УДК 004.043

СТИСНЕННЯ RGB-ЗОБРАЖЕНЬ БЕЗ ВТРАТ ІЗ ВИКОРИСТАННЯМ ПАЛІТРИ

О.В. ШПОРТЬКО

Розглянуто та обгрунтовано можливості стиснення RGB-зображень без втрат за допомогою палітрування. Описано варіант алгоритму для реалізації такого стиснення з розбиттям результату на трендову та шумову складові. Наведено результати застосування програми, розробленої згідно із запропонованим алгоритмом для стиснення зображень набору ACT.

У файлах, що передаються каналами зв'язку, найчастіше містяться дані одного з чотирьох основних типів: тексти, зображення, відео або звук. І якщо для зберігання однієї сторінки неформатованого тексту достатньо 4 Кб, то фотореалістичне растрове зображення 10×15 см без стиснення може займати кілька Мб, записи пісень — десятки Мб, а записи фільмів — декілька Гб. Добре, що більшість із цих файлів мають високий рівень надлишковості. Саме зменшення надлишковості дозволяє стискати файли, і тим самим підвищувати швидкість обміну інформацією по мережі та зменшувати об'єми використання дискового простору. Тому зараз, в епоху інформаційної цивілізації, коли значна частина інформації зберігається в електронному вигляді, проблема стиснення зображень залишається актуальною.

Більшість відомих алгоритмів, які дозволяють стиснути зображення в десятки разів (JPEG, фрактальних та вейвлет-перетворень), призводять до незначних втрат якості зображень, що непомітно для фотореалістичних зображень з високою роздільною здатністю, але впливає на якість зображень із невисокою роздільною здатністю з фрагментами ділової графіки чи дискретно-тоновими переходами. Крім того, існують класи зображень, для яких спотворення неприпустимі (наприклад рентгенівські знімки). Алгоритми стиснення зображень без втрат хоча й стискають зображення значно слабше, але не призводять до погіршення їхньої якості. Зараз найдинамічніше розвиваються алгоритми стиснення зображень із втратами, хоча й алгоритми стиснення зображень без втрат теж із кожним роком підвищують свої показники.

ОСНОВНІ ПІДХОДИ ДО СТИСНЕННЯ ЗОБРАЖЕНЬ БЕЗ ВТРАТ. АНАЛІЗ ОСТАННІХ ДОСЯГНЕНЬ І ПУБЛІКАЦІЙ. ПОСТАНОВКА ЗАДАЧІ

Більшість сучасних графічних форматів для збереження кольору кожного пікселя використовують три компоненти [1]. Наприклад, у кольоровій моделі RGB — це яскравість червоної (R), зеленої (G) та синьої (B) компоненти. Під кожну компоненту відводиться, як правило, 1 байт. Тому розмір файла зображення у нестисненому вигляді приблизно дорівнює потроєному добутку кількостей пікселів по вертикалі (*height*) та горизонталі (*width*).

Інший спосіб зберігання кольору пікселів зображення передбачає використання палітри. *Палітра* — це одномірний масив трибайтових елементів,

кожен з яких визначає колір [2]. При такому способі збереження зображення колір кожного пікселя задається індексом в палітрі (наприклад у форматі GIF). Зазвичай, використовуються палітри з 2, 16 чи 256 кольорів [3]; (відповідно 1, 4 чи 8 бітам на індекс кольору пікселя). Палітри з більшою кількістю кольорів хоча й можливі, але на практиці не використовуються, оскільки вимагають більше байтів для зберігання самої палітри і, головне, значно більше бітів для зберігання індексу кольору кожного пікселя.

Розмір файла палітрового зображення у нестисненому вигляді приблизно дорівнює сумі розміру палітри та добутку кількостей пікселів по горизонталі та вертикалі, помноженій на розмір індексу кольору пікселя. Наприклад, під час використання палітри з 256 кольорів розмір файла складає приблизно $256*3+height*width$ байт, тобто майже втричі менше, ніж при зберіганні згідно з кольоровою моделлю RGB. На жаль, кількість різних кольорів палітрового зображення не може перевищувати кількості кольорів у палітрі. Тому сучасні фотореалістичні зображення, що містять сотні тисяч різних кольорів, найчастіше використовують трикомпонентні кольорові моделі (RGB, YCrCb, HLS та ін.). Проблема використання палітри для стиснення таких зображень без втрат до цього часу залишалася невирішеною. Більшість сучасних графічних форматів стиснення без втрат (BMP, PNG) дозволяють зберігати зображення як із використанням палітри, так і без неї.

