рисунок 2.26 – Шаблони фільтра пікселів кольору R

У фільтрі для пікселів кольорів G,B,C,Y, як і у попередньому випадку, застосовуються певні шаблони, які показані на рис. 2.26.

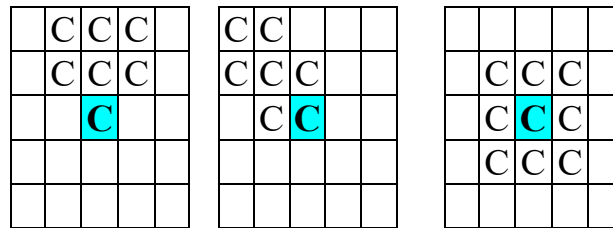


Рисунок 2.27 – Шаблони фільтрів для пікселів кольорів G,B,C,Y

Алгоритм MHN-фільтрації для пікселів кольорів G,B,C,Y.

1. Вибирається один з шаблонів як на рис. 2.26 чи повернутий на 90° або віддзеркалений відносно вертикалі чи горизонталі. Підраховується число пікселів, які попадають у шаблон і які співпадають з кольором центрального пікселя $N_{Ц}$ та число пікселів спорідненого кольору N_C .

Якщо $N_{Ц} + N_C < 6$, то вибирається наступний шаблон.

2. Якщо $N_{Ц} + N_C \geq 6$, то якщо $N_{Ц} \geq N_C$, то колір пікселя залишається тим самим, інакше – замінюється на споріднений.

3. Знаходяться різниці інтенсивності $\Delta_i = |I - I_i|$. Якщо якась з різниць $\Delta_i < \Delta$, то піксель замінюється на К.

4. Якщо шаблон не знайдений, то піксель замінюється на піксель кольору тла K .

Запропонований фільтр запрограмований мовою Java і застосований для ряду зображень $D(x,y)$. Деякі фрагменти зображення до і після фільтрації показані на рис. 2.28.

Порівняльна оцінка зображень на рис. 2.27 показує, що шуми фільтруються особливо ефективно на тлі зображення, тобто, на тих ділянках, де у вхідному зображенні відсутній суттєвий градієнт яскравості. При цьому краї та тонкі лінії зображення покращують свій вигляд — стають тоншими і чіткішими. У ділянках, де є незначний випадковий градієнт яскравості, шум відфільтровується гірше. Це наприклад, ділянки дещо нерівномірно зафарбованої форми або з дрібнозернистою фактурою. Але він залишається з малою амплітудою і може бути видалений відсіканням. Якщо вхідне зображення згладжується, наприклад, Гаусовим фільтром, то ці шуми відфільтровуються також, як це вже доведено у реалізаціях методу SIFT.

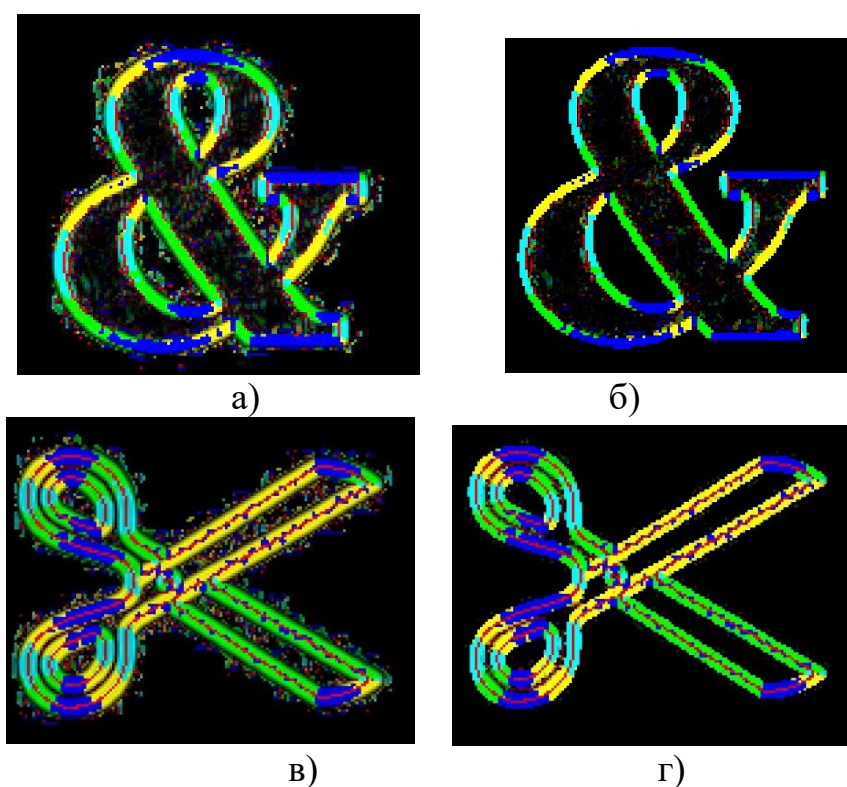


Рисунок 2.28 – Зображення ознак до (а), (в) і після МНН-фільтрації (б), (г)

Слід зауважити, що фільтри зображення, що побудовані за методом МНН, є нелінійними морфологічними фільтрами. Для таких фільтрів не існує математичного базису, такого як для лінійних фільтрів, згідно з яким можливо аналітично передбачити чи розрахувати характеристики того чи іншого фільтра. Тому перевірка такого фільтра виконується при виконанні його алгоритму з реальними даними з порівнянням якості зображення до і після фільтрації або з використанням суб'єктивних критеріїв якості.

2.4.3. Алгоритм побудови піраміди зображень ознак

Результатом обробки блоком аналізу зображення є зображення ознак у одному масштабі $D(x,y)$. Для одержання дескриптора характерної точки, який є інваріантним до масштабу, слід шукати характерні точки у зображеннях з різними масштабами. Досвід багатьох авторів при застосуванні методу SIFT свідчить про те, що достатньо аналізувати піраміду зображень, в якій зображення мають три масштаби.

Отже, у методі, що розробляється слід розробити алгоритм одержання зображень ознак з кількома масштабами, наприклад, $D_1(x,y)$, $D_2(x,y)$, $D_4(x,y)$. Для цього слід використати потоковий граф алгоритму, такий як на рис. 2.29.

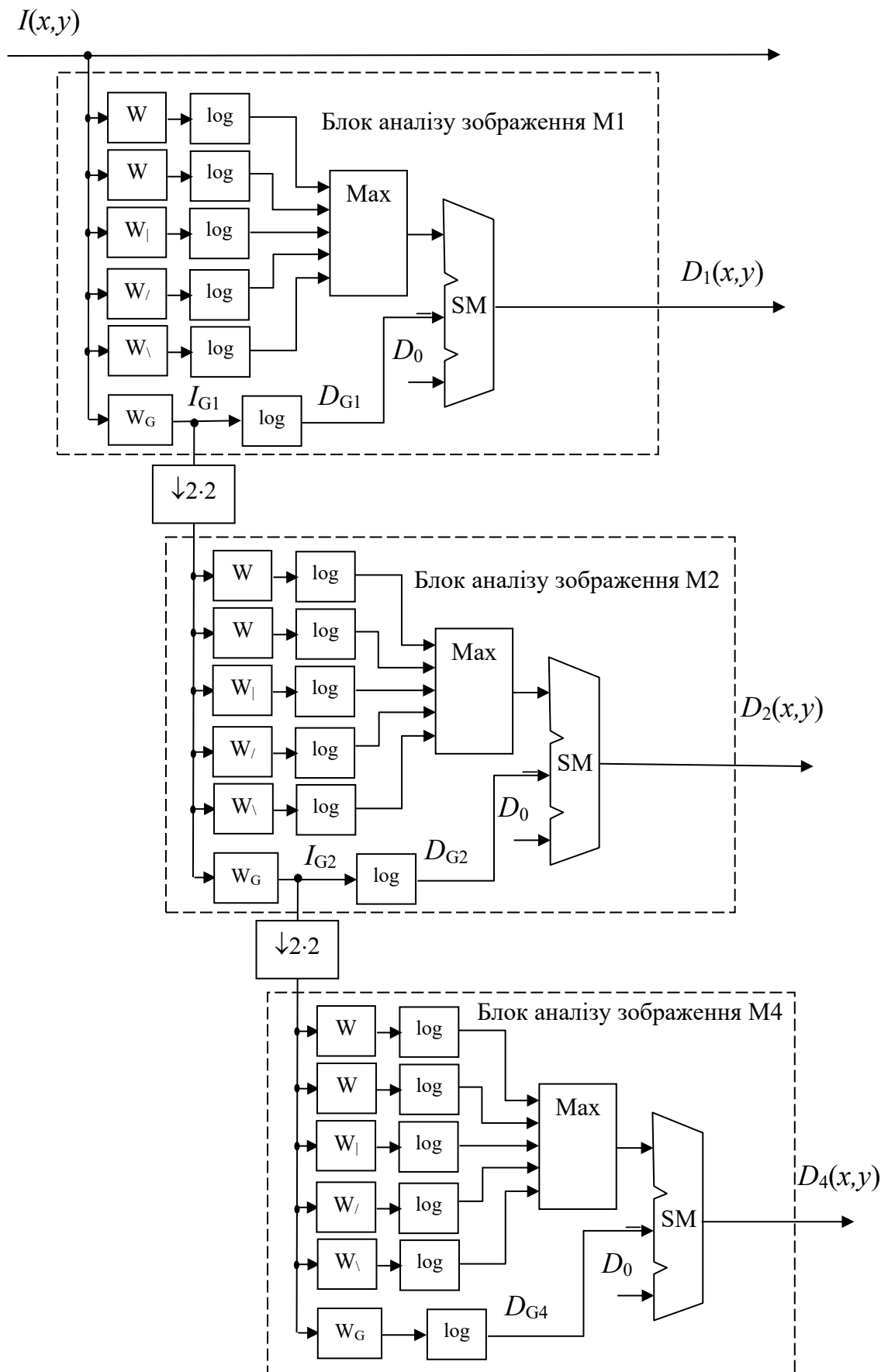


Рисунок 2.29 – Поточковий граф алгоритму для піраміди зображень ознак

Як видно з рис. 2.29, алгоритм складається з трьох однакових ступенів — блоків аналізу зображення M_1 , M_2 , M_4 . Між ними стоять вершини з оператором “ $\downarrow 2 \times 2$ ”, що виконує проріджування даних удвічі по горизонталі та вертикалі. Отже, кожен наступний ступінь обробляє кадр з масштабом, що удвічі менший. Причому даний алгоритм для прорідження зображень не потребує додаткової фільтрації нижніх частот крім тої, що вже існує в блоках аналізу зображення.

На рис. 2.29 показані зображення ознак $D_1(x,y)$, $D_2(x,y)$, $D_4(x,y)$ з різними масштабами одного і того самого фрагменту.

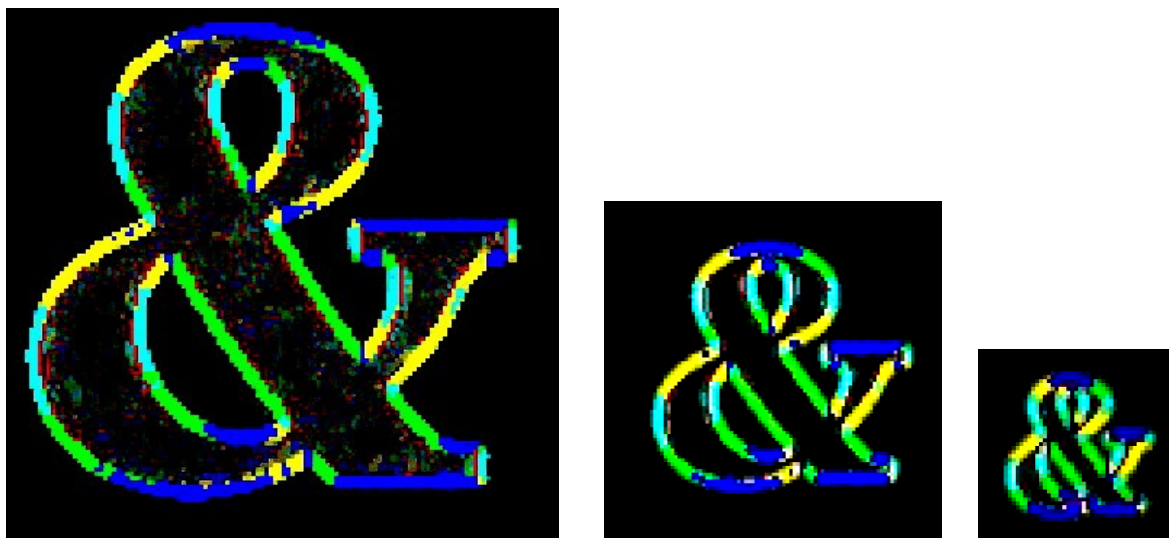


Рисунок 2.29 – Зображення ознак з різними масштабами

Огляд зображень на рис. 2.28 дає змогу зробити висновок, що даний алгоритм дає змогу визначити наявність характерної точки за максимумом інтенсивності одного і того самого кольору у пікселі з координатами (x,y) . Причому у зображеннях з різним масштабом центр характерної точки повинен мати координати

$$(x,y), (\lfloor x/2 \rfloor, \lfloor y/2 \rfloor) \text{ та } (\lfloor x/4 \rfloor, \lfloor y/4 \rfloor), \quad (2.18)$$

відповідно, де “ \lfloor ” — операція округлення до цілого.

2.4.4. Алгоритм знаходження характерних точок

Як і у методі SIFT, для прийняття рішення про наявність характерної точки крім критерія максимуму градієнту яскравості слід прийняти до уваги критерій приналежності пікселя до кута, великої кривизни, Т- чи Х-подібного торкання тонкої лінії чи краю форми. Для розрахунку цього критерія слід проаналізувати статистику кольорів та інтенсивностей пікселів у околі пікселя $D_i(x,y)$.

Характерною точкою (див. рис. 2.30) є точка, що відповідає пікселю у центрі локальної фігури типу:

- пляма,
- вершина кута форми чи лінії,
- кінець лінії,
- точка перетину ліній,
- лінія, яка дотична до форми,
- точка торкання двох форм.

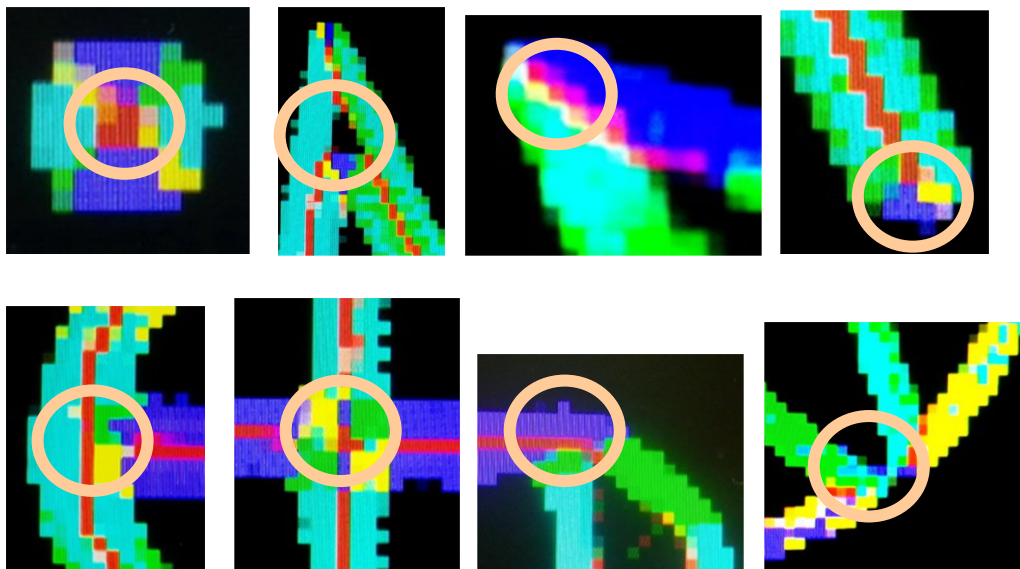


Рисунок 2.30 – Приклади характерних точок

Якщо орієнтуватись на метод SIFT, то треба прийняти до уваги наступні його особливості. За центр кандидата у характерні точки приймається піксель, в якому результат оператора Лапласа, налаштованого на пляму, приймає найбільше або найменше значення у певному околі. Це свідчить про те, що у цій

точці зображення має локальний максимум градієнту. Щоб відкинути точки, що належать тонким лініям чи краям форми, додатково розраховується детектор кутів Гарріса. Для спрощення складних розрахунків виконується апроксимація детектора Гарріса за допомогою обчислення критерія наявності кута (1.11) на основі обчислення матриці Гессе (1.9). Таким чином, за центр характерної точки приймається піксель, в якому великий градієнт яскравості за умови, що він належить до кута. Явним недоліком цього алгоритму є те, що критерій належності до кута є ненадійним, бо матриця Гессе найчастіше обчислюється лише для пікселів у апертурі розміром 3×3 .

Далі у околі знайденого пікселя розраховується матриця градієнтів, за якими будується гістограма розподілення напрямків по кутах. За цією гістограмою розраховується головний напрямок орієнтації характерної точки.

Аналогічно відбувається пошук характерних точок у різних шарах піраміди зображень (рис. 1.2), причому її координати мають відповідати одна одній згідно з (2.18).

На відміну від методу SIFT, у алгоритмі, що пропонується, обробляється зображення, у якому вже визначені як амплітуди, так і напрямки градієнтів, (які є ортогональними справжнім напрямкам). Але в окіл характерної точки можуть потрапити кілька пікселів з однаковою амплітудою через те, що вона має логарифмічне представлення, як, наприклад, на рис. 2.29. Тому щоб точніше визначити координати центру характерної точки, складову інтенсивності D_I зображення $D_I(x, y)$ (2.13) слід профільтрувати фільтром Гауса, апертура якого захоплює принаймні дві сусідні лінії градієнтів, як на рис. 2.29. Тоді можливо застосувати наступне ядро фільтра для апертури 7×7

$$G_{2.5} = \begin{bmatrix} 2 & 3 & 4 & 4 & 4 & 3 & 2 \\ 3 & 4 & 6 & 6 & 6 & 4 & 3 \\ 4 & 6 & 8 & 8 & 8 & 6 & 4 \\ 4 & 6 & 8 & 9 & 8 & 6 & 4 \\ 4 & 6 & 8 & 8 & 8 & 6 & 4 \\ 3 & 4 & 6 & 6 & 6 & 4 & 3 \\ 2 & 3 & 4 & 4 & 4 & 3 & 2 \end{bmatrix}. \quad (2.19)$$

Фільтр з таким ядром має рівень розмиття $\sigma = 2,5$ і виконує 52 додавання малорозрядних зсунутих операндів. Результатом фільтрації є згладжене зображення інтенсивності градієнта

$$D_G(x,y) = D_I(x,y) * G_{2.5}, \quad (2.20)$$

де $*$ — оператор згортки. Максимум результату такого фільтра у околі 3×3 свідчить про те, що знайдена вершина локального максимуму простору ознак, тобто, претендент на характерну точку.

Основною особливістю характерних точок, які перелічені вище, є наявність центру і характерний розподіл пікселів по вузьких секторах, які відходять від центру та мають різні кути нахилу. На рис. 2.31 показане можливе положення цих секторів.

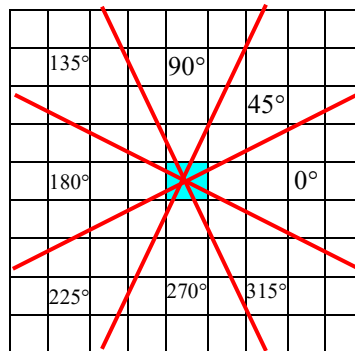


Рисунок 2.31 – Положення секторів при обчисленні розподілення пікселів по напрямках.

Побудова такого розподілу є ідеєю та основою відомого алгоритму Хафа (Hough) аналізу зображень [138].

Розподіл на n секторів виконується за наступним алгоритмом. Створюють n лічильників C_i , з нульовим початковим станом. Якщо в i -му секторі знайдеться піксель $D(x,y)$ відповідного йому кольору, то інтенсивність $D_I(x,y)$ пікселя додається до лічильника, тобто, $C_i = C_i + D_I(x,y)$. Окремий набір лічильників створюється для пікселів кольору R , які можуть попадати у довільний сектор. Таким чином обходять усі пікселі у околі даного пікселю.

На рис. 2.32 показано приблизний розподіл для випадків, як на рис.2.30. Як видно з рис. 2.32, плямі відповідає рівномірний розподіл пікселів при наявності пікселів R. Пляма представляє собою коло мінімального діаметру. Тому порядок кольорів у діаграмі для нього — протилежний. Решта характерних точок має один або кілька виражених піків на діаграмі, які відповідають лініям — променям, які виходять з точки. Якщо в характерну точку входить тонка лінія, то у діаграмі наявний відповідний пік кольору R. Можна передбачити, що якщо піксель, який розглядається, належить прямій лінії, але не характерній точці, то має бути 2 піки на діаграмах, причому такі, що знаходяться у протилежних секторах.

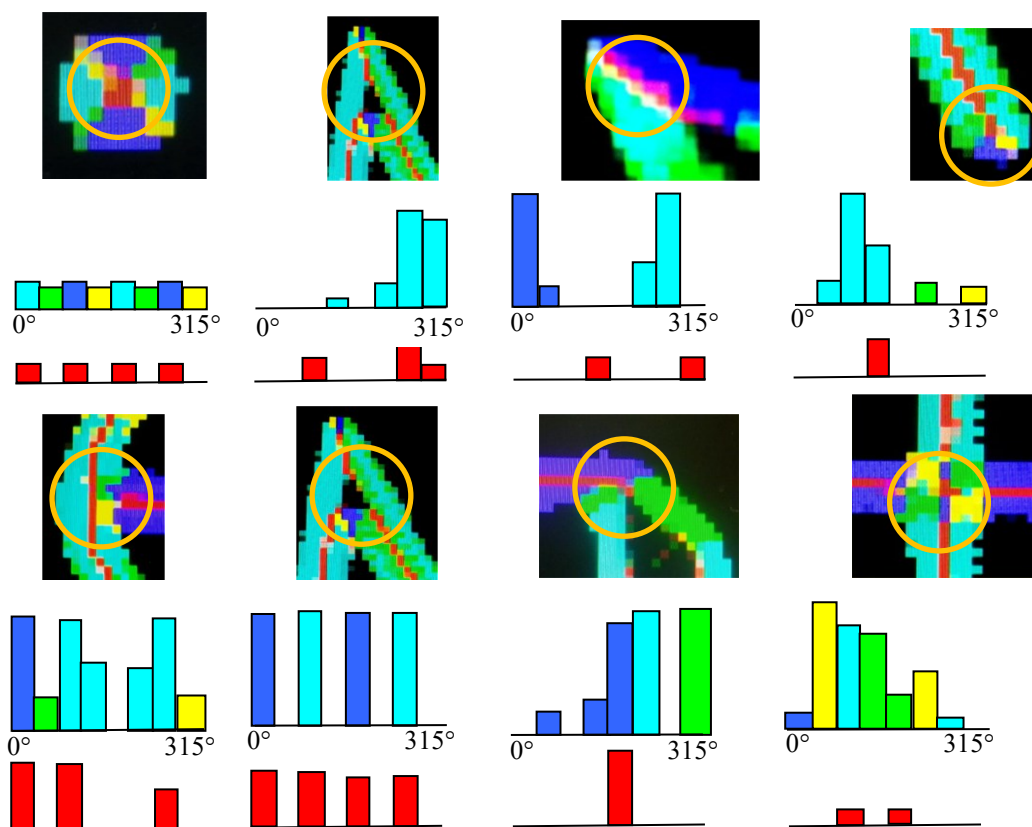


Рисунок 2.32 – Приклади характерних точок та відповідні їм діаграми розподілу

Кількість секторів вибирається, як компроміс між складністю обчислень та точністю пошуку та представлення характерних точок. Аналіз прикладів на рис. 2.32 показує, що кількість секторів 8 є недостатньою для виділення піків у діаграмі розподілу, що характеризують дану точку. Тому слід вибрати удвічі більшу кількість секторів, тобто, 16.

Для порівняння, у відомому методі SIFT для цієї мети вибрано 36 секторів. Крім того, слід мати на увазі, що ефективність методу SIFT ґрунтується на точності визначення відносної висоти піку та кута його нахилу. Причому, одну і ту саму точку розглядають як дві чи три незалежних точки за відповідними двома чи трьома вираженими піками на діаграмі розподілу. Слід відмітити, що згідно з [110], метод SIFT та його похідні методи задовільно виконують своє призначення, якщо градація можливих напрямків не перевищує 8 – 16.

На рис. 2.33 показане розбиття околу пікселю на 16 секторів, які використовуються для побудови діаграми розподілу.

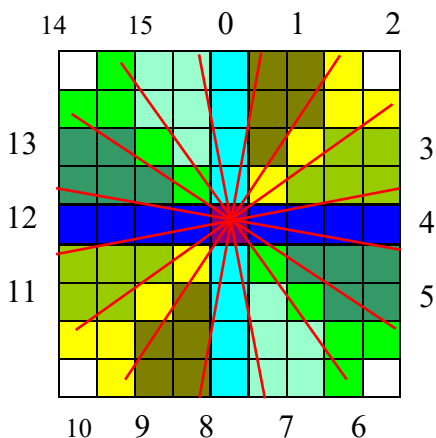


Рисунок 2.33 – Розбиття околу пікселя на 16 секторів

Алгоритм пошуку характерних точок

1) Нехай $i = 1$. Знаходять черговий піксель-точку $D_i(x,y)$ будь-якого кольору крім К, для якої відфільтрований піксель має максимальну амплітуду у зображенні $D_G(x,y)$ (2.20) у околі 3×3 . Обнуляють 16 лічильників C_i , $i = 0, \dots, 15$ розподілу кольорів С, G, В, Y, а також 16 лічильників кольору R

C_{Ri} , $i = 0, \dots, 15$. Розрядність лічильників – $3+k$, де k — розрядність складової амплітуди пікселя $D_i(x,y)$.

2) Для точки (x,y) будується розподіл по 16 напрямкам з використанням розбиття околу пікселя на шаблони-сектори, як на рис. 2.32. При цьому у лічильник C_0 сектору 0 додається амплітуда пікселів D_i (2.11), якщо вони кольору $N_M = C$, для сектору 1 — споріднених кольорів C, Y . у лічильник сектору 2 додається амплітуда пікселів D_i , якщо вони кольору Y для сектору 3 — споріднених кольорів Y, B і т.д.

3) Так само будується другий розподіл по 16 напрямкам, але до лічильників C_{Ri} додається інтенсивність кольору R .

4) У характерної точки знаходять пік з максимальною амплітудою C_{max} , а також інші піки. Піком вважається відлік розподілу, який є максимальним C_{max} або не нижче ніж на 70% від максимального $C_i \geq 0,7 C_{max}$. Точка-претендент відкидається, якщо $C_{max} < C_L$, де C_L — граничний поріг. Якщо у одному чи другому розподілах виявляються по 2 піки, причому такі, які знаходяться у протилежних секторах, наприклад, у 0 і 8-му, то така точка також відкидається (вона належить прямому краю форми або прямій ділянці тонкої лінії).

5) Якщо у одному чи другому розподілах виявляються по 1 – 4 піки, то такий піксель вважається центром характерної точки.

6) Пп. 1–5 повторюються для інших шарів піраміди зображень, тобто, для $i = 2, 3$. В результаті, одержується множина векторів характерних ознак k -ї характерної точки

$$F_k = \{ C_{11}, C_{12}, C_{21}, C_{22}, C_{31}, C_{32} \}, C_{ij} = (C_0, \dots, C_{15}), \quad (2.21)$$

де $j = 1$ – індекс вектору ознак C, G, B, Y , $j = 2$ – індекс вектору ознак R .

Розглянемо приклад знаходження характерної точки згідно з алгоритмом у зображенні на рис. 2.28. Піраміда цих зображень після фільтрації фільтром (2.20) показана на рис. 2.33. У зображенні вибраний черговий претендент на характерну точку, який має локальний максимум інтенсивності ознак

$D_G(x,y)$ (на рис. 2.33 помічений кружечком). Ця точка з околom на піраміді зображень (рис. 2.28) у збільшеному масштабі показана на рис. 2.34.



Рисунок 2.34 – Зображення інтенсивності ознак після згладжування з виділеною характерною точкою

На рис.2.35 показано окіл вибраної точки у шарах піраміди зображень $D_i(x,y)$. Згідно з п.2 алгоритму складено розподіл рівнів інтенсивності пікселів у околі, результати якого наведено у таблиці 2.3.

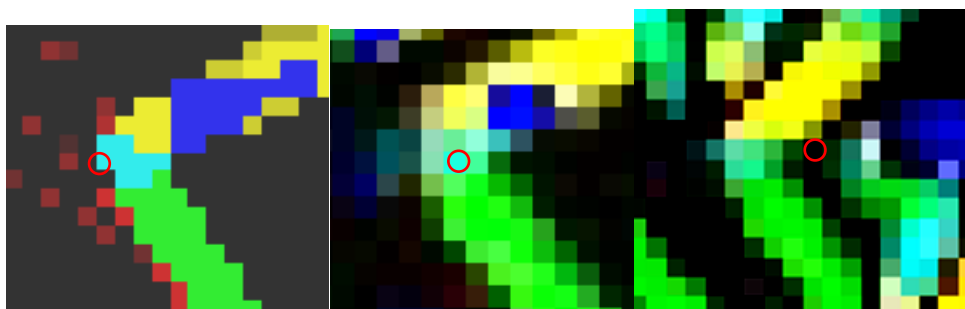


Рисунок 2.35 – Зображення околу знайденої точки

Таблиця 2.3 – Розподіл інтенсивностей ознак характерної точки по напрямках

Рівень піраміди	Напрямок															
	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	2	8	4	8	0	0	16	20	6	0	0	0	0	0	0	0
2	2	8	0	10	0	4	16	20	6	0	0	2	0	2	2	0
3	0	20	16	2	0	0	2	18	0	0	0	0	0	0	0	0

На рис. 2.36. умовно показаний одержаний розподіл ознак характерних точок по напрямках. Аналіз їх положення свідчить про те, що знайдена

характерна точка представляє собою кут у краю форми величиною приблизно 90° , причому компонента максимальної інтенсивності знаходиться у цьому секторі.

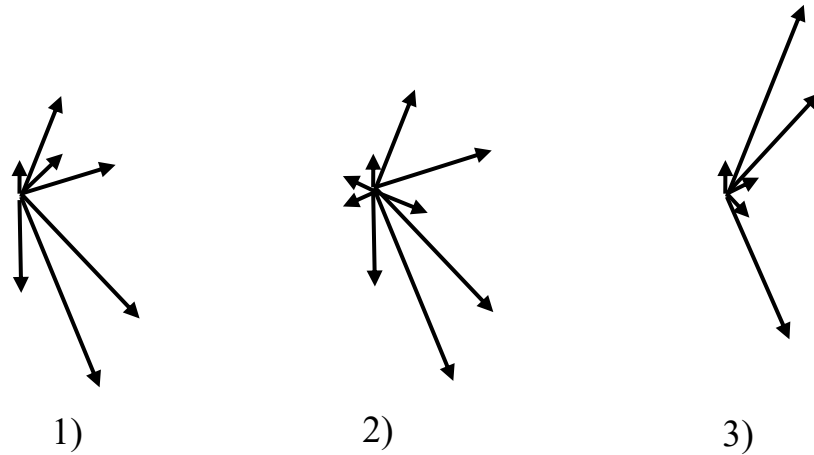


Рисунок 2.36 – Зображення розподілу ознак знайденої характерної точки

У даному прикладі пікселі червоного кольору не приймалися до уваги через те, що вони не проявились у зображенні, яке представляє форму без тонких ліній.

2.4.5. Формування дескриптора характерної точки

На основі побудованого розподілу напрямків поширення ознак навколо центру знайденої характерної точки (2.17) такого, як у табл.2.1, формується її дескриптор. Для цього знаходиться ведучий напрямок — індекс l максимальної компоненти

$$C_l = \max_k (C_k \in C_{ij}). \quad (2.22)$$

Усі компоненти векторів розподілів нормуються відносно компоненти C_l так що

$$C'_k = C_k / C_l. \quad (2.23)$$

Після цього вектори нормованих розподілів циклічно зсуваються на l позицій вліво

$$(C''_0, \dots, C''_{15}) = \text{ROL}((C'_0, \dots, C'_{15}), l). \quad (2.24)$$

Дескриптор k -ї характерної точки формується як множина

$$D_k = \{ C_l, C_{11}'', C_{12}'', C_{21}'', C_{22}'', C_{31}'', C_{32}'' \}, C_{ij}'' = (C_0'', \dots, C_{15}''). \quad (2.25)$$

Для характерної точки, знайденої у прикладі, описаному вище (табл. 2.3), дескриптор представлено у таблиці 2.4.

Таблиця 2.4 – Сформований дескриптор D_k характерної точки

Рівень піра- міди	(x,y)	Ведучий напрямок <i>l</i>	Напрямок															
			0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
1	66, 52	7	1,0	0,3	0	0	0	0	0	0	0	0,1	0,4	0,2	0,4	0	0	0,8
2	33, 42		1,0	0,3	0	0	0,1	0	0,1	0,1	0	0,1	0,4	0	0,5	0	0,2	0,8
3	17, 21		0,9	0	0	0	0	0	0	0	0	0	1,0	0,8	0,1	0	0	0,1

Отже, дескриптор займає $16 \times 3 \times 2 + 1 = 97$ чисел розміром один байт. Зважаючи на те, що більшість елементів – нульові, для зберігання та обробки дескриптор може бути в упакованому виді.

У порівнянні з дескриптором SIFT та аналогічними дескрипторами методів, що походять від SIFT, у даному дескрипторі ознаками виступають не розподілення напрямків градієнтів пікселів у околі характерної точки, а розподілення напрямків ліній зображення градієнтів, що виходять з центру цієї характерної точки. Тому дана процедура формування дескриптора є суттєво простішою, ніж у відомих алгоритмах пошуку характерних точок, таких як SIFT, за рахунок того, що зображення околу характерної точки не обертається і для нього не складається розподілення градієнтів яскравості.

Але для диференціації характерної точки використовується суттєво менший діаметр околу. Через це, запропонований дескриптор має меншу інформативність. Новий дескриптор несе таку інформацію, як наявність кута краю форми чи тонкої лінії, перетину тонких ліній, торкання тонкої лінії краю форми у характерній точці, а також величина знайдених кутів та кут повороту характерної точки у просторі.

