ОБРОБКА ЗОБРАЖЕНЬ ТА РОЗПІЗНАВАННЯ ОБРАЗІВ

УДК 004.043 https://doi.org/10.15407/vidbir2024.52.074

ЗАСТОСУВАННЯ РІЗНИЦЕВИХ КОЛІРНИХ МОДЕЛЕЙ ДО ФРАГМЕНТІВ RGB-ЗОБРАЖЕНЬ ПЕРЕД ПРОГРЕСУЮЧИМ ІЄРАРХІЧНИМ СТИСНЕННЯМ БЕЗ ВТРАТ

О. В. Шпортько[1], А. Я. Бомба[2]

[1] Міжнародний економіко-гуманітарний університет імені академіка Степана Дем'янчука, Рівне;
[2] Національний університет водного господарства та природокористування, Рівне

E-mail: ITShportko@gmail.com, ABomba@ukr.net

Запропоновано спосіб і відповідний алгоритм ієрархічного обходу мінімальних кодових одиниць (MCU – блоків 8 × 8 пікселів) під час препроцесингу стиснення зображень без втрат. Наведено алгоритм вибору для кожної MCU ефективної різницевої колірної моделі з цілими коефіцієнтами з переліку основних альтернативних моделей, яка прогнозовано мінімізує коефіцієнт стиснення зображення на основі аналізу ентропії. Описано та реалізовано алгоритм компактного ієрархічного зберігання номерів обраних різницевих колірних моделей для кожної MCU, який враховує високі ймовірності повторення цих номерів для суміжних MCU. На загальновідомому тестовому наборі Archive Comparison Test показано, що застосування різницевих колірних моделей з цілими коефіцієнтами до MCU-фрагментів зображень з можливістю переходу до єдиної різницевої колірної моделі для цілого зображення дає змогу зменшити коефіцієнти стиснення в середньому на 0,02 bpb, але при цьому сповільнює кодування в 2,88 раза та декодування на 3,39%, тому такий препроцесинг не рекомендовано використовувати в графічних форматах. Наголошено, що різницеві колірні моделі варто застосовувати до окремих MCU в архіваторах для забезпечення максимального стиснення зображень без втрат.

Ключові слова: *ієрархічне стиснення зображень, стиснення без втрат, MCU, різницеві колірні моделі з цілими коефіцієнтами.*

APPLICATION OF DIFFERENCE COLOR MODELS TO RGB IMAGE FRAGMENTS BEFORE PROGRESSIVE HIERARCHICAL LOSSLESS COMPRESSION

A. V. Shportko[1], A. Ya. Bomba[2]

[1]Academician Stepan Demianchuk International University of Economics and Humanities, Rivne;
[2]National University of Water and Environmental Engineering, Rivne

The method and corresponding algorithm for hierarchical bypass of minimum code units (MCU – blocks of 8 × 8 pixels) in the process of preprocessing of lossless image compression is proposed. The algorithm for selecting for each MCU an effective difference color model with integer coefficients from the list of basic alternative models, which predictable minimizes the image compression ratio based on entropy analysis is given. An algorithm for compact hierarchical storage of numbers of selected difference color models for each MCU, which takes into account the high probability of repetition of these numbers for adjacent MCUs is described and implemented. It is shown that both color models for individual MCUs and a single color model for the entire image can be effective for difference images. To evaluate the effectiveness of preprocessing algorithms during hierarchical compression, the feasibility of using the weighted average entropy by passes instead of the total entropy, which is calculated only by pixels processed by a context-independent algorithm is proposed. It is noted that the entropy by 4 passes is always less than the total entropy, because it takes into account the different unevenness of the probability distribution of elements on different passes, that is why it is advisable to store the

data of different passes in different compressed blocks. On the well-known Archive Comparison Test suite, it has been demonstrated that applying integer difference color models to MCU image fragments with the option of switching to a single difference color model for the entire image can reduce compression ratios by an average of 0.02 bpb but at the same time slows down the encoding by a factor of 2.88 and decoding by 3.39%, so such preprocessing is not recommended to be used in graphic formats. It is emphasized that difference color models should be applied to individual MCUs in archivers in order to ensure maximum lossless image compression.

Keywords: *hierarchical image compression, lossless compression, MCU, difference color models with integer coefficients.*

Вступ. Як відомо, зображення суттєво полегшують і прискорюють сприйняття інформації людиною. Саме тому на сьогодні вони є невід'ємною складовою мультимедійної інформації, яку найчастіше передають каналами зв'язку чи зберігають на електромагнітних носіях. Отже, підвищення ефективності стиснення зображень є актуальним завданням і буде в найближчому майбутньому.

Всі графічні формати та методи стиснення даних зображень поділяють на два основні класи: з втратами (наприклад, JPEG [1]) та без втрат (наприклад, PNG [2]). І якщо для більшості алгоритмів компресії зображень з втратами можна забезпечити потрібний коефіцієнт стиснення (відношення розмірів стиснутого до нестиснутого файлів зображення, виражене в bpb, надалі – КС), погіршуючи якість, то рівень стиснення зображень без втрат залежить, власне, лише від перепадів кольорів їх пікселів та самого алгоритму стиснення, він не регулюється програмно і становить в середньому тільки 30...70% [3]. Тому розроблення методів препроцесингу таких, як альтернативні різницеві колірні моделі з цілими коефіцієнтами [4], які дають змогу зменшити КС, на сьогодні є актуальним завданням.