Сучасні архіватори та формати графічних файлів для стиснення зображень без втрат найчастіше використовують як один із основних етапів алгоритмів словникового методу [1, 3]. Такі алгоритми намагаються замінити наступні незакодовані символи посиланням на аналогічні символи у вже закодованій частині. Фотореалістичні зображення стискаються словниковими алгоритмами неефективно, оскільки, як правило, вони містять багато подібних, але неоднакових кольорів. Словникові алгоритми належать до класу контекстно-залежних, тому що використовують опрацьовану раніше послідовність. Алгоритми цього методу покликані усувати залежності між різними послідовностями елементів.

Для покращення стиснення даних результати дії словникових алгоритмів доцільно стиснути одним із контекстно-незалежних алгоритмів (арифметичним чи Хафмана). Ідея використання цих алгоритмів полягає у заміні елементів з більшою частотою послідовностями меншої кількості бітів, ніж для елементів з меншою частотою. При цьому середня довжина коду елемента після застосування контекстно-незалежного методу згідно з теоремою Шеннона має наближатися до величини:

$$H = -\sum_i p(s_i) \log p(s_i), \qquad (1)$$

де $p(s_i)$ — ймовірність появи елемента s_i, а всі логарифми тут і надалі беруться за основою 2. Цю величину також називають *ентропією джерела* [1]. Вона зменшується під час збільшення нерівномірності розподілу ймовірностей між елементами.

Для аналізу ефективності запропонованого алгоритму ми використали програму WinRar версії 4.00 із застосуванням лише основного алгоритму, що послідовно виконує LZ-кодування та стиснення Хафмана (LZ+Huff).

Основну увагу при розробці алгоритмів стиснення без втрат приділяють показникам компресії: коефіцієнту стиснення (КС), тобто відсотку зменшення початкового розміру файла або усередненому значенню ентро-

пії, тобто кількості бітів, що всередньому витрачається на кодування одного пікселя (bpp). Оскільки між цими показниками існує взаємно-однозначна відповідність, то ми скористалися лише першим, тому що він використовується для оцінки стиснення не лише зображень.

Врахуємо, що згідно з [4]: «Відхилення між значеннями сусідніх елементів у зображеннях найчастіше зумовлені двома причинами: «сильними» коливаннями, що обумовлюються зображеними об'єктами — трендом, і «слабкими» фоновими коливаннями — шумом. Тому можливі два протилежні типи моделей:

- внесок шуму незначний у порівнянні з внеском тренду;
- внесок тренду незначний у порівнянні з внеском шуму.

Око людини насамперед орієнтується на контури об'єктів (тренд) і менш чутливе до фонових коливань (шуму). Тому під час передачі даних по мережі для забезпечення прогресуючого стиснення доцільно спочатку передавати дані тренду і лише після цього — шуму.

МЕТА ДОСЛІДЖЕННЯ

Обґрунтування можливості та опис відповідного алгоритму для стиснення довільних RGB-зображень без втрат із використанням палітри та словникових методів. Основною вимогою до алгоритму є досягнення максимальних КС у практично незмінних витратах часу.

АЛГОРИТМ СТИСНЕННЯ RGB-ЗОБРАЖЕНЬ БЕЗ ВТРАТ ІЗ ВИКОРИСТАННЯМ ПАЛІТРИ

З'ясуємо спочатку глибинні причини майже потрійного стиснення дискретно-тонових зображень у разі використання палітри з 256 кольорами. У кольоровій моделі RGB значення кожної компоненти лежить у межах від 0 до 255. Отже, загалом ця модель адресує $255 \times 256 \times 256 = 16777216$ кольорів. При використанні ж палітри індексується лише 256 кольорів, тобто для зберігання індексу кольору використовується 8 бітів замість 24.

