В.П. Майданюк

МЕТОДИ І ЗАСОБИ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ. КОДУВАННЯ ЗОБРАЖЕНЬ

Міністерство освіти і науки України
Вінницький державний технічний університет

В. П. Майданюк

МЕТОДИ І ЗАСОБИ КОМП'ЮТЕРНИХ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ. КОДУВАННЯ ЗОБРАЖЕНЬ

НАВЧАЛЬНИЙ ПОСІБНИК

Затверджено Ученою радою Вінницького державного технічного університету як навчальний посібник для студентів бакалаврського напрямку 6.0804 – "Комп'ютерні науки" всіх спеціальностей. Протокол № 5 від 27 грудня 2000 р.

Вінниця ВДТУ 2001

УДК 681.3.06
М 14

Р е ц е н з е н т и:

В.П. Кожем'яко, доктор технічних наук, професор
О.М. Мельников, кандидат технічних наук, професор
С.Г. Кривогубченко, кандидат технічних наук, доцент

Рекомендовано до видання Ученою радою Вінницького державного технічного університету Міністерства освіти і науки України

Майданюк В.П.

М 14 **Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень.** Навчальний посібник. - Вінниця: ВДТУ, 2001. - 65 с.

В посібнику розглянуті питання кодування зображень в комп'ютерних системах. Посібник розроблений у відповідності з планом кафедри прикладної математики та обчислювальних систем і програм до дисциплін "Методи і засоби комп'ютерних інформаційних технологій", "Обробка сигналів та зображень".

УДК 681.3.06

ЗМІСТ

ВСТУП

Технологія обробки цифрових зображень досягла в останні роки великих успіхів завдяки тому, що цифрові методи обробки зображень мають ряд переваг перед аналоговими [1,2,3]. Однак, значною проблемою є великі розміри файлів, які підлягають передачі та обробці. Наприклад, швидкість передачі символів об'єднаного цифрового потоку при передачі кольорових телевізійних зображень цифровими методами в форматі 4:2:2 [4] складе:

$$Q_Я = 13{,}5\ Мгц * 8\ біт = 108\ Мбіт/с$$

$$Q_К = 6{,}75\ МГц * 8\ біт = 54\ Мбіт/с,$$

$$Q_Σ = Q_Я + 2Q_К = 108\ Мбіт/с + 108\ Мбіт/с = 216\ Мбіт/с,$$

де $Q_Я$ - швидкість передачі символів цифрового сигналу яскравості, а $Q_К$ - швидкість передачі символів різницевих кольорових сигналів. Передати такий цифровий потік по лініях зв'язку, що існують вкрай важко. Для зберігання тільки одного телевізійного кадра в цифровій формі необхідно близько 0,5 Мбайта пам'яті, а для однохвилинного репортажу 750 Мбайтів.

Вирішенням проблеми є зменшення об'єму файлів шляхом стиснення даних, або іншими словами кодування зображень.

Актуальність стиснення зображень особливо зросла за останні десять років в зв’язку з появою розвинутих комп’ютерних засобів відображення зображень, представлених в цифровій формі. Це призвело до широкого застосування зображень в комп’ютерних системах та мережах.

Виділення кодування зображень в окремий розділ пов’язано з тим, що зображення особливий вид даних, орієнтований на зорове сприйняття людиною, що дозволяє створити спеціальні алгоритми стиснення, які призначені тільки для зображень. Крім того, зображення має надлишковість в двох вимірах, тобто це двовимірний сигнал, що також дає додаткові можливості для стиснення. Завдяки цьому алгоритми стиснення зображень можуть мати дуже високі характеристики.

РОЗДІЛ 1 ОСНОВИ КОДУВАННЯ ЗОБРАЖЕНЬ

1.1 Класи зображень

Растрові зображення представляють собою двовимірний масив чисел. Елементи цього масиву називають пікселами (від англійського pixel - picture element). Всі зображення можна поділити на дві групи - з палітрою та без неї. В зображень з палітрою в пікселі зберігається число - індекс в деякому одновимірному векторі кольорів, що називається палітрою. Частіше за все зустрічаються палітри з 16 та 256 кольорів.