Аналіз останніх досліджень і публікацій. Будь-яке стиснення даних можливе внаслідок зменшення чи ліквідації надлишковостей [5]. У зображеннях розрізняють три основні типи надлишковостей [6]: візуальну (полягає в наявності інформації, яку не сприймає зорова система людини), міжелементну або просторову (проявляється в кореляції яскравостей суміжних пікселів) та кодову (виявляється за умови використання кодів однакової довжини для елементів з різними ймовірностями). Зрозуміло, що більше видів надлишковостей кожного типу опрацьовують графічним форматом, то ефективніше стиснення. Але під час стиснення без втрат інформація не втрачається, ось чому перший тип надлишковостей не зменшується. Як наслідок, стиснення зображень без втрат в архіваторах та графічних форматах найчастіше відбувається максимум в чотири етапи, причому, власне, компресію виконують лише на першому та останньому з них: на першому – контекстно-залежне кодування зменшує надлишковості між однаковими фрагментами чи фрагментами з однаковою структурою (зменшує міжелементну надлишковість) [7], а на четвертому (останньому) – контекстно-незалежне кодування формує коди елементів з довжинами, залежними від їх ймовірностей (опрацьовує кодову надлишковість, наприклад, кодами Хафмана чи арифметичними кодами [2, 3, 5, 8]).

Основний принцип контекстно-незалежного кодування четвертого етапу сформулюємо так: **довжина коду довільного елемента з більшою ймовірністю не повинна перевищувати довжину коду будь-якого елемента з меншою ймовірністю**. Стосовно зображень, цей принцип базується на фундаментальному положенні теорії інформації, згідно з яким, для мінімізації довжини коду послідовності кожне значення елемента *i* (яскравість окремої компоненти *brightness* (для окремої компоненти кожного пікселя зображень True Color $brightness = \overline{0, 255}$) чи значення контекстно-залежного коду) з ймовірністю появи p_i у двійковій систе-

мі доцільно кодувати $l_i = -\log_2 p_i$ бітами [5], де l_i називають *довжиною ентропійного коду елемента i* (тут і надалі в роботі логарифм береться за основою 2). Тоді середня довжина коду елемента блоку, згідно з формулою of Shannon, рівна *ентропії джерела* [6]:

$$H = -\sum_i p_i \times \log p_i \ . \tag{1}$$

Нехай кожне зі значень i зустрічається n_i разів в послідовності довжини $N = \sum_i n_i$. Згідно зі статистичним означенням ймовірності, $p_i = n_i / N$, тому довжина ентропійного коду елемента, до якого близька довжина контекстно-незалежного коду, становить $l_i = -\log p_i = \log \frac{N}{n_i}$, а загальна довжина *ентропійного коду послідовності* [8], враховуючи формулу (1), наближається до

$$L = N \times H = N \log\left(N\right) - \sum_i n_i \log\left(n_i\right). \tag{2}$$

Після застосування будь-якого контекстно-незалежного алгоритму досягнути ентропійної довжини коду для всіх елементів вдається рідко, тому середня довжина контекстно-незалежного коду зазвичай незначно перевищує ентропію (1) [3].

Розробити кодування, яке забезпечить середню довжину коду, меншу від ентропії, неможливо, тому на практиці намагаються зменшити ентропію через збільшення нерівномірності розподілу ймовірностей (частот) між елементами просторовою декореляцією [5]. Для цього на другому етапі стиснення зображень без втрат найчастіше виконують перехід до альтернативної колірної моделі [4, 9], а на третьому – яскравості компонентів пікселів перетворюють за допомогою предикторів [10]. Нагадаємо принципи цих методів препроцесингу.

Предиктори під час обходу пікселів зображення прогнозують значення яскравості кожної компоненти (для найпоширеніших 24-бітних зображень – це яскравості червоної, зеленої та синьої компоненти, записані цілими числами в окремих байтах), використовуючи значення яскравостей тих же компонент опрацьованих раніше суміжних пікселів [10], оскільки ці яскравості мають між собою найбільшу кореляцію. При цьому обчислюють і далі кодують відхилення Δ_{uv} значення яскравості чергової компоненти пікселя $brightness_{uv}$ від прогнозованого обраним предиктором значення $predict_{uv}$:

$$\Delta_{uv} = brightness_{uv} - predict_{uv} \tag{3}$$

(u та v пробігають відповідно у всіх рядках та стовпцях компонентів пікселів зображення). Суміжні пікселі зображень найчастіше мають подібні кольори (близькі значення яскравостей відповідних компонентів), тому прогноз часто збігається зі значенням яскравості чергової компоненти, найчастіше – є близьким до цього значення і рідко – значно відрізняється від нього. Тобто більшість значень Δ_{uv} є близькими до нуля. У такий спосіб застосування предикторів найчастіше збільшують нерівномірність розподілу ймовірностей значень яскравостей і, як наслідок, зменшують ентропію (1).

Різницеві колірні моделі теж застосовують для зменшення ентропії. Справа в тому, що різні компоненти зображень відображають достатньо подібні за геометрично-просторовою структурою об’єкти (як, наприклад, на рис. 1). Зрозуміло, що коефіцієнти кореляції між парами компонентів у традиційній колірній моделі RGB для різних зображень можуть суттєво відрізнятися між собою і одну з компонентів довільної такої пари з сильною кореляцією можна замінювати різницею

з іншою компонентою [4]. Але на сьогодні сучасні архіватори та формати стиснення зображень опрацьовують яскравості пікселів переважно у фіксованій колірній моделі (наприклад, формат PNG – в моделі *R, G, B*; BMP – в моделі B, G, R; JPEG [1] – в моделі Y, Cb, Cr; архіватор RAR – в моделі R–G, G, B–G) і не використовують можливості вибору ефективної колірної моделі для кожного зображення, яка максимально зменшує ентропію (1) внаслідок міжкомпонентної декореляції.

