УДК 004.932 doi: 10.15421/322120

М.Є. Сердюк, А.О. Олексієнко, С.Ф. Сірик
Дніпровський національний університет імені Олеся Гончара

ДОСЛІДЖЕННЯ СПОСОБІВ ПІДВИЩЕННЯ ШВИДКОДІЇ МЕТОДІВ ВИДАЛЕННЯ ШУМУ З ЦИФРОВИХ ЗОБРАЖЕНЬ

Розглянуто проблема підвищення швидкодії методів обробки зображень на прикладі методів видалення шуму. Проаналізовані усереднюючі фільтри та досліджені алгоритмічні способи прискорення їх роботи. Розглянуто способи паралелізації методів фільтрації та запропоновано модифікації обробки з використанням векторних операцій. Наведено оцінки швидкодії розглянутих модифікацій методів знешумлення зображень за допомогою розробленої програмної системи.

Ключові слова: цифрове зображення, фільтрація зображень, усереднюючий фільтр, фільтр Гауса, SIMD технологія, паралельні алгоритми.

M.E. Serdiuk, A.O. Oleksiienko, S.F. Siryk
Oles Honchar Dnipro National University

THE RESEARCH OF WAYS TO INCREASE THE SPEED OF DIGITAL IMAGES DENOISING METHODS

The problem of ways to increase the speed of image processing methods on the example of noise removal methods is considered. Digital images are used in various systems of computer vision, video surveillance and objects recognition, geographic information and medical systems, etc. The low quality of input images makes it difficult to work with them in the future. Therefore, images pre-processing, in particular their denoising, is an important step in these systems operation. Often such systems operate in real-time. Due to the graphic information amount growth, the processing speed increase becomes a topical issue.

The purpose of this research is to study the effectiveness of the application of algorithmic and technological ways to speed up methods of removing noise from digital images.

The research analyzes averaging filters, the effect of which can be represented in the form of image convolution with a filter mask. In most cases, the masks of such filters are represented by a matrix that is symmetrical and has a square or circular shape. In addition to the classical implementations, the modifications of the methods are also considered. These modifications allow to improve the theoretical estimates of the computational complexity of the corresponding algorithms. The «horizontal-vertical» modification is considered for the Gaussian filter and the arithmetic mean filter. This modification is based on the separability properties of these filters. The modification based on recurrent relations is also considered. Technological ways of accelerating the methods operation are analyzed. The parallelism implementation at different levels is considered. For this purpose, modifications of noise removal methods using vector operations have been developed. The main attention is given to the methods of SIMD operations implementing. The practical result of this research is the software application, that implements the considered filtering methods and their modification, as well as a module for testing the methods productivity. The experimental results have shown that the conversion of algorithms to a vector basis and the use of the SIMD technology have increased the filtering productivity by 5-14 times depending on the filter and processor type.

Keywords: digital image, image filtering, averaging filter, Gaussian filter, SIMD technology, parallel algorithms.

М.Е. Сердюк, А.А. Алексеенко, С.Ф. Сирик
Днипровский национальный университет имени Олеся Гончара

ИССЛЕДОВАНИЕ СПОСОБОВ ПОВЫШЕНИЯ БЫСТРОДЕЙСТВИЯ МЕТОДОВ УДАЛЕНИЯ ШУМА С ЦИФРОВЫХ ИЗОБРАЖЕНИЙ

Рассмотрена проблема повышения быстродействия методов обработки изображений на примере методов удаления шума. Проанализированы усредняющие фильтры и исследованы алгоритмические способы ускорения их работы. Рассмотрены способы параллелизации методов фильтрации и предложены модификации обработки с использованием векторных операций. Приведены оценки быстродействия рассмотренных модификаций методов удаления шума с помощью разработанной программной системы.

Ключевые слова: цифровое изображение, фильтрация изображений, усредняющий фильтр, фильтр Гаусса, SIMD технология, параллельные алгоритмы.

Вступ. На сьогоднішній день спостерігається інтегрування інформаційних технологій у різноманітні сфери та галузі діяльності людини. Широкого розповсюдження набули системи, у яких інформація представляється у вигляді цифрових зображень. Прикладами таких систем є різноманітні системи комп'ютерного зору, відеоспостереження та розпізнавання об'єктів, геоінформаційні системи, системи медичної діагностики та ін. Певну проблему в роботі подібних систем може створювати недостатня якість зображень, які реєструються або поступають на вхід для подальшого опрацювання. Низька контрастність, погана розрізнюваність, зашумленість зображень ускладнюють подальшу роботу з ними. Тому попереднім етапом, як правило, є обробка вхідних зображень з метою покращення їх якості: підвищення контрасту, різкості, корекція кольорів, згладжування та ін. Зокрема, однією з найактуальніших і поширених проблем в області обробки як статичних зображень, так і відео, є проблема шумозаглушення. Шумозаглушення може використовуватись як для поліпшення візуального сприйняття зображення, так і для спеціалізованих цілей, наприклад, як етап передобробки в задачах розпізнавання.