Надлишкова індексація характерна для кольорової моделі RGB і у випадку зберігання фотореалістичних зображень, адже в кожному з них використовується лише невеличка частина спектру кольорів (наприклад, в зображеннях набору Archiv Comparison Test (ACT) — менше 1% (номер 6 у табл. 1)).

Таблиця 1. Характеристики зображень набору ACT

Номер файла	Назва файла	Розмір, Кб	Розміри пікселів	Кількість різних кольорів	Використання спектру, %	Різні кольори серед пікселів, %
1	Clegg.bmp	2101	814 × 880	127696	0,76	17,83
2	Frymire.bmp	3622	1118 × 1105	3622	0,02	0,29
3	Lena.bmp	769	512 × 512	148279	0,88	56,56
4	Monarch.bmp	1153	768 × 512	78617	0,47	19,99
5	Peppers.bmp	769	512 × 512	111344	0,66	42,47
6	Sail.bmp	1153	768 × 512	75748	0,45	19,26
7	Serrano.bmp	1464	629 × 794	1313	0,01	0,26
8	Tulips.bmp	1153	768 × 512	118233	0,70	30,07

Максимальна кількість різних кольорів у зображенні у найгіршому випадку не перевищує кількості пікселів. Наприклад, у фотореалістичному зображенні *Lena.bmp* (рис. 1) використовується 148279 кольорів 262144 пікселями, тобто 0,88% кольорів від можливих. Розглядаючи навіть проекцію кольорів пікселів цього зображення на площину RG (рис. 4), можна помітити, що використовується менше 30% індексованого простору площини.

Рис. 1. Зображення Lena.bmp

Рис. 2. Зображення Lena.bmp в палітрі з 63 кольорів

Рис. 3. Зображення Lena.bmp в палітрі з 256 кольорів

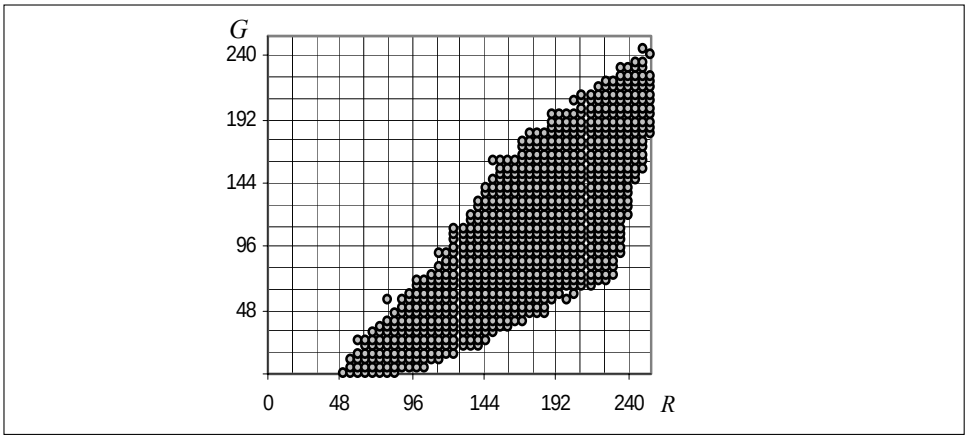

Рис. 4. RG-проекція кольорів пікселів зображення Lena.bmp

Для зменшення надлишкової індексації кольоровою моделлю *сформуємо палітру тренду*, виконуючи такі кроки:

1. Розіб'ємо множину допустимих значень кольорів на сегменти максимального фіксованого розміру [5], що не перекриваються між собою (наприклад, для RGB — це множина прямокутних паралелепіпедів). Оскільки наперед невідомо, в яких сегментах містяться кольори пікселів обраного зображення, то множина сегментів має повністю покривати множину допустимих значень кольорової моделі.

2. Визначимо кількість кольорів пікселів та межі їх знаходження у кожному сегменті. Звузимо межі кожного сегменту до меж знаходження кольорів пікселів у ньому.

3. Поєднаємо між собою сусідні сегменти, якщо їх сукупний розмір не перевищує максимального фіксованого.

4. Збережемо в палітрі координати лише тих сегментів, які містять кольори пікселів. Тоді колір кожного пікселя можна розбити на дві складові — індекс сегменту в палітрі (тренд) та зміщення кольору всередині сегменту (шум). Оскільки у найгіршому випадку кожен вихідний сегмент може містити кольори пікселів, то початкова кількість сегментів не може перевищувати максимально допустимої кількості кольорів палітри.