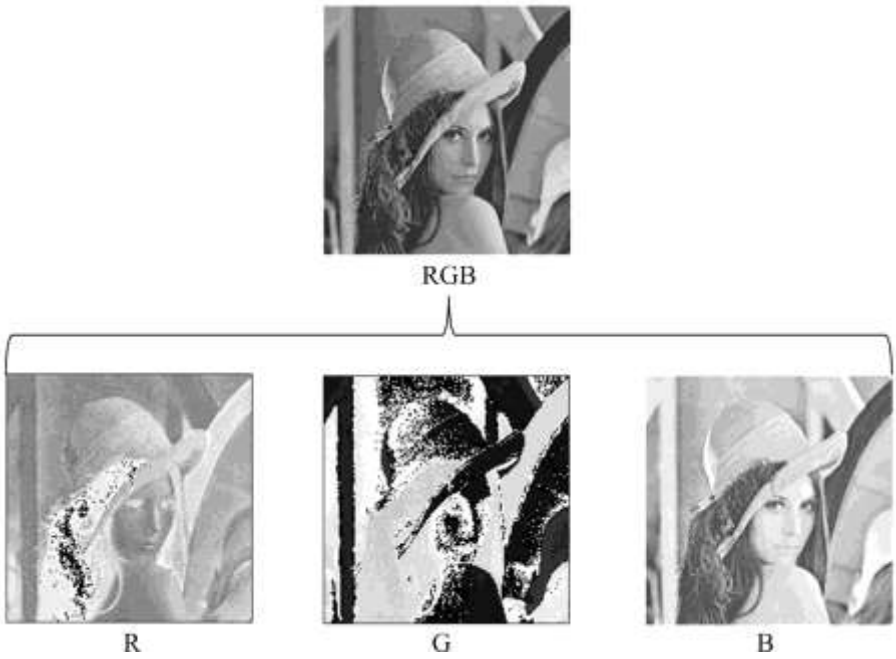

Рис. 1. Розклад зображення Lena.bmp з тестового набору ACT на компоненти колірної моделі RGB.

Крім цього, формати графічних файлів для стиснення зображень без втрат мають забезпечувати як швидке кодування, так і декодування, тому в них доцільно використовувати різницеві колірні моделі з цілими коефіцієнтами [4], обираючи їх з 16 основних альтернативних [11]:

0. R, G, B;
1. R, R–G–*middle*(R–G)+128, R–B–*middle*(R–B)+128;
2. R, R–G–*middle*(R–G)+128, B–G–*middle*(B–G)+128;
3. R, G–B–*middle*(G–B)+128, R–B–*middle*(R–B)+128;
4. G–R–*middle*(G–R)+128, G, G–B–*middle*(G–B)+128;
5. G–R–*middle*(G–R)+128, G, B–R–*middle*(B–R)+128;
6. R–B–*middle*(R–B)+128, G, G–B–*middle*(G–B)+128;
7. B–R–*middle*(B–R)+128, G–R–*middle*(G–R)+128, B;
8. R–G–*middle*(G–R)+128, B–G–*middle*(B–G)+128, B; (4)
9. B–R–*middle*(B–R)+128; B–G–*middle*(B–G)+128, B;
10. R, G, G–B–*middle*(G–B)+128;
11. R, G, R–B–*middle*(R–B)+128;
12. R, R–G–*middle*(R–G)+128, B;
13. R, B–G–*middle*(B–G)+128, B;
14. G–R–*middle*(G–R)+128, G, B;
15. B–R–*middle*(B–R)+128, G, B.

Тут *middle*() – функція для обчислення центру інтервалу з максимальною кількістю елементів. Але різні фрагменти зображень можуть мати неоднакову кореляцію між окремими компонентами, а окремі компоненти в них – різну ентропію. Тому для різних фрагментів зображень ефективними можуть виявитися відмінні різницеві колірні моделі. Проблему вибору ефективної колірної моделі для окремих фрагментів зображень не досліджували раніше і саме їй присвячено цю статтю.

З іншого боку, яскравості пікселів зображень у популярних графічних форматах, які виконують стиснення без втрат, найчастіше опрацьовують послідовно рядками зверху вниз, а у кожному рядку – зліва направо. Як наслідок – вивести стиснуте зображення у цих форматах можливо лише після завершення декодування, а декомпресія знімків чи малюнків з мільйонами пікселів за такого способу обходу може тривати декілька секунд незалежно від розміру області чи роздільної здатності пристрою виводу. Тому ми розробляємо формат HBF-LS [12] для прогресуючого ієрархічного стиснення зображень без втрат, який дасть змогу швидко отримувати зменшені копії зображення без декодування всього файла.

У форматі HBF-LS, як альтернативу послідовному обходу пікселів, ми запропонували ієрархічну схему [7–10], за якою на першому шарі пікселі зображення опрацьовують послідовно, починаючи з першого у верхньому лівому куті, рядками зверху вниз, а у кожному рядку – зліва направо з кроком $h_1 = 2^k$, де

$$k = \left\lfloor \log_2 \left(\frac{\max\left(\min\left(image\_row; image\_col\right); 16\right) - 1}{15} \right) \right\rfloor, \qquad (5)$$

image_row – кількість рядків, *image_col* – кількість стовпців пікселів зображення (рис. 2*а*). Цей крок забезпечує опрацювання на першому шарі принаймні 16 пікселів по кожній з осей, якщо зображення має не менші розміри.

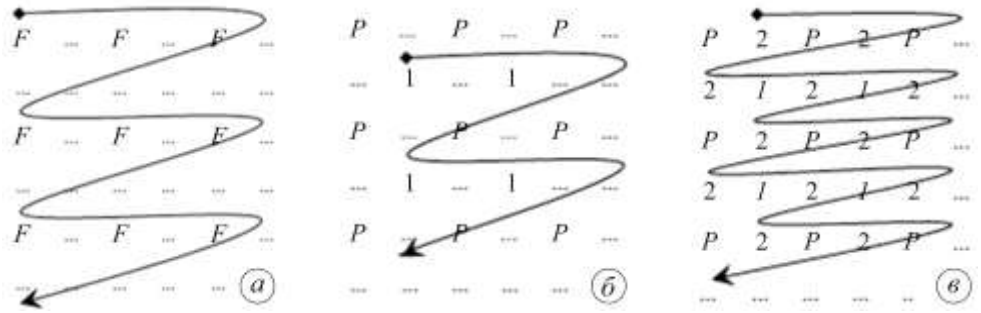

Рис. 2. Черговість обходу пікселів під час прогресуючого ієрархічного опрацювання пікселів першого шару (*а*); пікселів першого проходу чергового шару (*б*); пікселів другого проходу чергового шару (*в*).