2.4.6. Метод пошуку характерних точок у зображенні

Розроблений метод пошуку характерних точок у зображенні має як початкові дані чорно-біле зображення $I(x,y)$ з широким динамічним діапазоном (HDR) і полягає у виконанні ряду наступних етапів.

На першому етапі за допомогою блоку аналізу зображення алгоритму адаптивного фільтру (рис. 2.5) перетворюється у зображення ознак $D(x,y)$ (2.13), пікселі якого мають складову $D_I(x,y)$ інтенсивності градієнту яскравості у логарифмічному масштабі та номер знайденої ознаки у множині характерних ознак $N_M(x,y)$. До цієї множини належать такі ознаки, як пляма, горизонтальний, вертикальний, похилий чи обернено похилий край форми або тонкої лінії. Пікселі з цими ознаками умовно позначені кольорами R, B, C, G, Y, відповідно. Причому для нормалізації $D(x,y)$ розраховується оцінка локальної яскравості $I_G(x,y)$ за допомогою Гаусового фільтру і розраховується її логарифм $D_G(x,y)$, який віднімається від інтенсивності градієнту яскравості.

Формується піраміда зображень ознак, до якої входять кілька зображень $D(x,y)$ у різному масштабі, наприклад, $D_1(x,y)$, $D_2(x,y)$, $D_4(x,y)$, масштаби сусідніх з яких відрізняються удвічі. Причому зображення $D_2(x,y)$ формується за допомогою блоку аналізу зображення із зображення $I_{G1}(x,y)$, яке проріджене удвічі і яке одержане при формуванні зображення $D_1(x,y)$, а зображення $D_4(x,y)$ — із прорідженого зображення $I_{G2}(x,y)$, одержаного при формуванні зображення $D_2(x,y)$ (рис. 2.28).

На другому етапі виконується фільтрація зображень $D_i(x,y)$ піраміди з одержанням зображень $D'_i(x,y)$, в яких мінімізовані шумові пікселі. При цьому застосовується алгоритм фільтрації за методом максимуму однорідності зображення у околі (MHN), згідно з яким піксель має відповідати одному з можливих шаблонів у його околі. Особливостями запропонованого алгоритму фільтрації є те, що у шаблоні мають бути пікселі того самого кольору, що і у пікселя, який розглядається, а також те, що шаблони для пікселів кольорів G, B, C, Y відрізняються від шаблонів для пікселів кольору R.

На третьому етапі виконується пошук координат пікселів $D'_{iC}(x,y)$ — претендентів на центр характерної точки. Для цього зображення зі складовою інтенсивності D_I зображення $D_I(x,y)$ слід фільтрувати фільтром Гауса з апертурою, яка охоплює пікселі, що можуть належати характерній точці, а також з відповідним розмиттям. Наприклад, така апертура має розміри 7×7 , а розмиття $\sigma = 2,5$. У відфільтрованому зображенні $D'_1(x,y)$ відбувається пошук точки (x,y) локального максимуму $D'_{1Cp}(x,y)$. Причому координати точок-претендентів $D'_{iCp}(x,y)$ у інших шарах піраміди ($i = 2, 3$) розраховуються за (2.18).

На четвертому етапі серед точок-претендентів $D'_{iCp}(x,y)$ вибираються характерні точки $D'_{iC}(x,y)$. Для цього виконується алгоритм пошуку характерних точок, описаний у підрозділі 2.4.4. Згідно з ним, розглядаються пікселі у околі точки-претендента, для яких будуються розподіли $C_{ij} = (C_0, \dots, C_{15})$ (2.25) пікселів по напрямках з розбиттям околу, таким, як на рис. 2.32. Точка-претендент $D'_{iCp}(x,y)$ відкидається, якщо у розподілі пік з максимальною амплітудою C_{max} менший за встановлений поріг та коли знайдено рівно два піки, які належать до взаємно-протилежних секторів, наприклад, секторів 0 та 8, 1 та 9 і т.д. Тобто, відкидаються ті точки, які не проходять цей тест наявності ознак.

На п'ятому етапі формується дескриптор знайденої характерної точки. Для цього знаходиться ведучий напрямок, який закодовано індексом l максимальної компоненти $C_l = C_{max}$, усі компоненти векторів розподілів нормуються відносно компоненти C_l і вектори нормованих розподілів циклічно зсуваються на l позицій вліво. Дескриптор характерної точки формується як множина зсунутих нормованих компонентів векторів розподілів.

На відміну від існуючих методів пошуку характерних точок, таких як SIFT та похідних від нього новий метод виконує своє завдання у несприятливих умовах освітленості, зокрема, при обробці HDR-зображення з динамічним діапазоном до 120 дБ, за рахунок віднімання середнього рівня освітленості у околі характерної точки та морфологічної обробки зображення у

логарифмічному масштабі. Крім того, обсяг обчислень зменшений у кілька разів за рахунок того, що втричі менше формується зображень у піраміді зображень, не розраховуються трудомісткі функції обчислення градієнту в околі центрального пікселя, який має набагато більші розміри, а також не виконується поворот у просторі цього околу, а обчислення, в основному, виконуються з малорозрядними даними і коефіцієнтами. Завдяки такому зменшенню обсягів обчислень, зменшується латентна затримка виявлення характерних точок у зображеннях, які слід обробляти у реальному часі, а також спрощується апаратна реалізація нового методу.

Недоліком методу є той, що знайдені характерні точки несуть меншу інформацію про форму зображення цієї точки через те, що аналізується окіл меншого розміру. Але цей недолік мінімізується завдяки можливості групувати знайдені сусідні характерні точки у кластери, які є більш зручними для вирішення задачі розпізнавання образів, ніж характерні точки методу SIFT.

Даний метод розглянуто та проаналізовано автором у різних публікаціях у фахових виданнях [139-142].

2.4.7. Застосування метод пошуку характерних точок у зображенні

На рис. 2.37 показано приклад застосування нового методу до зображення символу “&”, яке розташоване прямо та повернуте на 10° навколо двох осей і зменшене.

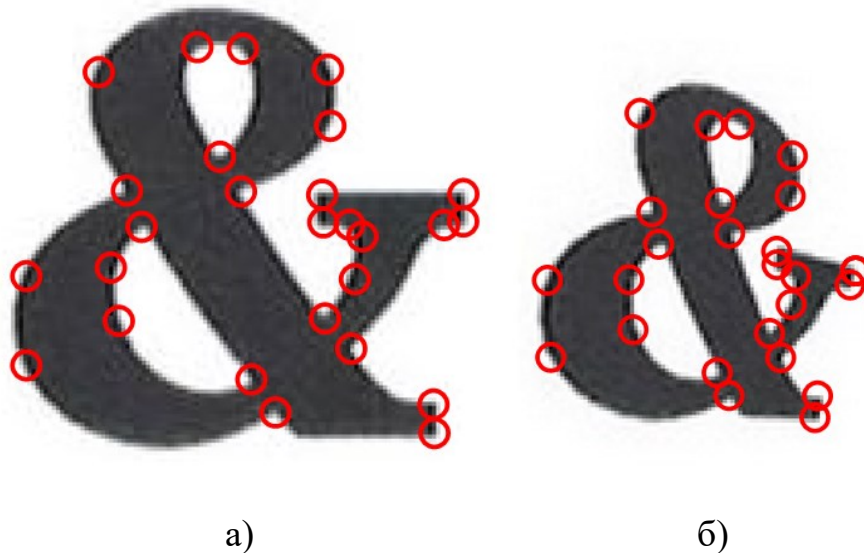


Рисунок 2.37 – Приклад знайдених характерних точок на зображенні, яке розташоване прямо (а) та повернуте на 10° (б)

Аналіз розташування знайдених характерних точок показує, що їх взаємне розташування залишається практично незмінним на об'єкті, який має різне положення у просторі. Отже, можна зробити висновок про дієвість запропонованого методу.

Крім того, групи характерних точок числом від двох до шести, які є локальними одна відносно одної, можуть формувати кластер, на основі якого можна будувати характерну точку більш високого рангу. При цьому дескриптор такої точки включатиме дескриптори точок, які він вміщує та інформацію про їх взаємне розташування і ведучий напрямок кластеру.

Даний метод може знайти багато різних застосувань для вирішення задач розпізнавання образів. Наприклад, на рис. 2.38 показане зображення зразка крові з атиповою формою еритроцитів та результат його обробки у блоці аналізу зображення (рис. 2.5).

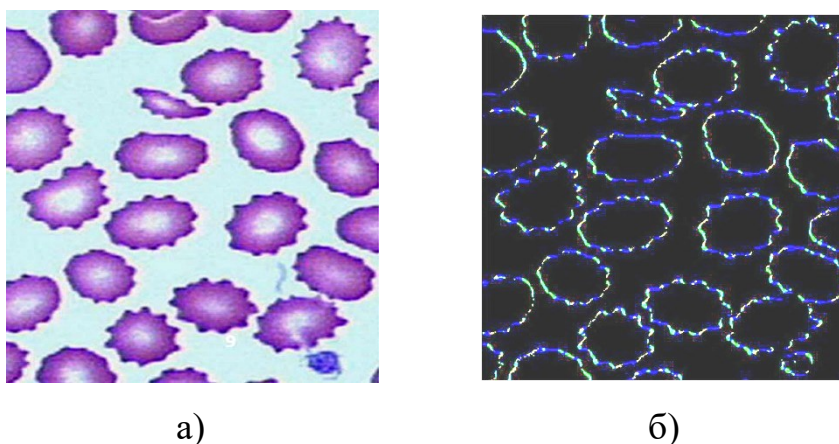


Рисунок 2.38 – Зображення зразка крові (а) та результат його обробки (б)

Отже, даний метод є дієвим при вирішенні задачі оброблення медичних зображень. Вперше його застосування з метою аналізу медичних зображень було опубліковане в [143].

2.4.8. Висновки до розділу

Проаналізовано методи стиснення HDR-зображення і вибрано метод Retinex на основі білатеральної фільтрації як найкращий за якістю стиснення. Підкреслено надмірну складність білатерального фільтра як недолік.

Запропоновано алгоритм адаптивної фільтрації на основі блоку аналізу зображення, який виявляє локальні градієнтні характеристики і формує з них зображення ознак $D(x,y)$. На відміну від алгоритму білатеральної фільтрації, новий алгоритм має учетверо менше операцій множення і не потребує обчислень з підвищеною точністю.

Запропоновано удосконалений алгоритм компресії HDR-зображення, в якому білатеральний фільтр замінено на алгоритм адаптивної фільтрації, завдяки чому зменшена складність алгоритму компресії. Даний алгоритм покращує розпізнавання у складних умовах освітлення за рахунок того, що він формує зображення, у якому вилучено інформацію про локальну освітленість об'єктів, що заважає їхньому розпізнаванню будь-якою системою розпізнавання графічних образів. Тому удосконалений алгоритм компресії є ефективним для використання у системах технічного зору.

Встановлено, що зображення ознак $D(x,y)$, що генерується блоком аналізу зображення алгоритму адаптивної фільтрації вміщує великий обсяг прорідженої інформації про форми і лінії об'єктів і може бути використане для побудови нових алгоритмів розпізнавання образів.

Створено новий алгоритм MHN-фільтрації, завдяки якому у зображенні ознак $D(x,y)$ вилучаються шумові пікселі, які заважають подальшому знаходженню характерних точок.

Створено алгоритм аналізу зображення, який обчислює піраміду зображень ознак $D_i(x,y)$ на основі алгоритму блоку аналізу зображення алгоритму адаптивної фільтрації, який для прорідження зображень не потребує додаткової фільтрації нижніх частот.

Розроблено новий алгоритм знаходження характерних точок, який має етапи знаходження центрального пікселя кандидата на характерну точку, побудови розподілу напрямків ліній, що через неї проходять і аналізу цього розподілу з вибором характерної точки. На відміну від відомих алгоритмів пошуку характерних точок, таких як SIFT, даний алгоритм забезпечує пошук у несприятливих умовах освітленості та має меншу складність.

Запропонована процедура формування дескриптора характерної точки за результатами виконання нового алгоритму знаходження характерних точок, яка є менш складною, ніж у відомих алгоритмах пошуку характерних точок, таких як SIFT, за рахунок того, що немає необхідності обертати зображення околу характерної точки і складати для нього розподілення градієнтів яскравості.

Розроблено новий метод пошуку характерних точок у зображенні, за яким на першому етапі вхідне зображення з широким динамічним діапазоном перетворюється у піраміду зображень ознак $D_i(x,y)$ за допомогою запропонованого алгоритму адаптивної фільтрації, на другому етапі виконується фільтрація зображень $D_i(x,y)$ піраміди з застосуванням нового алгоритму МНН-фільтрації, на третьому етапі виконується пошук координат пікселів — претендентів на центр характерної точки як локальний максимум зображення ознак згладженого фільтром Гауса, на четвертому етапі серед точок-претендентів відкидаються ті, що не проходять тест наявності ознак і на п'ятому етапі формується дескриптор знайденої характерної точки. На відміну від існуючих методів пошуку характерних точок, таких як SIFT, новий метод виконує пошук характерних точок несприятливих умовах освітленості та має обсяг обчислень зменшений у кілька разів.

Дієвість нового методу перевірена у прикладах його застосування. Показана можливість його реалізації при обробці медичних зображень.

Розроблені алгоритми та метод можуть бути реалізовані програмно як в комп'ютері з традиційною архітектурою, так і з підтримкою графічного

акселератора. Про таку можливість свідчать дослід, які проведені з використанням цих алгоритмів і методу при їх програмуванні мовою Java.

Розроблені алгоритми та метод мають мінімізовану складність при їх реалізації у арифметиці цілих чисел особливо якщо їх розрядність мінімізована. Тому у наступному розділі розглянуті особливості їх апаратної реалізації у ПЛІС, де ефективність цих алгоритмів (мінімізовані апаратні витрати при максимальній продуктивності) проявляється в повній мірі.

РОЗДІЛ 3. СПОСОБИ АПАРАТНОЇ РЕАЛІЗАЦІЇ АЛГОРИТМІВ ПОШУКУ ХАРАКТЕРНИХ ТОЧОК У ЗОБРАЖЕННІ

У першому розділі виконано огляд апаратних реалізацій різних методів пошуку характерних точок у зображеннях і зроблені висновки про їх удосконалення. Для цього необхідно створення способу побудови буферних схем для обробки одно- та двовимірних сигналів, який забезпечує заданий порядок слідування вхідних та вихідних даних і мінімізовані апаратні витрати при його реалізації у ПЛІС, удосконалення способів побудови обчислювачів елементарних функцій, а також процесорів, які виконують пост-обробку, таку як формування дескрипторів, їх пошук у базах даних.

У другому розділі було встановлено, що новий метод пошуку характерних точок може бути ефективно реалізований апаратно у такому середовищі, як ПЛІС. При цьому така розробка апаратних засобів також вимагає створення нових та удосконалення існуючих способів апаратної реалізації алгоритмів оброблення зображень.

Даний розділ присвячений виконанню завдань створення і удосконалення способів апаратної реалізації потокових алгоритмів, які використовуються при розробці систем розпізнавання образів на основі характерних точок.

3.1. Методи і способи проєктування апаратних засобів для цифрової обробки сигналів та зображень

3.1.1. Графові моделі потокових алгоритмів

Алгоритми цифрової обробки сигналів та зображень належать до періодичних чи потокових алгоритмів. Потік (dataflow) — це засіб передачі однорідних даних від одного обчислювального процесу до іншого. В потоковому алгоритмі обчислювальні процеси обмінюються даними через потоки даних, причому, як правило, передача даних виконується з певним періодом. У високоефективних системах цифрової обробки сигналів тривалість такого періоду складає період квантування сигналу [144].

Граф потоків даних (ГПД) – це напрямлений граф, вершини якого – актори – представляють операції, а дуги – потоки даних. Актори споживають дані, що називаються мітками або токенами, що відповідають даним і видають мітки на свої виходи. Найбільш поширеними і найбільш виразними є ГПД Денніса [145], мережа процесів Кана [146], граф Карпа й Міллера (граф обчислень, ГПД) [147]. Їхнім основним недоліком є те, що можливість їх блокувань перевіряється лише під час динамічного складання їх розкладу виконання — при безпосередньому виконанні алгоритму в певній обчислювальній системі.

У разі синтезу обчислювачів на базі НВІС і ПЛІС відповідні САПР дозволяють тільки повністю статичний розклад, тобто, усі етапи складання розкладу проходять до виконання алгоритму в обчислювальній системі [148]. Отже, слід вибирати різновид ГПД, який дозволяє саме такий розклад. Актор або процес, або просто оператор, який позначений вершиною ГПД, виконує деяку функцію під час свого запуску. Якщо актор споживає і виробляє деяку кількість міток, ця кількість стабільна під час виконання алгоритму й може бути визначена під час компіляції, то такий актор є регулярним. Якщо ця кількість може змінюватися залежно від даних у потоках, то це динамічний актор. ГПД, у якому всі актори регулярні, дістав назву графа синхронних потоків даних (ГСПД, *synchronous data flow*, SDF), який є мережею процесів Кана з обмеженнями [149].

У ГСПД до міток у потоках додають теги (індекси змінних), які строго відповідають номерам циклів, що є ознакою синхронності потоків. Якщо кількість спожитих із кожного входу міток і згенерованих кожною вершиною міток в одному циклі є тією самою, наприклад, дорівнює одиниці, то такий граф називають однорідним ГСПД (*homogeneous SDF*). Однорідний ГСПД взаємно однозначно відповідає сигнальному графу або обчислювальній схемі, які використовуються в цифровій обробці сигналів [150].

При розробці систем цифрової обробки сигналів та зображень використовуються також ГСПД більш загального виду, такі як неоднорідний ГСПД

(multirate SDF), у якому кількість поданих у потік міток відрізняється від кількості спожитих з нього міток у одному циклі [151], масштабований чи багатовимірний ГСПД, в якому міткам відповідають одно- чи багатовимірні масиви даних, параметричний ГСПД актори якого можуть мати деякі набори функцій, які можна динамічно змінювати [152], булевий ГПД та його узагальнення — цілочисельний ГПД з вершинами, що керуються булевими чи цілочисельними сигналами [153], масштабований ГСПД, в якому міткам відповідають масиви даних. Однак, за застосування вказаних графів значно ускладнюється аналіз графа на блокування.

В останні роки було запропоновано декілька видів моделей ГПД, які мають підвищену експресивну силу. Усі вони пов'язані з моделями скінченних автоматів та моделями, які ґрунтуються на ускладненні поведінки акторів. До них належать модель El Greco [154], яка була впроваджений у системі Synopsys System Studio, модель ГПД із дозволом вмикання акторів [155], модель ГПД за сценарієм (SADF) [156] та модель TensorFlow, що базується на ГСПД, які описуються мовою Lustre [157], які використовуються, зокрема, для проєктування систем обробки зображень та розпізнавання образів.

Отже, ГПД використовується для задання різних алгоритмів обробки потоків даних. Алгоритм, заданий як ГПД, показує, як одержується потік результатів і не задає точний порядок обчислень. Його відображення в обчислювальну систему полягає в знаходженні розкладу і відображенні вершин графа в її процесорні елементи (ПЕ). З метою використання для синтезу обчислювальної системи для обробки сигналів та зображень найбільш спрощеною та придатною є модель однорідного ГСПД. Такий ГСПД переважає завдяки простоті його аналізу й можливості еквівалентних перетворень у нього інших типів ГСПД [127].

Однорідний ГСПД та його аналог — сигнальний граф широко використовується у різних методах проєктування апаратних засобів для цифрової обробки сигналів та зображень. Ациклічні ГСПД відображаються у відношенні

1:1 у нерекурсивні цифрові фільтри, конвеєри процесорів швидкого перетворення Фур'є, а циклічні ГСПД — у схеми рекурсивних фільтрів [8]. Однорідний ГСПД оптимізують за допомогою метода конвеєризації або метода розрізання навпіл [158].

Параметричний ГСПД часто одержують склеюванням кількох вершин однорідного ГСПД у одну ієрархічну вершину. Нова вершина має складну функцію, що покриває функції вершин, які склеєні, і за один цикл алгоритму послідовно обчислює оператори цих вершин. Процедура такого перетворення ГСПД набула назви згортання ГСПД (folding), а одержаний граф – згорнутого ГСПД (folded SDF). Згорнутий ГСПД є ізоморфним графу структури конвеєрного обчислювача, а згортання ГСПД є методом синтезу таких обчислювачів [158, 159].

Деякі способи організації обробки зображень на основі ГСПД, в тому числі буферних схем описані в [160].

Отже, однорідний ГСПД є ефективним засобом завдання потокових алгоритмів та проєктування на його основі високопродуктивних апаратних засобів обробки сигналів та зображень. І тому його слід вибрати як засіб для проєктування апаратури для пошуку характерних точок.

3.1.2 Просторовий ГСПД

Довільний напрямлений граф алгоритму G_A , такий як ГСПД, задається матрицею інцидентності

$$A = ||a_{ij}||_{N \times M}, \quad (3.1)$$

де $a_{ij} = 1$, якщо i -та вершина інцидентна дузі j графа G_A та є її кінцем, або $a_{ij} = -1$, коли i -та вершина є початком дуги j , та $a_{ij} = 0$ – якщо i -та вершина не інцидентна дузі j , N і M – кількість вершин та дуг графа G_A .

У загальному випадку, ГСПД можна представити у n -вимірному просторі \mathbb{Z}^n . Вектори $K_i \in \mathbb{Z}^n$ ототожнюються з вершинами графа G_A . Такий ГСПД називається *просторовим ГСПД*.

Частина координат векторів K_i – це координати K_{Si} процесорного елементу (П)Е, у якому виконується i -й оператор, а інша частина K_{Ti} – кодує момент часу виконання цього оператора. Отже, $K_i = (K_{Si}^T, K_{Ti}^T)^T$.

Таким чином, вектор K_i грає роль тега для i -ї вершини ГСПД. Деякі координати K_i можуть кодувати відлік, наприклад, рівень ієрархії алгоритму і структури, тип оператора й процесорного елементу (ПЕ), в якому він виконується, затримку виконання оператора тощо.

Якщо ГСПД представляє собою алгоритм, що обробляє кадр піксель за пікселем, тоді він представляє собою решітку розміром $p \times n \times m$, де p – кількість ПЕ, які обробляють піксель, $n \times m$ співпадає з розмірами кадра. В цьому випадку вершина K_i має часову складову $K_{Ti} = (t_r, t_c)^T$, де $t_r < n$ – номер рядка, а $t_c < m$ – номер колонки зображення.

При розміщенні ГСПД алгоритму в $n = 3$ -вимірному цілочисельному просторі \mathbb{Z}^3 i -й оператор ототожнюється з вектором $K_i = (s, p, t)^T$, ($i = 1 \dots N$), де p – тип операції, s – просторова координата, t – часова координата.

Вважається, що вершини ГСПД генерують мітки, що означають окремі змінні x_j . Кожна змінна x_j з M змінних алгоритму ототожнюється з n -вимірним вектором D_j ; ($j = 1 \dots M$) $\in \mathbb{Z}^3$, причому

$$D_j = K_l - K_i, \quad (3.2)$$

де K_i відповідає оператору з результатом x_j , а K_l – оператор, для якого x_j – це операнд, тобто, K_i безпосередньо передує K_l .

Для просторового ГСПД задані просторова метрика $S(K_{Si})$, яка виражає абсолютні координати в просторі та часова метрика $T(K_{Ti})$, яка визначає конкретний момент виконання i -го оператора. Відповідно, $S(D_{Sj})$ дорівнює відстані між двома ПЕ, а $T(D_{Tj})$ – різниці моментів виконання суміжних операторів і є більшою або дорівнює тривалості виконання оператора, вершина якого інцидентна ребру D_j та є його початком.

Для того, щоби задати ГСПД через ці множини векторів K_i, D_j , введені наступні визначення.

Конфігурацією алгоритму (КА) є трійка $C_A = (K, D, A)$, де $K = (K_1, \dots, K_N)$ – матриця координат векторів-вершин $K_i = (s_i, p_i, t_i)^T \in \mathbb{Z}^3$; $D = (D_1, \dots, D_{M+1})$ – матриця координат векторів-дуг $D_j = (s_j, p_j, t_j)^T \in \mathbb{Z}^3$; $A = ||a_{ij}||_{N \times (M+1)}$ – матриця інцидентності графа G_A , яка доповнена $M+1$ -м стовпцем з елементом $a_{p, M+1} = 1$. Цей елемент відповідає базисному вектору $D_{M+1} = D_B$, який з'єднує початок системи координат простору \mathbb{Z}^3 з довільним вектором $D_B = K_p \in K$ [161].

3.1.3. Синтез конвеєрних обчислювачів на базі просторового ГСПД

Синтез конвеєрних обчислювачів на базі просторового ГСПД оснований на ін'єктивному проектуванні простору \mathbb{Z}^3 , в якому представлений ГСПД, у підпростори структур \mathbb{Z}^2 та підпростір часу (подій) \mathbb{Z} .

Конфігурацією структури (КС) є трійка $C_S = (K_S, D_S, A)$, де $K_S = (K_{S1}, \dots, K_{SN})$, $D_S = (D_{S1}, \dots, D_{S(M+1)})$ – матриці координат векторів-вершин $K_{Si} = (s_i, p_i)^T \in \mathbb{Z}^2$ і векторів-дуг $D_{Sj} = (s_j, p_j)^T \in \mathbb{Z}^2$, A – матриця інцидентності ГСПД. K_{Si} — це координати ПЕ, у якому виконується i -й оператор, а D_{Sj} — відносні координати лінії зв'язку, якою передається j -а змінна.

Конфігурацією передування (КП) є трійка $C_T = (K_T, D_T, A)$, де $K_T = (t_1, \dots, t_N)$, і $D_T = (d_1, \dots, d_{(M+1)})$. Однорядковій матриці K_T відповідає **вектор часової розгортки** $T = (t_1, \dots, t_N)^T$, що вказує, у які моменти часу виконуються операторні вершини.

Розглянемо приклад представлення ГСПД алгоритму обчислення полінома

$$y_i = ax_i^2 + bx_i + c. \quad (3.3)$$

На рис. 3.1 показані приклад КА (а) та відповідні КС (б) і КП (в). На ньому маленьким кружечком позначена вершина затримки на 1 такт, темним кружечком – вершина вводу, кружечком з крапкою – вершина виводу,

кружечком зі знаком «+» – вершина суматора, кружечком зі знаком « \times » – вершина множення.

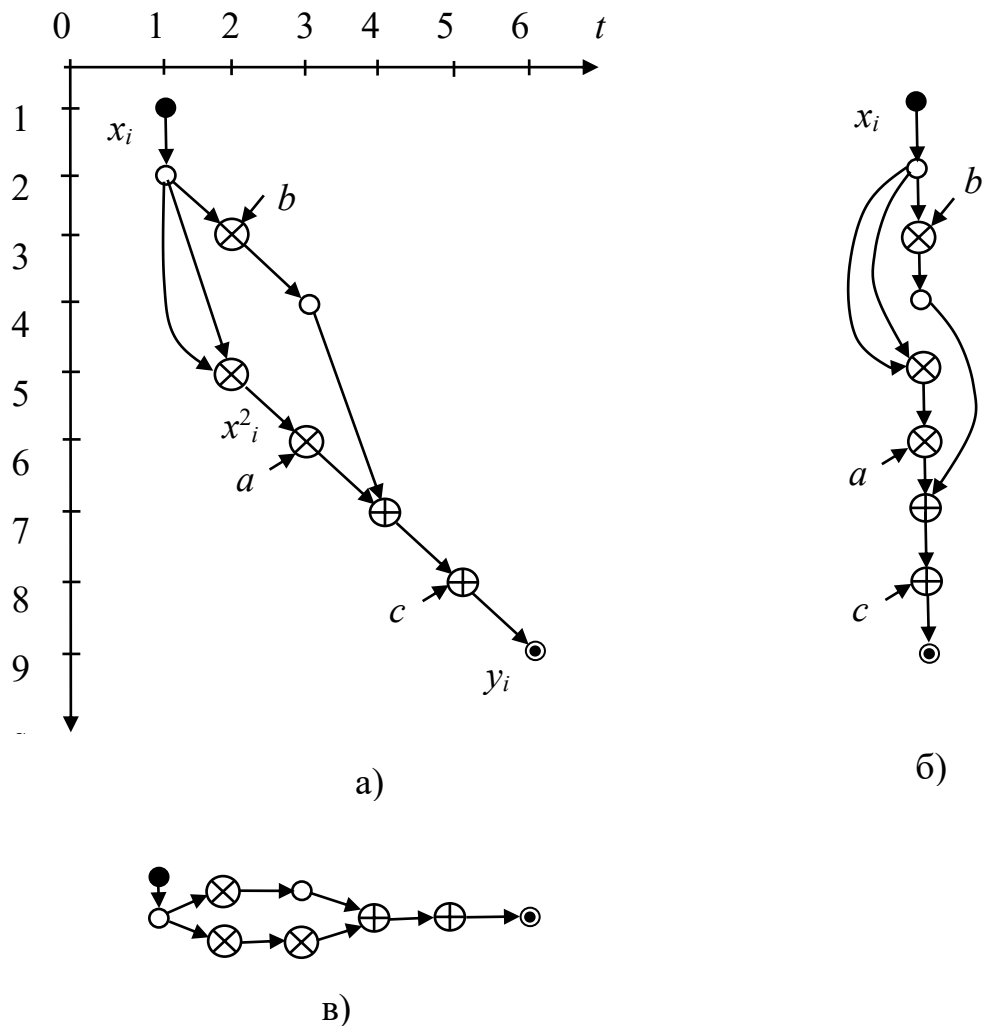


Рисунок 3.1 – Приклад КА (а) та відповідні КС (б) і КП (в) для обчислення (3.1)

Як видно з рис. 3.1, КС представляє собою проєкцію графа на вісь os і її граф співпадає з графом структури, що обчислює формулу (3.1). Відповідно, КП представляє собою проєкцію графа на вісь ot , причому положення вершин відповідає моментам виконання відповідних операторів, а те, що вектори-дуги напрямлені в один бік свідчить про те, що виконується правильне передування цих операторів.

На відміну від звичайного ГСПД, вершини просторового ГСПД повинні розміщуватись у просторі з урахуванням певних правил, які є наступні.

КА є коректною, якщо в матриці K відсутні пари однакових векторів, тобто

$$\forall K_i, K_j (K_i \neq K_j, i \neq j). \quad (3.4)$$

Завдяки цьому, ніякі два оператори не можуть бути виконані одночасно в одному і тому самому ПЕ.

Дуги міжітераційної залежності ГСПД, що навантажені числом w ,

$$D_{Dj} = (k_i, s_i, -wL)^T \quad (3.5)$$

означають затримку на w циклів (ітерацій). D_{Dj} — дуга міжітераційної залежності, що фактично передає дане з попередньої ітерації. Наприклад, якщо в цю дугу-потік, яка навантажена одною затримкою $w = 1$, входить змінна x_i , то з неї виходить змінна x_{i-1} . Отже, просторовий ГСПД є об'єднанням ациклічного графа, який виконує обчислення однієї ітерації, та множини дуг D_{Dj} , які зображують міжітераційні затримки змінних на wL тактів.

ГСПД і відповідна КА є коректними відносно часової функції R , якщо

$$\begin{aligned} K_{Tr} - K_{Ti} = D_{Tj} \in D_T \Rightarrow \\ \begin{cases} R(K_{Tr}) \geq R(K_{Ti}); & \text{— для ненавантажених дуг} \\ R(K_{Tr}) < R(K_{Ti}). & \text{— для дуг } D_{Dj}, \text{ що навантажені затримками} \end{cases} \quad (3.6) \\ i, r = 1, \dots, N; \quad j = 1, \dots, M+1; \end{aligned}$$

тобто, якщо вершини K_{Tr}, K_{Ti} сполучені дугою, то момент виконання оператора попередньої вершини K_{Ti} менший за момент виконання наступної K_{Tr} . Цим самим виконується принцип передування операторів в алгоритмі.

У простому випадку кожен оператор алгоритму виконується за один такт, тобто, $R(s_i, p_i, t_i)^T = t_i$. Такий випадок є натуральним у проектуванні конвеєрних пристроїв на RTL-рівні.

Тоді $R(D_j) = 0$, якщо відповідний ПЕ не має регістра результату й передає результат негайно в наступний ПЕ, $R(D_j) = p$, якщо на виході ПЕ стоїть буфер FIFO з p регістрами. Згідно з рис. 3.1, усі суматори і блоки множення мають регістри результату.

На рис. 3.2 показана модель ПЕ, у який відображаються операторні вершини. Вона має арифметико-логічний пристрій ALU, вихід якого підключений до буфера FIFO. Входи ALU підключені до виходів мультиплексорів, на входи яких поступають дані з інших ПЕ або з даного ПЕ. Причому кількість регістрів у ланцюгу затримки FIFO дорівнює довжині проєкції D_{Tj} найдовшого вектора D_j на вісь ot , який виходить з вершини K_i , яка відображається у даний ПЕ. У найпростішому випадку, ПЕ має єдиний регістр при $D_{Tj} = 1$.

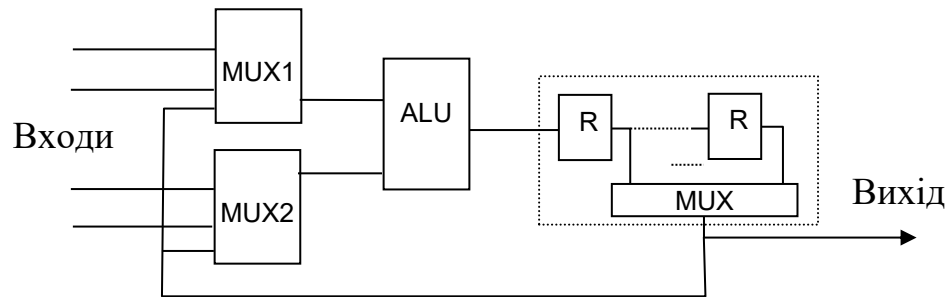


Рисунок 3.2 – Модель ПЕ з пам'яттю

Розклад для КА C_A , з періодом L тактів *є коректним*, якщо оператори, які відображаються в один і той самий ПЕ, виконуються в різних тактах, які розраховуються за модулем L , тобто:

$$\forall K_i, K_j (k_i = k_j, s_i = s_j) \Rightarrow \begin{cases} t_i \equiv t_j \bmod L - (K_i, K_j) - \text{дуга, що навантажена затримками} \\ t_i \not\equiv t_j \bmod L - \text{в решті випадків} \end{cases} \quad (3.7)$$

Лише за цієї умови результуюча структура коректно обчислює період алгоритму в конвеєрному режимі з періодом подання вхідних даних L тактів.