З розвитком цифрових технологій підвищується роздільна здатність зображень, а отже, зростають обсяги графічних даних, а це, в свою чергу, потребує більших ресурсів для методів їх обробки. Якщо такі методи працюють в системах реального часу, то особлива увага приділяється питанням швидкодії. Не дивлячись на розвиток обчислювальних потужностей, актуальним залишається питання підвищення ефективності методів та алгоритмів обробки зображень. Це можна робити різними способами: алгоритмічно, тобто вдосконалювати алгоритми, або технологічно, тобто впроваджувати сучасні технології, зокрема паралельні обчислення, в існуючі методи з використанням можливостей сучасної комп'ютерної техніки. Саме дослідженню цих двох аспектів на прикладі методів пригнічення шуму на цифрових зображеннях присвячена дана робота.

Огляд літератури. Важливе значення в системах, які працюють з графічною інформацією, має етап попередньої обробки зображення для підвищення його якості, зокрема видалення шуму. На сьогодні відомо багато методів шумозаглушення, які застосовуються для видалення різних видів шумів на різних типах зображень. В роботах [1-4] можна знайти порівняльний аналіз основних методів фільтрації. Основна увага в цих дослідженнях приділяється алгоритмічним особливостям методів та порівнянню якості результатів, отриманих для різних видів шумів. Але в багатьох випадках застосувань не менш важливим питанням є ефективність реалізації методів, оскільки неефективна реалізація фільтрів може привести до суттєвих затримок у роботі систем, які опрацьовують графічну інформацію, в цілому. У [5] відзначається, що існуючі реалізації методів фільтрації, які є у програмних бібліотеках, в більшості випадків є застарілими відносно сучасних архітектур процесорів та не задовольняють технічні потреби сучасних проектів. Найчастіше це пов’язано з поганою розпаралеленістю методів та поганою сумісністю з сучасними архітектурами процесорів.

До найбільш відомих фільтрів для зниження шуму можна віднести усереднюючі фільтри (арифметичний, геометричний, гармонічний), фільтр Гауса [6], білатеральний фільтр [7]. Хоча ці фільтри можна віднести до найпростіших, але без засобів прискорення навіть вони будуть виконуватись надто багато часу на великих зображеннях. Для збільшення швидкодії розробляються вдосконалені реалізації методів. Так, для усереднюючих фільтрів можлива рекурсивна реалізація. Для фільтра Гауса можливе підвищення ефективності за рахунок використання лінійної сепарабельності [6]. Крім того, реалізація гаусового розмиття може бути здійснена послідовністю усереднюючих фільтрів різних радіусів, а отже, можливим в цьому випадку є і рекурсивний підхід [8].

Інший спосіб підвищення ефективності роботи з графічною інформацією – використання можливостей графічного процесору (GPU). Однак цей спосіб не завжди може бути використаний, оскільки виділення як матеріальних ресурсів, так і енергетичних не завжди можливе для деяких проектів. Так, для веб-проектів використання графічних прискорювачів може бути не лише надмірно дорогим та неповоротким при потребі розширення потужностей, а й мало ефективним при необхідності обробляти велику кількість зображень. Ця проблема полягає у специфіці проекту та роботи з графічним прискорювачем [9]. Іншим прикладом є використання одноплатних комп’ютерів (SBC) в проектах різної спеціалізації. SBC в своїй більшості не мають в своєму розпорядженні потужного графічного адаптеру, а при його наявності потребують великої кількості енергії для підтримки роботи, чим суттєво зменшують свою мобільність.

Ще один шлях для зменшення часу роботи методів обробки зображень – паралелізація алгоритмів, що також є суттєвим завданням вдосконалення алгоритмів.

Метою даної роботи є дослідження ефективності застосування алгоритмічних та технологічних способів прискорення методів обробки цифрових зображень на прикладі методів видалення шуму.