Зображення без палітри можуть бути в будь-якій системі представлення кольорів та в градаціях сірого (grayscale). Для останніх значення кожного піксела інтерпретується як яскравість відповідної точки. Зустрічаються зображення з 2, 16 та 256 рівнями сірого. При використанні певної системи представлення кольорів кожний піксел представляє собою структуру, полями якої є компоненти кольору. Найбільш розповсюдженою є система RGB, в якій колір представлено значеннями інтенсивності червоної (R), зеленої (G) та синьої (B) компонент. Існують і інші системи представлення кольорів, такі як CMYK, CIE XYZccir60-1 і т.п.

Для того, щоб коректно оцінювати ступінь стиснення введемо поняття класу зображень. Під класом будемо розуміти деяку сукупність зображень, застосування до якої алгоритму архівації дає якісно однакові результати. Наприклад, для одного класу алгоритм дає дуже високу ступінь стиснення, для другого - майже не стискає, для третього - збільшує розмір файлу.

Розглянемо такі приклади неформального визначення класів зображень [6]:

1. Клас 1. Зображення з невеликою кількістю кольорів (4-16) і великими областями, заповненими одним кольором. Плавні переходи кольорів відсутні. Приклади: ділова графіка - гістограми, діаграми, графіки і т.п.
2. Клас 2. Зображення з плавними переходами кольорів, побудовані на комп’ютері. Приклади: графіка презентацій, ескізні моделі в САПР, зображення, побудовані за методом Гуро.
3. Клас 3. Фотореалістичні зображення. Приклад: відсканові фотографії.

4. Клас 4. Фотореалістичні зображення з накладанням ділової графіки. Приклад: реклама.

Розвиваючи дану класифікацію, в якості окремих класів можуть бути запропоновані неякісно відскановані в 256 градацій сірого кольору сторінки книг або растрові зображення топографічних карт. Формально будучи 8- або 24-бітовими, вони несуть навіть не растрову, а чисто векторну інформацію. Окремі класи можуть формувати і зовсім специфічні зображення: рентгенівські знімки або фотографії в профіль і фас з електронного досьє [6].

Актуальною залишається задача пошуку найкращого алгоритму для конкретного класу зображень.

Дослідження, які виконуються в області стиснення зображень, орієнтовані в першу чергу на зображення класу 3 – фотореалістичні зображення, оскільки їх стиснення вимагає найбільших витрат і при його виконанні виникають найбільші труднощі. Дійсно, стиснення, наприклад, комп'ютерної графіки може бути легко виконано передачею файлу, що використовується для формування графічного зображення, стиснення зображень з малим числом градацій яскравості також не представляє значних труднощів, оскільки при його стисненні можуть бути використані алгоритми з малою обчислювальною складністю.

1.2 Джерела надлишковості зображень

В зображенні розрізняють два основних види надлишковості [3]:

- статистична надлишковість;
- фізіологічна надлишковість.

Перша пов'язана з тим що будь-які величини отримані із зображення не є випадковими. Сусідні відліки часто мають подібні значення яскравості, в чому проявляється важлива властивість їх просторової кореляції. Якщо відповідним чином використати цю властивість, то можна значно зменшити число біт для подання зображення у цифровій формі.

Фізіологічна надлишковість пов'язана з тією частиною інформації, яка не сприймається оком людини. Скорочення фізіологічної надлишковості в значній мірі скорочує і статистичну надлишковість і навпаки.

Необхідно відзначити і так звану афінну надлишковість, на якій базуються фрактальні методи [5] стиснення зображень (більш детально розглядається нижче).