Однотипні оператори треба відображати в ПЕ того самого типу:

$$K_i, K_j \in K_{p,q} (k_i = k_j = p, s_i = s_j = q), |K_{p,q}| \leq L, \quad (3.8)$$

де $K_{p,q}$ – множина векторів-вершин операторів p -го типу, що відображаються в q -й ПЕ p -го типу ($q=1, 2, \dots, q_{\max}^p$).

КА може бути перетворена в КС і КП так:

$$K = \begin{pmatrix} K_s \\ K_T \end{pmatrix}; \quad D = \begin{pmatrix} D_s \\ D_T \end{pmatrix}, \quad (3.9)$$

якщо КА, КП і розклад є коректними, тобто, вони задовольняють умови (3.2) – (3.6), причому одержана модель обчислювача буде правильно виконувати заданий алгоритм у конвеєрному режимі з періодом L тактів.

Ефективна КА шукається за два етапи. На першому етапі ГСПД розміщується в тривимірному просторі як множини векторів \mathbf{K}_i та \mathbf{D}_j з урахуванням умов (3.4) – (3.9), тобто, формується початкова КА. Кількість ПЕ в структурі мінімізується через виконання умови $|K_{p,q}| \rightarrow L$, тобто, коли кількість вершин, що відображаються в один ПЕ, прямує до L . Також виконується перестановка вершин відносно осі часу ot , яка відповідає ресинхронізації ГСПД [162].

На другому етапі КА урівноважується. При цьому розглядається ациклічний підграф ГСПД без дуг \mathbf{D}_{Dj} , які навантажені затримками. У всі його дуги включаються проміжні вершини операторів затримки (регістрів). В урівноваженій КА всі вектори-дуги, крім навантажених дуг, дорівнюють $\mathbf{D}_j = (a_j, b_j, 1)^T$ або $\mathbf{D}_j = (a_j, b_j, 0)^T$. Водночас вершини-оператори формують яруси, відстань між якими за координатою ot дорівнює 1 такт. Урівноважена КА оптимізується через взаємні перестановки векторів-вершин з одного ярусу, які призводять до мінімізації кількості регістрів та входів мультиплексорів.

Розглянемо приклад проектування нерекурсивного фільтра за допомогою ГСПД, що відповідає рівнянню

$$y_i = ax_i + bx_{i-1} + cx_{i-2}. \quad (3.10)$$

На рис. 3.3 показані КА до урівноваження (а), КА після урівноваження та відповідна конвеєрна структура фільтра (в) для періоду виконання $L = 1$ такт. При цьому по осі os умовно відкладені назви відповідних ПЕ.

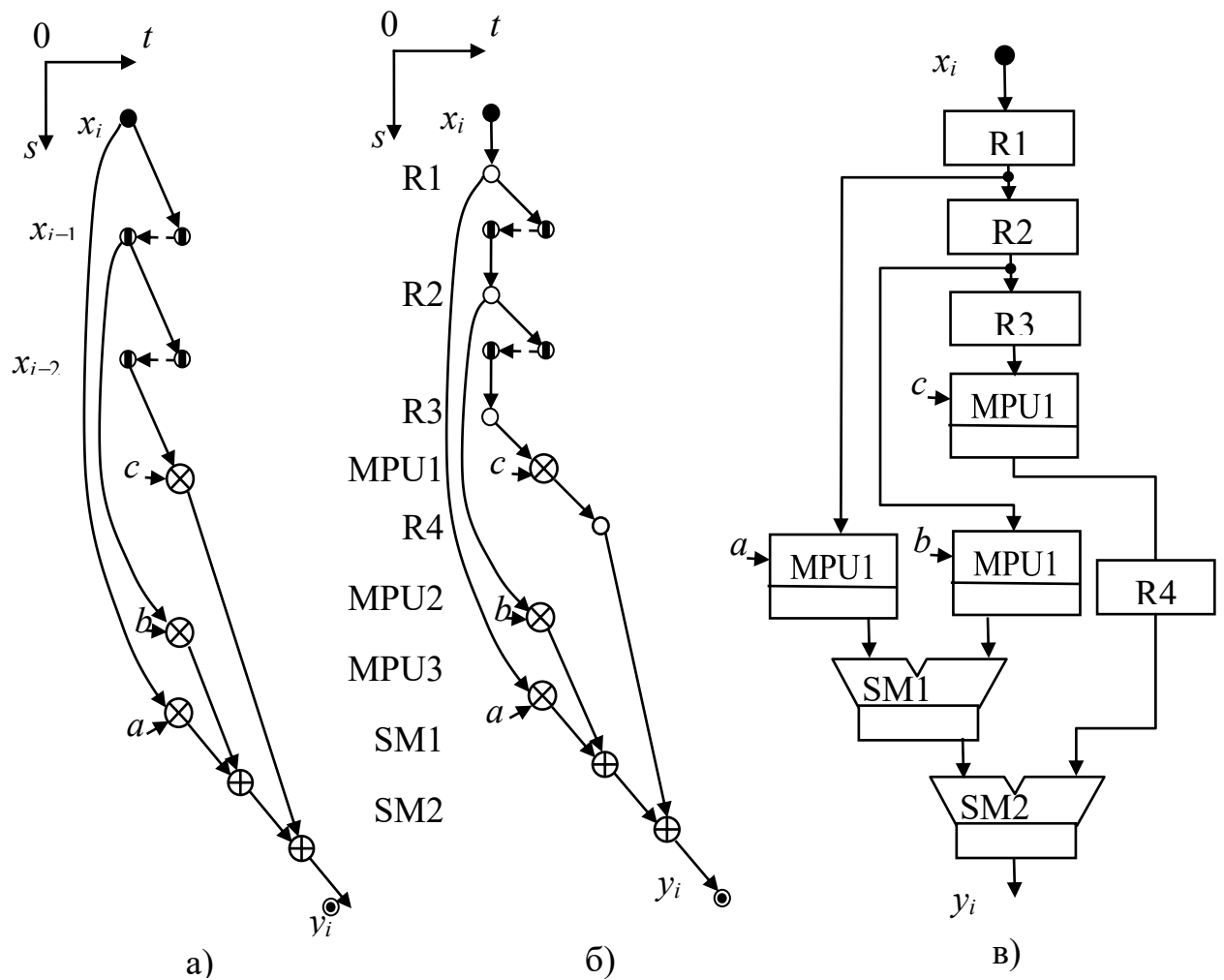


Рисунок 3.3 – КА до урівноваження (а), КА після урівноваження та відповідна конвеєрна структура фільтра (в)

Аналізуючи результати синтезу структури даного фільтра, можна встановити, що згідно з методом просторового ГСПД, одержуються працездатні структури, які виконують заданий потоковий алгоритм у конвеєрному режимі. Причому, переставляючи вершини ГСПД у просторі згідно з правил (3.4) – (3.9), виконується оптимізація як розкладу виконання алгоритму, так і апаратні витрати результуючої структури.

Крім того, побудований узгоджений ГСПД можна безпосередньо описати мовою VHDL з одержанням працездатного модуля без безпосереднього креслення відповідної функціональної схеми [163].

Отже, метод синтезу конвеєрних обчислювачів для цифрової обробки сигналів та зображень на основі просторового ГСПД є ефективним і далі він

буде застосований для проєктування апаратних засобів для пошуку характерних точок і розпізнавання образів.

3.2. Метод проєктування буферних схем для цифрової обробки зображень

3.2.1. Методи та способи проєктування буферних схем

Останнє покоління ПЛІС представляють собою привабливу альтернативу для прискорення обробки зображень та розпізнавання образів у них, оскільки ПЛІС містять програмовну логіку для прискорення інтенсивних обчислювальних операцій. Однак, ПЛІС мають не більше ніж десяток мегабайт вбудованої пам'яті. Через такі обмежені можливості вбудованої пам'яті, для розміщення кадрів зображення використовується зовнішня пам'ять DDR-RAM [164]. Це призводить до великої інтенсивності звертань до пам'яті, яка має потенційно велику затримку, та, як результат, до погіршення продуктивності або великих апаратних витрат через розпаралелювання доступу до кількох мікросхем пам'яті та збільшення енергоспоживання, бо пам'ять DDR енергоємна. Це перешкоджає застосуванню ПЛІС особливо при обробці великих кадрів зображення за алгоритмами, що мають інтенсивний обмін з пам'яттю.

Для задач обробки зображень характерна обробка відеопотоків з високою тактовою частотою та, бажано, з мінімізованою латентною затримкою. Для цього, з одного боку, у ПЛІС слід використовувати конвеєрне обладнання, яке здатне працювати з вхідними потоками пікселів зображення, а з іншого боку, буферизація проміжних зображень не може бути надмірною. Така апаратна архітектура може розпочинати обробку зображення, як тільки буде накопичена необхідна множина пікселів, і продовжувати обробку за конвеєрним принципом, забезпечуючи як високу пропускну здатність, так і низьку латентну затримку.

Алгоритми обробки зображень, як правило, складаються з кількох функцій, що виконуються послідовно так, як показано у першому розділі. Кожна

з цих функцій зчитує необхідну кількість пікселів з блоку пам'яті кадра зображення, обробляє їх і результуючий піксель записує у іншу область цієї пам'яті або у інший блок пам'яті. Оскільки кадр зображення займає порівняно великий об'єм, у системах обробки зображень на ПЛІС використовують ієрархічну пам'ять. Нижній рівень цієї пам'яті складають конвеєрні регістри, середній рівень - блоки буферної пам'яті BRAM розміром кілька кілобайт та зовнішня пам'ять (DRAM) складає вищий рівень.

Внутрішня пам'ять ПЛІС має низьку затримку доступу, але відносно невелику ємність. Навпаки, зовнішня пам'ять має більшу ємність, але більшу затримку та меншу пропускну здатність. Крім того, доступ до DRAM споживає значно більше енергії, ніж до пам'яті вбудованої у ПЛІС. Таким чином, у сфері архітектури ПЛІС для обробки зображень актуальною темою є пошук балансу між вартістю вбудованого буфера та продуктивністю системи.

Для потокової архітектури прийнято, що ПЛІС отримує пікселі зображення рядок за рядком так, як вони захоплюються датчиком зображення. Буфери на кристалі ПЛІС використовуються для зберігання кількох рядків кадру для доступу до певного вікна або апертури, що обробляється. Але найуживанішою та зрозумілою парадигмою залишається та, що алгоритм обробки читає дані з довільного місця в кадрі, обробляє їх і записує результати назад у пам'ять кадрів. Для цього необхідна зовнішня пам'ять з великою пропускну здатністю, а також блоки буферної пам'яті з конструкцією, що оптимізована за об'ємом.

Найпростіший спосіб збільшити пропускну здатність пам'яті — мати кілька паралельних блоків пам'яті. Аналогічно можна виконати пам'ять з надвеликою довжиною слова даних, яке зберігає кілька сусідніх пікселів. Але у цих випадках необхідно крім кількох зовнішніх мікросхем пам'яті мати багато окремих виходів з ПЛІС для адрес і даних, що часто є неприйнятним.

Дещо зменшити вплив цієї проблеми можна організувавши кілька блоків кеш-пам'яті у ПЛІС. За допомогою розділення адресного простору на кілька

банків, наприклад, використовуючи один банк пам'яті для непарних адрес і один для парних адрес, доступ до суміжних адрес може бути доступний одночасно. Наприклад, чотири банки можна використовувати для доступу до чотирьох пікселів у блоці розмірами 2×2 . Крім того, для ефективного доступу до пікселів у апертурі адреса може бути закодована так, як запропоновано у [165].

При конвеєрній обробці даних з довільним доступом один процес може писати результати в один банк пам'яті, а інший – читати дані з другого банку. Такий режим передбачує двопортова пам'ять BRAM у ПЛІС. Коли обробку чергового кадру завершено, банки міняються ролями. При цьому для кращої синхронізації використовують ще третій банк пам'яті [166]. Але таке перемикання банків додає один період кадру до суттєвої латентної затримки алгоритму і має наслідок збільшення апаратних витрат системи та використання більшої кількості контактів ПЛІС.

Більш практичним підходом є використання пам'яті на вищій тактовій частоті, ніж решта системи. Пам'ять із подвійною швидкістю передачі даних (DDR) є одним із прикладів пам'яті, яка дає змогу двічі передавати дані за такт. Як правило, сучасні ПЛІС великої ємності мають виділені виводи та вбудований контролер доступу до зовнішньої динамічної пам'яті DDR останніх поколінь [9]. При цьому в проекті імітується багатопортова пам'ять за рахунок часових інтервалів доступу. Крім того, необхідні блоки буферної пам'яті для запису та читання, оскільки динамічна пам'ять має високу пропускну здатність лише при пересилці рядків даних з сусідніх комірок. На жаль, у багатьох проектах пам'ять DDR також потрібна для підтримки роботи операційної системи процесора, вбудованого у ПЛІС і тому пропускну здатність цієї пам'яті падає при обробці зображень.

Буфери з конвеєрних регістрів та буфери типу first in – first out (FIFO) – це два поширених способи організації пам'яті, які використовуються в ПЛІС. Вони розрізняються у наступному. Архітектура буфера на конвеєрних регістрах також належить до категорії first in – first out (перший прийшов – перший

вийшов). Але операції прийому та видачі даних в ньому є синхронні, як і в регістрі зсуву. Буфер FIFO — це пам'ять, куди дані можна приймати та видавати з однаковим порядком даних, але ці операції можуть бути несинхронізованими. Спосіб проектування таких буферів пояснюється в [167]. Але розробник сам повинен організувати належний порядок запису та видачі даних.

Якщо дані слідують послідовно, то варто використовувати буфери типу FIFO, комірки якого зберігають блоки даних, а пікселі на виході вибираються за локальною адресою [168]. Для цього більшість конструкцій блоків BRAM мають програмовний режим FIFO.

Однією з поширених форм проміжного зберігання даних є буферизація рядків. Розглянемо обчислення функції від дев'яти значень пікселів у апертурі (рис.1.7). За алгоритмом, дев'ять пікселів потрібно зчитувати з пам'яті кадра для кожної позиції апертури у кожному тактовому інтервалі та кожен піксель має зчитуватися дев'ять разів, коли апертура сканує зображення. Пікселі, що розташовані поруч по горизонталі, потрібні в послідовних тактах, тому можуть бути буферизовані та затримані в регістрах. Це зменшує кількість читань до трьох пікселів за кожен такт. Буфер рядків зберігає значення пікселів попередніх рядків, щоб уникнути повторного читання значень пікселів.

Кожен буфер рядка фактично затримує введення пікселів на один рядок. Очевидною реалізацією такої цифрової затримки є використання N -ступеневого регістра зсуву, де N — ширина зображення. Блок BRAM у ПЛІС можна налаштувати як буфер FIFO за схемою циклічного буфера. Крім того, декілька паралельних буферів рядків можна реалізувати як один буфер за рахунок того, що в одне слово упаковані кілька пікселів [169].

Для різних розмірів зображення слід проектувати буфери різної довжини. В роботі [170] пропонується застосовувати універсальний буфер, який настраюється на розміри кадра й апертури з можливістю динамічного перенастроювання. Аналогічний буфер описано в [171], який додатково спроможний

транспонувати положення пікселів у вікні, а також виконувати корекцію зображення на його краях.

У роботах [172, 173] були представлені загальні методи проектування потокової структури для обробки зображень із апертурою як у прикладі на рис. 1.7. При цьому функції, що послідовно виконуються у алгоритмі, відображаються у відповідні блоки обробки, які відокремлені один від одного буферними блоками і зберігають кілька сусідніх рядків. Взаємозв'язки між блоками обробки і буферними блоками — це шини, які відповідають дугам графу потоків даних алгоритму. Слід відмітити, що у такій схемі виконується алгоритм, який задано на мережі процесів Кана [146], у якій дані між акторами передаються через потоки даних виконані FIFO. За рахунок цього, проміжні дані використовуються багаторазово без звертання до зовнішньої пам'яті. Отже, буфери FIFO служать ефективними буферами даних [158]. Оскільки ця модель передбачає, що дані вибираються з FIFO у довільному порядку, то буфер може містити кілька виходів зі своїх головних регістрів.

Така організація обчислень рекомендується також при програмуванні в ПЛІС графічних функцій бібліотеки OpenCL [174]. Але варто зауважити, що мережа процесів Кана не захищена від блокувань.

Уніфікована мова моделювання або UML забезпечує ефективне представлення мережі процесів Кана. Багато інструментів, таких як IBM Rational Rhapsody, забезпечують компіляцію опису UML у апаратне забезпечення [175]. Інструмент Matlab Real-Time Workshop (RTW) пропонує можливості генерації коду безпосередньо з описів графічної системи Simulink, яка є свого роду мережі процесів Кана [176]. Ці інструменти реалізують буфери FIFO, як це передбачено мережі процесів Кана. Але ці буфери повинні підкорятися правилам асинхронного читання та запису даних у них у відповідному порядку.

Коли алгоритм може бути представлений деякою мережею Петрі, тоді можна використовувати обчислювальну модель потокової обробки. У цій моделі вершина актора має буфер шаблону чи апертури та обчислювальний

модуль, що підключений до виходів буфера. Під час обчислювального процесу вхідні дані завантажуються в буфер асинхронно, і саме тоді, коли вони утворюють правильний шаблон, починаються обчислення [177]. Таким чином, буфер насправді є регістровим конвеєром з великим набором виходів, що часто є неефективним рішенням.

Широко використовується парадигма архітектури фон Неймана, в якій кожні дані мають власну адресу в загальному адресному просторі. У цій парадигмі буфери даних реалізовані як блоки кеш-пам'яті. Слід зауважити, що коли дані втрачають свої адреси в момент перед їх виконанням, то ця кеш-пам'ять може бути представлена як звичайний буфер FIFO. Тому такий буфер даних часто називають кеш-буфером [178]. Спосіб проектування кеш-буфера для ПЛІС з оптимізованою пропускнуою здатністю описана в [179]. Коли застосунок ПЛІС має справу з динамічним розподілом пам'яті, кеш-буфери можуть бути розроблені за допомогою методу аналізу алгоритму, який вибирає незалежні та спільні області пам'яті [180].

Як показано у підрозділі 3.1, ГСПД — це урізана модель мережі процесів Кана, у якій усі потоки даних є синхронними. Буфери FIFO в моделі ГСПД завжди є синхронними, і ця модель зазвичай вільна від взаємоблокувань. Ця модель надає просте відображення у апаратуру, забезпечуючи ефективні методи оптимізації структури, такі як конвеєр, ресинхронізація, згортання та розділене використання ресурсів [181]. Ця ідея розширена та реалізована в системі моделювання потоків даних Ptolemy [182]. За допомогою ГСПД також синтезуються оптимізовані буфери даних. Але результати синтезу можуть бути далеко не оптимальними, оскільки оптимізація виконується вручну або автоматично за жадібним алгоритмом. За допомогою такої оптимізації виконується пошук ефективного розкладу програмного виконання, який не узгоджується з апаратною мінімізацією.

Якщо алгоритм, заданий ГСПД, не має циклів і зворотного зв'язку, він зазвичай представлений графом потоку даних (ГПД). Тоді буфери даних

мінімального об'єму можуть бути синтезовані за допомогою методу, запропонованого в [183]. Цей метод поєднує розподіл реєстрів за допомогою планування лівого краю та згортання ГСПД.

При обробці зображень можна використовувати багатовимірний ГСПД, в якому дані мають вектори індексів, які розглядаються як координати пікселів у кадрі зображення [184]. Але розроблення буфера залишається складним завданням.

Багато алгоритмів, включаючи алгоритми обробки зображень, представлені гніздом циклу. Вектори індексів ітерацій гнізда циклу та самі дані утворюють багатовимірну сітку, а алгоритм виконує відповідну решітку графа залежностей даних. Метод проектування систолічного процесора широко використовується для відображення цих алгоритмів як у структуру процесора, так і в розклад виконання операторів [181]. Обов'язковим результатом такого відображення є конвеєрні буфери даних. Тому цей метод зараз широко використовується для розробки буферів даних у багатьох методах синтезу та САПР.

Розміщення операторів у просторі ітерацій та відображення їх у структурі та розклад також використовується в [185]. Для оптимізації буферів даних використовується система лінійних нерівностей, яка враховує залежності по даних, затримки переміщення даних і часові обмеження. Ця система розв'язується за допомогою програми розв'язку задачі цілочисельного лінійного програмування. У результаті оптимізується пропускна здатність і синтезуються конвеєрні буфери даних. Але процес синтезу стає дуже складним, коли розміри задачі збільшуються.

Цей метод розширено за допомогою поліедральної моделі представлення графу потоків даних алгоритму та його відображення [186]. Завдяки цьому методу виконані ітерації алгоритму та їхні дані утворюють поліедр у багатовимірному ітераційному просторі, який обмежує об'єм ґратчастого графу. Кожна ітерація в ньому займає певний цілочисельний вектор у просторі. Цей поліедр відображається в систолічну структуру комп'ютера та

розкладі за допомогою оптимізованих афінних перетворень цього простору. Коли гніздо циклу описує поведінку масиву даних, тоді результатом відображення є набір конвеєрних буферів даних. Подібний метод запропоновано в [187]. Метод під назвою *lattice-based partitioning* базується на тому ж принципі та виконує вибір набору розподілених буферів [188].

Апаратне забезпечення ПЛІС використовується ефективно, забезпечуючи високу пропускну здатність при частому повторному використанні даних. Метод проектування буферів, описаний у [189], забезпечує повторне використання даних, коли алгоритм виконує послідовну обробку масиву з використанням адресації за модулем періоду. Більш складний метод, що використовує повторне використання даних, запропоновано в [190]. У ньому реалізовано підхід розробки систолічного процесора, і дані, які вибираються з одно- або двовимірного масиву, повторно використовуються в алгоритмі.

Компілятор може створити такий буфер даних, який забезпечує повторне використання отриманих даних у апертурі, що переміщується по кадру. Структура буфера визначається розміром вікна, розміром масиву та кроком повторного використання в кожному вимірі [190]. Цей метод ефективно використовується в компіляторі *Riverside Optimizing Compiler for Configurable ComPEting (ROCCC)* [191].

3.2.2. Ресурси ПЛІС для проектування буферних схем

Мікросхема ПЛІС зазвичай містить достатній обсяг різних ресурсів пам'яті. Зазвичай основним будівельним блоком є логічна таблиця (ЛТ, LUT) у ПЛІС фірми Xilinx або адаптивний логічний модуль (ALM) у ПЛІС Intel-Altera. До кожної з них підключається додатково один або два тригери. Ці тригери зазвичай утворюють запам'ятовуючі елементи ступенів конвеєра, включаючи буфери конвеєра. Сама по собі ЛТ може бути налаштована як буферна пам'ять з об'ємом до 64 біт і з кількома можливими портами читання. Крім того, ЛТ можна налаштувати як конвеєрний буфер змінної глибини. На рис. 3.4 показана структура примітиву SRL16, який створюється на базі ЛТ і

містить 16-розрядний регістр зсуву. Кожен з його відводів вибирається статично або динамічно вихідним мультиплексором.

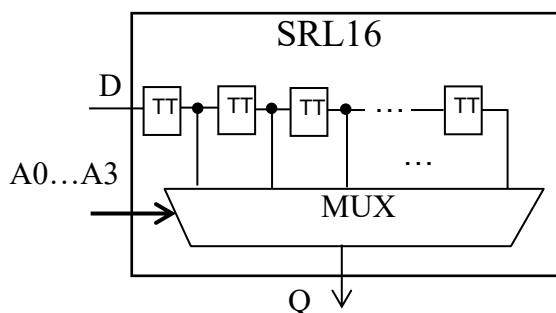


Рис. 3.4. Структура конвеєрного буфера SRL16

ПЛІС містить від десятків до тисяч двопортових блоків пам'яті BRAM. Кожен з них містить кілобайти пам'яті програмованої розрядності. Відношення числа BRAM до числа ЛТ у ПЛІС складає від 60 до 200. Зазвичай вони можуть бути налаштовані як буфери FIFO [192, 193].

Архітектура Intel Hyperflex ПЛІС забезпечує конвеєрні буфери довільної довжини в сегментах маршрутизації зв'язків між ALM. Ці буфери забезпечують найвищі тактові частоти в пристроях Intel Stratix® 10 і Intel Agilex™. Використання тригерів Hyperflex у проектах вимагає спеціальних знань про оптимізацію ГСПД і не виконується в більшості випадків, коли ГСПД містить цикли [194].

Зазвичай найефективніші структурні рішення виводяться при проєктуванні на рівні регістрових передач (RTL). Але при такому проєктуванні вибір буфера та його динамічне керування, яке залежить від модулів, які до нього приєднані, є складним завданням. Тому традиційним рішенням є вибір буфера FIFO на основі BRAM, який займає збільшений апаратний обсяг. Буфери SRL16 рідко використовуються, як наприклад, в деяких конкретних скінченних автоматах, фільтрах або блоках шифрування [195] і не використовуються у буферних схемах через складність проєктування.

3.2.3. Цілі та задачі дослідження проєктування буферних схем

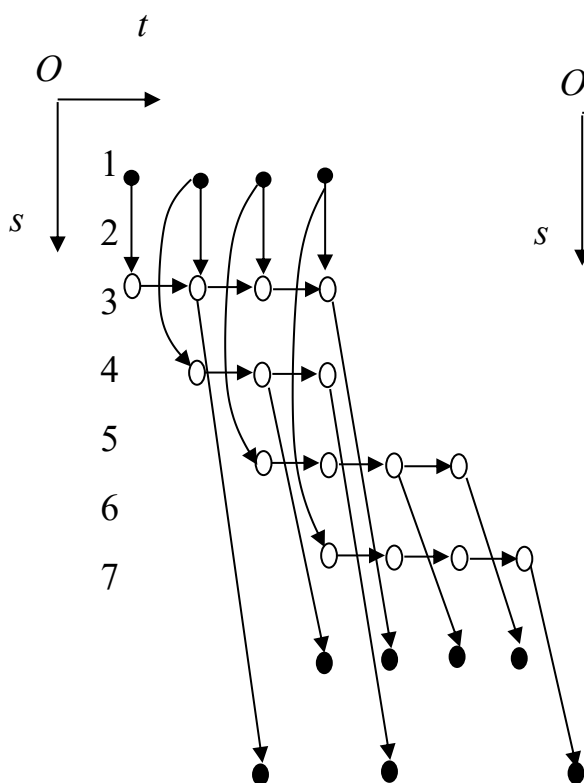
Отже, аналіз існуючих методів та способів проєктування буферних схем при обробці зображень і зокрема огляд буферів, реалізованих у системах пошуку характерних точок на базі ПЛІС, зроблений у підрозділі 1.4.2., показує наступне. Найкраща стратегія — виконувати обчислення даних, які якомога більше зберігаються в блоках буферної пам'яті всередині ПЛІС. При цьому обробка ведеться у конвеєрних блоках обробки, а блоки буферної пам'яті є буфери FIFO. Але залишається відкритим питання, яким чином краще організувати ввід даних і передачу проміжних результатів між конвеєрними блоками обробки. Крім того, не вирішена проблема організації обчислень, коли період слідування даних L не співпадає з періодом тактового сигналу, хоча є кратним йому. У ПЛІС, що виготовляються фірмами Xilinx та Lattice є можливість організувати буфери FIFO на базі ЛТ, які мають динамічне перестроювання глибини, тобто, на елементах SRL16. Але відсутній спосіб, за яким можна їх ефективно застосувати для обробки сигналів та зображень.

Отже, цілями та задачами дослідження проєктування буферних схем є створення методу побудови буферних схем для обробки одно- та двохвимірних сигналів, який забезпечує заданий порядок слідування вхідних та вихідних даних і мінімізовані апаратні витрати при його реалізації у ПЛІС, зокрема, схем, які забезпечують організацію обчислень, коли період слідування даних не співпадає з періодом тактового сигналу.

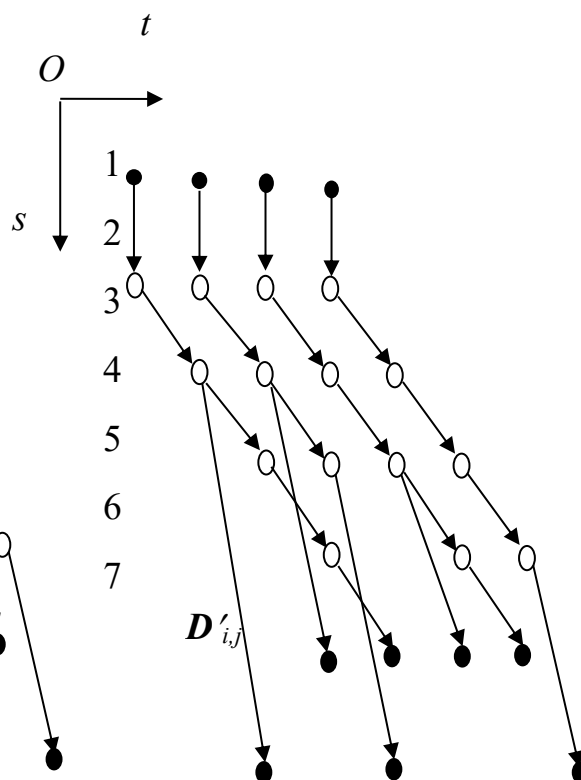
3.2.4. Проектування буферних схем для обробки сигналів

Розглянемо КА C'_{Av} , яка виконує ітераційний алгоритм з періодом $L = 4$ такти, і яка складається тільки з вершин вводу і виводу. Ця КА відображається в буфер даних. Розміщуючи вершини КА у просторі \mathbb{Z}^3 , слід використовувати деякі стратегії для мінімізації кількості з'єднань між ПЕ. Розташування вершин операторів затримки за стратегією розміщення дуг D_{ij} паралельно осі Ot на другому кроці синтезу показано на рис. 3.5, а. А конфігурацію C'_{Av} за стратегією розміщення дуг D_{ij} під кутом до осі Ot показано на рис. 3.5, б.

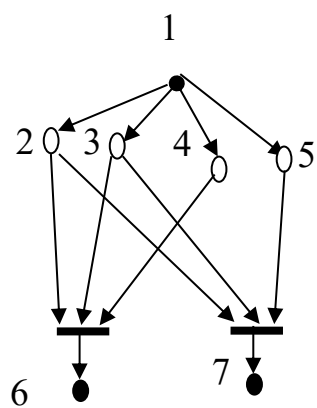
Конфігурації структур, що відповідають цим КА, показано на рис. 3.5, в і 3.5, г відповідно.



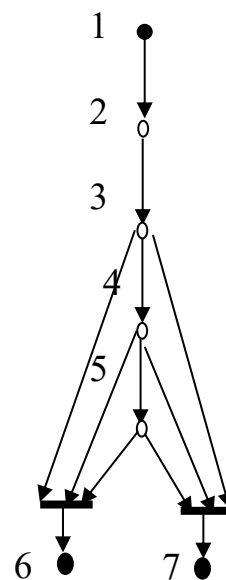
а)



б)



в)



г)

Рисунок 3.5 – КА, дуги якої розміщені відповідно до стратегії синтезу ОЗП (а) або конвеєрного буфера (б), а також відповідні конфігурації ОЗП (в) і FIFO (г)

Тут товсті лінії означають мультиплексори, підключені до входів ПЕ, які виконують вибір операнда, коли він зчитується з відповідного регістра.

Аналіз цих конфігурацій структури показує, що вони відповідають дво-портовому ОЗП (один порт для читання-запису, другий лише для читання) і конвеєрному буферу даних відповідно. Застосовуючи ту чи іншу стратегію мінімізації числа з'єднань, проектувальник може орієнтувати процес синтезу буфера даних на реалізацію у вигляді оперативної пам'яті або регістрового конвеєра. Стратегію слід обирати з урахуванням наступних особливостей.

При синтезі буфера на основі ОЗП змінна x_i розміщується у відповідному регістрі, тобто ланцюжок вершин затримки розташовується на прямій, яка паралельна осі Ot . Також один регістр присвоюється декільком змінним, періоди існування яких не перетинаються, тобто кілька ланцюгів вершин затримки розташовані на прямій, паралельній осі Ot , і ці ланцюжки не перекриваються. При цьому дуги D_{ij} , які примикають до виходів вершин K_{ij} КА до врівноваження повинні задовольняти співвідношення

$$\max_{ij}(t_{D_{ij}}) \leq L, \quad (3.11)$$

де $t_{D_{ij}}$ – часова складова дуги вектора D_{ij} . Якщо воно не дотримується, необхідно розрізати збалансований КА C'_{Av} на кілька підконфігурацій, кожна з яких буде відповідати своїй оперативній пам'яті або забезпечувати перезапис змінної x_i , для якої не дотримується нерівність (3.8), в другий регістр ОЗП після L тактів. Очевидно, що об'єм результуючого ОЗП для КА C'_{Av} з λ вхідними вершинами дорівнює

$$N_p = \lambda. \quad (3.12)$$

Коли буфер конвеєра розроблений, то змінна x_i надсилається в сусідній регістр конвеєра в кожному такті i , проходячи через ланцюжок регістрів $t_{D_{ij}}$,

виводиться з нього на вхід ПЕ, який отримує цю змінну. Це еквівалентно тому, що ланцюжки суміжних вершин K_{ij} операторів затримки при рівномірно зростаючих координатах s_{ij} і t_{ij} розміщуються вздовж паралельних прямих, розташованих під кутом до осі Ot (рис. 3.5, б). Отже, значення t_{Dij} в (3.8) може бути будь-яким, однак для мінімізації кількості ступенів реєстрового конвеєра кількість різних значень векторів D'_{ij} має бути мінімальною. Кількість реєстрів у конвеєрі дорівнює

$$N_P = \max_{ij}(t_{Dij}). \quad (3.13)$$

Таким чином, КА, яка виконує передачу даних між вхідним і вихідним портами після його балансування та оптимізації відповідно до однієї з двох стратегій, дає мінімізований обсяг пам'яті в результуючому буфері даних. Ми отримуємо буферну структуру з пам'яттю, організованою у вигляді оперативної пам'яті або реєстрового конвеєра. У той же час, кількість реєстрів в ОЗП менша, ніж у конвеєрі реєстрів, якщо кількість вхідних вершин, які відображаються на один вершина порту (кількість різних змінних, що входять в один ПЕ) в КА менше максимального затримка змінної, яка обчислюється в цьому ПУ, тобто при

$$\lambda = \max_{ij}(t_{Dij}). \quad (3.14)$$

Коли отриманий конвеєрний буфер виконується в примітиві SRL16, тоді метод повинен враховувати той факт, що буфер має єдиний вихід (див. рис. 3.4). Це додає додаткове обмеження для розміщення вершин у просторі, оскільки лише одне дуга має з'єднувати будь-яку вершину затримки з вершиною, яка відображається у ПЕ вихідного порт. КА на рис. 3.5, б не задовольняє цій умові. Таким чином, цю КА слід розбити на дві підконфігурації як на рис. 3.6, а, що задовольняє вимогам та відображається в структуру з двома буферами, реалізованими на примітивах SRL16 (рис. 3.6, б).