Постановка задачі. Нехай $I=g(x, y)$ – дискретна функція, яка визначає вихідне зображення з наявним шумом. (x, y) – координати пікселів піксельної сітки розміром $M\times N$, на якій визначено зображення, $x=1..M$, $y=1..N$. Значення функції g – це значення яскравості зображення у пікселі для напівтонових зображень або значення колірних *RGB*-каналів для повнокольорових зображень. Частіше за все, g – цілі числа з діапазону [0, 255] або дійсні числа з діапазону [0, 1]. Задача видалення шуму полягає в побудуванні перетворення $g(x, y)\rightarrow f(x, y)$, де $f(x, y)$ – результуюче знешумлене зображення. Для дослідження обрані методи фільтрації, які реалізують зазначене перетворення: усереднюючі фільтри та фільтр Гауса. Необхідно розглянути алгоритмічні та технологічні способи збільшення швидкодії зазначених методів, реалізувати їх у програмному додатку та провести тестування швидкості роботи методів за масового навантаження.

Метод розв'язання задачі. Для пониження шуму на цифрових зображеннях використовують низькочастотні фільтри, до яких належать і усереднюючі фільтри. Результатом такої фільтрації є розмиття зображення. А дію подібних фільтрів можна представити у вигляді згортки зображення, що обробляється, з фільтром, ядро якого представлено у вигляді матриці або маски з певними значеннями [6]. Загальна формула фільтрації має вигляд:

$$f(x,y)=\sum_{(s,t)\in S_{xy}} w(s,t)g(s,t), \qquad (1)$$

де $w(s, t)$ – вагові коефіцієнти матриці ядра фільтра, S_{xy} – окіл пікселя (x, y), для якого розраховується нове значення. Повна фільтрація зображення досягається застосуванням формули (1) до усіх пікселів. Характерною ознакою низькочастотних фільтрів є те, що всі коефіцієнти маски невід'ємні, а їх сума дорівнює одиниці [6]. У більшості випадків матриця, яка визначає ядро подібних фільтрів, симетрична і має квадратну або кругову форму.

Найпростішим усереднюючим фільтром є середньоарифметичний фільтр. Значення яскравості пікселя (x, y) відновленого зображення обчислюється за формулою:

$$f(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t),$$

де $m\times n$ – розміри прямокутного околу S_{xy} пікселя (x, y). Тобто вагові коефіцієнти матриці ядра фільтру однакові й дорівнюють $1/mn$. Заміна значень елементів зображення на середні значення по околу призводить до зменшення різких перепадів яскравості, якими характеризується шум. Отже результатом застосування такої фільтрації буде зменшення шуму.

Іншим усереднюючим фільтром є середньогеометричний фільтр, який описується виразом:

$$f(x,y)=\left[\prod_{(s,t)\in S_{xy}} g(s,t)\right]^{\frac{1}{mn}}.$$

Застосування середньогеометричного фільтру призводить до згладжування, яке можна порівняти з тим, що досягається використанням середньоарифметичного фільтру, але при цьому втрачається менше деталей зображення, хоча нові значення яскравості обчислюються довше.

Результатом дії усереднюючих фільтрів є зменшення різких перепадів яскравості. Проте контури на зображенні також характеризуються перепадами яскравості, отже негативним результатом застосування згладжувальних фільтрів є розфокусування контурів. Для зменшення цього негативного ефекту застосовують неоднорідні згладжувальні фільтри з різними ваговими коефіцієнтами маски, що показують ступінь впливу відповідного пікселя на результат згортки, а сума цих коефіцієнтів дорівнює одиниці. Прикладом такого фільтру є фільтр Гауса [6], у якому в центрі маски вагові коефіцієнти значно більші, ніж на границях. Для фільтрації зображень використовується двовимірний фільтр Гауса:

$$G_\sigma=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}},$$

де σ – стандартне відхилення розподілення Гауса. Коефіцієнти матриці дискретного ядра визначаються за формулою:

$$w(s,t)=\frac{1}{2\pi\sigma^2}e^{-\frac{(s-r-1)^2+(t-r-1)^2}{2\sigma^2}},$$

де *s*, *t* – значення індексів елемента в матриці (*w*(1, 1) – верхній лівий елемент матриці), *r* – радіус ядра фільтру у пікселях, тобто розмір квадратної матриці фільтру (2*r*+1)×(2*r*+1). Зазвичай радіус фільтра обирають рівним 3σ. При такому виборі співвідношення σ та *r* за межами околу, що визначається матрицею фільтра, значення функції Гаусу будуть дорівнювати нулю або нехтувано малі, що гарантує достатню точність наближення розподілення Гауса. Чим більше σ, тим більше розмивається зображення при застосуванні фільтра. Нижче наведено приклад фільтра 5×5 для σ=1:

$$\begin{bmatrix} 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \\ 0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\ 0.022 & 0.097 & 0.159 & 0.097 & 0.022 \\ 0.013 & 0.059 & 0.097 & 0.059 & 0.013 \\ 0.003 & 0.013 & 0.022 & 0.013 & 0.003 \end{bmatrix}$$