На наступних шарах ($l = \overline{2, k+1}$) проміжні пікселі зображення обробляють двома проходами: на першому послідовно опрацьовують ті, які містяться на перетині діагоналей квадратів з вершинами у суміжних пікселях попередніх шарів з кроком $h_l = 2^{k+2-l}$ як по рядках, так і стовпцях (рис. 2*б*), а на другому необроблені пікселі послідовно обходять між суміжними пікселями попередніх шарів і пікселями першого проходу з тим же кроком по стовпцях і з удвічі зменшеним – по рядках (рис. 2*в*). На рис. 2 символом *F* позначені пікселі першого шару, символом *P* – пікселі попередніх шарів, цифрою 1 – пікселі першого проходу чергового шару, цифрою 2 – пікселі другого проходу чергового шару. Пікселі, які опрацьовані раніше і тому не опрацьовуються на черговому проході шару, виділені курсивом.

Мета роботи – дослідити можливості застосування різницевих колірних моделей з цілими коефіцієнтами до фрагментів RGB-зображень під час прогресуючого ієрархічного стиснення без втрат.

Результати дослідження. ***Обхід фрагментів під час прогресуючого ієрархічного опрацювання пікселів зображень.*** Для форматів графічних файлів одним з основних показників ефективності, поряд з КС, є час декодування, тому врахування фрагментування зображень у них має реалізовуватися так, щоб істотно не сповільнювати процес. Отже, дослідимо ефективність застосування різних різницевих колірних моделей з цілими коефіцієнтами не до окремих пікселів чи фрагментів довільної форми, а до мінімальних кодових одиниць (MCU – Minimum Coding Unit) розміром 8×8 пікселів (на правому чи нижньому краї зображення – можливо менше), які використовують, зокрема, і у форматі JPEG [1]. У цих блоках вже може бути доцільно виконувати міжкомпонентну декореляцію [3], а перейти від індексів пікселя до індексів MCU можна цілочисловим діленням на 8, або, що те саме, побітовим зсувом вправо на 3 біти.

Всі MCU мають однакові визначені розміри, тому зберігати ці параметри у стиснутих даних не потрібно. Достатньо для кожної MCU визначити і зберегти номер різницевої колірної моделі з переліку (4), який прогнозовано забезпечить найменший КС всього зображення.

Обхід MCU під час збереження та зчитування номерів їх різницевих колірних моделей з переліку основних альтернативних (4) (надалі ці обрані колірні моделі з переліку альтернативних (4) назвемо “ефективними”) реалізували відповідно до схеми прогресуючого ієрархічного обходу пікселів, наведеної на рис. 2. При цьому MCU чергового шару і проходу у верхньому лівому куті містять пікселі цього шару і проходу (наприклад, MCU першого проходу третього шару містять у верхньому лівому куті пікселі четвертого від початку ієрархічного проходу). Тому за необхідності декодування, наприклад, лише пікселів чотирьох перших шарів, з послідовності номерів різницевих колірних моделей достатньо зчитати номери тільки для перших чотирьох шарів MCU, а решту MCU обходити не потрібно.

На першому шарі MCU зображення опрацьовують послідовно, починаючи з першого у верхньому лівому куті, рядками зверху вниз, а у кожному рядку – зліва направо з кроком $h_1' = h_1 / 8 = 2^{k-3}$, де h_1 – крок обходу пікселів на першому шарі зображення, а k визначають з умови (5). Цей крок забезпечує опрацювання на першому шарі принаймні 16 MCU на кожній з осей, якщо зображення має не менші розміри. Приклад розміщення MCU двох перших шарів для зображення Lena.bmp наведено на рис. 3. На ньому всі MCU розмежовані білими лініями.

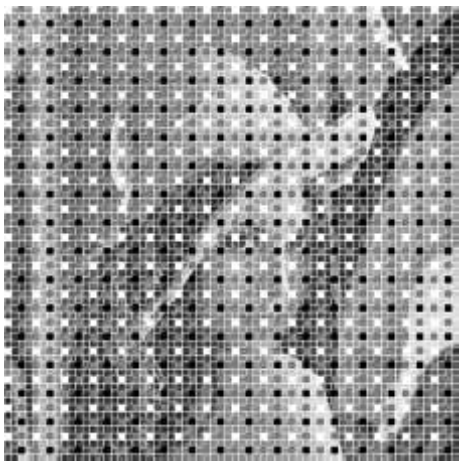

Рис. 3. MCU першого і другого шарів для зображення Lena.bmp
(MCU першого шару містяться в білих квадратах, першого проходу другого шару – в чорних, другого проходу другого шару – в сірих).

На наступних шарах ($l = \overline{2, k-2}$) проміжні MCU зображення обходять аналогічно окремим пікселям двома проходами: на першому послідовно опрацьовують ті з них, які містяться на перетині діагоналей квадратів з вершинами у суміжних MCU попередніх шарів, а на другому необроблені MCU послідовно обходять між суміжними MCU попередніх шарів і MCU першого проходу. На рис. 3 не зафарбовані MCU опрацьовують на третьому шарі. Кількість шарів для обходу всіх MCU на три менше кількості шарів для прогресуючого ієрархічного обходу пікселів, оскільки кожен MCU містить не більше восьми пікселів як по горизонталі, так і по вертикалі.