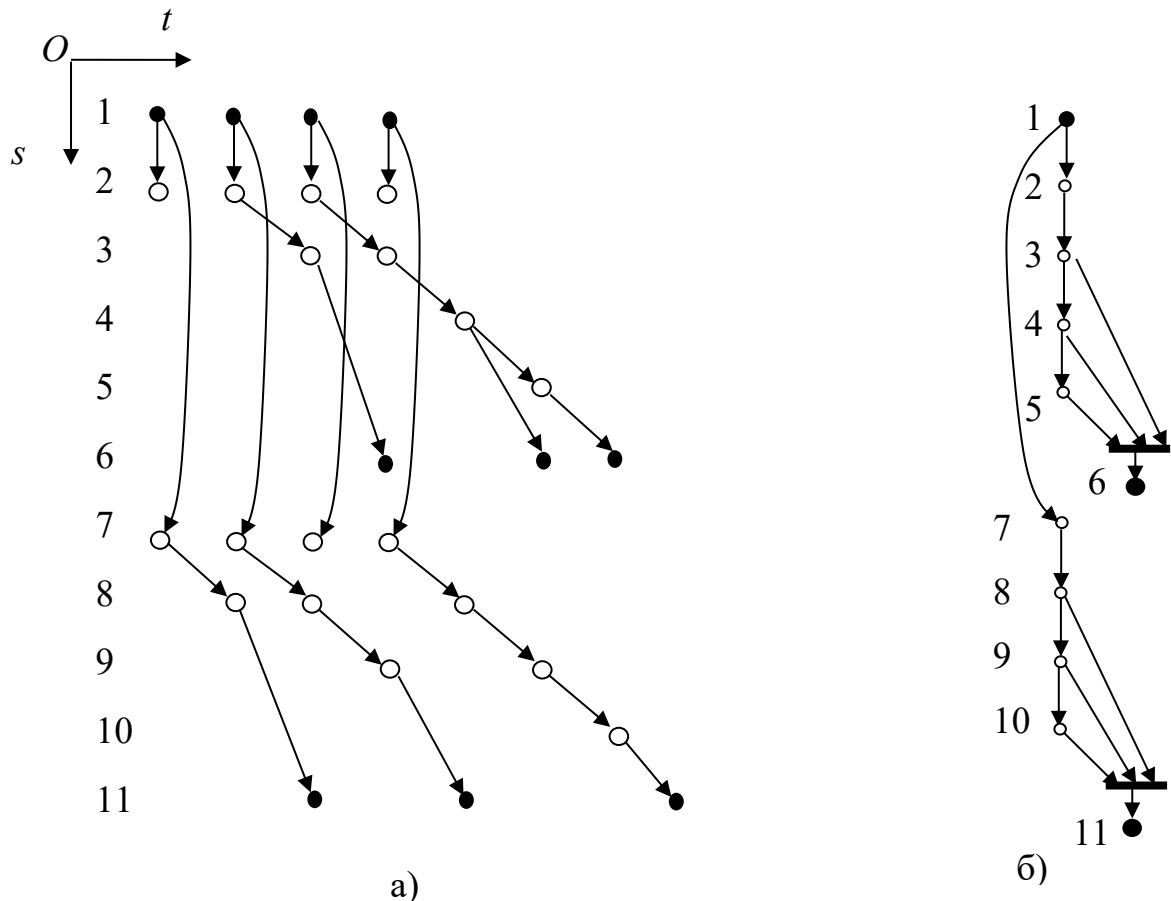


Рисунок 3.6 – Розділена КА (а) та її відображення в структури SRL16 (б)

Примітив SRL16 має додатковий вхід дозволу синхронізації, керування яким дозволяє уповільнити переміщення даних через конвеєрні регістри. При використанні цього входу кількість регістрів можна мінімізувати, якщо значення $R(D_j)$ більше, ніж кількість доступних регістрів у конвеєрі. На рис. 3.7 наведено приклад перетворення КА, зображеного на рис. 3.6, а, з метою додаткової затримки операндів. Таким затримкам відповідають вектори D_j , які розміщені паралельно осі Ot . Слід звернути увагу, що кількість вершин, які мають однакову координату s , не повинна перевищувати період обчислення L .

Аналіз на рис. 3.7 показує, що спосіб керування дозволом синхронізації дає змогу істотно мінімізувати кількість регістрів конвеєра та вихідних мультиплексорів. Це важливо, коли конвеєрні регістри виконуються на основі звичайних регістрів, оскільки це економить апаратне забезпечення та мінімізує тактовий період.

Якщо вершини-джерела розглянутої КА мають різні просторові координати s (у наведених прикладах $s = 1$), то на вході примітиву SRL16 виходить вхідний мультиплексор. Для мінімізації таких мультиплексорів можна використовувати метод, який описано в [196].

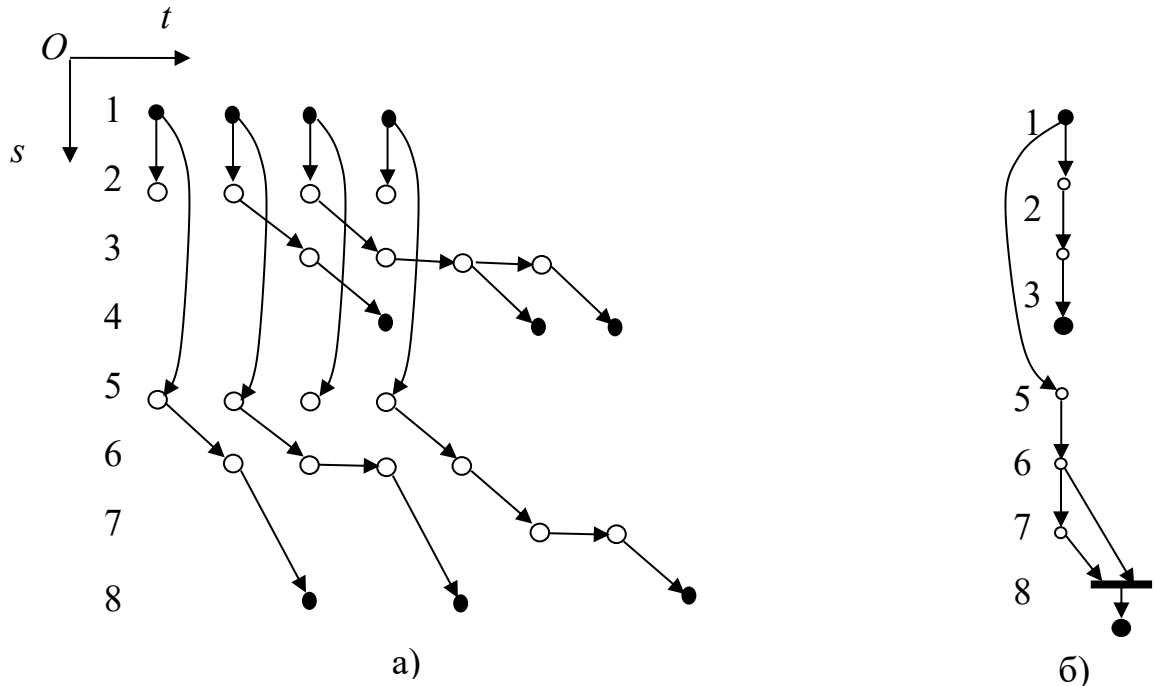


Рисунок 3.7 – Модифікований КА з рис. 3.6 (а) та його відображення в структурі SRL16 (б)

3.2.5. Спосіб проектування буферів з конвеєрних регістрів

Таким чином, спосіб проектування конвеєрних обчислювачів з буферами на основі примітивів SRL16 виглядає наступним чином. Вихідними даними є КА, період виконання алгоритму L та інші параметри оптимізації. Спосіб виконується так само, як метод проектування конвеєрних обчислювачів, який описано в [162, 197], за винятками, описаними нижче.

На першому етапі синтезу повинні бути обрані підконфігурації КА, що відповідають передачі операндів між ресурсами обчислювача з часовими затримками та/або перетасуванням операндів, які, як очікується, будуть відображені в окремі буфери даних.

На другому етапі синтезу дуги, по яких передаються буферизовані дані, урівноважуються додаванням в них проміжних вершин затримки. Вершини затримки цих дуг розміщуються на паралельних лініях, розташованих під кутом до осі часу або паралельно цій осі, таким чином, що сусідні вершини затримки відрізняються за часовими координатами на один такт. Повинні виконуватись вимоги до правильного розміщення вершин (3.2) – (3.7), а також вимога щодо реалізації буфера з одним входом і одним виходом. При неможливості отримання одного входу в буфер використовується евристика мінімізації кількості входів додаткового мультиплексора на вході буфера згідно з [196], а при неможливості отримання буфера з одним виходом ланцюжок вершин затримки розбивається так, що вони відображаються в додаткові буфери (див. рис. 3.7).

Якщо вдається одержати КА, у якій немає проміжних відводів, то така КА відображається у конвеєрний буфер ПЛІС Intel Hyperflex.

Дуги залежностей разом із відповідними вершинами затримки, які є інцидентними до вершин, які споживають буферизовані дані, повинні бути відображені в буфери даних. Якщо в буфер відображаються лише дуги, які знаходяться під кутом до осі часу, то операнди записуються в буфер у кожному такті. Якщо є дуги, які паралельні цій осі, то запис у буфер забороняється у відповідних тактах (див. рис. 3.7).

На третьому етапі буфер з конвеєрних регістрів описується мовою VHDL відповідно до методу, представленому в [197], і компілюється в конфігурацію ПЛІС, яка містить буфери на основі елементів SRL16, які відповідають вибраним підграфам КА.

3.2.6. Приклад синтезу буфера з конвеєрних регістрів

Розглянемо розробку вхідного буфера для конвеєрного блоку, що виконує 8-точкове дискретне косинусне перетворення (ДКП). ГПД цього алгоритму часто базується на алгоритмі Чена [198]. Цей алгоритм відрізняється тим, що його період конвеєрних обчислень дорівнює $L = 8$ тактів, а вісім вхідних даних одного ДКП-перетворення повинні бути затримані та переставлені у

вхідному буфері перед їх обчисленнями. ГПД першого етапу цього алгоритму, який потребує буфера даних, показаний на рис. 3.8.

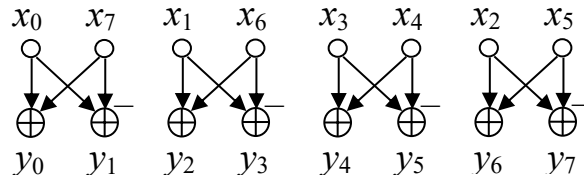
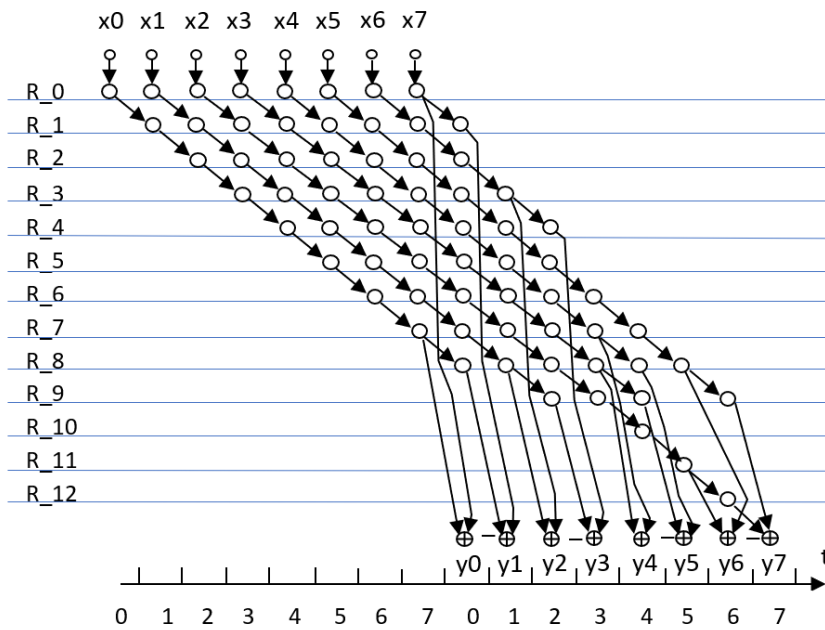
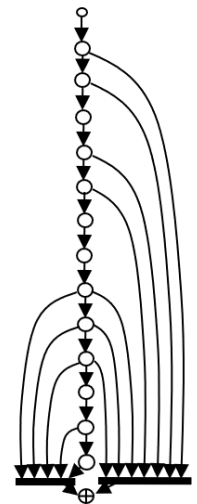


Рисунок 3.8 – ГПД першого етапу алгоритму ДКП

Оптимізований КА, який відображається в конвеєрному буфері та суматорі. КА і відповідна конфігурація структури показані на рис. 3.10. Тут назви ресурсів розміщені на осі Os , а номер тактового циклу за модулем $L = 8$ відображається на осі Ot . Цей КА описаний у VHDL на рис 3.11.



а)



б)

Рисунок 3.10. – Збалансований просторовий ГСПД для ГПД на рис. 3.8 (а) та відповідна конфігурація структури (б)

```

library IEEE;
use IEEE.STD_LOGIC_1164.all;
use IEEE.Numeric_STD.all;
entity DCT_BUF is
    port(
        CLK : in STD_LOGIC;
        RST : in STD_LOGIC;
        START : in STD_LOGIC;
        X : in SIGNED(8 downto 0);
        Y : out SIGNED(8 downto 0)
    );
end DCT_BUF;
architecture synt of DCT_BUF is
    type TARRAY16 is array (0 to 15) of SIGNED(8 downto 0);
    type TN is array(0 to 7) of natural range 0 to 15;
    constant a1: TN:=(7,8,8,9,8,9,11,12);
        constant ar: TN:=(0,1,3,4,7,8,8,9);
    signal r1,r2:TARRAY16;           -- register array of SRL16
    signal cycle:natural range 0 to 7;
    signal sm,l,r: SIGNED(8 downto 0);

begin
    CT8:process(CLK) begin           -- period counter
        if CLK'event and CLK='1' then
            if START='1' then
                cycle<=0;
            else
                cycle<= (cycle+1) mod 8;
            end if;
        end if;
    end process;

    l<= r1(a1(cycle));
    r<= r2(ar(cycle));
    SRL16_BUF:process(CLK) begin    -- SRL16 description
        if CLK'event and CLK='1' then
            r1<=X & r1(0 to 14);      -- FIFO shift
            r2<=X & r2(0 to 14);      -- FIFO shift
            case(cycle) is
                when 0|2|4|6 => sm<= l + r; -- adder
                when others => sm<= l - r; -- subtractor
            end case;
        end if;
    end process;

    Y<=sm;
end synt;

```

Рисунок 3.11. – Код программы 8-точкового дискретного косинусного перетворення

Тут сигнали r_1 , r_2 представляють два ланцюжки конвеєрних регістрів, які завантажують вхідні дані X в кожному такті. Вони синтезуються після поділу КА на рис. 3.9, а на дві підконфігурації, як це зроблено на рис. 3.8. Сигнали з них l , r зчитуються за адресами, які вибираються з ПЗП a_1 , a_r . Ці сигнали спрямовуються на лівий і правий входи суматора-віднімача з регістром sm , що отримує результат Y . Цикл лічильника періоду обчислення розраховує за модулем $L = 8$ і контролює як знак суматора sm , так і конвеєрні ланцюги регістрів через ПЗП a_1 , a_r .

Цей проект скомпільований пакетами Xilinx ISE і Vivado CAD в ПЛІС різних серій. Результати компіляції наведені в таблиці 3.1.

Таблиця 3.1. Результати конфігурації проекту буферу у ПЛІС

Серія ПЛІС	САПР	Тригерів	ЛТ	ЛТ використаних як логіка	ЛТ використаних як SRL16	Період синхросигналу, нс
Virtex-4	ISE 14.7	12	37	19	18	3.14
Spartan-3A	ISE 14.7	12	37	19	18	5.47
Spartan-6	ISE 14.7	12	39	29	10	4.72
Artix-7	Vivado2016	111	44	39	5	4.06

Аналіз цієї таблиці показує, що синтезатор ISE розпізнає шаблон примітиву SRL16, і результати синтезу є буферами даних з мінімальним апаратним обсягом і високою продуктивністю. Синтезатор Vivado спочатку намагається об'єднати обидві гілки буфера конвеєра в одну, а потім мінімізує число тригерів, замінюючи ланцюжки регістрів примітивами SRL16. Створена структура проілюстрована на рис. 3.12. Можна побачити, що, крім того, синтезатор не виконує спільне використання ресурсів суматора-від'ємника. Як наслідок, обсяг апаратних витрат у кількості тригерів значно вищий.

Отже, приклад показує, що застосовуючи новий спосіб, зменшуються апаратні витрати, за рахунок того, що p регістрів замінюються на k елементів SRL16 на основі логічних таблиць, де $p/k = 2 - 16$.

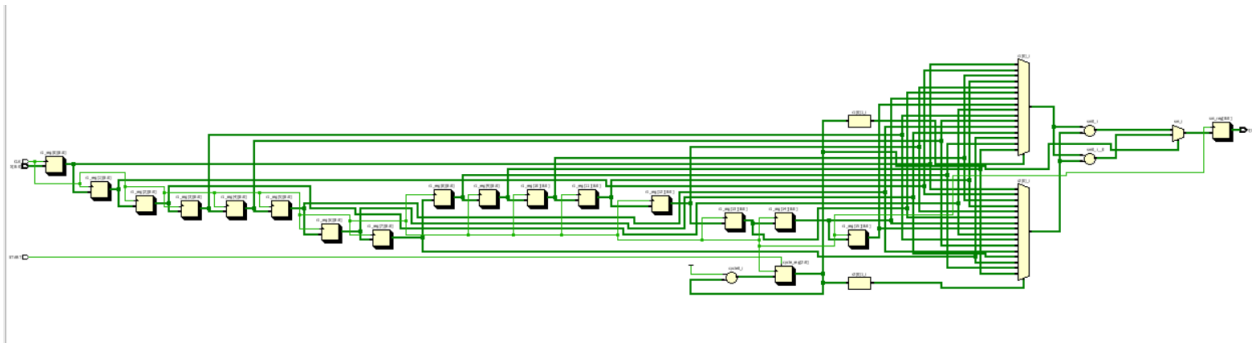


Рисунок 3.12 – Структура буфера даних, отримана за допомогою САПР Vivado

3.2.7. Метод синтезу буферних схем для обробки двовимірних потоків даних

У системі з ПЛІС середнього розміру внутрішньої пам'яті недостатньо для зберігання одного кадру зображення. Тому двовимірний масив, що представляє кадр зображення, поступає з зовнішнього датчика зображення або зберігається у зовнішньому блоці DRAM і пересилається для обробки у ПЛІС піксель за пікселем за законом сканування кадру або за іншим правилом. Цей потік даних має бути тимчасово збережений у внутрішньому буфері, з якого зчитуються дані, що належать певній апертурі (рис.1.7).

Згідно з методом просторового ГСПД, положення пікселя в DDR можна закодувати вектором $\mathbf{K}_i = (s_{1i}, s_{2i}, q, t)^T$, де s_1, s_2 — положення пікселя у кадру в рядку і колонці, відповідно, q — тип пристрою, що зберігає дані, t — момент часу, що розглядається.

При передачі кадру у буферну пам'ять ПЛІС його пікселі відображаються у елементи потоку даних $\mathbf{K}_j' = (s, q', t')^T$, де s — номер рядка кадру. Згідно з теорією проектування систолічних процесорів [199] та методу синтезу паралельних процесорних структур [200], таке відображення $\mathbf{K}_i' = R(\mathbf{K}_i)$, має бути ін'єктивним, лінійним і монотонним. Тут ін'єктивність означає те, що жодні два пікселі не можуть зберігатись в одній комірці пам'яті чи передаватись по лінії зв'язку одночасно. Лінійне і монотонне відображення зберігає залежність попередності пікселів.

Якщо використовується передача кадра за законом сканування по рядках при передачі даних через одну шину, а час відлічується по тактах, то функція відображення є такою:

$$R(s_{1i}, s_{2i}, q, t)^T = (0, q', Ns_{1i} + s_{2i} + t)^T \quad (3.15)$$

де N — ширина кадра.

Буфер даних дає змогу одержувати з пікселів K'_i , які належать до апертури, пікселі, що обробляються, з координатами K''_j , які видається через вихідну вершину буфера (рис. 1.7). Отже, алгоритм функціонування буфера описується множиною векторів-дуг

$$D_j = K''_j - K'_i. \quad (3.16)$$

Множини векторів K'_i , K''_j та D_j формують просторовий ГСПД, за яким синтезується буферна схема таким же чином, як описано у підрозділі 3.2.4.

Розглянемо приклад проектування буфера для фільтрації фільтром з апертурою 3×3 . Нехай кадр зображення має розміри 5×8 , тобто, $N=8$ і складається з пікселів $x_{i,j}$ (див. рис. 3.13).

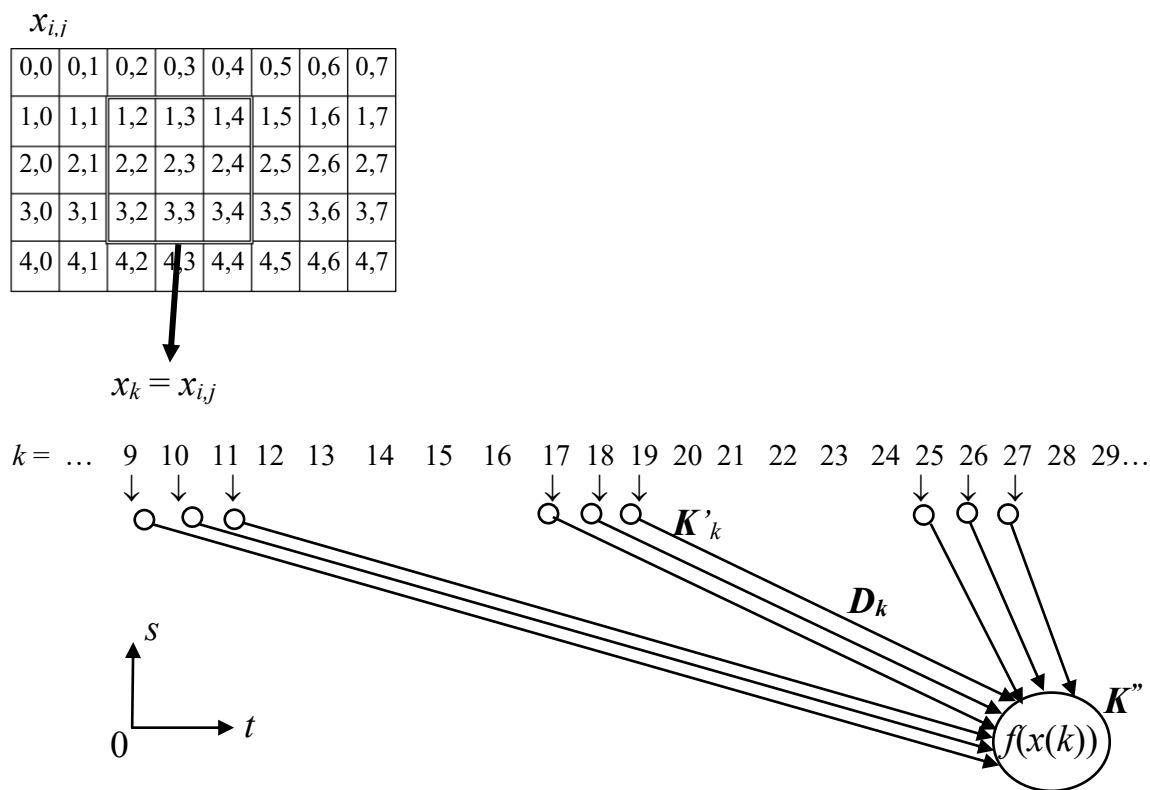


Рис. 3.13 – Відображення кадра у просторовий ГСПД буфера

Таким чином, пікселі кадру зчитуються по рядках і з них формується потік $x_k = x_{ij}$, де $k = Ni + j$. Елементом потоку відповідають вектори-вершини K_k' . Операторна вершина K'' збирає дані дев'яти пікселів і отже, вона зв'язана векторами-дугами $D_k = K_k' - K''$ з вершинами потоку, що позначають пікселі, які належать апертурі.

Урівноважений просторовий ГСПД алгоритму функціонування буфера, який підготовлена згідно з методикою, показаний на рис. 3.14. На ньому вершини затримки, які відображаються у два буфери рядків, виділені прямокутниками. Решта вершин затримки відображуються у відповідні конвеєрні регістри. Результуюча структура співпадає зі структурою, яка зображена на рис. 1.7. Параметр $N = 8$ можна замінити на довільне число так само, як і розміри апертури.

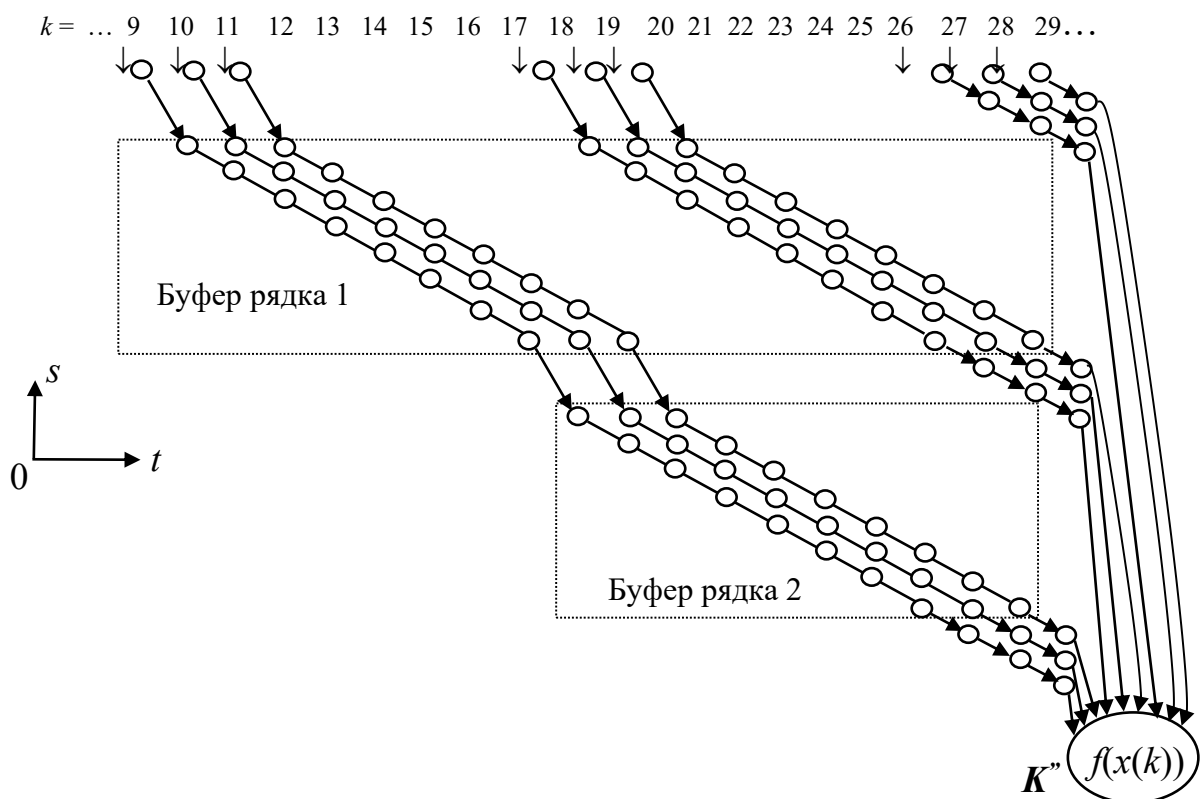


Рисунок 3.14 – Урівноважений просторовий ГСПД буфера

Отже, метод, що пропонується, дає змогу формалізовано будувати буферні схеми для обробки зображень. У даному прикладі блок обробки, що виконує функцію $f(x(k))$, одержує вхідні дані в одному такті. При потребі, метод може забезпечити подання даних у довільних тактах. Наприклад, замість

вершини K' можна вставити підконфігурацію, яка виконує затримки операндів на необхідну кількість тактів.

Запропонований метод синтезу блоків буферної пам'яті ґрунтується на методі відображення просторового ГСПД у обчислювальні ресурси. При цьому як ресурси застосовуються регістри буферної пам'яті. Новий метод полягає у перетворенні двовимірного представлення зображення у одновимірне, побудові просторового ГСПД та описі його мовою опису апаратури, такою як VHDL. На відміну від відомих методів проектування буферних схем, метод дає змогу виконувати їх розробку формалізовано з мінімізацією апаратних витрат, направляючи синтез на одержання буферів типу FIFO або пам'яті довільного доступу чи регістрової пам'яті, забезпечуючи наперед заданий порядок вводу-виводу даних. Рівень формалізації методу дає змогу його реалізувати в системах автоматизованого проектування.

Описи методу і способу опубліковані у статтях [201, 202].

3.2.8. Приклад застосування методу

Метод синтезу буферних схем для обробки двовимірних потоків даних було застосовано при проектуванні експериментального зразка інтелектуальної відеокамери. Вона була виконана на базі плати Lattice HDR-60, яка використовує ПЛІС ECP3-70. Як відеодатчик використано датчик Aptina MT9M024, який виробляє потік HDR-зображення 720×1280 зі швидкістю 60 кадрів на секунду з динамічним діапазоном 120 дБ. Зовнішній вид відеокамери показано на рис. 3.15.

Модуль стиснення HDR-зображення виконує стиснення 18-розрядного чорно-білого зображення до 8-розрядного. Структура модуля показана на рис. 3.16. Вона відтворює алгоритм Retinex, граф якого зображено на рис. 2.2. На вході модуля встановлено буфер, який формує апертуру зображення яке обробляється. Тобто, блок $ADelay$ зберігає у конвеєрних регістрах фрагмент 5 рядків зображення і видає на свої виходи або 25 відліків у апертурі розміром 5×5 ,

або центральний відлік апертури $I_{i,j}$ які затримані на певну затримку відносно один одного.

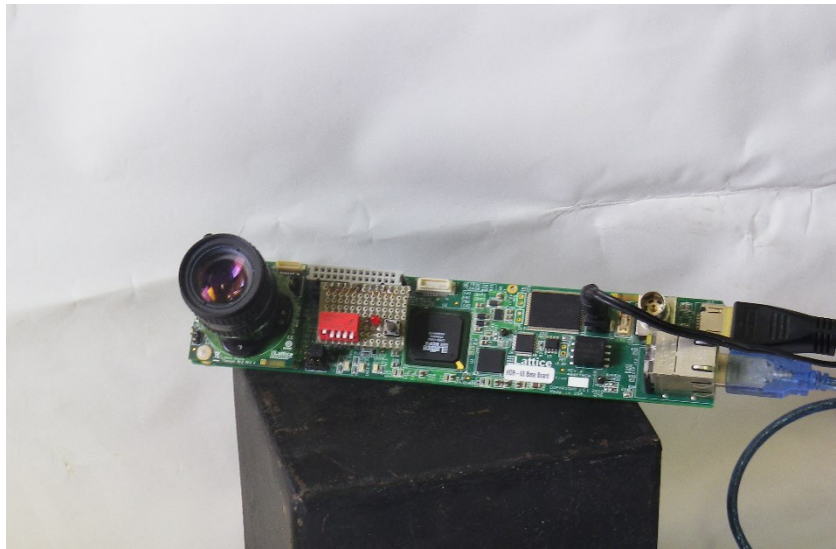


Рисунок 3.15 – Зовнішній вид інтелектуальної відеокамери

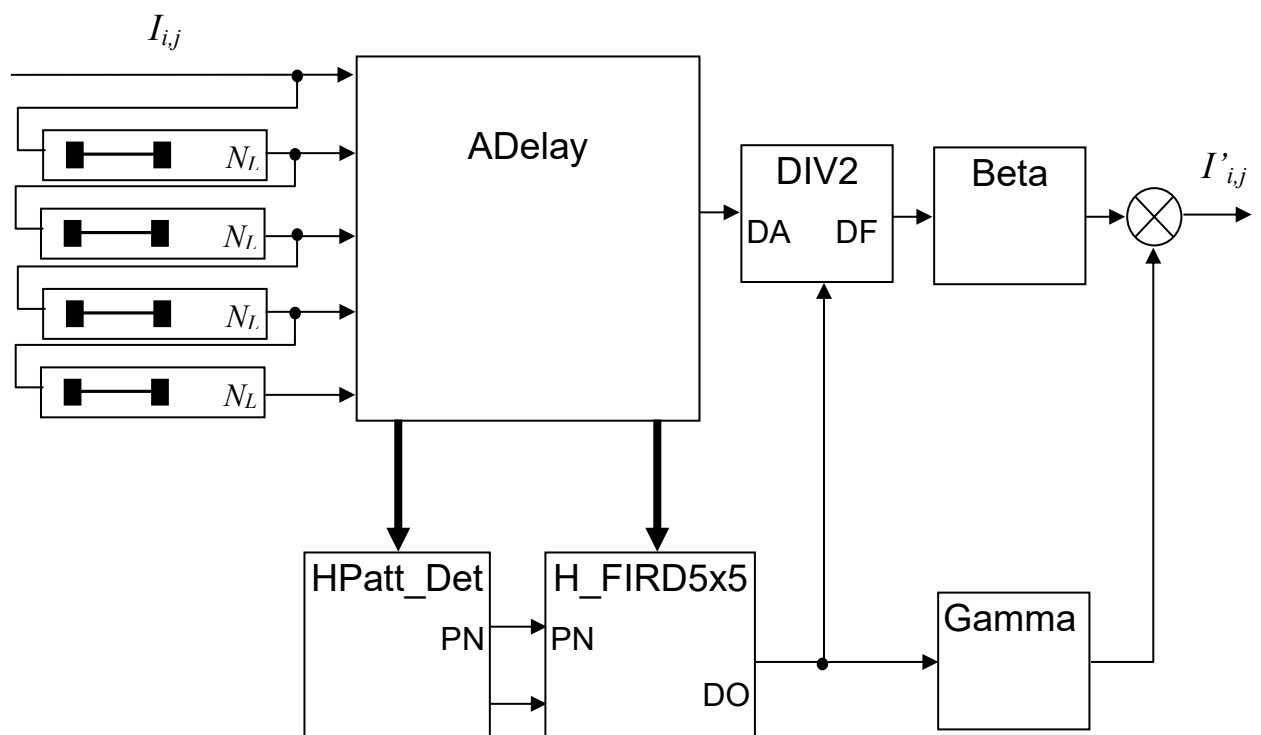


Рисунок 3.16 – Структура модуля стиснення HDR-зображення

Блок аналізатора зображення HPatt_Det аналізує відлік $I_{i,j}$ та його 24 найближчих сусідів на схожість з шістьма елементарними шаблонами (patterns). Це такі шаблони, як горизонтальна, вертикальна, діагональна лінія та точка. Результат порівняння з шостим шаблоном — круглою прямою —

використовується для нормалізації яскравості результатів порівняння. Сигнал результату фільтрації L_{PNij} дорівнює величині градієнта зміни яскравості у околі пікселя I_{ij} у напрямку PN.