Для того, щоб оперувати цілими числами, здійснюють масштабування коефіцієнтів матриці. Масштабування ядра виконується так, щоб усі коефіцієнти можна було б округлити до цілих чисел без істотних втрат у точності. Нормуючий множник згортки дорівнює зворотному значенню до суми усіх кое-

фіцієнтів, що дозволяє зберегти без змін області з постійною яскравістю. Далі наведені приклади масштабованих масок фільтра Гауса 3×3 та 5×5 (σ=1):

$$\frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}, \quad \frac{1}{273}\begin{bmatrix}1 & 4 & 7 & 4 & 1\\ 4 & 16 & 26 & 16 & 4\\ 7 & 26 & 41 & 26 & 7\\ 4 & 16 & 26 & 16 & 4\\ 1 & 4 & 7 & 4 & 1\end{bmatrix}.$$

Якщо виконувати фільтрацію зображення розміром $M\times N$ безпосередньо згорткою з фільтром розміру $n\times n$, то витрати часу на виконання фільтрації можна оцінити як $O(n^2\cdot M\cdot N)$.

Для апроксимації гаусової фільтрації може виконуватись багаторазова фільтрація з усереднюючим фільтром. В [8] показано, що прийнятне наближення до гаусового розмиття досягається вже при трикратному застосуванні середньоарифметичної фільтрації. При п'ятикратному застосуванні точність наближення зростає значною мірою. Проте подальше збільшення кількості проходів не має сенсу, оскільки приріст точності значно зменшується, при цьому час виконання суттєво збільшується. Крім того, у роботі [8] наведено метод визначення розмірів фільтрів усереднення для апроксимації гаусіану з заданим стандартним відхиленням.

Відзначимо, що описаний підхід до реалізації наближення фільтру Гауса не змінює теоретичну оцінку обчислювальної складності $O(n^2\cdot M\cdot N)$. Але за рахунок того, що вагові коефіцієнти маски середньоарифметичних фільтрів однакові, алгоритм буде працювати швидше, ніж звичайна гаусова фільтрація, незважаючи на те що згортку треба виконати декілька разів.

Важливою властивістю фільтра Гауса є його сепарабельність, тобто двовимірна функція Гауса може бути представлена у вигляді добутку двох одновимірних функцій:

$$G_\sigma(x,y)=\frac{1}{2\pi\sigma^2}e^{-\frac{x^2+y^2}{2\sigma^2}}=\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{x^2}{2\sigma^2}}\times\frac{1}{\sqrt{2\pi}\sigma}e^{-\frac{y^2}{2\sigma^2}}.$$

У дискретному випадку застосування двовимірної матриці можна замінити застосуванням двох одновимірних матриць: спочатку виконати згортку з однією одновимірною матрицею по всім горизонтальним напрямкам, потім з іншою одновимірною матрицею – по вертикальним. Схематично розкладення 2D фільтру на добуток двох 1D фільтрів можна представити так:

$$\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}=\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\times\begin{bmatrix}1 & 2 & 1\end{bmatrix}$$

а згортка буде виконуватись за принципом «горизонталь-вертикаль» таким чином:

$$f_h(x,y)=\sum_{s=x-r}^{x+r} w_h(s,y)g(s,y), \quad f(x,y)=\sum_{t=y-r}^{y+r} w_v(x,t)\,f_h(x,t),$$

де $w_h(s, t)$, $w_v(s, t)$ – вагові коефіцієнти горизонтальної та вертикальної одновимірних матриць. За рахунок такого підходу час виконання фільтрації знизиться до $O(n\cdot M\cdot N)$. Очевидно, що те ж саме справедливо і для середньоарифметичного фільтру, причому коефіцієнти одновимірних матриць однакові і дорівнюють $1/n$.