Ієрархічний спосіб зберігання номерів різницевих колірних моделей для окремих MCU. Для зберігання у стиснутих даних номера ефективної різницевої колірної моделі з переліку основних альтернативних (4) для кожної MCU достатньо 4 біти (2^4=16). Але врахуємо, що суміжні MCU тих же сцен найчастіше можуть мати однакові номери ефективних різницевих колірних моделей. Наприклад, для MCU фону зображення Lena.bmp найчастіше ефективною є альтернативна різницева колірна модель № 2 (на рис. 4 – темно-сірі квадрати). Ця колірна модель ефективна для всього зображення. Зрозуміло, що повторювати у стиснутих даних однаковий номер ефективної колірної моделі для суміжних MCU недоцільно. Тому під час ієрархічного обходу MCU початкових шарів намагаємося поширити номери обраних для них ефективних колірних моделей на всі суміжні MCU справа та знизу до MCU цього ж або вищого шару чи проходу (надалі називатимемо їх "підпорядкованими" MCU). Наприклад, для першої "білої" MCU рис. 3, розміщеної у верхньому лівому куті, як і для інших "білих" MCU, підпорядкованими будуть 15 суміжних MCU, які знаходяться правіше та нижче в квадраті 4×4 MCU, а для всіх "чорних" і "сірих" MCU підпорядкованими будуть 3 MCU (справа, знизу та справа знизу від них).

Якщо для всіх підпорядкованих MCU ефективна одна і та ж різницева колірна модель, то в стиснутих даних перед номером ефективної колірної моделі для чергової MCU зберігаємо біт-прапорець, рівний 1, що сигналізує про те, що номери колірних моделей для всіх підпорядкованих MCU такі ж і зберігати чи зчитувати їх надалі не потрібно. Інакше перед номером ефективної колірної моделі для чергової MCU зберігаємо біт-прапорець з нулем, який вказує на те, що підпорядковані MCU можуть мати інші ефективні колірні моделі і їх потрібно обробляти далі. Для MCU останнього шару біти-прапорці не зберігаються, оскільки його MCU не мають підпорядкованих MCU.

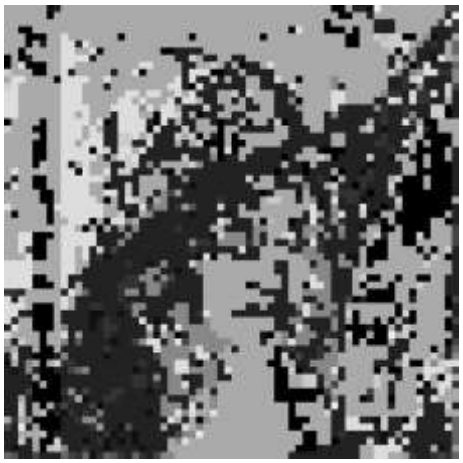

Рис. 4. Номери ефективних різницевих колірних моделей для окремих MCU зображення Lena.bmp у відтінках сірого (0 – чорний, 15 – білий колір).

Організація вибору різницевої колірної моделі для кожного MCU. Спочатку здається, що для вибору ефективної колірної моделі з переліку (4) для чергового MCU достатньо почергово застосувати кожну колірну модель з цього переліку до пікселів MCU, які прогнозовано опрацьовуватимуться контекстно-незалежним алгоритмом, і обрати з них ту альтернативну колірну модель, яка забезпечить мінімальну довжину ентропійного коду (2) після застосування предикторів (3). Але під час ієрархічного обходу проходів шарів пікселів зображень використовують пікселі з різних MCU. Тому довжина ентропійного коду (2) перетворених яскравостей пікселів зображення залежить, насамперед, не від частоти окремого елемента в черговому MCU, а загалом від розподілу частот елементів на черговому шарі чи проході. Отже, **визначати номер ефективної різницевої колірної моделі з переліку основних альтернативних (4) для чергового MCU будемо з врахуванням частот елементів після застосування обраних колірних моделей для попередніх MCU**. Поряд з цим, накопичуватимемо частоти компонентів та їх різниць в альтернативних колірних моделях для всього зображення, щоб після аналізу всіх MCU встановити доцільність використання колірних моделей для таких фрагментів щодо результатів застосування єдиної ефективної колірної моделі до цілого зображення.

Після визначення колірних моделей для окремих MCU порівняємо ефективність застосування різних колірних моделей для кожної MCU зі суміжних четвірок передостаннього шару з ефективністю єдиної колірної моделі для кожної такої четвірки. Адже встановлення єдиної колірної моделі для окремої четвірки суміжних MCU передостаннього шару хоча й погіршує компресію, але зменшує на 12 бітів розмір даних для ієрархічного зберігання номерів колірних моделей. Як показали експерименти, такий аналіз доцільності встановлення єдиних колірних моделей для четвірок суміжних MCU додатково зменшує розмір стиснутих зображень на сотні байтів.

Аналіз результатів застосування різницевих колірних моделей до окремих MCU зображень. Завершуючи, проаналізуємо результати застосування ефективних різницевих колірних моделей з цілими коефіцієнтами та з центруванням інтервалів різниць компонентів з переліку (4) до окремих MCU та для цілих зображень тестового набору ACT [13] у форматі HBF-LS [12] (табл. 1–3). Цей набір містить як синтезовані (№№ 1, 2, 7), так і фотореалістичні (решта) зображення. Вибір саме цього тестового набору зумовлений різноплановістю його зображень та наявністю у відкритих джерелах результатів тестувань на ньому алгоритмів інших дослідників.