Фільтри-аналізatori виконані як настроювані модулі H_An5. В них множення на коефіцієнти ядра фільтра виконується як зсув і додавання відповідного множеного. Результати фільтрації логарифмуються згідно з (2.8). Серед них вибирається максимальне значення, яке нормується згідно з (2.9), яке є вихідним 3-розрядним результатом PS (pattern strength) типу PN згідно з (2.11).

Блок цифрового фільтра H_FIRD5x5 виконує фільтрацію зображення з буфера ADelay фільтром з гаусовим ядром. Завдяки цьому, вихід фільтра L_{ij} є оцінкою яскравості у околі пікселя I_{ij} з певною формою. Причому у складі фільтра є 40 різних ядер (таблиця 2.2), які відрізняються між собою формою ядра та коефіцієнтом згладжування σ ($\sigma = 0,2 \dots 1,6$). Ядро фільтрації динамічно вибирається за сигналами PN і PS.

Таким чином, якщо зображення у околі пікселя L_{ij} не має великого градієнту яскравості, то воно згладжується сильніше, ніж зображення з великим градієнтом. Причому градієнтне зображення сильно згладжується поперек градієнта, наприклад, вздовж лінії та майже не згладжується вздовж градієнта. Завдяки цьому, відфільтроване зображення має розмиття крім областей з краями.

Блок ділення DIV2 виконує операцію

$$R_{ij} = \frac{I_{ij}}{L_{Oij}}, \quad (3.17)$$

де L_{Oij} — значення оцінки яскравості L_{ij} , яке модифіковане таким чином, щоби уникнути ситуації ділення на нуль.

Блок обчислення бета-функції **Beta** виконує обчислення за формулою (2.4) за допомогою кусково-лінійної інтерполяції (2.13) з графіком на рис. 2.6. Блок обчислення гама-функції **Gamma** виконує обчислення за формулою (2.4) за допомогою кусково-лінійної інтерполяції (2.13) з графіком на рис. 2.7.

Після помноження сигналів з виходів блоків бета- та гама-корекції одержується стиснутий чорно-білий 12-розрядний сигнал $I'_{ij} = L'_{ij} \cdot R'_{ij}$.

Блоки HPatt_Det, H_FIRD5x5, DIV2 розроблялись окремо один від одного за методом, описаним у підрозділі 3.1.3. Причому при розробці вважалося, що усі дані, що відносяться до одного результату, приходять на входи кожного з блоків у одному і тому самому такті. Кожен з цих блоків має конвеєрну структуру, яка виконує алгоритм з періодом $L = 1$ такт. Латентні затримки блоків HPatt_Det, H_FIRD5x5 дорівнюють $T_{L1} = 10$ та $T_{L2} = 5$ тактів, відповідно.

Отже, при розробці буфера ADelay приймається граф ГСПД подібний такому, як на рис. 3.12. Згідно з методом, обчислюються різниці векторів вершин пікселів \mathbf{K}'_i у одній і тій самій апертурі та вершин вхідних даних у вказаних блоках \mathbf{K}''_j за (3.18):

$$\mathbf{D}_j = \mathbf{K}''_j - \mathbf{K}'_i = (ds_{1j}, ds_{2j}, 0, t_j)^T, \quad (3.18)$$

де $ds_{1j}, ds_{2j} \in [-2, -1, \dots, 2]$ — різниці координат взятого та центрального пікселів у апертурі, dt — затримка між запуском вершини оператора та появою центрального пікселя апертури. При цьому для векторів-дуг, що заходять в операторну вершину HPatt_Det $t_j = 0$, в вершину H_FIRD5x5 — $t_j = T_{L1}$, а в DIV2 — $t_j = T_{L1} + T_{L2}$.

За часовою функцією (3.19) одержується відповідна затримка

$$d_j = R_T(\mathbf{D}_j) = N ds_{1j} + ds_{2j} + t_j. \quad (3.19)$$

Зважаючи на ці затримки, формується урівноважений ГСПД, подібний до ГСПД на рис. 3.13. При цьому затримка d_j дорівнює проєкції відповідного вектора \mathbf{D}_j на вісь часу. В урівноваженому ГСПД склеюються вершини \mathbf{K}_i , які відповідають поширенню одного і того самого даного і мають одну і ту саму часову координату. Таким чином, один і той самий FIFO буде повторно використаний. Рис. 3.17 ілюструє одержану буферну схему.

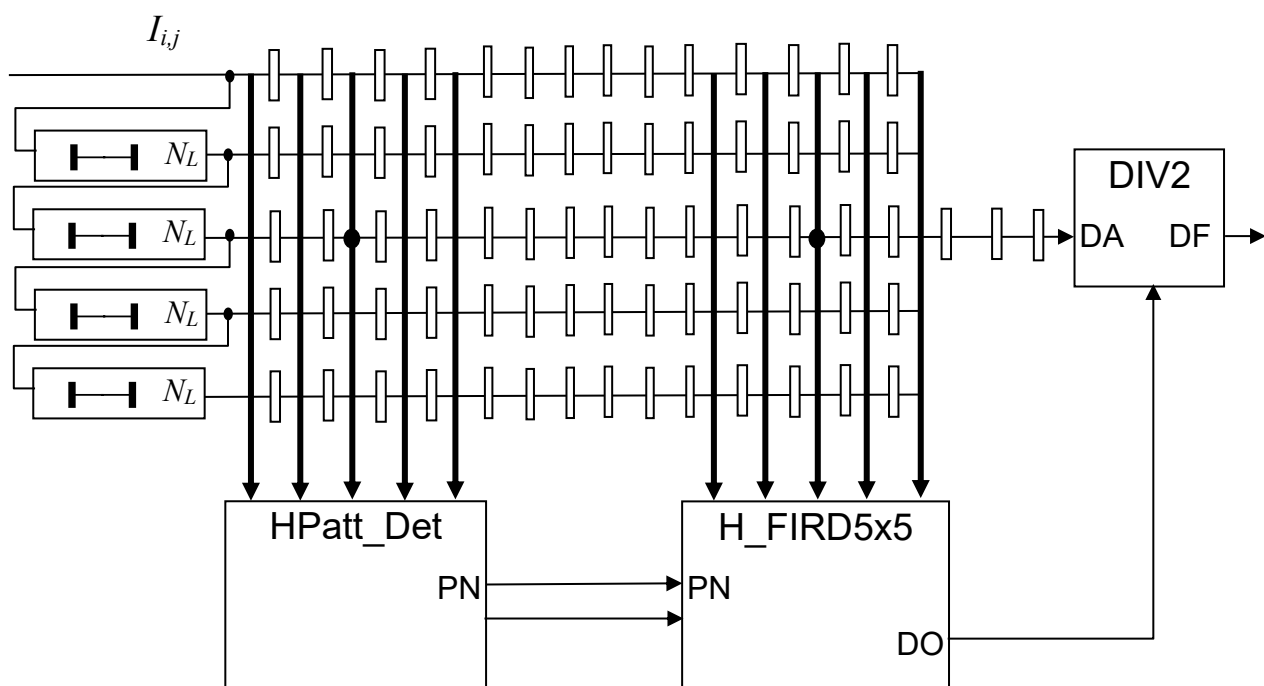


Рисунок 3.17 – Буферна схема для обробки двовимірних потоків даних

На рис. 3.17 тонкими прямокутниками позначені конвеєрні регістри затримки, товстими лініями позначені шини, по яких подаються групи з п'яти даних одночасно, товстою крапкою показано відвід затримки, який відповідає центральному пікселю апертури розміром 5×5 . Аналіз одержаної схеми показує, що три блоки обробки зображення використовують одні й ті самі буфери FIFO на BRAM, що затримують відеосигнал на один рядок. Таким чином, в порівнянні з аналогічними схемами у системах пошуку характерних точок, опублікованих у [113, 115–117], кількість високовартісних буферів FIFO у даному прикладі скорочено у 2,5 рази. Крім того, конвеєрні регістри, від яких немає проміжних відводів, відображаються у регістрові затримки на елементах SRL16, що додатково зменшує число задіяних регістрів. У даному прикладі за рахунок додавання 108 елементів SRL16 зекономлено 594 тригери. Отже, разом з мінімізацією буферів FIFO на BRAM одержано ефективну організацію передачі проміжних результатів між блоками обробки.

Отже, даний приклад свідчить про те, що запропонований метод є дієвим і дає змогу формальним чином проєктувати буферні схеми для обробки двовимірних сигналів з мінімізованими апаратними витратами.

3.3. Проектування конвеєрних пристроїв для обчислення квадратного кореня

3.3.1. Функція квадратного кореня у системах обробки зображень

Функція квадратного кореня є необхідною елементарною функцією у наукових розрахунках, додатках для цифрової обробки сигналів та зображень [158]. Так, вона необхідна при визначенні градієнту у околі характерних точок (1.3), (1.6), для відновлення кольорів при HDR-стисненні (2.1). Останнім часом набувають поширення штучні нейронні мережі, у яких ця функція є необхідною під час їх навчання [203].

Для вирішення задач обробки зображень використовують модулі обчислення \sqrt{x} , які пропонуються фірмами-виробниками ПЛІС та іншими фірмами, що поширюють ліцензії на такі модулі для їхнього конфігурування у ПЛІС [204]. Але ці модулі були розроблені один-два десятиліття тому і в них, як правило, не прийняті до уваги особливості нових ПЛІС, які з'явилися на ринку кілька років тому. Отже, такі модулі потребують удосконалення.

3.3.2. Особливості апаратних ресурсів ПЛІС та складність модулів

Основною конфігурованою логічною одиницею у ПЛІС є ЛТ. У ПЛІС найчастіше використовуються 4-, 6- та 8-входові ЛТ. Останні впроваджені у ПЛІС нових серій, що випускаються фірмами Intel-Altera та AMD-Xilinx.

Мінімальна тривалість тактового інтервалу T_C модуля, який сконфігуровано у ПЛІС, дорівнює затримці його критичного шляху, в який входять послідовно з'єднані ЛТ, мережа поширення переносу та мережа конфігурованих міжз'єднань. Затримка останньої може досягати 50-80% від тривалості T_C . Значення T_C досягає мінімуму тоді, коли у будь-який маршрут від одного тригера до іншого вставлено лише одну ЛТ. Це можливо, коли максимально задіяно тригери, на яких сформовано конвеєрні регістри.

Розглянемо можливості реалізації ступенів конвеєра, коли лише одна ЛТ стоїть у критичному шляху. У таких умовах на основі чотиривходових ЛТ

можлива побудова лише двовходового мультиплексора або двовходового суматора-від'ємника. На основі шестивходових ЛТ можливості побудови різних арифметико-логічних схем значно зростають. Це, наприклад, суматор-від'ємник, до одного входу якого підключено двовходовий мультиплексор, або суматор трьох операндів. Крім того, такий ЛТ конфігурується у блок постійної пам'яті об'ємом до 64 біта. Також у ПЛІС є вбудовані блоки пам'яті об'ємом, в середньому, 18 кілобіт, як наприклад, BRAM у ПЛІС Xilinx. Ці блоки можна використати як постійну пам'ять для збереження таблиці констант.

У сучасних ПЛІС є також від одиниць до тисяч вбудованих блоків множення типу DSP48. Складність апаратного блока множення DSP48 оцінюється у 208 CLBs або у 20 шістнадцятирозрядних суматорів. Також на один блок множення припадає один блок пам'яті BRAM [205].

У пристрої на ПЛІС з апаратними ресурсами, які ефективно використовуються, блоки множення повинні бути зайняті корисними обчисленнями, а решта ресурсів — розподілені між суматорами і мультиплексорами. Тоді на блок множення чи блок пам'яті припадає 20 шістнадцятирозрядних суматорів.

Отже, апаратну складність алгоритму Θ_s , що реалізується у ПЛІС, доцільно оцінювати у кількості необхідних суматорів. При цьому слід приймати до уваги, що складність блоку множення у ПЛІС для розрядності операндів у межах 16-32 оцінюється як апаратні витрати двадцяти суматорів такої самої розрядності. Як часову складність Θ_T алгоритму, слід брати сумарну затримку комбінаційної схеми блоку, яка приведена до затримки одного суматора [196].

Далі розглядатимуться алгоритми добування квадратного кореня з оцінкою їх ефективності для 24-розрядних вхідних даних та результатів з фіксованою комою, які можуть претендувати на реалізацію в ПЛІС. Така розрядність прийнятна для більшості алгоритмів обробки сигналів та зображень.

3.3.3. Алгоритми добування квадратного кореня

Традиційне рішення розрахунку елементарної функції – це обчислення полінома, який є, наприклад, рядом Тейлора [206]. При цьому неможливо

досягти похибки обчислення меншої за 0,2%, якщо $x \in (0;1)$. Крім того, алгоритм потребує виконання багатьох множень. Тому він неприйнятний для реалізації у ПЛІС, хоча, він придатний для кусково-поліноміальної апроксимації.

Відомий ітеративний алгоритм на основі формули Ньютона-Рафсона, який не потребує операцій ділення, який використовує початкове наближене значення, що зберігається в таблиці ПЗП [207]. Причому кожна наступна ітерація алгоритму приблизно подвоює кількість правильних розрядів результату. Тому для обчислення коректного 24-розрядного результату необхідно виконати $n = 2$ ітерації алгоритму та одержати початкове значення з таблиці об'ємом 2^7 . Алгоритм можна виконати за одну ітерацію, якщо таблиця має об'єм 2^{13} слів.

Алгоритм Волдера що належить до класу “цифра за цифрою”, при обчисленні функції $\text{arctgh}(x/y)$ має як побічний результат функцію $x_n = K\sqrt{x^2 - y^2}$ і при підстановці $x = A + 1, y = A - 1$ дає $x_n = K\sqrt{A}$ [208]. Цей алгоритм успішно реалізований у багатьох проектах ПЛІС, як наприклад, у [209]. Недоліками цього алгоритму є необхідність додаткового множення на коефіцієнт $1/K \approx 1,204$, а також повторення деяких ітерацій для збіжності алгоритму.

Більш конструктивним та простішим є алгоритм “цифра за цифрою”, який націлений на одержання саме функції \sqrt{x} [210]. Він оснований на ітеративному процесі, у якому одна змінна прямує до \sqrt{x} , в той час коли інша змінна — прямує до одиниці.

Алгоритм на основі зсуву та віднімання найбільш поширений в апаратних пристроях добування квадратного кореня [211]. Метою алгоритму є одержання наближення двійкового числа $B_i = 0, b_1 b_2 \dots b_i 0 \dots 0$ до значення \sqrt{x} . Він оснований на перевірці знаку остачі $R_i = x - B_i^2$, у результаті чого вибирається чергова цифра b_i . Нехай вважається, що $b_i = 1$, тоді

$$R_i = x - (B_{i-1} + 2^{-i})^2 = R_{i-1} - 2^{-(i-1)}(0, b_1 \dots b_{i-1} 01), \quad (3.18)$$

а якщо виявиться, що $R_i \leq 0$, то

$$R_i = R_{i-1} ; b_i = 0 ; \quad (3.19)$$

і шукається наступна цифра b_{i+1} .

При його реалізації в ПЛІС він має n ступенів, кожен з яких має суматор в режимі віднімання та мультиплексор для вибору R_i , який керується знаковим розрядом суматора [158]. Використовують також алгоритм без відновлення остачі, коли R_i є знакозмінним числом. Але він не має суттєвих переваг, оскільки в ньому використовується знакозмінний суматор, який за складністю такий самий, як суматор з мультиплексором.

Часова Θ_T та апаратна Θ_S оцінки складності цього та попереднього алгоритмів приведені в табл.3.2. При розрахунку Θ_S вважалось, що згадані таблиці реалізовані у ПЛІС як постійна пам'ять, яка має приблизну складність як складність двох та шістдесяти суматорів, відповідно.

Таблиця 3.2 – Витрати для обчислення \sqrt{x}

Алгоритм	Блоків множення	Θ_S	Θ_T
Ітеративний, 1 ітерація	2	102	7
Ітеративний, 2 ітерації	4	86	13
“Цифра за цифрою”	–	52	50
Зі зсувом та відніманням	–	24	24
Модифікований зі зсувом та відніманням	–	22	21

Отже, аналізуючи параметри розглянутих алгоритмів у табл.3.2, можна зробити висновок, що алгоритм на основі зсуву та віднімання реалізується у ПЛІС з мінімальними апаратними витратами. Якщо його виконати у конвеєрному блоці, то такий блок матиме високу пропускну спроможність з періодом тактового інтервалу, що дорівнює затримці одного суматора та мультиплексора.

3.3.4 Модифікація алгоритму зі зсувом та відніманням

Одна i -та ітерація алгоритму зсуву та віднімання полягає у наступних обчисленнях:

$$r_i = R_i - P_i ;$$

```

bi = not sign(ri) ;
Bi+1 = (Bi, bi);
Pi+1 = 2-i+1 (Bi+1, 01);
if (bi == 1)
    Ri+1 = ri;
else
    Ri+1 = Ri;

```

де $\text{sign}(x) = 1$ при $x < 0$, інакше -0 ; $i = 1, \dots, n$; (a, b) – операція конкатенації.

Недолік алгоритму полягає у великій кількості ітерацій, яка дорівнює числу розрядів результату. Помітно скоротити цю кількість можна, якщо перші k ітерацій не виконувати, а результати $k - i$ ітерації знаходити за допомогою табличних функцій.

Такими табличними функціями є

$$B_k = f_B(x_k) \approx \sqrt{x_k} \quad ; \quad R'_k = f_R(x_k) , \quad (3.20), (3.21)$$

де $x_k - 2k$ старших розряди вхідного данного x , причому

$$R_k = R'_k + x - x_k. \quad (3.22)$$

Наприклад, на рис. 3.18 показана функціональна схема, яка виконує модифікований алгоритм для 12-розрядних вхідних даних, $n = 6$ – розрядного результату та $k = 3$. При цьому табличні функції f_B та f_R реалізовані у блоках постійної пам'яті ROMB та ROMR, відповідно.

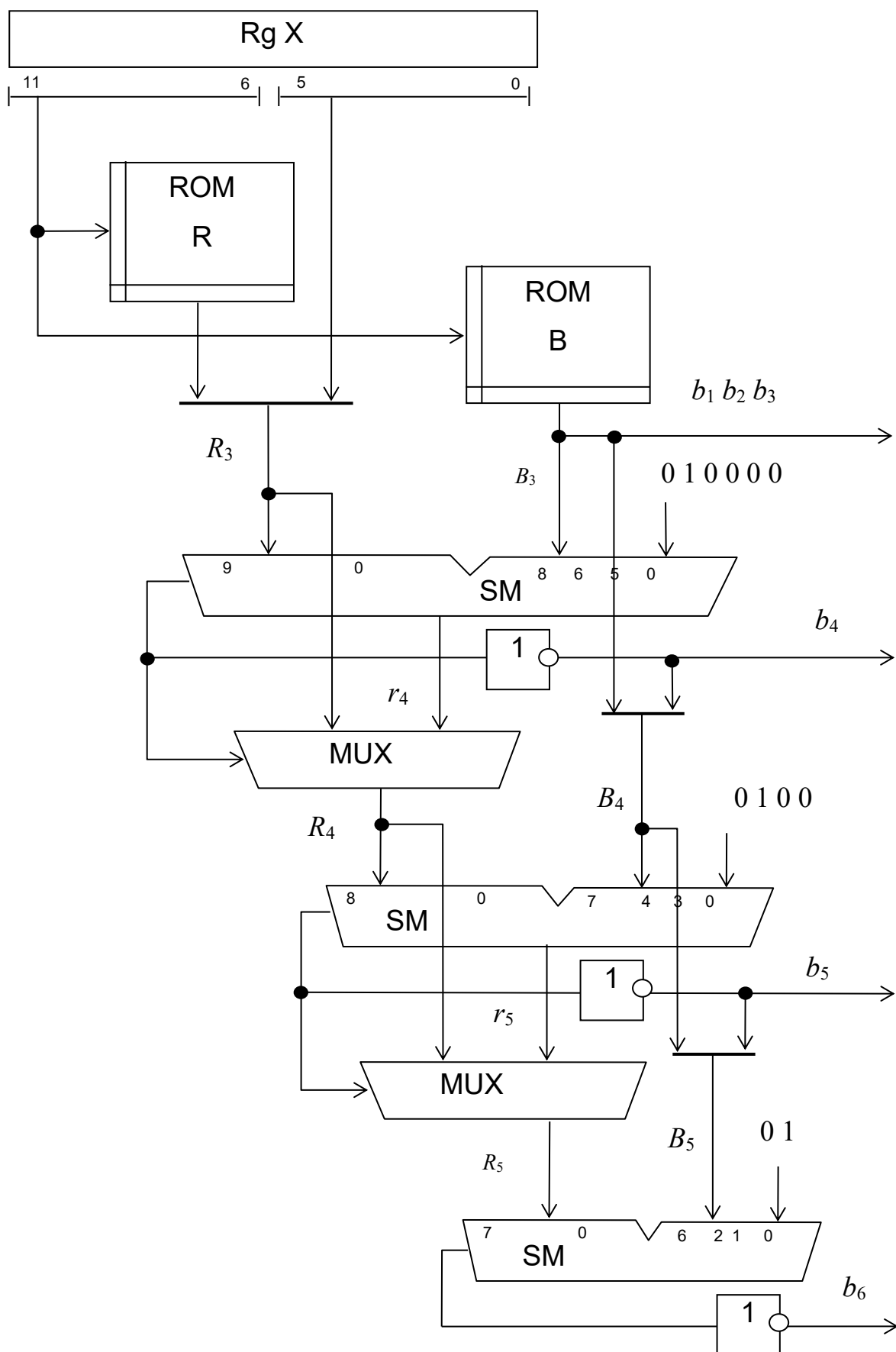


Рисунок 3.18 – Схема блоку добування квадратного кореня

Завдяки застосуванню блоків постійної пам'яті, у даному випадку вдалось зменшити кількість ітерацій з шести до чотирьох, перша з яких – це звертання до цих блоків. Відповідно, схема добування кореня стала меншою на три суматори та три мультиплексори.

Недолік модифікованого алгоритму – блоки постійної пам'яті повинні мати $2k$ -розрядні адресні входи, тому мають значний об'єм при великому k .

Проте, цей алгоритм доцільно використовувати в сучасних ПЛІС при $k \leq 6$. Так, при $k = 3$ для $n = 24$ -розрядних даних параметри блоку квадратного кореня наведені в табл.1., тобто з застосуванням цього алгоритму параметри блоку квадратного кореня покращуються.

Детально про цей алгоритм, його модернізацію та реалізацію викладено у роботі [212].

3.3.5. Реалізація модифікованого алгоритму

Модифікований алгоритм добування квадратного кореня реалізовано у Web-додатку, генераторі віртуальних модулів мовою VHDL, впровадженому на сайті www.kanyevsky.kpi.ua. У ньому організовано ввід параметрів розрядності вхідних та вихідних n даних, розрядності $2k$ адрес блоків постійної пам'яті, наявності вхідного та вихідного регістрів, побудови повністю чи частково конвеєризованої схеми, тобто з $n-k$ або з $\lfloor (n-k)/2 \rfloor$ ступенями конвеєра. При цьому обчислюються оцінки апаратних витрат модуля та максимальної тактової частоти. Наприклад, для 12-розрядних вхідних даних, 6-розрядного результату, $k = 3$ при відсутності конвеєризації генерується поведінковий опис модуля квадратного кореня.

Після синтезу та розміщення модулів блоку квадратного кореня у ПЛІС Xilinx Spartan-6 було одержано ряд характеристик апаратних витрат (рис.3.19) у кількості ЛТ та максимальної тактової частоти, МГц (рис.3.20) для різних значень k при вхідній розрядності 24 та розрядності результату $n = 12$. Розглядались модулі у вигляді комбінаційної схеми, повністю конвеєризовані та частково конвеєризовані модулі.

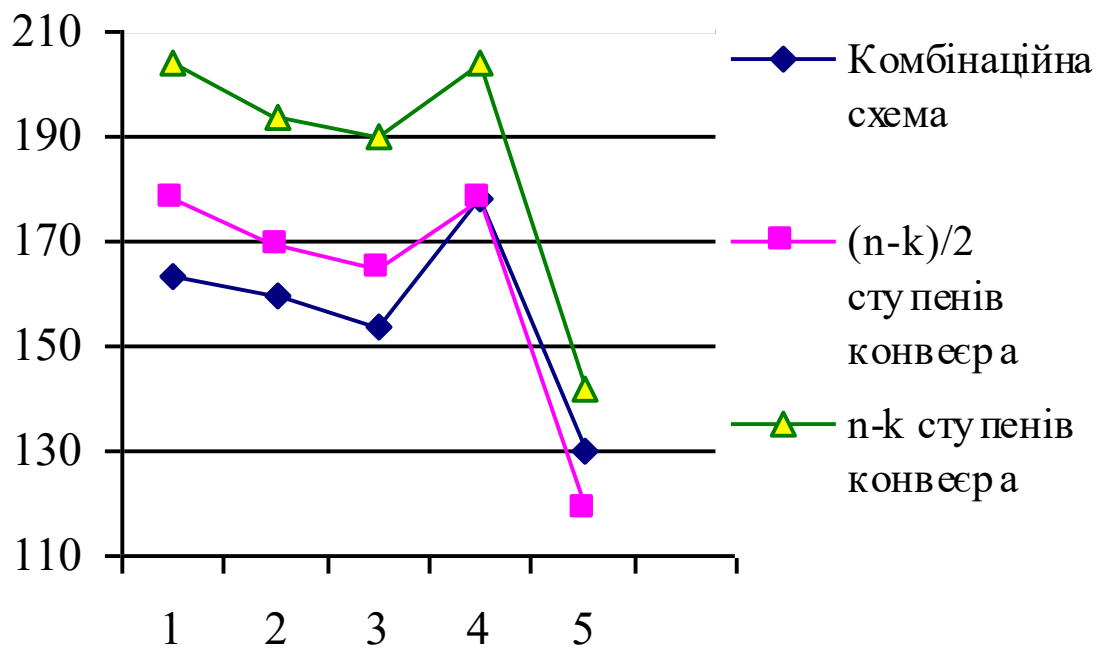


Рисунок 3.19 – Апаратні витрати блоку \sqrt{x} в залежності від k

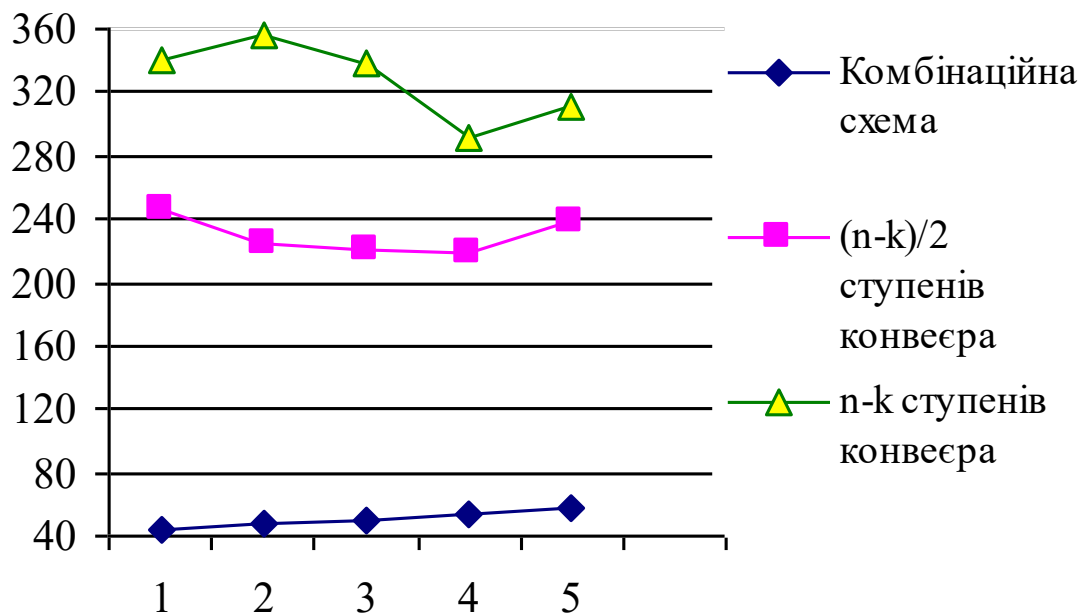


Рисунок 3.20 – Максимальна тактова частота блоку \sqrt{x} в залежності від k

Аналіз цих характеристик показує, що оптимальним за апаратними витратами та за швидкістю є блок з $k = 3$. Блок з $k = 5$ має помітно менші апаратні витрати за рахунок застосування окремого блоку пам'яті BlockRAM для реалізації таблиць. Використання значень $k > 6$ недоцільне через значне зростання необхідного об'єму пам'яті та зменшення тактової частоти.

На рис. 3.21, 3.22 показані характеристики блоків для $k = 3$ в залежності від вихідної розрядності n , яка дорівнює вхідній розрядності. Там же для порівняння приведені характеристики блоків, які пропонуються фірмою Xilinx.

Отже, синтезовані модулі для добування квадратного кореня мають помітно кращі показники за модулі, що пропонуються фірмою Xilinx.

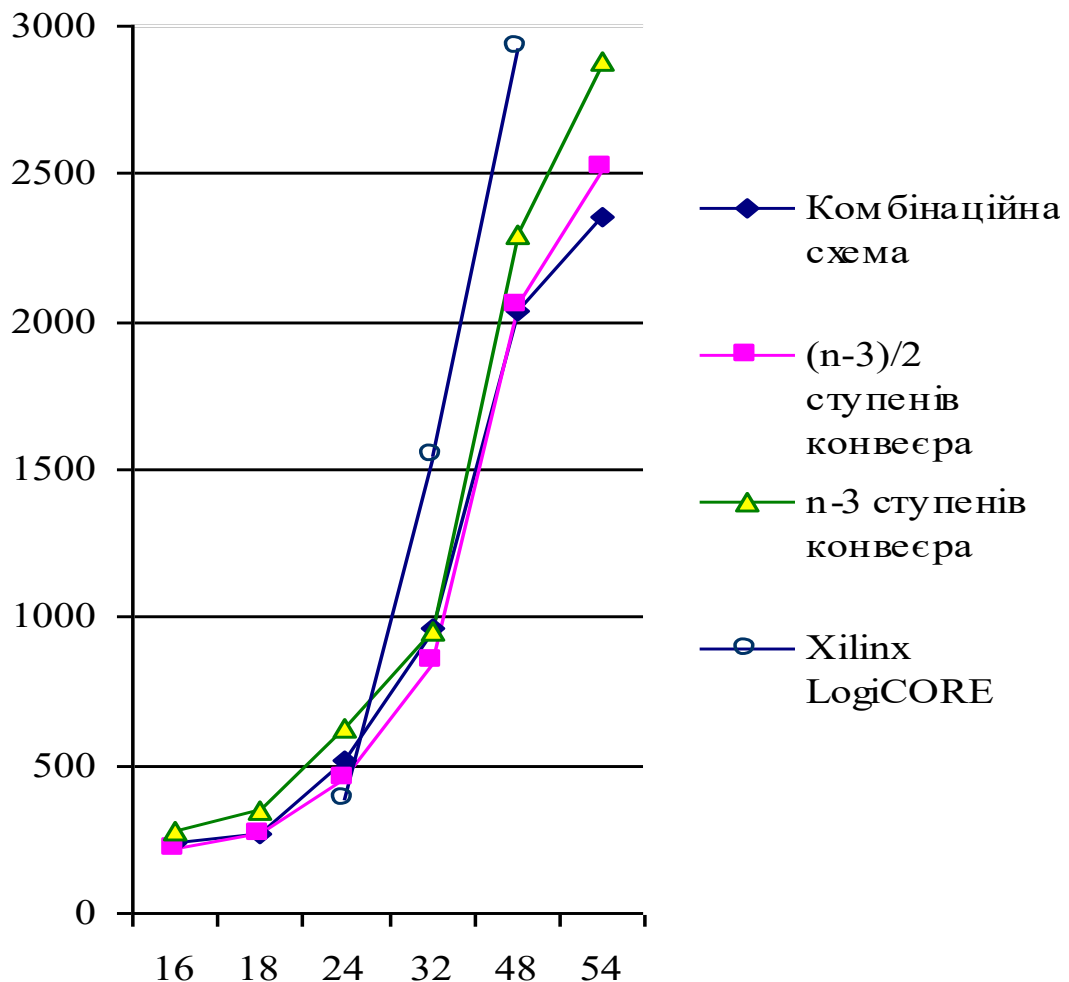


Рисунок 3.21 – Апаратні витрати блоку \sqrt{x} в залежності від n

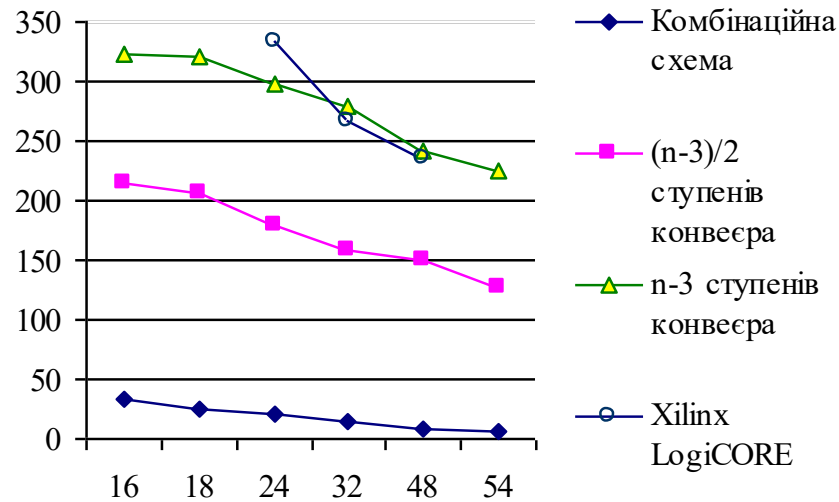


Рисунок 3.22 – Максимальна тактова частота блоку \sqrt{x} в залежності від n

Модулі обчислення \sqrt{x} часто використовуються у блоках для обчислень з плаваючою комою, для яких критичною є латентна затримка між входом та виходом. Цей параметр дорівнює $T_{\text{л}} = \lceil (n - k)/2 \rceil / f_{\text{с}}$ для модулів з частковою конвеєризацією та $T_{\text{л}} = (n - k) / f_{\text{с}}$ – для модулів з повною конвеєризацією, де $f_{\text{с}}$ – тактова частота. Так, для $n = 24$ і $k = 3$ ця затримка дорівнює $T_{\text{л}} = 61,4$ нс і $70,4$ нс, відповідно. Для порівняння, модуль фірми Xilinx має $T_{\text{л}} = 71,9$ нс. Отже, з точки зору зменшення латентної затримки доцільно застосовувати модулі з частковою конвеєризацією, які до того ж мають менші апаратні витрати.