1.3 Класи програм і вимоги до алгоритму стиснення зображень

Розглянемо таку класифікацію програм, що використовують алгоритми архівації [6]:

1. Клас 1. Характеризуються високими вимогами до часу архівації та розархівації. Нерідко потрібний перегляд зменшеної копії зображення та пошук в базі даних зображень. Приклади: видавничі системи в широкому розумінні цього слова. Причому, як системи, де необхідні публікації (журнали) з високою якістю зображень і використанням алгоритмів стиснення без втрат, так і видавничі системи для газет, і інформаційних вузлів WWW - де є можливість оперувати зображеннями меншої якості і використовувати алгоритми стиснення з втратами. В подібних стистемах приходиться мати справу з повнокольоровими зображеннями різного розміру (від 640×480 - формат цифрового фотоапарата, до 3000×2000) і з великими двокольоровими зображеннями. Оскільки ілюстрації займають велику частину від загального об’єму матеріалу в документі, проблема зберігання стоїть дуже гостро. Проблеми також створює велика різнорідність ілюстрацій (приходиться використовувати універсальні алгоритми). Єдине, що можна сказати завчасно, це те, що переважно це будуть фотореалістичні зображення та ділова графіка.
2. Клас 2. Характеризується високими вимогами до ступеня архівації та часу розархівації. Час архівації значення не має. Іноді подібні програми також вимагають від алгоритму стиснення можливості масштабування зображення під конкретну роздільну здатність монітора користувача. Приклад: довідники та енциклопедії на CD-ROM. З появою великої кількості комп’ютерів, оснащених цим приводом (у США - у 50% машин) досить швидко сформувався ринок програм, що випускаються на лазерних дисках. Не дивлячись на те, що ємність одного диска достатньо велика (приблизно 650 Мбайт), її, як правило, не вистачає. При створенні енциклопедій та ігор велику частину диска

займають зображення та відео. Таким чином, для цього класу програм актуальність мають суттєво асиметричні за часом алгоритми (симетричність за часом - відношення часу архівації до часу розархівації).

3. Клас 3. Характеризується дуже високими вимогами до ступеня архівації. Приклад: нова широко розповсюджена система «Всесвітня інформаційна паутина» - WWW. В цій системі достатньо активно використовуються ілюстрації. При оформленні інформаційних або рекламних сторінок хочеться зробити їх більш яскравими та якісними, що звичайно впливає на розмір зображень. Більш за все при цьому страждають користувачі, підключені до мережі за допомогою повільних каналів зв'язку. Якщо сторінка WWW перенасичена графікою, то очікування її повної появи на екрані може затягнутися. Оскільки при цьому навантаження на процесор мале, то тут можуть знайти застосування ефективні складні алгоритми стиснення з порівняно великим часом розархівації.

Можна привести багато більш вузьких класів програм. Так своє застосування машинна графіка знаходить і в різноманітних інформаційних системах. Наприклад, вже стає звичним досліджувати ультразвукові та рентгенівські знімки не на папері, а на екрані монітора. Поступово до електронного вигляду переводять і історії хвороб. Зрозуміло, що зберігати ці матеріали логічно в єдиній картотеці. При цьому без використання спеціальних алгоритмів велику частину архівів займуть фотографії. Тому при створенні ефективних алгоритмів розв'язання цієї задачі потрібно врахувати специфіку рентгенівських знімків – наявність великої кількості областей в зображенні з плавними переходами.

В геоінформаційних системах - при зберіганні аерофотознімків - специфічними проблемами є великий розмір зображення і необхідність вибірки лише частини зображення за вимогою. Крім цього, може знадобитися масштабування. Це, безумовно, накладає свої обмеження на алгоритм стиснення.

В електронних картотеках і досьє різних служб для зображень характерна подібність між фотографіями в профіль та подібність між фотографіями в фас, яку також потрібно враховувати при створенні алгоритму архівації. Подібність між фотографіями спостерігається і в

інших спеціалізованих довідниках. В якості прикладу можна привести енциклопедії птахів або квітів.