Ще одним алгоритмічним способом прискорення фільтрації зображення усереднюючим фільтром та фільтром Гауса є застосування рекурентних співвідношень для обчислень одновимірних згорток. Розглянемо середньоарифметичний фільтр, для якого $w_h(s, t)=w_v(s, t)= w=$const. Обчислення з одновимірною горизонтальною матрицею у пікселі (x, y) буде мати вигляд:

$$f_h(x,y)=w(g(x-r,y)+g(x-r+1,y)+...+g(x+r-1,y)+g(x+r,y)).$$

Аналогічна формула для наступного по горизонталі пікселя $(x+1, y)$ буде отримуватися зсувом по рядку матриці на одну позицію вліво. Отже, значення наступного пікселя може бути обчислено за формулою:

$$f_h(x+1,y)=f_h(x,y)-w\cdot g(x-r,y)+w\cdot g(x+r+1,y).$$

Відзначимо, що таке 1D розмиття має обчислювальну складність $O(M\cdot N)$ (незалежно від розміру маски фільтра). Враховуючи, що подібну операцію треба виконати і по вертикальному напрямку, отримуємо, що загальна складність такого розмиття середньоарифметичним фільтром складатиме $2O(M\cdot N)$. Для Гаусового розмиття, яке отримується k-кратним повторенням середньоарифметичної фільтрації, обчислювальна складність описаного підходу становить $2k\cdot O(M\cdot N)$.

Отже, ми розглянули алгоритмічні способи підвищення швидкості реалізацій методів фільтрації зображень. Ще одним інструментом прискорення методів є їх паралелізація, що потребує додаткових доопрацювань алгоритмів.

Використання паралелізації у методах фільтрації. Можна визначити декілька способів паралельної роботи у комп'ютерних системах [5]:

– на рівні додатку – операції виконуються в різних потоках, пропускна здатність системи зростає пропорційно кількості ядер;

– на рівні графічних операцій – графічна бібліотека отримує якусь операцію, всередині себе створює необхідну кількість потоків, розбиває одну операцію на кілька дрібніших і виконує їх. При цьому реальний час виконання зменшується, одна операція виконується швидше. Але пропускна здатність зростає далеко не лінійно від кількості ядер, оскільки є операції, які не паралеляться;

– на рівні команд процесора і даних – метод полягає у векторизації даних перед відправкою на опрацювання процесору. Це підхід SIMD (англ. single instruction, multiple data – одиночний потік команд, множинний потік даних). Реальний час виконання зменшується, пропускна здатність зростає.

Підхід SIMD добре поєднується з розпаралелюванням всередині операції та SIMD добре поєднується з розпаралелюванням всередині програми. Але розпаралелювання всередині програми і всередині операції один з одним не поєднуються, тобто переваг від цього не буде отримано [5].

Проведемо дослідження того, як використання SIMD впливає на ефективність виконання фільтрації зображення. Для використання SIMD необхідно здійснити векторизацію метода фільтрації, тобто від роботи з елементами матриці пікселів перейти до роботи з векторами і всі операції звести до векторних. Першим очевидним шляхом векторизації методів обробки повноколірних *RGB*-зображень є векторизація на рівні пікселя, тобто перехід від розгляду окремих значень яскравості кожного з колірних каналів *R, G, B* до здійснення операцій над трикомпонентними векторами.

Інший більш ефективний підхід з використанням SIMD технології полягає у виділенні векторних операцій всередині методу. Розглянемо модифікацію «горизонталь-вертикаль» середньоарифметичного фільтра, тобто розглянемо метод послідовних горизонтальної та вертикальної згорток з одновимірними матрицями. Перший етап методу середньоарифметичної фільтрації не викликає проблем з векторизацією: в якості першого вектору треба взяти частину рядка зображення, яка відповідає розміру регістра процесора, за інші вектори обираються відповідні частини наступних рядків, за рахунок чого операції будуть підтримуватись командами SIMD. Кількість векторів визначається розміром фільтру. Другий етап фільтрації, який реалізує згортку стовпців, є проблемним для безпосереднього впровадження SIMD, оскільки ця технологія має обмеження: за один запит можна обирати тільки послідовно розташовані елементи. Виходом з цієї ситуації є транспонування матриці зображення та проведення обчислень аналогічно першому етапу. По завершенні – ще одне транспонування, яке поверне матрицю до необхідного вигляду. Отже, впровадження SIMD дозволить досягти підвищення продуктивності, але будуть додані дві операції транспонування.