Отже, вибраний алгоритм зі зсувом та відніманням було модифіковано таким чином, що його перші k ітерацій обчислюються табличним способом. За рахунок цього дещо зменшуються апаратні витрати та час виконання алгоритму. Цей алгоритм було реалізовано у Web-додатку, який генерує модулі, описані мовою VHDL стилем для синтезу. Одержані характеристики ряду згенерованих модулів свідчать про те, що ці модулі є кращими за аналогічні модулі, що пропонуються фірмою-виробником ПЛІС.

3.4. Засоби компресії даних в системах машинного зору

3.4.1. Використання компресії в системах оброблення даних

Стиснення інформації дає змогу зменшити об'єм запам'ятовуючих пристроїв. Воно також може збільшити пропускну спроможність каналів передачі даних, але й мінімізувати енергоспоживання. Така мінімізація досягається за рахунок зменшення інтенсивності обмінів даних особливо між віддаленими передавачем та приймачем [213]. Наприклад, на обміни даними між процесором і пам'яттю при виконанні програм може витратись до 75% споживаної енергії за рахунок нагріву провідників та перемикання високострумових буферів [214]. Тому зберігання даних та програм у стисненому виді та їх декомпресія в реальному часі є ефективними як з точки зору зменшення об'єму пам'яті, так і пришвидшення завантаження даних і зменшення енергоспоживання.

В системах розпізнавання образів існує проблема зберігання інформації про образи. У системах з навчанням, основаних на нейронній мережі, десятки мільйонів коефіцієнтів мережі, а також десятки тисяч навчальних образів мають зберігатись у пам'яті великого об'єму, яка є зовнішньою відносно процесорів, що обчислюють цю мережу. Тому для навчання та використання нейронної мережі необхідна організація швидкоплинних потоків даних між пам'яттю та процесорами. Так само у системах розпізнавання образів на основі характерних точок зображення база даних цих точок має зберігатися у пам'яті великого об'єму. У обох випадках застосування безвтратної компресії дає змогу зменшити пропускну здатність каналів читання даних з пам'яті. Іноді можна навіть помістити цю пам'ять всередину ПЛІС, яка виконує задачу розпізнавання, значно підвищивши цим загальну продуктивність.

Для цього слід впроваджувати апаратні засоби безвтратної компресії даних [215]. Так, компресія програми і даних, які завантажуються у внутрішню пам'ять мікроконтролера, дає змогу майже удвічі знизити енергоспоживання системи при несуттєвих накладних витратах на реалізацію декомпресії [216]. Так само, компресія прошивки ПЛІС прискорює її завантаження [217].

Якщо виконати декомпресор як спеціалізований блок, його можливості значно розширюються. Наприклад, декомпресор даних зараз є невід'ємною частиною дисплеїв з надвеликою роздільною здатністю. Завдяки ньому, стає можливою швидкісна передача великих обсягів даних через інтерфейс дисплея [218]. Декомпресор тестових даних, якщо він має невеликі апаратні витрати, суттєво підвищує ефективність систем самодіагностування та діагностики складних систем за рахунок зменшення об'єму тестових даних, що зберігаються [219].

У системах імітаційного моделювання та управління складними об'єктами використовуються апаратні генератори графічних елементів, які в реальному часі виводяться на екран [220]. Те саме стосується систем машинного зору, в яких особливим чином виділяються знайдені об'єкти у зображенні. Однак впровадження відображення цих елементів в реальному часі вимагає значних обчислювальних витрат із використанням графічного прискорювача та часто має надмірну приховану затримку. Тому застосування декомпресора у форматі GIF, в якому реалізовано алгоритм LZW, дає змогу впроваджувати апаратні генератори різноманітних динамічних графічних елементів для відображення на дисплеї у реальному часі [221].

Отже, безвратно апаратна чи апаратно-програмна компресія має корисний ефект у системах технічного зору. Але вона недостатньо досліджена і рідко використовується при реалізації в ПЛІС.

3.4.2. Алгоритми і засоби декомпресії

Загальною метою стиснення даних є зменшення кількості бітів, необхідних для представлення інформації. Стиснення без втрат дозволяє стискати програмний код, цифрові дані, дескриптори характерних точок та інший контент, який є чутливим до викривлення.

Метод стиснення характеризується коефіцієнтом компресії η , який є відношенням довжин файлу до та після компресії. При стисненні за методом Хафмана бітовій послідовності призначаються кодові слова, довжина яких

обернено пропорційна ймовірності таких рядків [222]. Складніший за нього метод арифметичного кодування [223] дає дещо більший коефіцієнт компресії η . Обидва методи вимагають статистичного аналізу даних що обробляються [224].

Методи, що ґрунтуються на динамічному словнику, придатні для стиснення даних без їх попереднього аналізу. Так, алгоритм LZ77 використовує буфер словника та буфер попереднього перегляду [225]. Однак, він не підходить для апаратної реалізації, оскільки об'єми цих буферів є великими для апаратної реалізації. Алгоритм LZRW3 є варіантом алгоритму LZ77, в якому обмежена довжина рядків у словнику. Тому він одержав поширення у ПЛІС [226].

Алгоритм LZ77 разом з кодуванням Хафмена використовується у методі Deflate і застосовується у апаратних декомпресорах GZIP-файлів [227, 228]. Алгоритм LZ78 створює таблицю словника та знаходить у ній найдовший рядок, що відповідає вхідному рядку [229].

Алгоритм LZW є варіантом алгоритму LZ78, який виводить у стиснений файл лише індекс відповідного рядка таблиці словника [223]. В роботі [230] показано, що алгоритм LZW не поступається методу Deflate при стисненні зображень з восьмибітними пікселями. При цьому декомпресії є значно меншою, ніж за алгоритмами LZ77, LZ78 за рахунок того, що робота зі словником значно простіше. Через це алгоритм LZW було вибрано для реалізації.

Існують кілька апаратних реалізацій LZW-декомпресора на ПЛІС, в яких висока швидкість досягнута за рахунок розпаралелювання процесів формування словника, читання з нього рядка і формування вихідної послідовності, а також збільшеної кількості блоків пам'яті. [234, 235]

У таблиці 3.3 порівнюються параметри декомпресорів за алгоритмами LZ77, Deflate і LZW при їх реалізації у ПЛІС. Таке порівняння дає змогу зробити висновок, що декомпресори з алгоритмом LZW мають виграв у об'ємі використаних ресурсів, які припадають на одиницю пропускну здатності

більше, ніж у сім разів а також вимагають менший об'єм оперативної пам'яті у порівнянні з іншими декомпресорами.

Таблиця 3.3. Параметри декомпресорів, приведених у посиланнях

Параметр	Джерело			
	[230]	[231]	[232]	[233]
Алгоритм	LZ77	Deflate	Deflate	LZW
Апаратні витрати, ЛТ	56000	3362	15691	307
ОЗП, BRAM	50	16	30	13
Пропускна здатність, МБ/с	7200	5.4	97.4	280
Кількість ЛТ, яка припадає на одиницю пропускної здатності, ЛТ/МБ/с	7,8	623	161	1,1

Варто згадати, що алгоритм LZW був мало поширений протягом десятирічч через те, що він був запатентований і права на інтелектуальну власність не дозволяли його впроваджувати особливо в апаратних пристроях [233]. Зараз дія патенту скінчилась і з'явилися умови щоб впровадити ширше цей алгоритм.

У багатьох випадках не потрібна екстремальна пропускна здатність розпакувальника. Тому пропонується ідея реалізувати LZW-декомпресію на спеціалізованому компактному мікропроцесорному ядрі у ПЛІС, архітектура якого налаштована на апаратне виконання трудомістких операцій алгоритму. При цьому таке мікропроцесорне ядро крім розпакування здатне виконувати інші алгоритми, наприклад виконувати обробку дескрипторів характерних точок, реалізувати протоколи обміну даними, тестувати систему.

3.4.3. Декомпресія за алгоритмом LZW

Алгоритм декомпресії LZW описується наступним текстом:

```
i = 0;
while (yi ≠ 257){
    if (yi = 256) InitT(C);
    if(yi ∈ C){
        Out(C(yi)); // читається рядок зі словника i
```

```

AddT(C(yi-1) + C1(yi)); // додається
                        //рядок у словник
//запам'ятовується вказівник входу yi
}
else {
    Out(C(yi-1) + C1(yi-1));
    AddT(C(yi-1) + C1(yi-1));
    }
    i++;
}
}

```

Тут $Y = y_0, y_1, \dots, y_{m-1}$ — вхідна послідовність кодів,

$X = x_0, x_1, \dots, x_{n-1}$ — вихідний нестиснений рядок, з символами з алфавіту

$\alpha_0, \alpha_1, \dots, \alpha_{k-1}$,

$C1(y_i)$ — перший символ рядка, що закодовано кодом y_i ,

$Out(C(y_i))$ — функція додавання слова $C(y_i)$ до вихідного файлу,

$AddT()$ — функція додавання до словника нового слова,

«+» — операція конкатенації,

$InitT(C)$ — функція ініціалізації словника C .

Як правило, $k = 256$. Наприклад, якщо індекс 264 означає послідовність АВ, тоді $C(264) = АВ$. Коди 0 — 255, як правило, представляють символи таблиці ASCII. Коли трапляється код 256, то словник очищується, а за кодом 257 закінчується розпакування файлу.

Нехай стиснутий файл — це послідовність 256, 66, 65, 95, 258, 258, 95, 65, 261, 257. Тоді розпакування файлу показане у таблиці 3.4.

Таблиця 3.4. Приклад розпакування за алгоритмом LZW

y_i	y_{i-1}	Читання словника	Запис у словник	Вихід декомпресора
256	—		Очищення	

66	256	B		B
65	66	A	258->BA	BA
95	65	_	259->A_	BA_
258	95	BA	260->_B	BA_ BA
258	258	BA	261->BAB	BA_ BABA
95	258	_	262->BA_	BA_ BABA_
65	95	A	263->_A	BA_ BABA_ A
261	65	BAB	264->AB	BA_ BABA_ ABAB
257	261	(eof)	Кінець	BA_ BABA_ ABAB (eof)

Довжина рядків у словнику S є змінною і досягає сотень символів. Тому недоцільно словник реалізувати як ОЗП, що зберігає одне слово у комірці, а краще виконати як список.

На рис. 3.23 показано зміст ОЗП словника-списку, який має адресу-індекс i , поле $P(i)$ покажчика на попередній символ рядка і поле $C(i)$ чергового символу рядка. Нехай за прикладом в табл. 3.4 на вхід словника поступає код-індекс 261. Тоді з ОЗП вибирається останній символ рядка B та індекс попереднього символу 258, за яким вибирається символ рядка A та індекс першого символу B. Останнім вибирається перший символ B. При цьому чергове слово AB формується як запис в таблицю за індексом 264 символу B та індексу символу A, який є вхідним кодом y_{i-1} що затриманий на один період виконання алгоритму.

Для оптимізації компресії коди y_i мають розрядність, що динамічно змінюється. Спочатку вона дорівнює 9, а наступний код після коду 511 приймає розрядність 10 і т.д. У більшості компресорів ця розрядність не перевищує 12. Тому, як правило, після коду 4095 ставиться код очищення словника.

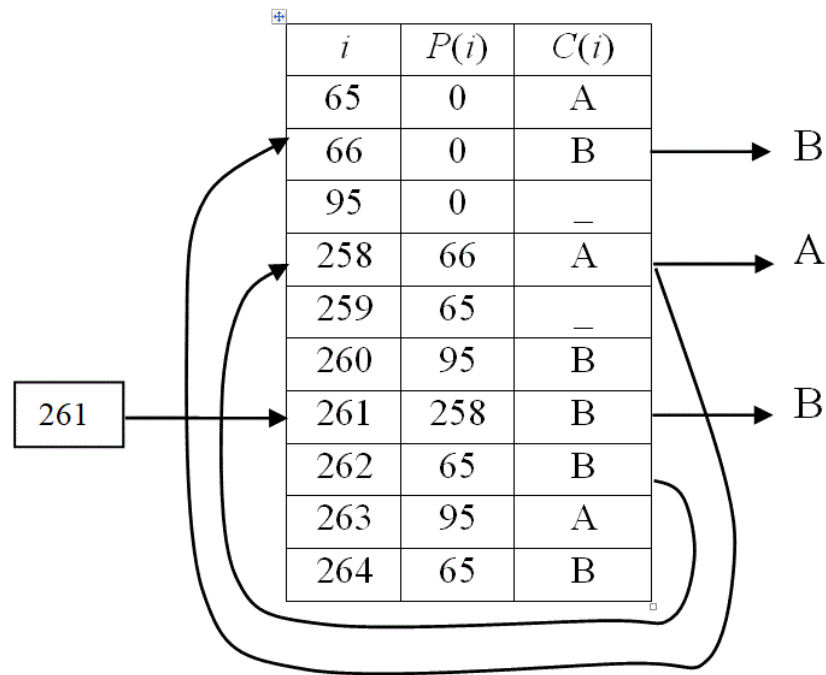


Рисунок 3.23 – Зміст словника і читання з нього слова, що відповідає черговому коду u_i

Отже розпакування файлу Y полягає у виділенні з нього кодів u_i за якими зі словника S зчитується чергове слово, яке приєднується до рядка вихідного файлу X . Причому прочитане зі словника слово має зворотний порядок символів. Він корегується або записом у вихідний буфер з індексною адресацією [235], або записом у стек та читанням з нього. Коефіцієнт компресії LZW залежить від якості словника компресора та змісту файлу, що стискається. У реальних застосуваннях він дорівнює $\eta = 1,6 - 38$ і є задовільним у багатьох застосунках [236].

3.4.4. Модуль для LZW-розпакування

Спочатку, для реалізації модуля було використане мікропроцесорне ядро RISC-процесора для реалізації у ПЛІС [237, 238]. Така реалізація займає порівняно великі апаратні витрати — 845 ЛТ. При цьому досягнута пропускна здатність розпакування складає 4,8 МБ/с. Аналіз продуктивності такого модуля показав, що велика частка часу витрачається на розпакування файлу, а не

на власне декомпресію [239, 240]. Тому для побудови модуля було вибрано більш компактне мікропроцесорне ядро.

В основі модуля для LZW-розпакування лежить мікропроцесорне ядро SM16, описане в [241]. Структура 16-розрядного мікропроцесорного ядра SM16 у складі декомпресора показана на рис. 3.24. Це мікропроцесорне ядро має двостекову архітектуру [242]. Мікропроцесорне ядро містить лічильник команд (PC), ОЗП даних DataRAM, ОЗП програм ProgramROM, регістр команд (IR), стек адрес повернення (RStack), стек даних (DStack), арифметико-логічний пристрій ALU. Регістри T, N є крайніми регістрами стека DStack. Регістр R є вершиною стека RStack, який також служить лічильником ітерацій. Регістри A і B є покажчиками адреси для пересилки масивів даних.

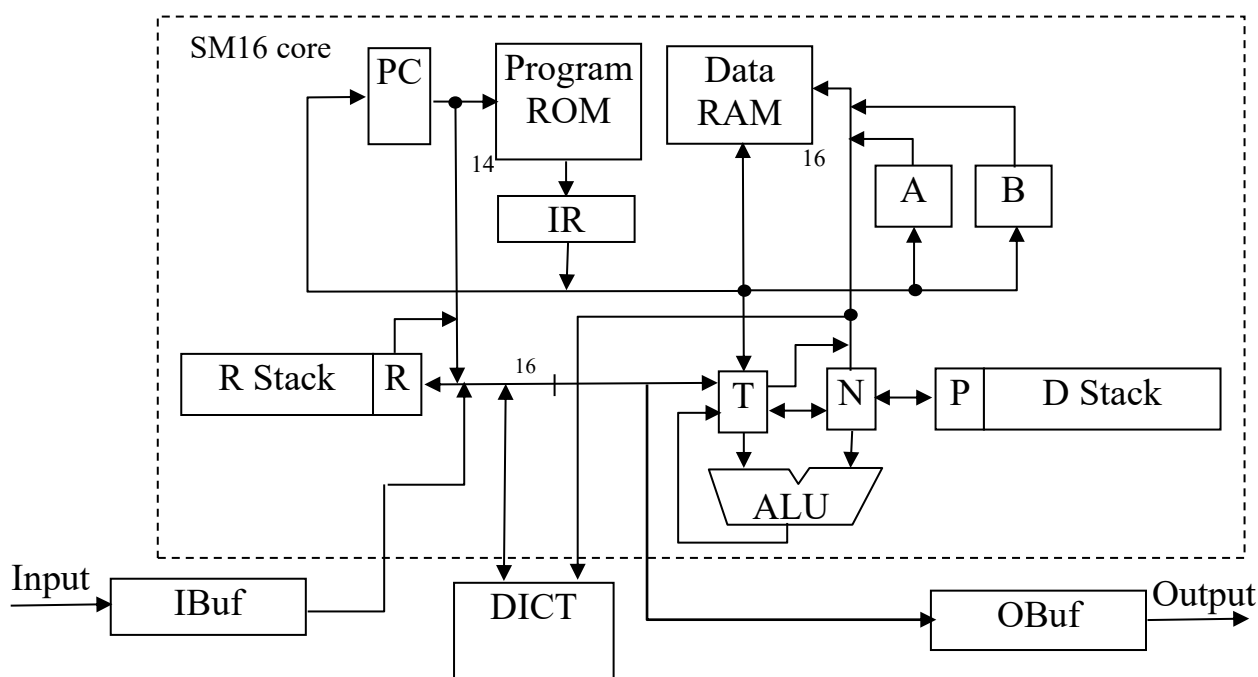


Рисунок 3.24 – Структура мікропроцесорного ядра SM16 у складі LZW-декомпресора

Вхідний файл Y поступає у буфер FIFO IBuf. В ньому реалізована апаратна схема селекції окремих кодів y_i з послідовності байтів файлу Y . Він має виходи сигналів переривання за подіями заповнення буфера і виводу з нього кодів очищення словника і кінця файлу.

Словник DICT зберігає до 4095 індексів $P(i)$ та символів $C(i)$ і виконаний як ОЗП, з якого за адресою i з регістра N зчитується вибране поле $P(i)$ чи $C(i)$. У пам'яті даних DataRAM зберігаються змінні та організований стек за допомогою регістра покажчика A. Його глибина визначається максимальною довжиною упакованого рядка.

Команди ядра виконуються за один такт за винятком команд переходу, завантаження константи що потребують два такти. Завдяки тому, що ядро має стекову архітектуру, виклик підпрограми виконується лише за два такти. При цьому контекст зберігається, а параметри передаються натуральним чином через стеки RStack і DStack. Так само швидко реалізується переривання від сигналів заповнення Ibuf та приходу коду очищення словника.

У 16-розрядній команді є від одного до трьох полів коду операції F1, F2, F3 і поле D, яке зберігає адресу переходу чи константу. Ядро може виконувати одночасно до трьох операцій F1, F2, F3. До системи команд мікропроцесорного ядра були додані кілька команд, які прискорюють звертання до словника і обміни даними. Так, за командою

:L1 OUTB @ab- L1 DJNZ

символ з регістра T записується у вихідний буфер OBuf, а на його місце записується наступний символ з ОЗП даних за адресою індексного регістра A, який після цього декрементується. Якщо зміст регістра R ненульовий, то керування передається на мітку L1, а регістр R декрементується. В іншому випадку відбувається вихід із циклу. Отже, за цією командою у циклі зі стека в ОЗП даних у вихідний буфер переписується розпаковане слово.

Слід відмітити, що користувач ядра SM16 може додати нові команди до системи команд та апаратне забезпечення, якщо вони сприяють пришвидшенню виконання алгоритму. Програма, написана на мові асемблера, яка за синтаксисом співпадає з мовою Forth, компілюється і моделюється у розробленому симуляторі з графічним інтерфейсом. Така програма займає мало

пам'яті в порівнянні з програмами на інших мовах, що є важливим при реалізації ядра у ПЛІС [241, 244].

3.4.5. Результати реалізації в ПЛІС

Мікропроцесорне ядро SM16 описано мовою VHDL. При розміщенні в ПЛІС Xilinx Spartan-6 воно займає лише 632 ЛТ і має максимальну тактову частоту 95 МГц. Ці витрати втричі менше, ніж у 16-розрядного ядра OpenMSP430, яке у такій самій ПЛІС має вчетверо меншу максимальну тактову частоту [243]. Апаратно реалізовані виділення кодів y_i з вхідного потоку, а також фіксація очищення словника, кінця файлу, події зміни довжини коду у буфері IBuf, словник DICT, що має прямий доступ по індексу, вихідний буфер OBuf забезпечують пришвидшення декомпресії удвічі у порівнянні з чисто програмною реалізацією.

Одержані параметри модуля приведені в табл. 3.5. Там же представлений показник якості K , що дорівнює добутку апаратних витрат і об'єму пам'яті, який поділений на тактову частоту. Тобто, чим нижчий показник K – тим ефективніший проєкт. Запрограмований розпакувальник за алгоритмом LZW витрачає, в середньому, 15 тактів, щоб розпакувати один символ. При тактовій частоті ядра 190 МГц декодування здійснюється зі швидкістю 12,6 Мбайт/с.

Для порівняння, була випробувана модель розпакувальника, в якій алгоритм LZW виконувався програмно лише мікропроцесорним ядром SM16 в ПЛІС. При цьому для програмного виділення кодів змінної розрядності із файлу необхідно було до системи команд додати команду зсуву даних. Така команда вимагає додавання схеми паралельного зсуву, яка потребує багато апаратури. Як результат, збільшились апаратні витрати ядра до 239 CLBs, тобто, на 24% в порівнянні з апаратно-програмною реалізацією. Отже, цей приклад свідчить на користь апаратно-програмної реалізації алгоритму LZW.

Таблиця 3.5. Параметри ядер розпакувальників

Ядро розпакувальника	Helion [226]	Джерело [234]	Джерело [235]	Пропонується без IBuf, DICT	Пропонується	Пропонується
Алгоритм	LZRW3	Модифікований LZW	LZW	LZW	LZW	LZW
Мікросхема ПЛІС Xilinx	Virtex-5	Virtex-2	Virtex-7	Spartan-6	Spartan-6	Kintex-7
Апаратні витрати, CLB _s	166	247	139	239	193	224
ОЗП, BRAM	4	8	13	7	7	7
Об'єм ОЗП, кБ	9	18	29.25	15.75	15.75	15.75
Тактова частота, МГц	226	50	300	95	95	190
η, Мбайт/с	180	140	280	3.1	6,3	12,6
Показник якості К	6.61	88.9	6.02	39.6	32.0	32.0
Можливість додавання функціональності	немає	немає	немає	є	є	є

Даний декомпресор програє у пропускній здатності у порівнянні з чисто апаратними декомпресорами. Зате він має вдвічі менші витрати у числі блоків BRAM [235] і здатний розпакувати файли з довгими рядками x_i , тобто, він забезпечує потенційно більший коефіцієнт компресії η .

Вказані у таблиці апаратні розпакувальники спроектовані за технологією опису мовою VHDL і компіляції у САПР ПЛІС. Тому для їх модернізації вони потребують трудомісткі повторні цикли проєктування, які пов'язані з додаванням об'ємів апаратури, що пропорційні доданий функціональності. У запропонованому модулі декомпресора нескладно додавати функціональність без зміни структури модуля. Наприклад, він може служити для оброблення дескрипторів характерних точок. Отже, його безумовною перевагою є можливість перенастроювання на виконання багатьох інших алгоритмів, таких як розпакування GIF-файлів, протоколи обміну даними, керуючі алгоритми з

великою кількістю розгалужень. При цьому незначне збільшення об'єму пам'яті програм є властивістю стекової архітектури [242].

3.5. Висновки до розділу 3

У даному розділі проаналізовані способи представлення алгоритмів, що обробляють потоки даних і вибрано метод проєктування конвеєрних пристроїв для обробки сигналів на основі просторового графа синхронних потоків даних як формалізований метод для структурного синтезу спеціалізованих конвеєрних пристроїв з заданою пропускнуою спроможністю.

Проаналізовані відомі методи проєктування буферних схем для пристроїв обробки зображень і встановлено що методи розглядають побудову буферних схем окремо для кожного конвеєрного блока обробки і не розглядають організацію передачі проміжних результатів між цими блоками. Також методи не приймають до уваги можливість мінімізації числа конвеєрних регістрів за рахунок їх заміни на елементи SRL16.

Вперше запропоновано спосіб проєктування буферів з конвеєрних регістрів у ПЛІС, який відрізняється від існуючих способів формальною побудовою функціональної схеми, в якій використовуються елементи SRL16, за рахунок чого p регістрів замінюються на k логічних таблиць, де $p/k = 2 - 16$.

Розроблено новий метод синтезу буферних схем для обробки двовимірних потоків даних, який дає змогу будувати буферні схеми, використовуючи метод відображення просторового ГСПД у обчислювальні ресурси. При цьому метод дає змогу виконувати розробку схем формалізовано з мінімізацією апаратних витрат, направляючи синтез на одержання буферів типу FIFO або пам'яті довільного доступу чи регістрової пам'яті, забезпечуючи наперед заданий порядок вводу-виводу даних.

Зроблено огляд алгоритмів обчислення квадратного кореня і вибрано алгоритм зі зсувом та відніманням як найефективніший алгоритм для реалізації у ПЛІС.

Удосконалено алгоритм та структура модуля обчислення квадратного кореня, який на відміну від відомого алгоритму зі зсувом та відніманням має меншу затримку обчислення за рахунок застосування блоків постійної пам'яті, які зберігають результати обчислень перших ітерацій алгоритму.

Вказано на важливість впровадження безвтратних декомпресорів у системи розпізнавання образів. Проаналізовано алгоритми безвтратної компресії і вибрано алгоритм LZW як такий, що може забезпечити досить великий коефіцієнт компресії, високу пропускну спроможність та невеликі апаратні витрати при його реалізації в ПЛІС. Встановлено, що апаратні модулі для безвтратної декомпресії даних дають можливість як зменшити обсяги даних, що зберігаються чи пересилаються по каналах зв'язку, так і зменшити енергоспоживання пристроїв для вбудованого застосування.

Вперше запропоновано апаратно-програмний безвтратний декомпресор за алгоритмом LZW на основі процесорного ядра зі стековою архітектурою для реалізації у ПЛІС, який у порівнянні з програмною реалізацією має вдвічі більшу пропускну здатність, а в порівнянні з апаратною реалізацією даного алгоритму забезпечує багатофункціональне використання процесорного ядра.

Розроблено апаратно-програмний модуль для LZW-декомпресії, який конфігурується у ПЛІС різних серій. Завдяки апаратно-програмній реалізації, модуль декомпресора, який при апаратних витратах у 193 CLBs ПЛІС технології Xilinx має швидкість розпакування 12.6 МБ/с і на відміну від апаратних декомпресорів має можливість переналаштовуватись та збільшувати кількість алгоритмів, що виконуються при відсутніх або невеликих додаткових апаратних витратах. Зокрема, він налаштовується на розпакування GIF-файлів. Пропускна здатність декомпресії підвищується кратно при збільшенні кількості таких модулів, які працюють паралельно.

ВИСНОВКИ

Дана робота присвячена вирішенню технічної проблеми створення обчислювачів для розпізнавання образів у зображеннях. Для цього досліджені сучасні методи та засоби розпізнавання образів у зображеннях, на основі чого розроблений новий метод пошуку характерних точок у зображенні та розроблені нові способи апаратної реалізації алгоритмів пошуку характерних точок у зображенні для втілення нового методу. На основі виконаних дослідження та розробок зроблено наступні висновки.

1. Запропоновано новий алгоритм адаптивної фільтрації на основі блоку аналізу зображення, який детектує локальні градієнтні характеристики і формує з них зображення ознак. На відміну від алгоритму білатеральної фільтрації, новий алгоритм має учетверо менше операцій множення і не потребує обчислень з підвищеною точністю.

2. Розроблено новий метод пошуку характерних точок у зображенні з етапами формування піраміди зображень ознак за допомогою запропонованого алгоритму адаптивної фільтрації, фільтрації зображень піраміди з застосуванням нового алгоритму MHN-фільтрації, пошуку координат характерних точок, вибору характерних точок і формування дескрипторів знайдених характерних точок. На відміну від існуючих методів пошуку характерних точок, таких як SIFT, новий метод виконує пошук таких точок у несприятливих умовах освітленості та має обсяг обчислень зменшений у кілька разів.

3. Запропоновано спосіб проектування буферів з конвеєрних регістрів у ПЛІС, який відрізняється від існуючих способів формальною побудовою функціональної схеми, в якій використовуються елементи SRL16, за рахунок чого p регістрів замінюються на k логічних таблиць, де $p/k = 2 - 16$.

4. Розроблено новий метод синтезу буферних схем для обробки двовимірних потоків даних, який дає змогу проєктувати буферні схеми формалізовано з мінімізацією апаратних витрат, направляючи синтез на одержання

буферів типу FIFO або пам'яті довільного доступу чи реєстрової пам'яті, забезпечуючи наперед заданий порядок вводу-виводу даних.

5. Удосконалено алгоритм та структура модуля обчислення квадратного кореня, який на відміну від відомого алгоритму зі зсувом та відніманням має меншу затримку обчислення за рахунок застосування блоків постійної пам'яті. Удосконалений алгоритм впроваджено у Web-застосунку, який генерує модулі обчислення квадратного кореня для реалізації у ПЛІС.

6. Результати дисертаційного дослідження впроваджені у 2 НДР: НДР ЗОТ/2027 «Методи і засоби відображення потокових алгоритмів у конфігуровані комп'ютери» № держреєстрації 0117U005087; 2. НДР «Методи, моделі та комп'ютерні засоби виявлення деструктивного впливу в медіапросторі», № держреєстрації 0121U110662; та у освітній процес кафедри СПіСКС, при підготовці та викладанні курсу лекцій «Цифрова обробка сигналів та зображень».

7. Проведені дослідження можуть бути продовженні у напрямках оптимізації детекторів ознак у блоках аналізу зображення, апаратної реалізації етапів формування дескрипторів та їх порівняння з еталонами, комбінування нового методу з глибокою нейронною мережею.

8. Розроблені методи, способи і апаратні пристрої можуть знайти широке впровадження як в системах технічного зору, таких як системи для розпізнавання образів у зображеннях, так і в інших системах оброблення сигналів та зображень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Machine Vision Handbook : reference work / Batchelor B. G, Ed. Springer, 2012. 2270 p.
2. Reisslein M., Rinner B., Roy-Chowdhury A. K. Smart Camera Networks. *Computer*. 2014. Vol. 47, No 5. P. 23–25.
3. Sha A. Google Tensor G2 vs Snapdragon 8+ Gen 1 vs A16 Bionic: It's No Longer About the CPU. Beebom : website. October 11, 2022. URL: <https://beebom.com/google-tensor-g2-vs-snapdragon-8-gen-1-vs-a16-bionic/> (дата звернення: 12.12.2022).
4. Bailey D. Implementing Machine Vision Systems Using FPGAs / *Machine Vision Handbook* : reference work / B. G. Batchelor – Ed. Springer, 2012. P. 1104–1132.
5. Maggiani L., Salvadori C., Petracca M., Pagano P., Saletti R. Reconfigurable FPGA architecture for computer vision applications in Smart Camera Networks. *2013 International Conference on Distributed Smart Cameras (ICDSC)* : proceedings of the Seventh International Conference, Palm Springs, CA, USA, 29 Oct. 2013 – 01 Nov. IEEE, 2013. P. 1–6. DOI: <https://doi.org/10.1109/ICDSC.2013.6778212>.
6. Hannig F. A Quick Tour of High-Level Synthesis Solutions for FPGAs / *FPGAs for Software Programmers* / D. Koch, F. Hannig, D. Ziener, Ed-s. Springer, 2016. P. 49–59.
7. Micheli G. D. Synthesis and Optimization of Digital Circuits. McGraw-Hill, 1994. 576 p.
8. Meyer-Baese U. Digital Signal Processing with Field Programmable Gate Arrays : 4th Ed. Springer, 2014. 930 p.
9. Designing with Xilinx FPGAs: Using Vivado. / S. Churiwala — Ed. Switzerland : Springer, 2017. 270 p. DOI: <https://doi.org/10.1007/978-3-319-42438-5>.

10. Balarin F., Kondratyev A., Watanabe Y. High Level Synthesis / *Electronic Design Automation for IC System Design, Verification, and Testing* / Ed-s: Lavagno L. et al. CRC Press, 2016. P. 229–274.
11. Börger E. Abstract State Machines: A Method for High-Level System Design and Analysis / *Formal Methods: State of the Art and New Directions* / P. Boca, J. P. Bowen, J. Siddiqi, Ed-s. Springer, 2009. P. 79–116.
12. Camposano R. Path-based scheduling for synthesis. *IEEE Trans. on Computer-Aided Design*. 1991. № 10. P. 85–93.
13. Haubelt C. Falk J., Keinert J., Schlichter T., et al. A SystemC-based design methodology for digital signal processing systems. *EURASIP Journal on Embedded Systems*. Hindawi Pub., 2007. Article ID 47580, 22 p.
14. Distanto A., Distanto C. Handbook of Image Processing and Computer Vision : in 3 vol. Springer, 2020. V. 3 : From Pattern to Object. 676 p.
15. Fieguth P. An Introduction to Pattern Recognition and Machine Learning. Springer, 2022. 471 p.
16. Computer vision : a reference guide / K. Ikeuchi, Ed. Springer, 2021. 1406 p.
17. Serhiienko P., Orlova M., Loutskii H. Overview of Algorithms for Graphic Pattern Recognition in Images. *Security, Fault Tolerance, Intelligence* : proceedings of the International Conference ICSFTI'2020, Kyiv, Ukraine, May 13 – June 15, 2020. P. 250–257.
18. Das R. Content-Based Image Classification: Efficient Machine Learning Using Robust Feature Extraction Techniques. CRC Press, 2021. 179 p.
19. Russell S., Norvig P. Artificial Intelligence: A Modern Approach. 4th ed. Pearson, 2022. 1166 p.
20. Goodfellow I., Bengio Y., Courville A. Deep Learning. MIT Press, 2016. 785 p.
21. Fukushima K., Miyake S. Neocognitron: A new algorithm for pattern recognition tolerant of deformations and shifts in position. *Pattern Recognition*. 1982. Vol. 15, No 6. P. 455–469.