5. Доповнимо палітру до максимально можливої кількості кольорів, використовуючи поділ окремих сегментів так, щоб максимально збільшити КС.

Як зазначалося вище, трендову та шумову складові доцільно зберігати окремо для забезпечення прогресивної передачі даних по мережі. Варіант програми, що реалізує пропонований алгоритм, взагалі розбиває зображення на два файли, які містять відповідно дані трендової та шумової моделі. При цьому множина всіх допустимих значень кольорів на початку програми розбивається на 256 сегментів-паралелепіпедів розміром $32 \times 32 \times 64$ значення. Саме такі сегменти будемо розглядати надалі. Кількість сегментів-паралелепіпедів, що реально використовуються окремими зображеннями, після четвертого кроку алгоритму, як правило, не перевищує 100. Наприклад, зображення *Lena.bmp* використовує 63 паралелепіпеда з 256 (рис. 2) кольорів.

Розглянемо результати застосування запропонованого алгоритму для стиснення різнотипних зображень згаданого вище набору АСТ. Файл тренду, отриманий у результаті виконання запропонованого алгоритму, під час використання палітри з 256 кольорів без стиснення займає третю частину вхідного файла. Його доцільно стиснути алгоритмом LZ+Huff. Причому розмір стиснутого файла тренду завжди буде меншим від розміру стиснутого аналогічного вхідного зображення (номери 2, 5, 8 у табл. 2). Це пов'язано як із застосуванням палітри, так і зі зведенням кольорів сегментів до єдиного кольору палітри, тобто ліквідації шумів.

Таблиця 2. Результати стиснення файлів набору АСТ (Кб) при застосуванні та без застосування палітри

Номер файла	Без поділу паралелепіпедів			З поділом паралелепіпедів			LZ+Huff файла зображення
	LZ+Huff файла тренду	Файл шуму	Всього	LZ+Huff файла тренду	Файл шуму	Всього	
1	103	1398	1501	116	1186	1302	500
2	138	2288	2426	140	1073	1213	235
3	78	510	588	186	370	556	701
4	87	766	853	216	542	758	789
5	58	508	566	140	387	527	641
6	134	767	901	289	548	837	867
7	46	863	909	60	285	345	102
8	113	767	880	232	603	835	938
Разом	**757**	**7867**	**8624**	**1379**	**4994**	**6373**	**4773**

Зовсім інші підходи застосовуються для стиснення файлів шумів. Використання контекстно-залежних алгоритмів типу LZ тут неефективне, оскільки повторення послідовностей практично відсутні. Файли шумів можна відразу закодувати ентропійним алгоритмом, але такий підхід несуттєво підвищує КС.

Нами пропонується спосіб зберігання шумів шляхом безпосереднього запису зміщень кольорів усередині сегменту з урахуванням його розміру. Якщо, наприклад, максимальний сегмент-паралелепіпед має розміри $32 \times 32 \times 64$ значення, то для зберігання зміщення в ньому необхідно $\lceil \log(32) \rceil + \lceil \log(32) \rceil + \lceil \log(64) \rceil = 16$ біт. Тобто, навіть у випадку максимальних розмірів усіх паралелепіпедів (для хаотичних зображень) результат палітрування перевищить розмір вхідного файла лише на опис цих сегментів. Наприклад, для опису паралелепіпедів, що не перевищують згаданого максимального, достатньо 40 бітів: 24 — для опису базового кольору паралелепіпеда в палітрі і 16 — його розмірів. Для запису зміщень в паралелепіпедах меншого розміру знадобиться менша кількість біт. Саме завдяки цьому і досягається стиснення файлів шумів. Інша перевага безпосереднього запису зміщень кольорів усередині паралелепіпеда полягає у максимальній швидкості обробки даних, оскільки при цьому не виконуються ніякі перетворення. Результати стиснення шумів зображень набору ACT лише завдяки врахуванню розмірів сегментів-паралелепіпедів (без п'ятого кроку алгоритму) наведено в табл. 2, у номері 3.