Транспонування матриці може потребувати доволі багато часу, що знищить досягнуті переваги, тому цю операцію також можна вдосконалити за рахунок SIMD команд. Для цього пропонується розбивати матрицю на підматриці та транспонувати їх без відношення одна до одної з подальшою побудовою нової матриці. Розмір підматриць обирається автоматично в залежності від платформи виконання програмної реалізації (в даному дослідженні 16×16 для першої тестової машини та 8×8 – для інших двох). Для того, щоб матриця поділилася на підматриці обраного розміру без залишків, можна доповнити вихідну матрицю до потрібного розміру повторенням граничних значень або використати інший спосіб: фрагменти, що залишаться після розбиття вихідної матриці на підматриці, транспонувати класичним методом. У даній роботі використовується другий підхід.

Програмна реалізація. Описані методи були реалізовані у програмному забезпеченні, яке було використано для дослідження ефективності розглянутих рішень. Основними інструментами розробки були: мова C#, в якій є підт-

римка SIMD; фреймворк .Net Core 3.1.; бібліотека BenchmarkDotNet, за допомогою якої реалізовані можливості для тестування ефективності методів у різних умовах використання.

До структури розробленого програмного додатку увійшли:

– VisualWorkSpace – модуль для візуальної перевірки коректності реалізованих методів та експериментування з ними;

– Methods – методи фільтрації зображень;

– BenchmarkCore – модуль тестування продуктивності методів.

До набору методів, які реалізовані в системі, увійшли середньоарифметичний, середньогеометричний фільтри та фільтр Гауса (різні модифікації, описані вище). Крім того, були додані середньогармонічний та медіанний фільтри. Отже, система дозволяє здійснити видалення шуму з зображення будь-яким з зазначених методів. Крім того, за допомогою модуля тестування можна дослідити продуктивність реалізованих методів.

Експериментальне дослідження. Перевірка продуктивності проводилась на трьох комп'ютерах з різними архітектурами та операційними системами, параметри яких наведені у таблиці 1.

Перше дослідження було проведено відносно середньогеометричного фільтра з метою перевірки доцільності використання вбудованої бібліотеки System.Drawing для реалізації методів у програмному додатку, а також для перевірки переваг поєднання паралельності та SIMD. Для цього були використані такі модифікації:

– RootMeanSquareNaive – звичайна реалізація фільтра з використанням Bitmap та роботою з пікселем через засоби System.Drawing;

Таблиця 1

Параметри тестових комп'ютерних систем

Параметри комп'ютера	Тестова машина № 1	Тестова машина № 2	Тестова машина № 3
Операційна система	Windows 10.0.17763.1577	Linuxmint 20	Ubuntu 20.10
Процесор	AMD Ryzen 5 3550H with Radeon Vega Mobile Gfx 2.2 – 3.2 GHz	Celeron Dual-Core CPU T3000 1.80GHz	Broadcom BCM2711 с ядрами Cortex-A72 1.60GHz
Логічні ядра	8	2	4
Фізичні ядра	4	2	4
Архітектура процесору	AMD64	AMD64	ARM64
Операційна пам'ять, GB	8	4	4

– RootMeanSquare_MOD – той самий алгоритм, але без використання засобів System.Drawing для опрацювання пікселів;

– RootMeanSquareParallel –модифікація з розбиттям розрахунків за стовпцями з використанням Parallel.For для розпаралелення обчислень;

– RootMeanSquareParallel_SIMD – модифікація з використанням SIMD у вже розпаралеленому методі. Векторизація виконана на рівні пікселів, за-

мість окремих значень яскравості кожного з колірних каналів *R*, *G*, *B* використані трикомпонентні вектори.

Результати тестування роботи різних реалізацій методу на тестовій машині 1 наведені в таблиці 2.

Таблиця 2

Результати вимірів часу виконання (у мілісекундах) різних реалізацій середньогеометричного фільтру

Модифікація методу	Радіус ядра фільтру	
	1	2
RootMeanSquareNaive	2,193.97 ms	5,677.13 ms
RootMeanSquare_MOD	102.49 ms	137.74 ms
RootMeanSquareParallel	44.34 ms	53.87 ms
RootMeanSquareParallel_SIMD	29.90 ms	39.00 ms

Експерименти показали, що при відмові від використання System.Drawing швидкість обробки зображення збільшується в декілька разів та прибирається проблема геометричного ускладнення алгоритму при збільшенні радіусу, тобто різниця між часом фільтрації для радіусів 1 і 2 не перевищує 40%. Отже, решта методів у програмній системі реалізовано без використання System.Drawing. Розпаралелення обчислень та використання SIMD надають певні переваги перед класичною реалізацією, але результати не є близькими до теоретичних. Головним чином це пов’язано з низьким коефіцієнтом продуктивності використання трикомпонентних векторів.