22. LeCun Y., Boser B., Denker J. S., Henderson D., Howard R. E., Hubbard W., Jackel L. D. Backpropagation applied to handwritten zip code recognition. *Neural Computation*. 1989. Vol. 1, No 4. P. 541–551.
23. Simonyan K., Vedaldi A., Zisserman A. Learning Local Feature Descriptors Using Convex Optimisation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Aug. 2014. Vol. 36, No 8. P. 1573–1585. DOI: <https://doi.org/10.1109/TPAMI.2014.2301163>.
24. Chen L.-C., Papandreou G.; Kokkinos I., Murphy K., Yuille A. L. DeepLab: Semantic Image Segmentation with Deep Convolutional Nets, Atrous Convolution, and Fully Connected CRFs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018, 1 April. Vol. 40, No 4. P. 834–848., doi: 10.1109/TPAMI.2017.2699184.
25. Girshick R., Donahue J., Darrell T., Malik J. Region-Based Convolutional Networks for Accurate Object Detection and Segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2016, 1 Jan. Vol. 38, No 1. P. 142–158. DOI: <https://doi.org/10.1109/TPAMI.2015.2437384>.
26. He K., Gkioxari G., Dollár P., Girshick R. Mask R-CNN. *2017 IEEE International Conference on Computer Vision (ICCV)* : proceedings of the international conference, Venice, Italy, 22–29 Oct. 2017. IEEE, 2017. P. 2980–2988. DOI: <https://doi.org/10.1109/ICCV.2017.322>.
27. Uijlings J. R. R., Koen E. A., Sande V. D., Gevers L., Smeulders A. W. M. Selective search for object recognition. *International Journal of Computer Vision*. 2013. Vol. 104, No 2. P. 154–171. DOI: <https://doi.org/10.1007/s11263-013-0620-5>.
28. Zitnick C. L., Dollar P. Edge boxes: Locating object proposals from edges. *European Conference on Computer Vision* : proceedings of the 13th European Conference, Zurich, Switzerland, September 6–2, 2014. Springer, 2014. P. 391–405. DOI: http://dx.doi.org/10.1007/978-3-319-10602-1_26.

29. He K., Zhang X., Ren S., Sun J. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2015. Vol. 37, No 9. P. 1904–1916. DOI: <https://doi.org/10.1109/tpami.2015.2389824>.
30. Arandjelovic R., Gronat P., Torii A., Pajdla T., Sivic J. NetVLAD: CNN Architecture for Weakly Supervised Place Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* : proceedings of the international conference, Las Vegas, NV, USA, 27–30 June 2016. IEEE, 2016. P. 5297–5307. DOI: <https://doi.org/10.1109/CVPR.2016.572>.
31. Jaderberg M., Simonyan K., Zisserman A., Kavukcuoglu K. Spatial transformer networks. *NIPS'15 : Proceedings of the 28th International Conference on Neural Information Processing Systems*, Montreal, Canada, December 7–12, 2015. MIT Press, 2015. Vol. 2. P. 2017–2025.
32. Khan S., Rahmani H., Shah S. A. A., Bennamoun M. A Guide to Convolutional Neural Networks for Computer Vision. Springer, 2018. 207 p.
33. Chopra S., Hadsell R., LeCun Y. Learning a similarity metric discriminatively, with application to face verification. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)* : proceedings of the international conference, San Diego, CA, USA, 20–25 June 2005, IEEE, 2005. Vol. 1. P. 539–546. DOI: <https://doi.org/10.1109/CVPR.2005.202>.
34. Saxe A. M., McClelland J. L., Ganguli S. Exact solutions to the nonlinear dynamics of learning in deep linear neural networks. ArXiv, 2013. 22 p. (Preprint. ArXiv; 1312.6120). DOI: <https://doi.org/10.48550/arXiv.1312.6120>.
35. Hinton G. E., Osindero S., Teh Y. W. A fast learning algorithm for deep belief nets. *Neural Computation*. 2006. Vol. 18, No 7. P. 1527–1554. DOI: <https://doi.org/10.1162/neco.2006.18.7.1527>.
36. Khan S. H., Hayat M., Bennamoun M., Sohel F. A., Togneri R. Cost-Sensitive Learning of Deep Feature Representations From Imbalanced Data. *IEEE*

- Transactions on Neural Networks and Learning Systems*. Aug. 2018. Vol. 29, No 8. P. 3573–3587. DOI: <https://doi.org/10.1109/TNNLS.2017.2732482>.
37. Azizpour H., Razavian A. S., Sullivan J., Maki A., Carlsson S. Factors of Transferability for a Generic ConvNet Representation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2016, 1 Sept. Vol. 38, No 9. P. 1790–1802. DOI: <https://doi.org/10.1109/TPAMI.2015.2500224>.
 38. Anderson A., Shaffer K., Yankov A., Corley C. D., Hodas N. O. Beyond fine tuning: A modular approach to learning on small data. ArXiv, 2016. (Preprint. ArXiv; 1611.01714). DOI: <https://doi.org/10.48550/arXiv.1611.01714>.
 39. Krizhevsky A., Sutskever I., Hinton G. E. Imagenet classification with deep convolutional neural networks. *Communications of the ACM*. June 2017. Vol. 60, No 6. P. 84–90. DOI: <https://doi.org/10.1145/3065386>.
 40. Rahmani H., Mian A., Shah M. Learning a Deep Model for Human Action Recognition from Novel Viewpoints. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2018, 1 March. Vol. 40, No 3. P. 667–681. DOI: <https://doi.org/10.1109/TPAMI.2017.2691768>.
 41. Srivastava N., Hinton G. E., Krizhevsky A., Sutskever I., Salakhutdinov R. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*. 2014. Vol. 15, No 1. P. 1929–1958.
 42. Wan L., Zeiler M., Zhang S., Cun Y. L., Fergus R. Regularization of neural networks using dropconnect. *ICML'13 : Proc. of the 30th International Conference on Machine Learning, Atlanta GA USA, June 16–21, 2013*. JMLR.org, 2013. P. 1058–1066.
 43. Bailey B. Machine Learning's Growing Divide. *Semiconductor Engineering* : website. 11 Jan. 2018. URL: <https://semiengineering.com/machine-learning-growing-divide> (дата звернення: 13.10.2021).
 44. Hareth S., Mostafa H., Shehata K. A. Low power CNN hardware FPGA implementation. *2019 31st International Conference on Microelectronics (ICM)* : proceedings of the international conference, Cairo, Egypt, 15–18 December

2019. IEEE, 2019. P. 162–165, DOI: <https://doi.org/10.1109/ICM48031.2019.9021904>.
45. Kyriakos A., Papatheofanous E.-A., Charalampos B., Petrongonas E., Soudris D., Reisis D. Design and Performance Comparison of CNN Accelerators Based on the Intel Movidius Myriad2 SoC and FPGA Embedded Prototype. *2019 International Conference on Control, Artificial Intelligence, Robotics & Optimization (ICCAIRO)* : proc. of the international conf., Athens, Greece, 08–10 December 2019. IEEE, 2019. P. 142–147, DOI: <https://doi.org/10.1109/ICCAIRO47923.2019.00030>.
46. Simonyan K., Zisserman A. Very Deep Convolutional Networks for Large-Scale Image Recognition. *ICLR2015* : Proc. of the 3rd International Conference on Learning Representations, San Diego, CA, USA, May 7–9 2015. URL: <https://arxiv.org/abs/1409.1556>.
47. Szegedy C., Liu W., Jia Y., Sermanet P., Reed S., Anguelov D., Erhan D., Vanhoucke V., Rabinovich A. Going deeper with convolutions. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* : proceedings of the international conference, Boston, MA, USA, 07–12 June 2015. IEEE, 2015. P. 1–9. DOI: <https://doi.org/10.1109/CVPR.2015.7298594>.
48. He K., Zhang X., Ren S., Sun J. Deep Residual Learning for Image Recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* : proceedings of the international conference, Las Vegas, NV, USA, 27–30 June 2016. IEEE, 2016. P. 770–778. DOI: <https://doi.org/10.1109/CVPR.2016.90>.
49. Leng C. C., Xu W., Cheng I., Basu A. Graph matching based on stochastic perturbation. *IEEE Transactions on Image Processing*. 2015. Vol. 24, No 12. P. 4862–4875.
50. Belongie S., Malik J., Puzicha J. Shape matching and object recognition using shape contexts. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2002. Vol. 24, No 4. P. 509–522.

51. Yang J. F., Liang J., Shen H., Wang K., Rosin P. L., Yang M. H. Dynamic match kernel with deep convolutional features for image retrieval. *IEEE Transactions on Image Processing*. 2018. Vol. 27, No 11. P. 5288–5302.
52. Pernici F., Del Bimbo A. Object tracking by oversampling local features. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2014. Vol. 36, No 12. P. 2538–2551.
53. Khan S. A., Hussain A., Usman M., Nazir M., Riaz N., Mirza A. M. Robust face recognition using computationally efficient features. *Journal of Intelligent and Fuzzy Systems*. 2014. Vol. 27, No 6. P. 3131–3143.
54. Zhang Z. Y., Tian Z., Ding M. T., Basu A. Improved robust kernel subspace for object-based registration and change detection. *IEEE Geoscience and Remote Sensing Letters*. 2013. Vol. 10, No 4. P. 791–795.
55. Brown L. G. A survey of image registration techniques. *ACM Computing Surveys*. 1992. Vol. 24, No 4. P. 325–376.
56. Amiri M., Rabiee H. R. RASIM: A novel rotation and scale invariant matching of local image interest points. *IEEE Transactions on Image Processing*. Dec. 2011. Vol. 20, No 12. P. 3580–3591.
57. Weng D. W., Wang Y. H., Gong M. M., Tao D. C., Wei H., Huang D. DERF: Distinctive efficient robust features from the biological modeling of the P ganglion cells. *IEEE Transactions on Image Processing*. Aug. 2015. Vol. 24, No 8. P. 2287–2302.
58. Nixon M. S., Aguado A. S. Feature Extraction and Image Processing for Computer Vision. Fourth Edition. London : Academic Press, 2020. 650 p.
59. Morrone M. C., Owens R. A. Feature Detection from Local Energy. *Pattern Recognition Letters*. 1987. Vol. 6, No 5. P. 303–313.
60. Kovesi P. Image Features from Phase Congruency. *Videre: Journal of Computer Vision Research*. 1999. Vol. 1, No 3. P. 1–27.
61. Kass M., Witkin A., Terzopoulos D., Snakes: Active Contour Models. *International Journal of Computer Vision*. 1988. Vol. 1, No 4. P. 321–331.

62. Harris C., Stephens M. A Combined Corner and Edge Detector. *AVC88 : Proceedings of Fourth Alvey Vision Conference*, University of Manchester, 31st August – 2nd September 1988. University of Sheffield Printing Unit, 1988. P. 147–151.
63. Lowe D. G. Distinctive Image Features from Scale-Invariant Key Points. *International Journal of Computer Vision*. 2004. Vol. 60, No 2. P. 91–110.
64. Mikolajczyk K., Tuytelaars T. Local Invariant Feature Detectors: A Survey. *Foundations and Trends in Computer Graphics and Vision*. 2007. Vol. 3, No 3. P. 177–280.
65. Bay H., Eas A., Tuytelaars T., Van Gool L. Speeded-Up Robust Features (SURF). *Comput. Vis. Image Und.* 2008. Vol. 110, No 3. P. 346–359.
66. Mikolajczyk K., Schmid C. A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Oct. 2005. Vol. 27, No 10. P. 1615–1630.
67. Morel J. M., Yu G. ASIFT: a new framework for fully affine invariant image comparison. *SIAM J. Imaging Sci.* 2009. Vol. 2, No 2. P. 438–469.
68. Pang Y. W., Li W., Yuan Y., Pan J. Fully affine invariant SURF for image matching. *Neurocomputing*. 2012. Vol. 85. P. 6–10.
69. Tola E., Lepetit V., Fua P. DAISY: an efficient dense descriptor applied to wide baseline stereo. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2010. Vol. 32, No 5. P. 815–830.
70. Cui C. H., Ngan K. N. Scale- and affine-invariant fan feature. *IEEE Transactions on Image Processing*. 2011. Vol. 20, No 6. P. 1627–1640.
71. Sedaghat A., Ebadi H. Remote sensing image matching based on adaptive binning SIFT descriptor. *IEEE Transactions on Geoscience and Remote Sensing*. Oct. 2015. Vol. 53, No 10. P. 5283–5293.
72. Strecha C., Bronstein A. M., Bronstein M. M., Fua P. LDAHash: Improved matching with smaller descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. Jan. 2012. Vol. 34, No 1, P. 66–78.

73. Baber J., Dailey M. N., Satoh S., Afzulpurkar N., Bakhtyar M. BIG-OH: Binarization of gradient orientation histograms. *Image and Vision Computing*. 2014. Vol. 32, No 11. P. 940–953.
74. Tombari F., Franchi A., Di Stefano L. BOLD features to detect texture-less objects. *2013 IEEE International Conference on Computer Vision (ICCV)* : proceedings of the international conference, Sydney, NSW, Australia, 1–8 Dec. 2013. IEEE, 2013. P. 1265–1272.
75. Su X. Q., Lin W. Y., Zheng X. Z., Han X. T., Chu H., Zhang X. Y. A new local-main-gradient-orientation HOG and contour differences based algorithm for object classification. *IEEE International Symposium on Circuits and Systems (ISCAS)* : proceedings of the international conference, Beijing, China, 19–23 May 2013. IEEE, 2013. P. 2892–2895.
76. Rosten E., Drummond T. Machine Learning for High-Speed Corner Detection. *Computer Vision — ECCV 2006* : Proceedings of the 9th European Conference on Computer Vision, Graz, Austria, May 7–13 2006. Springer, 2006. P. 430–443.
77. Alahi A., Ortiz R., Vandergheynst P. Freak: Fast Retina Keypoint. *CVPR '12: Proceedings of the 2012 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 16–21 2012. IEEE, 2012. P. 510–517.
78. Calonder M., Lepetit V., Strecha C., Fua P. BRIEF: Binary Robust Independent Elementary Features. *Lecture Notes in Computer Science*. Springer-Verlag, 2010. Vol. 6314, Computer Vision – ECCV 2010. P. 778–792.
79. Rublee E., Rabaud V., Konolige K., Bradski G. ORB: An efficient alternative to SIFT or SURF. *2011 IEEE International Conference on Computer Vision (ICCV)* : proceedings of the international conference, Barcelona, Spain, 6–13 Nov. 2011. IEEE, 2011. P. 2564–2571.
80. Hasanbelliu E., Giraldo L. S., Principe J. C. Information theoretic shape matching. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2014. Vol. 36, No 12. P. 2436–2451.

81. Litman R., Bronstein A. M. Learning spectral descriptors for deformable shape correspondence. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2014. Vol. 36, No 1. P. 171–180.
82. Wang B., Brown D., Gao Y. S., Salle J. L. MARCH: Multiscale-arch-height description for mobile retrieval of leaf images. *Information Sciences*. 2015. Vol. 302. P. 132–148.
83. Hong B. W., Soatto S. Shape matching using multiscale integral invariants. *IEEE Transactions on Pattern Analysis and Machine Intelligence*. 2015. Vol. 37, No 1. P. 151–160.
84. Kadir T., Brady M. Scale, saliency and image description. *Int. J. Comput. Vis.* 2001. Vol. 45, No 2. P. 83–105.
85. Borji A., Cheng M. M., Jiang H., Li J. Salient Object Detection: A Benchmark, *IEEE Transactions on IP*. 2015. Vol. 24, No 12. P. 5706–5722.
86. Wang W., Shen J., Shao, L. Video Salient Object Detection via Fully Convolutional Networks. *IEEE Transactions on IP*. 2018. Vol. 27, No 1. P. 38–49.
87. Virmani J., Dey N., Kumar V. PCA-PNN and PCA-SVM based CAD systems for breast density classification. / *Applications of Intelligent Optimization in Biology and Medicine* / Hassanien A.-E., Grosan C., Tolba M. F. — Eds. Springer, Cham, Switzerland, 2016. P. 159–180.
88. Chaki J., Parekh R., Bhattacharya S. Plant leaf recognition using a layered approach. *2016 International Conference on Microelectronics, Computing and Communications (MicroCom)* : proceedings of the international conference, Durgapur, India, 23-25 January 2016. IEEE, 2016. P. 1–6.
89. Chaki J., Parekh R., Bhattacharya S. Plant leaf classification using multiple descriptors: A hierarchical approach. *Journal of King Saud University-Computer and Information Sciences*. 2020. Vol. 32, No 10. P. 1158–1172.
90. Automated system for crops recognition and classification / A. M. AlShahrani et al. *Computer Vision: Concepts, Methodologies, Tools, and Applications* : in 4 vol. / IRMA. Hershey, PA : IGI Global, 2018. V. 2. P. 1208–1223.

91. Chaki J., Parekh R. Designing an automated system for plant leaf recognition. *International Journal of Advances in Engineering & Technology*. 2012. Vol. 2, No 1. P. 149–158.
92. Govindaraju V., Shi Z., Schneider J. Feature Extraction Using a Chaincoded Contour Representation of Fingerprint Images. *Lecture Notes in Computer Science* / Kittler J., Nixon M.S. — Eds. Springer, Berlin, Heidelberg, 2003. Vol. 2688, Audio- and Video-Based Biometric Person Authentication. P. 268–275.
93. Huerta-Hernández C. E., Sánchez-Cruz H. Chain Code Histograms for Rotation Invariance. *2014 International Conference on Computational Science and Computational Intelligence (CSCI 2014)* : proceedings of the international conference, Las Vegas, NV, USA, 10–13 March 2014. IEEE, 2014. P. 190–193.
94. Chaki J., Parekh R., Bhattacharya S. Plant leaf recognition using texture and shape features with neural classifiers. *Pattern Recognition Letters*. 2015. Vol. 58. P. 61–68.
95. Chaki J. An efficient two-stage Palmprint recognition using Frangi-filter and 2-component partition method. *2018 Fifth International Conference on Emerging Applications of Information Technology (EAIT)* : proceedings of the international conference, Kolkata, India, 12–13 January 2018. IEEE, 2018. P. 1–5.
96. Huang D., Zhu C., Wang Y., Chen L. HSOG: A novel local image descriptor based on histograms of the second-order gradients. *IEEE Transactions on Image Processing*. Nov. 2014. Vol. 23, No 11. P. 4680–4695.
97. Zhang D., Lu G. Generic Fourier descriptor for shape-based image retrieval. Proceedings. *2002 IEEE International Conference on Multimedia and Expo (ICME)* : proceedings of the international conference, Lausanne, Switzerland, 26–29 August 2002. IEEE, 2002. Vol. 1. P. 425–428.

98. Nabout A. A., Tibken B. Wavelet Descriptors for Object Recognition Using Mexican Hat Function. *IFAC Proceedings Volumes*. 2005. Vol. 38, Issue 1. P. 1107–1112.
99. Sermanet P., Eigen D., Zhang X., Mathieu M., Fergus R., LeCun Y. Overfeat: Integrated Recognition, Localization and Detection Using Convolutional Networks. ArXiv, 2013. (Preprint. ArXiv; 1312.6229).
100. Gul M. S. K., Gunturk B. K. Spatial and Angular Resolution Enhancement of Light Fields Using Convolutional Neural Networks. *IEEE Transactions on IP*. 2018. Vol. 27, No 5. P. 2146–2159.
101. Zhang K., Zuo W., Chen Y., Meng D., Zhang L. Beyond a Gaussian Denoiser: Residual Learning of Deep CNN for Image Denoising. *IEEE Transactions on IP*. 2017. Vol. 26, No 7. P. 3142–3155.
102. Zhang Q. S., Zhu S. C. Visual Interpretability for Deep Learning: a Survey. *Frontiers of Information Technology & Electronic Engineering*. 2018. Vol. 19, No 1. P. 27–39.
103. Leutenegger S., Chli M., Siegwart R. Y. BRISK: binary robust invariant scalable keypoints. *2011 IEEE International Conference on Computer Vision (ICCV)* : proceedings of the international conference, Barcelona, Spain, 6–13 Nov. 2011. IEEE, 2011. P. 2548–2555.
104. Beis J., Lowe D. G. Shape indexing using approximate nearest-neighbour search in highdimensional spaces. *CVPR '97: Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition (CVPR '97)*, Puerto Rico, June 17–19 1997. IEEE, 1997. P. 1000–1006.
105. Arya S., Mount D. M., Netanyahu N. S., Silverman R., Wu A.Y. An optimal algorithm for approximate nearest neighbor searching. *Journal of the ACM*. 1998. Vol. 45. P. 891–923.
106. Khan N., McCane B., Mills S. Better than SIFT? *Machine Vision and Applications*. 2015. Vol. 26, No 6. P. 819–836. DOI: <https://doi.org/10.1007/s00138-015-0689-7>.

107. Qasaimeh M., Sagahyroon A. Shanableh T. A Parallel Hardware Architecture for Scale Invariant Feature Transform (SIFT). *2014 International Conference on Multimedia Computing and Systems (ICMCS)* : proc. of the int. conf., Marrakech, Morocco, 14–16 April 2014. IEEE, 2014. P. 1–6.
108. Yao L., Feng H., Zhu Y., Jiang Z., Zhao D., Feng W. An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher. *2009 International Conference on Field-Programmable Technology* : proceedings of the international conference, Sydney, NSW, Australia, 9–11 Dec. 2009. IEEE, 2009. P. 30–37. DOI: <https://doi.org/10.1109/FPT.2009.5377651>.
109. Mizuno K., Noguchi H., He G., Terachi Y., Kamino T., Kawaguchi H., Yoshimoto M. Fast and Low-Memory-Bandwidth Architecture of SIFT Descriptor Generation with Scalability on Speed and Accuracy for VGA Video. *2010 International Conference on Field Programmable Logic and Applications* : proc. of the international conference, Milan, Italy, 31 Aug. – 02 Sept. 2010. IEEE, 2010. P. 608–611. DOI: <https://doi.org/10.1109/FPL.2010.119>.
110. Kerowsky P., Stabernack B. A Full-Featured FPGA-Based Pipelined Architecture for SIFT Extraction. *IEEE Access*. 2021. Vol. 9. P. 128564–128573. DOI: <https://doi.org/10.1109/ACCESS.2021.3104387>.
111. Bonato V., Marques E., Constantinides G. A. A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection. *IEEE Transactions on Circuits and Systems for Video Technology*. Dec. 2008. Vol. 18, No 12. P. 1703–1712. DOI: <https://doi.org/10.1109/TCSVT.2008.2004936>.
112. Zhong S., Wang J., Yan L., Kang L., Cao Zh. A real-time embedded architecture for SIFT. *Journal of Systems Architecture*. 2013. Vol. 59, Issue 1. P. 16–29. DOI: <https://doi.org/10.1016/j.sysarc.2012.09.002>.
113. Vourvoulakis J., Kalomiros J., Lygouras J. Fully pipelined FPGA-based architecture for real-time SIFT extraction. *Microprocessors Microsyst.* Feb. 2016. Vol. 40. P. 53–73. DOI: <https://doi.org/10.1016/j.micpro.2015.11.013>.

114. Li S.-A., Wang W.-Y., Pan W.-Z., Hsu C.-C. J., Lu C.-K. FPGA-Based Hardware Design for Scale-Invariant Feature Transform. *IEEE Access*. 2018. Vol. 6. P. 43850–43864. DOI: <https://doi.org/10.1109/ACCESS.2018.2863019>.
115. Fischer J., Ruppel A., Weißhardt F., Verl A. A rotation invariant feature descriptor O-DAISY and its FPGA implementation. *2011 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)* : proc. int. conf., San Francisco, CA, USA, 25–30 Sept. 2011. P. 2365–2370, DOI: <https://doi.org/10.1109/IROS.2011.6094813>.
116. Shin G., Sung J., Kim Y.-H., Lee Y. H. Hardware Design of Feature Point Extraction using SIFT Algorithm. *Advanced Science and Technology Letters*. 2014. Vol. 48, CIA 2014. P. 42–46.
117. Ni Q., Wang F., Zhao W., Gao P. FPGA-based Binocular Image Feature Extraction and Matching System. *ICMSSP '19* : Proceedings of the 2019 4th International Conference on Multimedia Systems and Signal Processing, May 2019. Association for Computing Machinery, 2019. P. 182–187. <https://doi.org/10.1145/3330393.3330429>.
118. Chiu L.-C., Chang T.-S., Chen J.-Y., Chang N. Y.-C. Fast SIFT Design for Real-Time Visual Feature Extraction. *IEEE Transactions on Image Processing*. Aug. 2013. Vol. 22, No 8. P. 3158–3167. DOI: <https://doi.org/10.1109/TIP.2013.2259841>.
119. Jiang J., Li X., Zhang G. SIFT Hardware Implementation for Real-Time Image Feature Extraction. *IEEE Transactions on Circuits and Systems for Video Technology*. July 2014. Vol. 24, No 7. P. 1209–1220. DOI: <https://doi.org/10.1109/TCSVT.2014.2302535>.
120. Banterle F., Artusi A., Debattista K., Chalmers A. Advanced High Dynamic Range Imaging : 2nd Ed. CRC Press, 2018. 315 p.
121. Debevec P. E., Malik J. Recovering high dynamic range radiance maps from photographs. *SIGGRAPH'97* : Proc. of the 24th annual conference on

- Computer graphics and interactive techniques,.1997. ACM Press/Addison-Wesley, 1997. P. 369–378.
122. Guenter B. K., Hodgins J. K., Tumblin J. Two methods for display of high contrast images. *ACM Transaction on Graphics*. 1999. V. 18, No 1. P. 56–94.
 123. Larson G. W., Rushmeier H., Piatko C. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Trans. on Visualization and Computer Graphics*. 1997. V. 3, No 4. P. 291–306.
 124. Pattanaik S. N., Ferwerda J. A., Fairchild M. D., Greenberg D. P. A multiscale model of adaptation and spatial vision for realistic image display. *SIGGRAPH '98 : Proc. 25th annual conference on Computer graphics and interactive techniques*, 1998. ACM, 1998. P. 287–298.
 125. Meylan L., Susstrunk S. High dynamic range image rendering with a retinex-based adaptive filter. *IEEE Trans. on Image Processing*. 2006. V.15, No 9. P. 2820–2830.
 126. Land E. Recent advances in retinex theory. *Vision Research*. 1986. V. 19, No 1. P. 7–21.
 127. Сергиенко А. М., Симоненко В. П. Алгоритмические модели обработки потоков данных. *Электронное моделирование*. 2008. Т. 30, № 6. С. 49–60.
 128. Saponara S., Fanucci L., Marsi S., Ramponi G., Kammler D., Witte E. M. Application-Specific Instruction-Set Processor for Retinex-Like Image and Video Processing. *IEEE Trans. On CAS — II: Express Briefs*. 2007. V. 54, No 7. P. 596–600.
 129. Provenzi E. Computational Color Science: Variational Retinex-like Methods. Wiley, 2017. 126 p.
 130. Paris S., Kornprobst P., Tumblin J., Durand F. Bilateral Filtering: Theory and Applications. *Foundations and Trends in Computer Graphics and Vision*. 2008. V. 4, No 1. P. 1–73.

131. Choudhury P., Tumblin J. The trilateral filter for high contrast images and meshes. *EGRW '03* : Proc. 14th Eurographics workshop on Rendering, Aire-la-Ville, Switzerland, June 25–27, 2003. Eurographics Association, 2003. P. 186–196.
132. Fattal R., Lischinski D., Werman M. Gradient domain high dynamic range compression. *SIGGRAPH'02* : Proc. 29th annual conference on Computer graphics and interactive techniques, San Antonio Texas, July 23–26, 2002. ACM, 2002. P. 249–256.
133. Yee H., Pattanaik S. Segmentation and adaptive assimilation for detail-preserving display of high-dynamic range images. *The Visual Computer*. 2003. V. 19, No 7/8. P. 457–466.
134. Krawczyk G., Myszkowski K., Seidel H.-P. Lightness perception in tone reproduction for highdynamic range images. *Computer Graphics Forum*. Dublin, Ireland: Blackwell, 2005. Vol. 24, Issue 3 (EUROGRAPHICS2005). P. 635–645.
135. Hassaballah M., Abdelmgeid A. A., Alshazly H. A. Image features detection, description and matching / *Image Feature Detectors and Descriptors: Foundations and Applications* / A. I. Awad, M. Hassaballah, Eds. Springer, 2016. P. 11–46.
136. Lattice HDR-60 Camera Development Kit. *Helion Vision* : website. URL: <https://www.helion-vision.com/ionos-gui> (last access: 10.02.2023).
137. Nagao M., Matsuyama T. Edge preserving smoothing. *Computer Graphics and Image Processing*. Apr 1979. Vol. 9, No 4. P. 394–407.
138. Duda R. O., Hart P. E. Use of the Hough Transformation to Detect Lines and Curves in Pictures. *Comm. ACM*. 1972. Vol. 15, No 1. P. 11–15.
139. Serhiiienko P. A., Romankevich V. O., Sergiyenko A. M. Local Feature Extraction in High Dynamic Range Images. *Elektronnoe modelirovanie*. 2022. Vol. 44, No 4. P. 125–139. DOI: <https://doi.org/10.15407/emodel.44.04.041>.

140. Serhiienko P. A., Orlova M. M, Sergiyenko A. M. Local Feature Extraction in Images. *Information, Computing and Intelligent systems*. 2021. No 2. P. 1–13. DOI: <https://doi.org/10.20535/2708-4930.2.2021.244191>.
141. Сергієнко А. М., Зорін Ю. М., Сергієнко П. А. Пошук характерних ознак у зображенні з широким динамічним діапазоном. *ПМК-2018* : зб. тез 10-ї наук. конф. магістрантів та аспірантів «Прикладна математика та комп'ютинг», Київ, 2018. К.: НТУУ «КПІ», ВПІ ВПК «Політехніка», 2018. С. 65–69.
142. Сергієнко П. А., Зорін Ю. М. Детектування характерних точок у зображенні. *ПМК-2017* : зб. тез 9-ї наук. конф. Магістрантів та аспірантів «Прикладна математика та комп'ютинг», Київ, 19–21 квіт. 2017 р. К.: НТУУ «КПІ», ВПІ ВПК «Політехніка», 2017.
143. Sergiyenko A., Serhiienko P., Zorin Ju. High Dynamic Range Video Camera with Elements of the Pattern Recognition. *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)* : proc. int. conf., Kyiv, Ukraine, 24–26 April 2018. IEEE, 2018. P. 435–438, DOI: <https://doi.org/10.1109/ELNANO.2018.8477556>.
144. Schaumont P. R. Data Flow Modeling and Transformation / P. R. Schaumont. *Practical Introduction to Hardware/Software Codesign*. Springer, 2012. P. 31–59.
145. Алгоритмы, математическое обеспечение и архитектура многопроцессорных вычислительных систем / А. В. Вальковский, В. Е. Котов, Й. Миклошко и др; ред. А. П. Ершов. М. : Наука, 1982. 340 с.
146. Kahn G. The semantics of a simple language for parallel programming. *Information Processing 74* : Proc. IFIP Congress'74, 1974. North-Holland Pub. Comp., 1974. P. 471–475.
147. Карп Р. М., Миллер Р. Е. Параллельные схемы программ / перев. с англ. В. Иткин / *Кибернетический сборник. Новая серия* : сборник переводов / ред. Лупанов О. Б. М.: Мир, 1976. Вып. 13. С. 5–61.

148. Camposano R., Saunders L. F., Tabet R. M. VHDL as Input for High-Level Synthesis. *IEEE Design & Test of Computers*. March 1991. Vol. 8, Issue 1. P. 43–49.
149. Ли Е. А., Мессершмитт Д. Г. Вычисления с синхронными потоками данных. *ТИИЭР* / пер. с англ. под общ. ред. В. С. Хавкина. 1987. Т. 75, № 9. С. 107–119.
150. Оппенгейм А. В. Шафер Р. В. Цифровая обработка сигналов. М.: Связь, 1979. 416 с.
151. Lee E. A. Recurrences, Iteration, and Conditionals in Statically Scheduled Block Diagram Languages / *VLSI Signal Processing III* / R.W. Brodersen, H. S. Moscovitz, Ed-s. NY : IEEE Press, 1988. P. 330–340.
152. Bhattacharya B., Bhattacharyya S. S. Parameterized Dataflow Modeling for DSP Systems. *IEEE Trans. on Signal Processing*. 2001. Vol. 49, No 10. P. 2408–2421.
153. Buck J. T., Ha S., Lee E. A., Messerschmitt D. G. Ptolemy: A framework for simulating and prototyping heterogeneous systems. *Int. J. Computer Simul.* 1994. No 4. P. 155–182.
154. Buck J., Vaidyanathan R. Heterogeneous modeling and simulation of embedded systems in El Greco. *CODES '00* : Proc. 8th Int. Workshop on Hardware/Software Co-Design, San Diego California USA, 2000. P. 142–146.
155. Plishker W., Sane N., Kiemb M., Anand K., Bhattacharyya S. S. Functional DIF for rapid prototyping. *2008 The 19th IEEE/IFIP International Symposium on Rapid System Prototyping (RSP)* : Proc. Int. international conference, Monterey, CA, 2–5 June 2008. IEEE, 2008. P. 17–23.
156. Theelen B. D., Geilen M. C., Basten T., Voeten J. P., Gheorghita S. V., Stuijk S. A scenario-aware dataflow model for combined long-run average and worst-case performance analysis. *2006 4th IEEE/ACM International Conference on Formal Methods and Models for Co-Design* : Proc. Int. Conf., Washington DC, 2006. IEEE, 2006. P. 185 – 194.