Підвищити КС файлів шумів можна завдяки поділу окремих паралелепіпедів. Навіть безпосереднє розбиття довільного паралелепіпеда навпіл зменшує кількість значень по осі поділу вдвічі, а кількість бітів — на один для кожного зміщення паралелепіпеда. Тому розбиття таким чином паралелепіпедів із максимальною кількістю точок покращує показники стиснення. Досягнути ж кращих КС можна завдяки оптимальному розбиттю паралелепіпедів. Розглянемо, наприклад, RG-проекцію зміщень кольорів пікселів окремого умовного паралелепіпеда (рис. 5), розбиття якого посередині ребра по осі R (штрих-пунктирна лінія) створить два паралелепіпеда: перший — у діапазоні [0; 15] і другий — [16; 31] по цій осі. Тобто сумарного зменшення розміру від такого поділу не відбудеться. Якщо ж розбити цей паралелепіпед по значенню $R = 7$ (пунктирна лінія), то можна буде створити лівий паралелепіпед у діапазоні [0; 5] та правий — [10; 31], що дозволить сумарно зменшити розмір по цій осі на 4 значення.

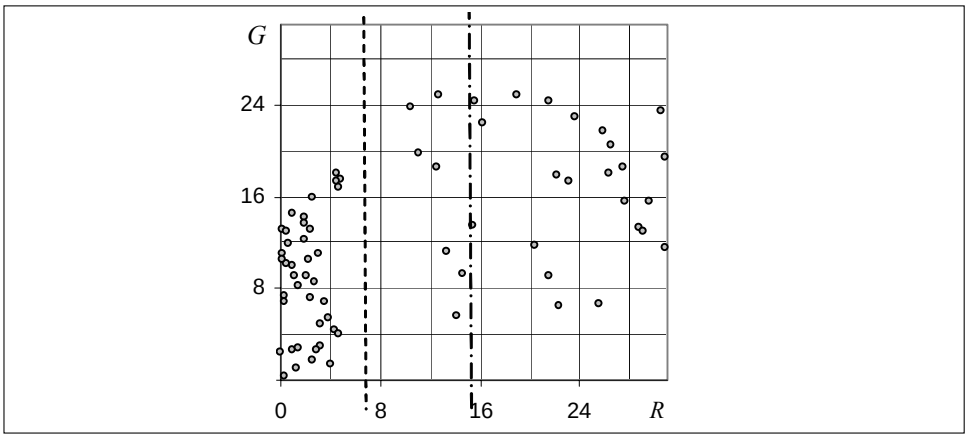

Рис. 5. RG-проекція зміщень кольорів пікселів окремого умовного паралелепіпеда

Отже, досягнути додаткового зменшення розмірів сегментів-паралелепіпедів у результаті поділу можна завдяки врахуванню положення

зміщень у ньому. Зменшення розмірів паралелепіпедів призводить до зменшення кількості бітів для збереження зміщень, тобто до покращення показників стиснення.

Виведемо формулу для підрахунку зменшення кількості бітів у записі зміщень кольорів у результаті поділу паралелепіпеда по значенню i осі R на два паралелепіпеди — правий і лівий. Нехай діапазон значень ребра паралелепіпеда до поділу по осі R знаходиться в межах $[\min R; \max R]$, по осі G — $[\min G; \max G]$, а по осі B — $[\min B; \max B]$. Позначимо countPoint загальну кількість зміщень у паралелепіпеді, а $\text{count}R(j)$, $\text{count}G(j)$, $\text{count}B(j)$ — кількості зміщень, які мають значення j відповідно по R, G та B. Очевидно, що:

$$\text{countPoint} = \sum_{j=\min R}^{\max R} \text{count}R(j) = \sum_{j=\min G}^{\max G} \text{count}\,G(j) = \sum_{j=\min B}^{\max B} \text{count}B(j). \qquad (2)$$

Позначимо також $\underline{i}$ праву межу лівого, а $\bar{i}$ ліву межу правого паралелепіпедів при поділі по осі R.

$$\underline{i} = \max_{\substack{j=\min R,\, i \\ \text{count}R(j)>0}} j, \quad \bar{i} = \min_{\substack{j=i+1,\, \max R \\ \text{count}R(j)>0}} j.$$