Таблиця 1. Коефіцієнти стиснення зображень набору ACT після застосування ефективних різницевих колірних моделей з переліку (4), bpb

Колірні моделі	№ файла								Середній КС
	1	2	3	4	5	6	7	8	
RGB	1,34	0,58	4,65	3,81	4,14	5,16	0,61	4,32	3,07
Єдина з 16-ти альтернативних (4) для всього зображення	1,34	0,568	4,45	3,27	3,67	4,05	0,59	3,60	2,69
Різницеві з 16-ти альтернативних для окремих MCU	1,30	0,574	4,42	3,21	3,62	4,11	0,60	3,61	2,68
Різницеві з 16-ти альтернативних для окремих MCU та для всього зображення	1,30	0,568	4,42	3,21	3,62	4,05	0,59	3,60	2,67

Таблиця 2. Час кодування файлів зображень набору ACT з використанням ефективних різницевих колірних моделей з переліку (4), с

Колірні моделі	№ файла								Середній час
	1	2	3	4	5	6	7	8	
RGB	2,19	3,05	1,06	1,93	1,16	2,04	1,25	1,87	1,82
Єдина з 16-ти альтернативних (4) для всього зображення	2,45	3,52	1,41	2,33	1,56	2,41	1,61	2,14	2,18
Різницеві з 16-ти альтернативних для окремих MCU	8,66	14,45	2,97	5,16	3,59	5,56	5,36	5,66	6,43
Різницеві з 16-ти альтернативних для окремих MCU та для всього зображення	8,72	14,19	2,95	5,14	3,52	5,34	5,25	5,14	6,28

Таблиця 3. Час декодування файлів зображень набору ACT, закодованих з використанням ефективних різницевих колірних моделей з переліку (4), с

Колірні моделі	№ файла								Середній час
	1	2	3	4	5	6	7	8	
RGB	0,58	1,21	0,30	0,54	0,32	0,49	0,49	0,42	0,54
Єдина з 16-ти альтернативних (4) для всього зображення	0,58	1,23	0,32	0,60	0,31	0,61	0,44	0,60	0,59
Різницеві з 16-ти альтернативних для окремих MCU	0,75	1,56	0,36	0,64	0,36	0,53	0,50	0,55	0,66
Різницеві з 16-ти альтернативних для окремих MCU та для всього зображення	0,75	1,23	0,36	0,64	0,36	0,53	0,44	0,53	0,61

Зіставляючи результати двох перших рядків табл. 1–3, бачимо, що застосування ефективних різницевих колірних моделей з переліку альтернативних (4) до цілих зображень в середньому зменшило КС на 0,38 bpb, сповільнило кодування на 19,88% та декодування – на 9,26%, здебільшого завдяки фотореалістичним зображенням. Водночас, порівнюючи КС другого та третього рядків табл. 1, робимо висновок, що застосування ефективних різницевих колірних моделей до окремих MCU, а не до цілих зображень додатково зменшило КС в середньому лише на 0,01 bpb. При цьому для 4-х файлів спостерігаємо зменшення, а для інших 4-х – збільшення КС. Найістотніше (на 0,04 bpb) розмір стиснутого файла зменшився для дискретно-тонового зображення з шумами № 1, оскільки для його окремих фрагментів ефективні різні колірні моделі з переліку основних альтернативних (4). Ці колірні моделі для пікселів зображення, які не входять у довгі заміни контекстно-залежного алгоритму LZ77 [14], сумарно забезпечують меншу ентропію, ніж колірна модель RGB чи різницева колірна модель для всього зображення. А ось час кодування (другий та третій рядки з табл. 2) внаслідок застосування різницевих колірних моделей до окремих MCU, а не до всього зображення зріс в середньому аж у 2,95 раза. Причому сповільнення кодування спостерігаємо як для дискретно-тонових малюнків, так і для фотореалістичних знімків. Це пов’язано з необхідністю додаткових розрахунків по кожному MCU ентропійних довжин кодів (2) накопичених частот для 16-ти основних альтернативних різницевих колірних моделей (4). Зате декодування сповільнилося в середньому лише на 12% (другий та третій рядки табл. 3) і пов’язане воно з необхідністю визначення індекса MCU (побітових зсувів вправо на 3 біти) для кожного пікселя під час повернення від обраної різницевої колірної моделі до моделі RGB.

Щоб встановити причини зменшення КС лише для 50% файлів набору ACT внаслідок застосування колірних моделей до фрагментів замість однієї різницевої колірної моделі для всього зображення, зіставимо ентропію пікселів, які безпосередньо опрацьовуються ієрархічними предикторами та контекстно-незалежним алгоритмом після застосування цих двох підходів (табл. 4). Дані цієї таблиці відрізняються від КС у табл. 1, оскільки стосуються лише пікселів, попередньо не опрацьованих контекстно-залежним алгоритмом LZ77.

У табл. 4 наведені результати як загальної ентропії (1), так і ентропії, розрахованої з врахуванням стиснення за проходами окремих шарів

$$H' = \frac{\sum\limits_{pass} H'_{pass} N'_{pass}}{\sum\limits_{pass} N'_{pass}} = \frac{\sum\limits_{pass} L'_{pass}}{N'}, \qquad (6)$$

де " ′ " вказує на використання під час проходів лише пікселів, неопрацьованих контекстно-залежним алгоритмом, а сумування відбувається на всіх проходах. Назвемо цю ентропію "за проходами". Фактично вона дорівнює відношенню суми прогнозованих довжин ентропійних кодів пікселів окремих проходів до загальної кількості опрацьованих елементів.