157. Halbwachs N., Caspi P., Raymond P., Pilaud D. The synchronous data flow programming language LUSTRE. *Proc. IEEE*. 1991. Vol. 79, No 9. P. 1305–1320.
158. Woods R., McAllister J., Lightbody G., Yi Y. *FPGA-based Implementation of Signal Processing Systems*. 2d Ed. Wiley, 2017. 447 p.
159. Parhi K. K., Wang C. Y., Brown A. P. Synthesis of control circuits in folded pipelined DSP architectures. *IEEE J. Solid-St. Circ.* 1992. Vol. 27. P. 29–43.
160. Bailey D. G. *Design for Embedded Image Processing on FPGAs*. Wiley-Blackwell, 2011. 482 p.
161. Сергиенко А. М. Пространственный граф синхронных потоков данных. *Вісник НТУУ «КПІ». Інформатика, управління та обчислювальна техніка* : зб. наук. праць. Київ, 2010. №. 51. С. 40–46.
162. Serhienko A., Sergiyenko A., Simonenko A. A method for synchronous data-flow retiming. *UKRCON : IEEE 1-st Ukraine Conference on Electrical and Computer Engineering*, Kyiv, 29 May – 02 June 2017. IEEE, 2017. P. 1015–1018. DOI: <https://doi.org/10.1109/UKRCON.2017.8100404>.
163. Сергиенко А. М. *VHDL для проектирования вычислительных устройств*. Киев: Диасофт, 2004. 205 с.
164. Dessouky G., Klaiber M. J., Bailey D. G., Simon S. Adaptive Dynamic On-chip Memory Management for FPGA-based reconfigurable architectures. *2014 24th International Conference on Field Programmable Logic and Applications (FPL) : Proceedings of the 24th International Conference*, Munich, Germany, 2–4 Sept. 2014. IEEE, 2014. P. 1–8. DOI: <https://doi.org/10.1109/FPL.2014.6927471>.
165. Kim K., Kumar V. K. P. Parallel memory systems for image processing. *CVPR '89 : IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, San Diego, California, USA, 4–8 June 1989. IEEE, 1989. P. 654–659. DOI: <https://doi.org/10.1109/CVPR.1989.37915>

166. Khan S., Bailey D., Sen Gupta G. Simulation of triple buffer scheme (comparison with double buffering scheme). *ICCEE '09* : Proceedings of the 2nd International Conference on Computer and Electrical Engineering, Dubai, UAE, December 28–30, 2009. IEEE, 2009. Vol. 2, P. 403–407. DOI: <https://doi.org/10.1109/ICCEE.2009.226>.
167. Sadrozinski H. F.-W., Wu J. Applications of Field-Programmable Gate Arrays in Scientific Research. CRC Press and Taylor & Francis, 2011. 144 p.
168. Sedcole, P., Cheung P. Y. K., Constantinides G. A., Luk W. Run-time integration of reconfigurable video processing systems. *IEEE Transactions on VLSI Systems*. 2007. Vol. 15, No 9. P. 1003–1016. DOI: <https://doi.org/10.1109/TVLSI.2007.902203>.
169. ChipScope Pro 10.1 Software and Cores User Guide. Vol. UG029. Xilinx Inc., 2008. 194 p. URL: https://docs.xilinx.com/v/u/en-US/chip-scope_pro_sw_cores_10_1_ug029 (last access: 25.12.2022).
170. Shi R., Wong J. S. J., So H. K. High-Throughput Line Buffer Microarchitecture for Arbitrary Sized Streaming Image Processing. *Journal of Imaging*. 2019. Vol. 5, No 3 : 34. P. 1–20. DOI: <https://doi.org/10.3390/jimaging5030034>. PMID: 34460462; PMCID: PMC8320917.
171. Bailey D. G., Ambikumar A. S. Border Handling for 2D Transpose Filter Structures on an FPGA. *Journal of Imaging*. 2018. Vol. 4, No 12 : 138. P. 1–21. DOI: <https://doi.org/10.3390/jimaging4120138>.
172. Ikarashi Y., Ragan-Kelley J., Fukusato T., Kato J., Igarashi T. Guided Optimization for Image Processing Pipelines. *2021 IEEE Symposium on Visual Languages and Human-Centric Computing (VL/HCC)* : proceedings of the international conference, St Louis, MO, USA, 10–13 October 2021. IEEE, 2021. P. 1–5. DOI: <https://doi.org/10.1109/VL/HCC51201.2021.9576341>.
173. Özkan M.A., Reiche O., Hannig F., Teich J. FPGA-based accelerator design from a domain-specific language. *2016 26th International Conference on Field Programmable Logic and Applications (FPL)* : Proc. of the 26th Int.

- Conference, Lausanne, Switzerland, 29 August – 02 September 2016. IEEE, 2016. P. 1–9. DOI: <https://doi.org/10.1109/FPL.2016.7577357>.
174. Waidyasooriya H. M. A., Haiyama M., Uchiyama K. Design of FPGA-based computing systems with OpenCL. Springer, 2018. DOI: <https://doi.org/10.1007/978-3-319-68161-0>.
 175. Granado L., Berreteaga O. Creating Rich Human-machine Interfaces with Rational Rhapsody and Qt for Industrial Multi-core Real-time Applications. *Procedia Manufacturing*. 2015. Vol. 3. P. 1903 – 1909. DOI: <https://doi.org/10.1016/j.promfg.2015.07.233>.
 176. Hwang J, Milne B, Shirazi N., Stroome J. D. System Level Tools for DSP in FPGAs. *FPL '01 : Proc. 11th Int. Conf. on Field Programmable Logic and Applications*, 2001. Springer-Verlag, 2001. P. 534–543. DOI: https://doi.org/10.1007/3-540-44687-7_55.
 177. Sano K., Nakahara H. Hardware Algorithms / *Principles and Structures of FPGAs* / H. Amano, Ed. Springer, 2018. P. 137–177.
 178. Sklyarov V., Skliarova I., Barkalov A., Titarenko L. Synthesis and Optimization of FPGA-Based Systems. Springer, 2014. 432 p.
 179. Sass R., Schmidt A. G. Embedded Systems Design with Platform FPGAs. Principles and Practices. Morgan Kaufmann Pub., 2010. 389 p.
 180. Winterstein F., Fleming K., Yang H.-J., Bayliss S., Constantinides G. MATCHUP: Memory Abstractions for Heap Manipulating Programs. *FPGA '15: Proceedings of the 2015 ACM/SIGDA International Symposium on Field-Programmable Gate Arrays*, Monterey California USA, February 22–24, 2015. ACM, 2015. P. 136–145. DOI: <https://doi.org/10.1145/2684746.2689073>.
 181. Parhi K. K. VLSI Digital Signal Processing Systems: Design and Implementation. Wiley, 1999. 784 p.
 182. Lee E. A., Neuendorffer S., Wirthlin M. J. Actor-Oriented Design of Embedded Hardware and Software Systems. *Journal of Circuits System and*

- Computers*. 2003. Vol. 12. P. 231–260. DOI: <https://doi.org/10.1142/S0218126603000751>.
183. Pedersen M. R., Madsen J. Optimal register allocation by augmented left-edge algorithm on arbitrary control-flow structures. *NORCHIP'2012* : proceedings of the international conference, Copenhagen, Denmark, 12–13 November 2012. IEEE, 2012. P. 1–6. DOI: <https://doi.org/10.1109/NORCHIP.2012.6403107>.
 184. Murthy P. K., Lee E. A. Multidimensional Synchronous Dataflow. *IEEE Transactions on Signal Processing*. 2002. Vol. 50, No 8. P. 2064–2079. DOI: <https://doi.org/10.1109/TSP.2002.800830>.
 185. Cong J., Jiang W., Liu B., Zou Y. Automatic memory partitioning and scheduling for throughput and power optimization. *2009 IEEE/ACM International Conference on Computer-Aided Design* : Proc. Int. Conf., San Jose, CA, USA, 02–05 Nov. 2009. IEEE, 2009. P. 697–704. DOI: <https://doi.org/10.1145/1687399.1687528>.
 186. Yu Y. W., Li P., Cong J. Theory and algorithm for generalized memory partitioning in high-level synthesis. *FPGA '14* : Proceedings of the 2014 ACM/SIGDA international symposium on Field-programmable gate arrays, Monterey California USA, Feb. 26–28 2014. P. 199–208. DOI: <http://dx.doi.org/10.1145/2554688.2554780>
 187. Cong J., Wang J. PolySA: Polyhedral-Based Systolic Array Auto-Compilation. *2018 IEEE/ACM International Conference on Computer-Aided Design (ICCAD)* : proc. int. conference, San Diego, CA, USA, November 5–8, 2018. IEEE Press, 2018. P. 1–8. DOI: <http://doi.org/10.1145/3240765.3240838>.
 188. Gallo L., Cilaro A., Thomas D., Bayliss S., Constantinides G. A. Area implications of memory partitioning for high-level synthesis on FPGAs. *2014 24th International Conference on Field Programmable Logic and Applications (FPL)* : proc. int. conference, Munich, Germany, 02–04 September 2014. IEEE, 2014. P. 1–4. DOI: <http://doi.org/10.1109/FPL.2014.6927417>.

189. Wang Y., Zhang P., Cheng X., Cong J. An integrated and automated memory optimization flow for FPGA behavioral synthesis. *17th Asia and South Pacific Design Automation Conference* : proc. int. conf, Sydney, NSW, Australia, 30 Jan. 2012 – 02 Feb. 2012. IEEE, 2012. P. 257–262, DOI: <http://doi.org/10.1109/ASPDAC.2012.6164955>.
190. Guo Z., Najjar W., Buyukkurt B. Efficient hardware code generation for FPGAs. *ACM Transactions on Architecture and Code Optimization*. 2008. Vol. 5, No 1. P. 1–26. DOI: <https://doi.org/10.1145/1369396.1369402>.
191. Najjar W. A., Villarreal J., Halstead R. ROCCC 2.0 / *FPGAs for Software Programmers* / D. Koch, F. Hannig, D. Ziener – Ed-s. Springer, 2016. P. 191–204.
192. UltraFast Design Methodology Guide for the Vivado Design Suite. UG949 (v2013.3). October 23, 2013. Xilinx, 2013. 361 p. URL: <https://docs.xilinx.com/r/en-US/ug949-vivado-design-methodology> (last access: 20.12.2022).
193. 7 Series FPGAs Memory Resources User Guide. UG473 (v1.14). July 3, 2019. Xilinx, 2019. 88 p. URL: https://docs.xilinx.com/v/u/en-US/ug473_7Series_Memory_Resources (last access: 16.11.2022).
194. Intel Hyperflex™ Architecture High Performance Design Handbook. Ver.: 2021.10.04. Intel, 2021. 147 p. URL: <https://www.intel.com/programmable/technical-pdfs/683353.pdf> (last access: 14.12.2022).
195. Chu J., Benaissa M. Low area memory-free FPGA implementation of the AES algorithm. *22nd International Conference on Field Programmable Logic and Applications (FPL)* : proc. int. conf, Oslo, Norway, 29–31 August 2012. IEEE, 2012. P. 623–626, DOI: <https://doi.org/10.1109/FPL.2012.6339250>.
196. Сергиенко А. М., Симоненко В. П. Отображение периодических алгоритмов в программируемые логические интегральные схемы. *Электронное моделирование*. 2007. Т. 29, № 2. С. 49–61.

197. Sergiyenko A. M., Simonenko V. P. Method of synchronous dataflow scheduling. *System research and information technologies*. 2016. № 1. P. 51–62. DOI: <https://10.20535/SRIT.2308-8893.2016.1.06>.
198. Nikara J., Takala J., Akopian D., Saarinen J. Pipeline Architecture for DCT/IDCT. *ISCAS 2001* : Proc. The 2001 IEEE International Symposium on Circuits and Systems, Sydney, Australia, May 6–9, 2001. IEEE, 2001. P. 902–905.
199. Quinton P., Robert Y. Systolic Algorithms & Architectures. Prentice Hall, 1991. 364 p.
200. Sergiyenko A., Kanievsky Ju., Maslennikov O., Wyrzykowski R. A Method for Mapping DSP Algorithms into Application Specific Structures. *Euromicro '98* : Proceedings of the 24th Euromicro Conference, Vasteras, Sweden, 27 August 1998. IEEE, 1998. P. 365–371.
201. Serhiienko P., Sergiyenko A., Mozgovyi I., Molchanova A. Design of Data Buffers in Field Programmable Gate Arrays. *Information, Computing and Intelligent systems*. 2022. No 3. P. 1–16. DOI: <https://doi.org/10.20535/2708-4930.2.2021.244191>.
202. Serhiienko P. A., Romankevich V. O., Sergiyenko A. M. Image buffering in application specific processors. *Applied Aspects of Information Technology*. 2022. V. 6, No 5. P. 228–239. DOI: <https://doi.org/10.15276/aait.05.2022.16>.
203. FPGA Implementations of Neural Networks / A. R. Omondi, J. C. Rajapakse, eds. Springer, 2006. 360 p.
204. 7 Series FPGAs CLB User Guide. UG474 (v1.7). Xilinx, 2014. 58 p. URL: https://docs.xilinx.com/v/u/en-US/ug474_7Series_CLB.
205. Series FPGAs Data Sheet: Overview. Xilinx, 2017. 18 p. URL: https://www.xilinx.com/support/documentation/data_sheets/ds180_7Series_Overview.pdf (last access: 10.12.2022).

206. Люстерник Л. А., Червоненкис О. А., Янпольский А. Р. Математический анализ. Вычисление элементарных функций. М. : ГИЗ физ.-мат. лит., 1963. 247 с.
207. Благовещенский Ю. В., Теслер Г. С. Вычисление элементарных функций на ЭВМ. Киев: Техніка, 1977. 208 с.
208. Байков В. Д., Смолов В. Б. Специализированные процессоры: Итерационные алгоритмы и структуры. М.: Радио и связь, 1985. 288 с.
209. Бікташева С. Р., Мороз Л. В., Стахів М. Ю. CORDIC-метод обчислення квадратного кореня. *Вісн. Нац. Ун-ту «Львівська політехніка»*. Сер.: Електроніка : зб. наук. пр. Львів : Вид-во Нац. ун-ту «Львів. політехніка», 2006. С. 152–155.
210. Chen T. C. Automatic computations of exponentials, logarithms, ratios and square roots. *IBM J. Res. and Develop.* 1972. № 4. P. 380–388.
211. Поспелов Д. А. Арифметические основы вычислительных машин дискретного действия. 1970. 308 с.
212. Сергієнко А. М., Сергієнко П. А. Реалізація функції квадратного кореня у ПЛІС. *Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка* : зб. наук. праць. Київ, 2014. № 60. С. 40–45. ISSN 0135-1729.
213. Ritter D., Dann J., May N., Rinderle-Ma S. Hardware Accelerated Application Integration Processing: Industry Paper. *DEBS '17 : Proceedings of the 11th ACM International Conference on Distributed and Event-based Systems*, Barcelona Spain, June 19–23, 2017. ACM, 2017. P. 215–226. DOI: <https://doi.org/10.1145/3093742.3093911>.
214. Lafond S, Lilius J. An Energy Consumption Model for Java Virtual Machine. *TUCS Technical Report*. 2004. No 597.
215. Li X., Mu L., Zang Y., Qin Q. Study on performance degradation and failure analysis of machine gun barrel. *Defence Technology*. 2020. Vol. 16, No 2. P. 362–373. DOI: <https://doi.org/10.1016/j.dt.2019.05.008>.

216. Zervas N. Firmware Compression for Lower Energy and Faster Boot in IoT Devices. *Design & Reuse* : web portal. October 2015. URL: <https://www.design-reuse.com/articles/38541/firmware-compression-for-lower-energy-and-faster-boot-in-iot-devices.html> (accessed: Jan. 2023).
217. Beckhoff C., Koch D., Torresen J. Portable module relocation and bitstream compression for Xilinx FPGAs. *2014 24th International Conference on Field Programmable Logic and Applications (FPL)* : proc. int. conf., Munich, Germany, 2–4 Sept. 2014. IEEE, 2014. P. 1–8. DOI: <https://doi.org/10.1109/FPL.2014.6927480>.
218. Walls F. G., MacInnis A. S. VESA Display Stream Compression for Television and Cinema Applications. *IEEE Journal on Emerging and Selected Topics in Circuits and Systems*. 2016. Vol. 6, No 4. P. 460–470. DOI: <https://doi.org/10.1109/JETCAS.2016.2602009>.
219. Touba N. A. Survey of Test Vector Compression Techniques. *IEEE Design & Test of Computers*. 2006. Vol. 23, No 4. P. 294–303. DOI: <https://doi.org/10.1109/MDT.2006.105>.
220. Kovačec D. FPGA IP Cores for Displays / *Handbook of Visual Display Technology* / J.Chen, W. Cranton, M. Fihn, Eds. Springer, 2012. P. 512–530. DOI: https://doi.org/10.1007/978-3-540-79567-4_40.
221. Сергиенко П. А., Сергиенко А. М., Молчанов А. А., Мозговой И. В. Система динамической визуализации на базе аппаратного GIF-декомпрессора. *Simulation-2018* : Збірник праць міжнародної конференції, Київ, 12–14 вересня 2018. Київ, 2018. С. 221–223.
222. Gallager R. Variations on a theme by Huffman. *IEEE Transactions on Information Theory*. 1978. Vol. 24, No 6. P. 668–674. DOI: <https://doi.org/10.1109/TIT.1978.1055959>.
223. Welch T. A Technique for High-Performance Data Compression *Computer*. 1984. Vol. 17, No 6. P. 8–19. DOI: <https://doi.org/10.1109/MC.1984.1659158>.

224. Salomon D., Motta G. Handbook of Data Compression : 5th Ed. Springer, 2010. 1360 p. ISBN: 978-1-84882-903-9.
225. Ziv J., Lempel A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory*. 1977. Vol. 23, No 3. P. 337–343. DOI: <https://doi.org/10.1109/TIT.1977.1055714>.
226. LZRW3 Data Compression Core for Xilinx FPGA: Full Datasheet. Helion Technology, 2008. 3 p. URL: https://www.heliontech.com/downloads/lzrw3_xilinx_datasheet.pdf (accessed: Jan. 2023).
227. Hwang G. B., Cho K. N., Han C. Y., Oh H. W., Yoon Y. H., Lee S. E. Lossless Decompression Accelerator for Embedded Processor with GUI. *Micromachines*. 2021. Vol. 12, No 2. DOI: 10.3390/mi12020145.
228. Ledwon M., Cockburn B. F., Han J. High-Throughput FPGA-Based Hardware Accelerators for Deflate Compression and Decompression Using High-Level Synthesis. *IEEE Access*. 2020. Vol. 8. P. 62207–62217. DOI: <https://doi.org/10.1109/ACCESS.2020.2984191>.
229. Ziv J., Lempel A. Compression of individual sequences via variable-rate coding. *IEEE Transactions on Information Theory*. 1978. Vol. 24, No 5. P. 530–536. DOI: <https://doi.org/10.1109/TIT.1978.1055934>.
230. Fang, J., Chen, J., Lee, J. et al. An Efficient High-Throughput LZ77-Based Decompressor in Reconfigurable Logic. *J Sign Process Syst* 92, 931–947 (2020). <https://doi.org/10.1007/s11265-020-01547-w>
231. Hwang G. B., Cho K. N., Han C. Y., Oh H. W., Yoon Y. H., Lee S. E. Lossless Decompression Accelerator for Embedded Processor with GUI. *Micromachines*, 2021. Vol. 12. No. 2. DOI: 10.3390/mi12020145
232. Ledwon M., Cockburn B. F., Han J. High-Throughput FPGA-Based Hardware Accelerators for Deflate Compression and Decompression Using High-Level Synthesis, in *IEEE Access*, vol. 8, pp. 62207-62217, 2020, doi: <https://doi.org/10.1109/ACCESS.2020.2984191>

233. May P., Davies K. Practical Analysis of TIFF File Size Reductions Achievable Through Compression. *iPRES 2016* : proc. 13th International Conference on Digital Preservation, Bern, Switzerland, October 3–6, 2016. P. 1–10.
234. Navqi S., Naqvi R., Riaz R. A, Siddiqui F. Optimized RTL design and implementation of LZW algorithm for high bandwidth applications. *Przegląd Elektrotechniczny (Electrical Review)*. 2011. No 4. P. 279–285.
235. Zhou X., Ito Y., Nakano K. An Efficient Implementation of LZW Decompression in the FPGA. *2016 IEEE International Parallel and Distributed Processing Symposium Workshops (IPDPSW)* : proc. int. conf., Chicago, IL, USA, 23–27 May 2016. IEEE, 2016. P. 599–607. DOI: <https://doi.org/10.1109/IPDPSW.2016.33>.
236. Funasaka S., Nakano K., Ito Y. A Parallel Algorithm for LZW Decompression, with GPU Implementation. *LNCS / R. Wyrzykowski, E. Deelman, J. Dongarra, K. Karczewski, J. Kitowski, K. Wiatr, eds. Springer, 2016. Vol 9573, Parallel Processing and Applied Mathematics: PPAM 2015. P. 228–237. DOI: https://doi.org/10.1007/978-3-319-32149-3_22.*
237. Сергієнко П. А., Сергієнко А. М. Ядро RISC-процесора для реалізації у ПЛІС. *InfoCom '2015* : праці 1 міжнародної конференції, Київ, 24–25 листопада 2015 р. К.: НТУУ “КПІ”, ВПІ “Політехніка”, 2015. С. 54–55.
238. Сергієнко П. А., Кліменко І. А., Сергієнко А. М. Реконфігурована багатопроцесорна обчислювальна система на ПЛІС. *Вісник НТУУ «КПІ»: Інформатика, управління та обчислювальна техніка* : зб. наук. пр. 2016. № 64. С. 47–50.
239. Сергієнко П. А., Лепеха В. Л. Ядро 16-розрядного RISC-процесора. *Інформатика та обчислювальна техніка* : матеріали наук. конф. студентів, магістрантів та аспірантів ІОТ-2016, Київ, 25–27 квітня, 2016. К.: НТУУ «КПІ», 2016. С. 118–119.
240. Сергієнко П. А., Сергієнко А. М. Багатопроцесорна система на ПЛІС для синхронної обробки потоків даних. *Прикладна математика та*

комп'ютинг : зб. тез доп. восьмої наук. конф. магістрантів та аспірантів ПМК-2016, Київ, 20–22 квіт. 2016 р. К.: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка», 2016. С. 124–127.

241. Sergiyenko A., Orlova M., Molchanov O. Hardware/Software Co-design for XML-Document Processing. *Advances in Intelligent Systems and Computing* / Z. Hu, S. Petoukhov, I. Dychka, M. He, eds. Springer, 2021. Vol 1247, Advances in Computer Science for Engineering and Education III. P. 373–383. DOI: https://doi.org/10.1007/978-3-030-55506-1_34.
242. Koopman P. Stack computers: the new wave. Ellis Horwood, Mountain View Press, CA, 1989. 231 p.
243. Oliver J. P., Aclé J. P., Boemo E. Power estimations vs. power measurements in Spartan-6 devices. *2014 IX Southern Conference on Programmable Logic (SPL)* : proc. int. conf, Buenos Aires, Argentina, 5–7 Nov. 2014. IEEE, 2014. P. 1–5. DOI: <https://doi.org/10.1109/SPL.2014.7002214>.
244. Serhiienko P. A., Sergiyenko A. M., Romankevich V. O., Molchanov O. A., Mozghovyi I. V., Zacharioudakis L. Decompressor for hardware applications. *Applied Aspects of Information Technology*. 2023. V. 6, No.1: pp.74-83.

ДОДАТОК А

Список публікацій здобувача

СПИСОК ПУБЛІКАЦІЙ ЗДОБУВАЧА

Наукові праці в яких опубліковано основні наукові результати дисертації:

1. Serhiienko P. A., Sergiyenko A. M., Romankevich V. O., Molchanov O. A., Mozghovyi I. V., Zacharioudakis L. Decompressor for hardware applications. *Applied Aspects of Information Technology*. 2023. V. 6, No.1: pp.74-83. (фахове вид. кат. Б)
2. Serhiienko P. A., Romankevich V. O., Sergiyenko A. M. Image buffering in application specific processors. *Applied Aspects of Information Technology*. 2022. V.6. No. 5, pp 228–239.
<https://doi.org/10.15276/aait.05.2022.16> (фахове вид. кат. Б)
3. Serhiienko P. A., Romankevich V. O., Sergiyenko A. M. Local Feature Extraction in High Dynamic Range Images. *Elektronnoe modelirovanie*, 2022, v. 44. No. 4, pp. 125–139.
<https://doi.org/10.15407/emodel.44.04.041> (фахове вид. кат. Б)
4. Сергієнко П.А., Кліменко І.А., Сергієнко А.М. Реконфігурована багатопроцесорна обчислювальна система на ПЛІС. Вісник НТУУ «КПІ». Ін форматика, управління та обчислювальна техніка: Зб. наук. пр. К.: Век+, 2016, № 64, с. 47-50. (фахове вид. кат. Б)
5. Сергієнко А.М., Сергієнко П.А. Реалізація функції квадратного кореня у ПЛІС. Вісник НТУУ «КПІ»: “Інформатика, управління та обчислювальна техніка”: зб. наук. праць. 2014. Т.60, с. 40–45. ISSN 0135-1729. (фахове вид. кат. Б)

В яких засвідчено апробацію матеріалів дисертації

6. Serhiienko P., Orlova M., Sergiyenko A., Molchanov O. System for Video Pattern Recognition on FPGA. *2019 IEEE 2nd Ukraine*

Conference on Electrical and Computer Engineering, Lviv, Ukraine, July 2-6, 2019, pp. 1175–1178.

<https://doi.org/10.1109/UKRCON.2019.8879958> (реф Scopus).

7. Sergiyenko A., Serhienko P., Zorin Ju. High Dynamic Range Video Camera with Elements of the Pattern Recognition. *2018 IEEE 38th International Conference on Electronics and Nanotechnology (ELNANO)*, Kyiv, Ukraine, 2018, pp. 435-438, doi: 10.1109/ELNANO.2018.8477556 (реф Scopus).
8. Сергієнко П.А. Проблеми розвитку комп'ютерного зору. *Філософські засади креатосфери у контексті творчості: Матеріали 15 Міжнародної науково-практичної конференції*, Київ, Україна, 30 травня 2019, С. 178-181.
9. Сергієнко П. А. Сергиенко А. М., Молчанов А. А., Мозговой И. В. Система динамической визуализации на базе аппаратного GIF-декомпрессора. *Збірник праць міжнародної конференції Simulation-2018*, 12-14 вересня 2018, Київ, - с. 221–223.
10. Serhienko P. A., Hasan M. J., Sergiyenko A. M. Square root calculations in FPGA. *System analysis and information technology: 20-th International conference SAIT`2018*, Kyiv, Ukraine, May 21 – 24, 2018. *Proceedings. – ESC “IASA” NTUU “Igor Sikorsky Kyiv Polytechnic Institute”*, 2018.p. 163-164.
11. Serhienko P., Sergiyenko A., Molchanov O. Reconfigurable Many-core System. *Праці 5-ї міжнародної конференції «High Performance Computing» HPC-UA`2018*. 2018, с. 127 – 130.
12. Сергієнко А.М., Зорін Ю.М., Сергієнко П.А. Пошук характерних ознак у зображенні з широким динамічним діапазоном. *10 наукова конференція магістрантів та аспірантів "Прикладна математика та комп'ютинг", ПМК-2018*. Київ: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка». — 2018, С. 65-69.

- 13.Сергієнко П.А., Зорін Ю. М. Детектування характерних точок у зображенні. *Прикладна математика та комп'ютинг. ПМК, 2017 : 9 наук. конф. магістрантів та аспірантів*, Київ, 19-21 квіт. 2017 р. : – К.: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка».
- 14.Сергієнко А.М., Сергієнко П.А. Багатопроцесорна система на ПЛІС для синхронної обробки потоків даних. *Прикладна математика та комп'ютинг. ПМК, 2016 : восьма наук. конф. магістрантів та аспірантів*, Київ, 20-22 квіт. 2016 р. : зб. тез доп. К. ПМК-2016. – К.: Вид. НТУУ «КПІ» ВПІ ВПК «Політехніка». 2016, с. 124-127.
- 15.Сергієнко П.А., Лепеха В.Л., Ядро 16-розрядного RISC–процесора». *Матеріали наук. конф. студентів, магістрантів та аспірантів «Інформатика та обчислювальна техніка – ІОТ-2016»*, 25 – 27 квітня, 2016. — К: НТУУ «КПІ», 2016. — С. 118 – 119.
- 16.Сергієнко П. А., Сергієнко А. М. Ядро RISC-процесора для реалізації у ПЛІС. *InfoCom'2015 : праці 1 міжнародної конференції*, Київ, 24–25 листопада 2015 р. К.: НТУУ “КПІ”, ВПІ “Політехніка”, 2015. С. 54–55.

Які додатково відображають наукові результати дисертації

- 17.Serhiienko P., Sergiyenko A., Mozgovyi I., Molchanova A. Design of Data Buffers in Field Programmable Gate Arrays. *Information, Computing and Intelligent systems*, 2022, No. 3, pp. 1–16.
<https://doi.org/10.20535/2708-4930.2.2021.244191>
- 18.Serhiienko P. A., Orlova M. M, Sergiyenko A. M. Local Feature Extraction in Images. *Information, Computing and Intelligent systems*, 2021, No. 2, pp. 1–13. <https://doi.org/10.20535/2708-4930.2.2021.244191>

ДОДАТОК Б

Акти про впровадження результатів дисертаційного дослідження

ЗАТВЕРДЖУЮ

Проректор з науково-педагогічної роботи

Національного

технічного університету України

«Київський політехнічний інститут

імені Ігоря Сікорського»



Анатолій МЕЛЬНИЧЕНКО

» _____ 2023 р.

А К Т

впровадження результатів дисертаційного дослідження асистента кафедри Системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики Національного технічного університету України «КПІ імені Ігоря Сікорського» Сергієнка Павла Анатолійовича на тему «Методи і засоби проектування обчислювачів для розпізнавання образів у зображеннях» на здобуття освітньо-наукового ступеня доктора філософії.

Комісія у складі: голова – декан факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського», д.т.н., проф. Дичка І.А.; члени комісії – професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Романкевич О.М., професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Терейковський І.А. цим Актом засвідчує, що результати дисертаційного дослідження Сергієнка Павла використані співробітниками кафедри СПіСКС КПІ ім. Ігоря Сікорського при підготовці та викладанні курсу лекцій «Цифрова обробка сигналів та зображень». Зокрема впроваджено огляд алгоритмів технічного зору, що включає новий метод пошуку характерних точок у зображенні, розроблена лабораторна робота по вивченню пошуку характерних точок у зображенні.

Голова комісії

д.т.н., проф.

Іван ДИЧКА

Члени комісії

д.т.н., проф.


Олексій РОМАНКЕВИЧ

д.т.н., проф.

Ігор ТЕРЕЙКОВСЬКИЙ

ЗАТВЕРДЖУЮ
Проректор Національного
технічного університету України
«Київський політехнічний інститут
імені Ігоря Сікорського»



 **Анатолій МЕЛЬНИЧЕНКО**
» _____ 2023 р.

АКТ

про використання результатів дисертаційного дослідження асистента кафедри Системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики Національного технічного університету України «КПІ ім. Ігоря Сікорського» Сергієнка Павла Анатолійовича на тему «Методи і засоби проектування обчислювачів для розпізнавання образів у зображеннях» на здобуття наукового ступеня доктора філософії.

Комісія у складі: голова – завідувачий кафедрою ОТ КПІ ім. Ігоря Сікорського, д.т.н., проф. Стіренко С.Г.; члени комісії – доцент кафедри ОТ КПІ ім. Ігоря Сікорського, к.т.н., доц., Роковий О.П., професор кафедри ОТ КПІ ім. Ігоря Сікорського, д.ф.-м.н., проф. Гордієнко Ю. Г. цим Актом засвідчує, що результати дисертаційної роботи Сергієнка Павла Анатолійовича отримані їм особисто та використані у:

- НДР ЗОТ/2017 «Методи і засоби відображення потокових алгоритмів у конфігуровні комп'ютери», що виконується за ініціативою авторів, № держреєстрації 0117U005087, закінчення у 2023 р.,

- Веб-застосунку «Генератор модулів квадратного кореня», що розміщений на сайті, що належить кафедрі ОТ, URL: https://kanyevsky.kpi.ua/GEN_MODUL/SQRT/index_sqrt_ukr.php.

В темі та Веб-застосунку використані наступні результати дисертаційної роботи Сергієнка П.А.:

- метод синтезу буферних схем для обробки двовимірних потоків даних, в якому формалізована побудова таких схем, завдяки чому зменшуються апаратні витрати (до 2,5 разів у об'ємі пам'яті) і прискорюється проектування;

- удосконалений алгоритм та структура модуля обчислення квадратного кореня, у якому мінімізовані апаратні витрати та латентна затримка обчислень.

Голова комісії
д.т.н., проф.



Стіренко С.Г.

Члени комісії
д.ф.-м.н., проф.
к.т.н., доц.,



Гордієнко Ю. Г.
Роковий О.П.,

ЗАТВЕРДЖУЮ

Проректор Національного

технічного університету України

«Київський політехнічний інститут

імені Ігоря Сікорського»



Анатолій МЕЛЬНИЧЕНКО

2023 р.

А К Т

про використання результатів дисертаційного дослідження асистента кафедри Системного програмування і спеціалізованих комп'ютерних систем факультету прикладної математики Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» Сергієнка Павла Анатолійовича на тему «Методи і засоби проектування обчислювачів для розпізнавання образів у зображеннях» на здобуття освітньо-наукового ступеня доктора філософії.

Комісія у складі: голова – декан факультету прикладної математики «КПІ ім. Ігоря Сікорського», д.т.н., проф. Дичка І.А.; члени комісії – професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Романкевич О.М., професор кафедри СПіСКС КПІ ім. Ігоря Сікорського, д.т.н., проф. Терейковський І.А. цим Актом засвідчує, що результати дисертаційної роботи Сергієнка Павла Анатолійовича отримані їм особисто та використані у:

- НДР «Методи, моделі та комп'ютерні засоби виявлення деструктивного впливу в медіапросторі», що виконується за ініціативою авторів, № держреєстрації 0121U110662, закінчення у 2023 р.,

В НДР використані наступні результати дисертаційної роботи Сергієнка П.А.:

- метод пошуку характерних точок у зображенні, який завдяки використанню запропонованого алгоритму адаптивної фільтрації виконує пошук характерних точок у несприятливих умовах освітленості та має зменшений обсяг обчислень.

Голова комісії
д.т.н., проф.

Іван ДИЧКА

Члени комісії
д.т.н., проф.

Олексій РОМАНКЕВИЧ

д.т.н., проф.

Ігор ТЕРЕЙКОВСЬКИЙ