Кількість бітів, необхідних для запису зміщень кольорів паралелепіпеда до поділу, дорівнює:

$$\text{countBit} = \text{countPoint}\begin{pmatrix} \lceil \log(\max R - \min R + 1) \rceil + \\ + \lceil \log(\max G - \min G + 1) \rceil + \\ + \lceil \log(\max B - \min B + 1) \rceil \end{pmatrix}. \qquad (3)$$

Якщо знехтувати зменшенням розмірів лівого та правого паралелепіпедів після поділу по інших осях, то кількість бітів, яка знадобиться для запису зміщень кольорів у цих сегментах, відповідно складе:

$$\text{countLeftBit}_i = \sum_{j=\min R}^{\underline{i}} \text{count}R(j) \begin{pmatrix} \lceil \log(\underline{i} - \min R + 1) \rceil + \\ + \lceil \log(\max G - \min G + 1) \rceil + \\ + \lceil \log(\max B - \min B + 1) \rceil \end{pmatrix} \qquad (4)$$

та

$$\text{countRightBit}_i = \sum_{j=\bar{i}}^{\max R} \text{count}R(j) \begin{pmatrix} \lceil \log(\max R - \bar{i} + 1) \rceil + \\ + \lceil \log(\max G - \min G + 1) \rceil + \\ + \lceil \log(\max B - \min B + 1) \rceil \end{pmatrix}. \qquad (5)$$

Тоді очевидно, що зменшення кількості бітів від поділу сегменту по значенню i осі R складе:

$$\text{price}_i = \text{countBit} - \left(\text{countLeftBit}_i + \text{countRightBit}_i\right). \qquad (6)$$

Умовою оптимального розбиття сегменту-паралелепіпеда по осі R є максимальне зменшення кількості бітів:

$$\text{price}_R = \max_{i=\min R,\, \max R} \text{price}_i. \qquad (7)$$

Підставимо послідовно (3) – (5) в (6) та (6) в (7), розкриємо дужки та, врахувавши (2), після зведення отримаємо:

$$\text{price}_R = \underset{i=\min R,\,\max R}{\max} \left(\begin{array}{l} \text{countPoint}\left\lceil \log\,(\max R - \min R + 1)\right\rceil - \\ \left(\begin{array}{l} - \sum\limits_{j=\min R}^{\underline{i}} \text{count}\,R(j)\left\lceil \log\,(\underline{i} - \min R + 1)\right\rceil + \\ + \sum\limits_{j=\bar{i}}^{\max R} \text{count}R(j)\left\lceil \log\,(\max R - \bar{i} + 1)\right\rceil \end{array} \right) \end{array} \right) \qquad (8)$$

або

$$\text{price}_R = \text{countPoint}\left\lceil \log(\max R - \min R + 1)\right\rceil -$$

$$- \underset{i=\min R,\,\max R}{\min} \left(\begin{array}{l} - \sum\limits_{j=\min R}^{\underline{i}} \text{count}\,R(j)\left\lceil \log(\underline{i} - \min R + 1)\right\rceil + \\ + \sum\limits_{j=\bar{i}}^{\max R} \text{count}\,R(j)\left\lceil \log(\max R - \bar{i} + 1)\right\rceil \end{array} \right).$$

Формули для знаходження оптимального розбиття паралелепіпеда по осях G та B отримуються аналогічно. Умовою загального оптимального розбиття сегменту-паралелепіпеда є максимальне зменшення кількості бітів від поділу по довільній осі

$$\text{price} = \max\left(\text{price}_R, \text{price}_G, \text{price}_B\right),$$

а місцем — площина, щодо якої цей максимум досягається. Оскільки для зберігання опису паралелепіпеда витрачається 40 бітів, то поділ доцільно виконувати лише тоді, коли виграш від оптимального розбиття перевищує це значення.

Розглянемо тепер практичні аспекти реалізації знаходження оптимального розбиття паралелепіпеда по кожній з осей, які дозволяють пришвидшити виконання обчислень, на прикладі осі R згідно з (8).

1. Зменшення кількості бітів від розбиття паралелепіпеда по довільному значенню осі не залежить від його розмірів та від значень зміщень, а залежить, насамперед, від значень проекцій точок на аналізовану вісь.