Фактично саме характер використання зображень задає тон в виборі того чи іншого алгоритму стиснення. І при цьому задається ступінь важливості таких суперечливих вимог до алгоритму:

1. Високий коефіцієнт стиснення (архівації). Актуальний далеко не для всіх програм. Деякі алгоритми дають кращу якість зображення при високих коефіцієнтах стиснення, однак програють іншим алгоритмам при низьких.
2. Висока якість зображень. Виконання цієї вимоги напряму суперечить виконанню попередньої.
3. Висока швидкість архівації. Ця вимога для деяких алгоритмів з втратами інформації є взаємновиключною з першими двома. Інтуїтивно зрозуміло, що чим більше часу ми будемо аналізувати зображення, намагаючись отримати найбільший ступінь архівації, тим кращим буде результат. І, відповідно, чим менше часу витрачається на архівацію (аналіз), тим нижчою буде якість зображення і більшим його розмір.
4. Висока швидкість розархівації. Достатньо універсальна вимога, актуальна для багатьох програм. Однак можна привести приклади програм, де час виконання розархівації далеко на критичний.
5. Масштабування зображень. Дана вимога означає легкість зміни розмірів зображення до розмірів вікна активної програми. Справа в тім, що одні алгоритми дозволяють легко масштабувати зображення прямо під час розархівації, в той час як інші не тільки не дозволяють легко масштабувати, але і збільшують імовірність появи неприємних спотворень після застосування стандартних алгоритмів масштабування до розархівованого зображення. Наприклад, можна привести приклад «поганого» зображення для алгоритму JPEG - це зображення з достатньо дрібним регулярним малюнком (дрібна клітинка). Характер внесених алгоритмом JPEG змін такий, що зменшення або збільшення зображення може призвести до появи неприємних ефектів.
6. Можливість показати зображення низької розподільчої здатності, використавши тільки початок файлу. Дана можливість актуальна для різного роду мережевих програм, де зчитування зображень може

зайняти достатньо багато часу, і бажано, отримавши початок файлу, коректно показати preview.

7. Стійкість до помилок. Дана вимога означає локальність порушень в зображенні при зруйнуванні або втраті фрагмента файлу, що передається. Дана можливість використовується при передачі за багатьма адресами зображень в мережі, тобто в тих випадках, коли неможливо використати протокол передачі, який повторно дає запит на дані у сервера при виникненні помилок. Наприклад, якщо передається відеоряд кадрів, то було б неправильно використовувати алгоритм, в якого збій приводив би до зупинки правильного показу всіх наступних кадрів. Дана вимога суперечить високому ступені архівації, оскільки інтуїтивно зрозуміло, що ми повинні вводити в потік надлишкову інформацію. Однак для різних алгоритмів об'єм цієї надлишкової інформації може суттєво відрізнятися.
8. Врахування специфіки зображення. Більш високий ступінь архівації для класу зображень, які статистично частіше будуть застосовуватись в нашій програмі.
9. Редагованість. Під редагованістю розуміється мінімальний ступінь погіршення якості зображення при його повторному зберіганні після редагування. Багато алгоритмів з втратою інформації можуть суттєво спотворити зображення за декілька ітерацій редагування.
10. Ефективність програмно-апаратної реалізації.

Дані вимоги до алгоритму реально пред'являють не тільки виробники ігрових приставок, але і виробники багатьох інформаційних систем [6].

1.4 Критерії вірності кодування

Задача відновлення зображення після кодування полягає в тому, щоб одержати вихідне зображення, яке в найменшій мірі відрізняється від вхідного зображення.

Широко розповсюдженою мірою вірності кодування зображень є усереднена середньоквадратична помилка [6,13]:

$$e^2 = \frac{1}{M*N} \sum_{i=1}^{N} \sum_{J=1}^{M} E(U_{ij} - \hat{U}_{ij})^2, \qquad (1.1)$$

де U_{ij} - значення відліків початкового зображення, $\hat{U}_{ij}$ - значення відліків відновленого зображення; M, N – розміри сторін зображення; E(•) – математичне очікування.