Детальні дослідження були проведені для модифікації «горизонталь-вертикаль» середньоарифметичного фільтру:

– BlurRudeMethod_HV – звичайна реалізація,

– BlurRudeMethod_SIMD – реалізація з інтеграцією SIMD команд.

Тестування проводилось на зображенні розміром 1920×1080 з глибиною кольору 32 біта. Були застосовані фільтри з радіусами 1, 2, 5, 10. Результати замірів швидкості виконання обробки зображення наведені у таблиці 3.

Таблиця 3

Результати тестування швидкодії середньоарифметичного фільтру

Метод обробки	Радіус ядра фільтру	Час виконання на тестовій машині		
		№ 1	№ 2	№ 3
BlurRudeMethod_HV	1	1,399.52 ms	2,743.8 ms	2,065.3 ms
BlurRudeMethod_SIMD	1	101.13 ms	485.0 ms	376.2 ms
BlurRudeMethod_HV	2	1,412.59 ms	2,875.4 ms	2,069.7 ms
BlurRudeMethod_SIMD	2	96.56 ms	485.9 ms	374.2 ms
BlurRudeMethod_HV	5	1,458.63 ms	2,794.0 ms	2,066.2 ms
BlurRudeMethod_SIMD	5	95.89 ms	489.0 ms	364.2 ms
BlurRudeMethod_HV	10	1,430.56 ms	2,821.4 ms	2,068.9 ms
BlurRudeMethod_SIMD	10	99.81 ms	485.3 ms	374.5 ms

Ще один експеримент був проведений з обробкою одразу декількох зображень:

– BlurRudeMethod_SIMD_NotPar – 30 послідовно оброблених кадрів методом BlurRudeMethod_SIMD;

– BlurRudeMethod_SIMD_Par – 30 паралельно оброблених кадрів методом BlurRudeMethod_SIMD.

Результати замірів швидкості виконання обробки з фільтрами різних радіусів наведені у таблиці 4. Для порівняння у таблиці також наведений час обробки одного зображення.

Таблиця 4

Результати тестування швидкодії фільтрації на низці зображень

Метод обробки	Радіус ядра фільтру	Час виконання на тестовій машині	
		№ 1	№ 3
BlurRudeMethod_HV	1	1,399.52 ms	2,065.3 ms
BlurRudeMethod_SIMD	1	101.13 ms	376.2 ms
BlurRudeMethod_SIMD_NotPar	1	3,098.92 ms	10,022.7 ms
BlurRudeMethod_SIMD_Par	1	1,210.57 ms	7,035.8 ms
BlurRudeMethod_HV	2	1,412.59 ms	2,069.7 ms
BlurRudeMethod_SIMD	2	96.56 ms	374.2 ms
BlurRudeMethod_SIMD_NotPar	2	3,188.40 ms	10,083.8 ms
BlurRudeMethod_SIMD_Par	2	1,258.05 ms	5,524.2 ms
BlurRudeMethod_HV	5	1,458.63 ms	2,066.2 ms
BlurRudeMethod_SIMD	5	95.89 ms	364.2 ms
BlurRudeMethod_SIMD_NotPar	5	3,050.18 ms	10,020.6 ms
BlurRudeMethod_SIMD_Par	5	1,301.23 ms	5,549.8 ms
BlurRudeMethod_HV	10	1,430.56 ms	2,068.9 ms
BlurRudeMethod_SIMD	10	99.81 ms	374.5 ms
BlurRudeMethod_SIMD_NotPar	10	3,027.88 ms	10,013.8 ms
BlurRudeMethod_SIMD_Par	10	1,093.78 ms	5,572.8 ms

Аналіз результатів. Результати тестування, які наведені у таблиці 3, свідчать про те, що впровадження SIMD операцій пришвидшує виконання методу в 5-14 разів в залежності від SSE процесору. В процесі тестувань різних варіантів методів було виявлено деякі технічні особливості кожного з методів. Так, однопоточна реалізація фільтру без SIMD модифікацій була чутлива до навантаження усієї системи, значно погіршуючи результати швидкодії. Навіть гірша ситуація була при розпаралеленні на декілька потоків, оскільки усі ресурси системи були зайняті методом, який конкурував з іншими процесами, в той час, як реалізації на SIMD зазвичай не були настільки чутливі до навантаження системи. Це пояснюється тим, що SIMD операції проводяться циклічно з довгими перервами між операціями, головним чином у зв’язку з потребою завантаження великих об’ємів даних за один раз, через що процесор в моменти простою опрацьовував інші потоки. Отже, можна зазначити, що подібні методи більш ефективно використовують ресурси в багатопоточних програмах. Однак виключенням є проведення SIMD процесів на усіх яд-

рах, що призводить до відхилення від теоретично можливих результатів. Дана тенденція прослідковується в результатах експерименту, які наведені у таблиці 4. Як можна зазначити з результатів прискорення при розбитті на декілька потоків, на AMD64 процесорі відбувся приріст продуктивності трохи більше, ніж у 2 рази, при наявності чотирьох ядер процесору. В свою чергу, на ARM64 результати були гірші: приріст був від 1,5 до 2-х разів при наявності чотирьох фізичних ядер.