2. Для всіх i, таких, що $\text{count}\,R(t) = 0$, $\text{price}_i = \text{price}_{\underline{i}}$, проводити розрахунки зменшення кількості біт згідно з (8) не потрібно.

3. Послідовно змінюючи i, перераховувати щоразу дві внутрішні суми у (8) не потрібно, адже відносно попередньої перша сума збільшується на $\text{count}\,R(i)$, а друга — зменшується на цю ж величину.

У мові С алгоритм знаходження оптимального розбиття паралелепіпеда може бути реалізовано так:

```
//розрахуємо кількість проекцій зміщень паралелепіпеда на всі значення осей
memset(countR, 0,32 * sizeof(UBYTE4));
memset(countG, 0,32 * sizeof(UBYTE4));
memset(countB, 0,64 * sizeof(UBYTE4));
```

```
 p = firstPoint; //індекс першої точки паралелепіпеда
//перебираємо зміщення паралелепіпеда
for (index = 0 ; index < countPoint; index++)
  {count R[imageData[3 * p] − min R] + +} ;
  {count G[imageData[3 * p + 1] − min G] + +} ;
  {count B[imageData[3 * p + 2] − min B] + +} ;
  p = nextPoint[p] ; //перехід до наступного зміщення паралелепіпеда
  plusBit = 0 ; //ініціалізуємо змінну максимального зменшення кількості біт
if (max R > min R) //якщо розбиття паралелепіпеда по осі R можливе
  {countLeft = 0 ; //ініціалізація першої внутрішньої суми
  {countRight = countPoint ; //ініціалізація другої внутрішньої суми
  p = countPoint * countBit(max R − min R + 1) ; //стале зменшуване осі
   for (index = 0; index < max R − min R; index + +) //перебираємо всі значення
осі R
    {if (count R[index] > 0) //index вказує на праву межу лівого паралелепі-
педа
    {countLeft+ = count R[index] ;
    {countRight − = count R[index] ;
    index1 = index + 1 ; //index1 вказує на ліву межу правого паралелепіпеда
    while (count R[index1] = 0)
    index1+ + ; //посуваємо нижню межу правого паралелепіпеда
   //підрахуємо зменшення кількості біт від чергового розбиття
    plus = p − (countLeft * countBit (index + 1) + countRight*)
        countBit (max R − min R − index1 + 1)) ;
        if (plus > 0)
        if (plusBit < plus) //знайдено оптимальніше розбиття
        (plusBit = plus) ; //оптимальніше зменшення кількості біт
        tupMega = 1 ; //індекс осі з оптимальнішим розбиттям
        mega = min R + index ; //значення осі, де досягається оптимальніше
розбиття.
```

Описаний алгоритм оптимального поділу сегментів-паралелепіпедів, хоча й дозволяє цілком зменшити розміри файлів шумів, проте негативно впливає на стиснення файлів трендів (номери 5–7, у табл. 2). Це пов'язано зі створенням нових значень у палітрі, що зменшує кількості повторень і тим самим негативно впливає на стиснення алгоритмами LZ. З іншого боку, поділ сегментів зменшує нерівномірність розподілу, що призводить до збільшення ентропії (1) і негативно впливає на стиснення алгоритмом Хафмана. Загалом, використання алгоритму оптимального поділу паралелепіпедів підвищує КС по набору ACT на 18,47% (номери 2, 3, у табл. 3). У цій же таблиці наведено КС при безпосередній компресії зображення алгоритмом LZ+Huff та кращою на сьогодні програмою для цього набору ERI 5.1

(за даними http://www.compression.ru/arctest/act/act-tif.htm). Крім того, збільшення кількості кольорів палітри до максимально можливої значно покращує якість трендового зображення (див. рис. 2, 3). Час палітрування та виконання програми згідно з описаним алгоритмом наведено в табл. 4.