Таблиця 4. Ентропія пікселів зображень набору ACT, неопрацьованих контекстно-залежним алгоритмом, після застосування різницевих колірних моделей, сформованих різними способами, та ієрархічних предикторів, bpb

Колірні моделі	Тип ентропії	№ файла								Середній КС
		1	2	3	4	5	6	7	8	
RGB	Загальна	6,10	5,68	4,70	4,16	4,22	5,50	4,73	4,54	4,95
	За проходами	6,05	5,39	4,67	4,10	4,16	5,40	4,55	4,44	4,85
Єдина різницева для всього зображення	Загальна	6,10	5,02	4,50	3,50	3,75	4,18	4,34	3,81	4,40
	За проходами	6,05	4,98	4,47	3,39	3,67	**4,05**	4,26	3,65	4,32
Різницеві для окремих MCU	Загальна	5,82	4,59	4,45	3,40	3,68	4,16	3,89	3,76	4,22
	За проходами	5,77	4,55	4,42	3,32	3,61	**4,08**	3,81	3,64	4,15

Під час стиснення для зменшення КС зображень зберігаємо результати різних проходів у різних стиснутих Deflate-блоках, оскільки вони мають різну ентропію [8]. Тому саме ентропію "за проходами" варто застосовувати для оцінки ефективності стиснення різних алгоритмів препроцесингу. Ентропія "за проходами" завжди менше загальної ентропії, адже вона враховує різну нерівномірність розподілу ймовірностей елементів на різних проходах. Саме про це свідчать результати порівняння парних і непарних рядків табл. 4. Бачимо, що внаслідок застосування різницевих колірних моделей до цілих зображень ентропія "за проходами" пікселів, які не входять у довгі заміни алгоритму LZ77, зменшилася у середньому на 0,53 bpb, а після застосування цих моделей до окремих фрагментів – ще додатково на 0,17 bpb. Але ентропія "за проходами" після застосування різницевих колірних моделей до окремих MCU може виявитися навіть більшою від цієї ентропії після використання єдиної різницевої колірної моделі для цілого зображення (№ 6 у табл. 4). Як вже зазначали, це відбувається, бо вибрана різницева колірна модель для окремого MCU ефективно зменшує довжину ентропійного коду саме цього MCU, але під час ієрархічного обходу в дані чергового проходу входять результати застосування предикторів до пікселів з різних MCU. **Однакові перепади яскравостей у відмінних колірних моделях різних MCU можуть спричинити різні результати застосування предикторів (3) і тому призвести**

до розсіювання значень елементів відхилень Δ_{ji}, яке відсутнє для єдиної колірної моделі, та, як наслідок, до збільшення ентропії (1). Крім цього, застосовуючи різницеві колірні моделі для окремих MCU, потрібно ще й зберігати номери обраних моделей з переліку (4) для кожного MCU і сигнальні біти ієрархічного обходу (в середньому – 5 бітів для кожного MCU). **Тому для чергового окремого зображення ефективнішою може виявитися як єдина різницева колірна модель, так і сукупність різних колірних моделей для окремих MCU.**

В реалізації вибору різницевих колірних моделей для окремих MCU, наведеній у попередньому пункті, можна порівняти між собою лише загальну ентропію після застосування єдиної різницевої колірної моделі до всього зображення зі загальною ентропією після застосування колірних моделей до окремих MCU. Для об'єктивнішого порівняння цих величин домножимо їх на середні коефіцієнти зменшення загальної ентропії (1) до ентропії "за проходами" (6). Визначимо ці коефіцієнти за фотореалістичними зображеннями, для яких, здебільшого, ефективні різницеві колірні моделі. З даних табл. 4 бачимо, що відхилення загальної ентропії від ентропії "за проходами" відрізняється для різних способів формування колірних моделей. Після використання єдиної колірної моделі до цілих фотореалістичних зображень середня ентропія "за проходами" менша від загальної ентропії на 2,5% (третій та четвертий рядки), а після застосування ефективних моделей до окремих MCU це зменшення становить 2% (п'ятий та шостий рядки). Тому у першому випадку прогнозовану довжину ентропійного коду (2) домножуватимемо на 0,98, а у другому – на 0,975. Відмовлятися від використання різницевих колірних моделей для MCU на користь єдиної колірної моделі для всього зображення будемо тоді, коли прогнозоване зменшення довжини ентропійного коду менше від прогнозованої кількості бітів для зберігання номерів вибраних різницевих колірних моделей для окремих MCU:

$$\begin{aligned}&(1-\min(0{,}5;\ fractReplaceSumignPixel))\times\\&\times(0{,}975\times minLenImageCM\ -0{,}98\times minLenMCU)<countMCU\times 5\,.\end{aligned}\qquad(7)$$

Результати переходу до єдиних різницевих колірних моделей для цілих зображень, якщо вони прогнозовано забезпечують менший КС від різницевих моделей для MCU, наведені в останніх рядках табл. 1–3. Бачимо, що для 50% зображень набору ACT (двох фотореалістичних знімків та двох дискретно-тонових малюнків) вигідніше застосовувати єдину колірну модель. Реалізація можливості такого додаткового переходу до колірної моделі цілого зображення забезпечує додаткове зменшення КС в середньому для набору ACT на 0,01 bpb, прискорює кодування на 2,33% та декодування на 7,58%, і тому має використовуватися на практиці для визначення колірних моделей з використанням MCU.

Застосування різницевих колірних моделей для окремих MCU з можливістю переходу до єдиної різницевої колірної моделі для всього зображення зменшує КС в середньому лише на 0,02 bpb, але при цьому сповільнює кодування в 2,88 раза та декодування на 3,39%, **тому й не рекомендуємо використовувати в графічних форматах**. Таке використання різницевих колірних моделей варто реалізовувати в архіваторах, щоб забезпечити максимальне стиснення зображень без втрат.

ВИСНОВКИ

Зменшити КС зображень у трикомпонентних колірних моделях можливо не лише завдяки декореляції даних окремих компонентів, а й за допомогою міжкомпонентної декореляції шляхом переходу до різницевих колірних моделей. Її доцільно виконувати так, щоб підсилити властивості зображення, які використовуються алгоритмами препроцесингу та безпосереднього стиснення обраного гра-

фічного формату, мінімізуючи, наприклад, прогнозовану довжину ентропійного коду (2).