Оскільки середньоарифметичний фільтр лежить в основі розглянутих вище реалізацій фільтру Гауса, то можна очікувати, що подібні результати будуть отримані і для цього фільтру. Отже, разом з алгоритмічними вдосконаленнями методів доцільно використовувати і технологічні способи підвищення швидкодії методів.

Висновки. У роботі проаналізовані найбільш розповсюджені методи видалення шуму з цифрових зображень, розглянуто способи їх прискорення та розпаралелення на різних рівнях. Для аналізу ефективності різних способів інтегрування паралельності у методи були розроблені модифікації алгоритмів. Головна увага була приділена методам впровадження SIMD операцій у існуючі модифікації фільтрів. Практичним результатом роботи є програмний додаток, в якому реалізовані розглянуті методи фільтрації та їх модифікації, а також модуль для тестування продуктивності методів. Як результат дослідження отримано, що найпростіше впровадження SIMD, яке полягає у представленні пікселя як вектора, надало приріст продуктивності фільтру у розмірі 1,2 – 2 разів в залежності від типу фільтру та процесору. В той же час, повне переведення алгоритму на векторну основу дає приріст швидкості більш ніж в 14 разів для сучасних AMD64 процесорів та більш ніж в 5 разів для ARM64 процесорів, і це не зважаючи на необхідність додати двократне транспонування в алгоритм фільтрації. Отже, можна зробити висновок, що для підвищення ефективності доцільними є не тільки алгоритмічні вдосконалення методів, а й розробка модифікацій з використанням векторних операцій для ефективної роботи з сучасними процесорами, зокрема для впровадження технології SIMD.

Бібліографічні посилання

1. Patil J., Jadhav S. A Comparative Study of Image Denoising Techniques. *International Journal of Innovative Research in Science, Engineering and Technology*, Vol. 2, No. 3, 2013.
2. Hambal A.M., Pei Z., Ishabailu F. L. Image Noise Reduction and Filtering Techniques. *International Journal of Science and Research (IJSR).* Vol. 6, Issue 3, March 2017. [Electronic resource]. Access mode: https://www.ijsr.net/archive/v6i3/25031706.pdf. DOI: 10.21275/25031706.
3. Jasim M.K., Najm R.H., Kanan E.H., Alfaar H.E. Image Noise Removal Techniques: A Comparative Analysis. *International Journal of Science and Applied Information Technology*, Vol.8, No.6, November-December 2019. [Electronic resource]. Access mode: http://www.warse.org/ijsait/static/pdf/file/ijsait01862019.pdf.

4. Чочиа П.А. Сглаживание изображений: сравнительный анализ методов фильтрации, основанных на парзеновском оценивании. *Современные информационные технологии и ИТ-образование*. Т. 12. № 2. 2016. С. 216-222.
5. Карпинский А. Работа с изображениями на Python. 2018. [Електронний ресурс]. Режим доступу: https://www.pvsm.ru/python/296175
6. Старовойтов В.В., Голуб Ю.И. Цифровые изображения: от получения до обработки. Минск: ОИПИ НАН Беларуси, 2014. 202 с.
7. Yang, Q., Tan K. H., Ahuja N. Real-time O(1) bilateral filtering. *Proc. IEEE Conference on Computer Vision and Pattern Recognition.* Pp. 557-564. 2009. [Electronic resource]. Access mode: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.224.4602&rep=rep1&type=pdf
8. Kovesi P. Fast Almost-Gaussian Filtering. IEEE Computer Society. (2010). Pp. 121-125. [Electronic resource]. Access mode: https://www.peterkovesi.com/papers/FastGaussianSmoothing.pdf
9. Brain N. GPGPU via C#: краткий обзор. 2018. [Електронний ресурс]. Режим доступу: https://dou.ua/lenta/articles/gpgpu-via-csharp/

Надійшла до редколегії 29.07. 2021.