Таблиця 3. КС файлів набору ACT (%) при та без застосування палітри

Номер файла	КС без поділу паралелепіпедів	КС з поділом паралелепіпедів	КС LZ+Huff файла зображення	КС ERI 5,1
1	28,56	38,03	76,20	83,48
2	33,02	66,51	93,51	94,81
3	23,54	27,70	8,84	39,92
4	26,02	34,26	31,57	61,67
5	26,40	31,47	16,64	56,31
6	21,86	27,41	24,80	53,95
7	37,91	76,43	93,03	94,67
8	23,68	27,58	18,65	55,85
Разом	**29,22**	**47,69**	**60,83**	**76,26**

Таблиця 4. Час палітрування та виконання програми (С) для файлів набору ACT (на комп'ютері з частотою процесора 300 МГц)

Номер файла	Без поділу паралелепіпедів		З поділом паралелепіпедів	
	Тривалість палітрування	Тривалість програми	Тривалість палітрування	Тривалість програми
1	2	5	2	5
2	2	8	4	10
3	1	2	2	2
4	1	2	2	3
5	1	1	1	3
6	1	2	2	4
7	1	3	2	3
8	1	3	2	3
Разом	**10**	**26**	**17**	**33**

Запропонований алгоритм дозволяє досягнути кращих КС стосовно базового алгоритму для фотореалістичних зображень із десятками тисяч різних кольорів (номери 3–6, 8, у табл. 3), хоча й значно програє на зображеннях із розсіяним спектром (номер 1), великими однаковими ділянками (номери 2, 7) і, як наслідок, загалом по набору ACT. Виконується запропонований алгоритм майже стільки ж часу, скільки триває зчитування і запис файлів на диск.

Для подальшого вдосконалення наведеного алгоритму пропонуємо:

- забезпечити можливість необмеженого розширення палітри;
- зберігати опис та використовувати поділені сегменти-паралелепіпеди у файлах шумів для запобігання погіршення стиснення файлів тренду;
- реалізувати стиснення за допомогою палітрування з використанням предикторів;

- використовувати описаний алгоритм після LZ-алгоритму, що дозволить компактніше зберігати однакові ділянки зображення.

ВИСНОВКИ

1. Запропонований алгоритм стиснення RGB-зображень без втрат за допомогою палітрування виконує розбиття результату на трендову та шумову складові. Трендова складова містить наближену палітрову копію зображення. У шумовій — зберігаються відхилення оригіналу від палітрової копії. Описаний алгоритм дозволяє стиснути дані при створенні палітри головним чином завдяки зменшення надлишкової індексації трикомпонентної кольорової моделі.

2. Трендова складова розбиття займає менше третини стиснених даних, і, тому може бути використана для ефективної прогресивної передачі даних по мережі. Ефективне стиснення трендової частини можливе традиційними алгоритмами (наприклад LZ+Huff).

3. Шумова складова розбиття ефективно стискується за допомогою запропонованого алгоритму оптимального розбиття сегментів-паралелепіпедів, що загалом підвищує КС у півтора рази. Таке розбиття дозволяє не тільки підвищити якість, але й збільшити розміри трендової складової результату.

4. Розроблений алгоритм пропонує кращі результати стиснення стосовно базового алгоритму на фотореалістичних зображеннях із десятками тисяч кольорів, хоча й поступається на інших типах зображень і відстає від найкращих на сьогодні алгоритмів стиснення. Ідея палітрування фотореалістичних зображень у комбінації з іншими алгоритмами стиснення дозволяє досягти кращих загальних показників компресії і буде втілюватися в життя й надалі.

ЛІТЕРАТУРА

1. *Сэломон Д.* Сжатие данных, изображений и звука. — М.: Техносфера, 2006. — 336 с.
2. *Миано Дж.* Форматы и алгоритмы сжатия изображений в действии: Учеб. пособ. — М.: Триумф, 2003. — 336 с.
3. *Методы* сжатия данных. Устройство архиваторов, сжатие изображений и видео / Д. Ватолин, А. Ратушняк, М. Смирнов, В. Юкин. — М.: ДИАЛОГ–МИФИ, 2003. — 384 с.
4. *Бредихин Д.Ю.* Сжатие графики без потерь качества. — 2004. — http://www.compression.ru/download/articles/i_lless/bredikhin_2004_lossless_image_compression_doc.rar.
5. *Ратушняк О.А.* Алгоритмы сжатия изображений без потерь с помощью сортировки параллельных блоков // Тез. докл. конф. молодых ученых по математике, мат. моделированию и информатике. — Новосибирск, 2001. — С. 48–49.

Надійшла 23.04.2008