У графічних форматах доцільно використовувати різницеві колірні моделі з цілими коефіцієнтами, оскільки вони забезпечують швидке декодування. Застосування під час прогресуючого ієрархічного стиснення без втрат єдиної різницевої колірної моделі з цілими коефіцієнтами до кожного зображення з набору ACT зменшує КС в середньому на 0,38 bpb, а для фотореалістичних зображень – на 0,6 bpb. Таким чином, різницеві колірні моделі з цілими коефіцієнтами дають змогу суттєво підвищити ефективність стиснення без втрат трикомпонентних фотореалістичних зображень у форматах, які використовують предиктори, і тому можуть бути впроваджені в подальші версії цих форматів на рівні стандартів.

Застосування різницевих колірних моделей з цілими коефіцієнтами до окремих MCU з можливістю переходу до єдиної різницевої колірної моделі для всього зображення додатково зменшує КС в середньому лише на 0,02 bpb, але при цьому сповільнює кодування в 2,88 раза та декодування на 3,39%, тому й не рекомендуємо їх до використання в графічних форматах. Різницеві колірні моделі варто застосовувати до окремих MCU в архіваторах для забезпечення максимального стиснення зображень без втрат.

Для оцінки ефективності алгоритмів препроцесингу під час ієрархічного стиснення доцільно використовувати ентропію “за проходами” (6) замість загальної ентропії (1) та розраховувати її лише за пікселями, які опрацьовуються контекстно-незалежним алгоритмом. Ентропія “за проходами” завжди менша загальної ентропії, адже вона враховує різні нерівномірності розподілу ймовірностей елементів на різних проходах, через що ці дані доцільно зберігати в різних стиснутих блоках.

У подальшому плануємо зменшити КС зображень у графічному форматі HBF-LS модифікацією контекстно-залежного алгоритму LZ77 [14] для пошуку однакових яскравостей пікселів як в горизонтальному, так і в вертикальному напрямках та дослідити ефективність цього графічного формату для інших тестових наборів зображень [15].

1. Wallace, G. K. The JPEG still picture compression standard. *Communication of ACM.* **1991**, 34 (4), 30–44. https://doi.org/10.1145/103085.103089
2. Boutell, T. et. all. *PNG Specification. Version 1.0. RFC 2083*; RFC, 1997. https://doi.org/10.17487/RFC2083
3. Miano, J. *Compressed Image File Format: JPEG, PNG, GIF, XBM, BMP*; Addison Wesley Professional, 1999.
4. Shportko, A. V. The use of differences of colors models for compression RGB-images without losses. *Information extraction and processing*. **2009**, 31 (107), 90–97 (in Ukrainian).
5. Selomon, D. *A Guide to Data Compression Methods*; Springer, 2002. https://doi.org/10.1007/978-0-387-21708-6
6. Gonzalez, R.; Woods, R. *Digital Image Processing, 4th et*; Pearson, 2017.
7. Shportko, A. V.; Bomba, A. Ya.; Postolatii, V. A. Rejection of the Inefficient Replacements while Forming the Schedule of the Modified Algorithm LZ77 in the Process of Progressive Hierarchical Compression of Images without Losses. In *Computational Linguistics and Intelligent Systems, Proceedings of the 6th International Conference COLINS* 2022. Glivice, Poland, May 12–13, 2022; ceur-ws.org; Vol. 3171, pp. 1594–1605. http://ceur-ws.org/Vol-3171/paper113.pdf (accessed 2024-01-05)
8. Shportko, A. V.; Bomba, A. Ya.; Shportko, L. V. Features of application arithmetic encoding in the process of progressing hierarchical compression of images without losses. *Proceedings of the National University "Lviv Polytechnic". Series: Information Systems and Networks*. **2014**, 783, 12–22.
9. Shportko, A. V.; Bomba, A. Ya.; Postolatii, V. A. Programming the Formation of Difference Color Models for Lossless Image Compression. In *Computational Linguistics and Intelligent Systems, Proceedings of the 7th International Conference COLINS 2023*. Kharkiv, Ukraine, Apr 20–21, 2023; Vol. 3, pp. 53–68. http://ceur-ws.org/Vol-3403/paper5.pdf (accessed 2024-01-05)

10. Shportko, A. V.; Postolatii; V. A. Development of Predictors to Increase the Efficiency of Progressive Hierarchic Context-Independent Compression of Images Without Losses. In *Computational Linguistics and Intelligent Systems, Proceedings of the 5th International Conference COLINS 2021*. Kharkiv, Ukraine, Apr 22-23, 2021; Vol. 1, pp. 1026–1038. http://ceur-ws.org/Vol-2870/paper77.pdf (accessed 2024-01-05)
11. Shportko, A. V.; Bomba, A. Ya. Formation of color models with centering of component difference intervals in the process of progressive hierarchical lossless image compression In *Modeling, control and information technologies, Proceedings of VI International scientific and practical conference.* Rivne, Ukraine, Now 9–11, 2023; National university of water and environmental engineering; pp. 194–197. https://doi.org/10.31713/MCIT.2023.060
12. Shportko, A. V. *Author's certificate #58216 of Ukraine. HBF-LS Graphics Format Specification. Version 1.0*, 2015.
13. *ACT – Test Files.* http://www.compression.ca/act/act-files.html (accessed 2024-01-05).
14. Ziv, J.; Lempel, A. A universal algorithm for sequential data compression. *IEEE Transactions on Information Theory.* **1977**, 23 (3), 337–343. https://doi.org/10.1109/TIT.1977.1055714
15. Vemuri, B. C.; Sahni, S.; Chen, F.; Kapoor, C.; Leonard, C.; Fitzsimmons, J. *Lossless Image Compression*; 2022. URL: https://www.researchgate.net/publication/2555508_Lossless_Image_Compression (accessed 2024-01-05)

Одержано 09.01.2024