В експериментах мірою середньоквадратичної помилки служить середнє значення поелементних середньоквадратичних помилок:

$$\bar{e}^{\,2} = \frac{1}{M*N} \sum_{i=1}^{N} \sum_{J=1}^{M} (U_{ij} - \hat{U}_{ij})^2 . \qquad (1.2)$$

На основі приведених вище залежностей можна визначити відношення сигнал/шум:

$$SNR = 10\lg \frac{255^2}{e^2}, \qquad (1.3)$$

У чисельнику заданий розмах значень відеоданих, що звичайно задаються у виді дискретних відліків, проквантованих на 256 рівнів.

Критерій середньоквадратичної помилки - природна міра спотворень з фізичної і математичної точки зору. Але якщо зображення призначені для візуального спостереження, то перевага цьому критерію віддається не завжди. Це пов'язано з тим, що зорова система не обробляє зображення елемент за елементом, а витягає з нього в процесі нейронного кодування деякі просторові, часові ознаки, а також ознаки кольору.

Найбільш розповсюдженим і найнадійнішим способом визначення якості зображення є суб'єктивна експертиза [20].

Як експертів рекомендується залучати спостерігачів-неспеціалістів, їх оцінки визначають якість зображення саме так, як його сприймає "середній спостерігач".

Відповідно до рекомендації 654 МККР рекомендується п’ятибальна шкала абсолютних оцінок:

Якість	Погіршення
5. Відмінна	5. Непомітне
4. Добра	4. Помітне, але не заважає
3. Задовільна	3. Злегка заважає
2. Погана	2. Заважає
1. Дуже погана	1. Дуже заважає

За результатами експертних оцінок звичайно визначається середній бал:

$$\overline{C} = \Sigma(n_k C_k) / \Sigma n_k \ , \qquad (1.4)$$

де n_k - число зображень, віднесених до К-ої категорії, а $C_к$ – відповідний їй бал.

1.5 Етапи кодування зображень

Ефективне кодування зображень звичайно виконується за три етапи:

1. На першому етапі знаходиться представлення зображення в вигляді набору коефіцієнтів деякого перетворення. Ця операція як правило зворотна.
2. На другому етапі зменшується точність представлення компонент зображення, але так, щоб виконувались задані вимоги до якості зображення. Така операція призводить до втрат інформації, тому не є зворотною.
3. На третьому етапі усувається статистична надлишковість в зображенні, отриманому після виконання перших двох етапів. Для виконання цього етапу може застосовуватись кодування Хаффмана, арифметичне кодування та інші. Ця операція зворотна [13].

Найбільш інтенсивні дослідження виконуються по пошуку нових методів для виконання першого і другого етапів, оскільки при цьому витрачається найбільше обчислювальних ресурсів і дослідження пов’язані не тільки з пошуком математичного перетворення, але і з дослідженням особливостей зорового сприйняття зображення і особливостей завадостійкої передачі цього зображення по каналах зв’язку.

1.6 Аналіз основних методів кодування зображень

Базовим методом цифрового кодування джерел зображень є імпульсно-кодова модуляція (ІКМ). Вона характеризується тим, що кожному закодованому в цифрову форму слову відповідає квантований в часі і за амплітудою відлік відеоінформації. При цьому повинні виконуватись вимоги теореми дискретизації $f_д > 2W_0$, де W_0 максимальна частота, яка міститься в сигналі. Щоб запобігти появі фальшивих контурів на однокольоровому зображенні необхідно більше 50 рівнів квантування, що відповідає 6-8 розрядному кодовому слову на кожний елемент (піксел) зображення. Через великі об'єми інформації ІКМ застосовується лише при внутрістудійній передачі телевізійних зображень паралельним кодом і в якості базового канонічного подання зображення у цифровій формі[1,2,3,4].

Серед методів кодування з передбаченням найбільш досліджена диференційно-імпульсна кодова модуляція (ДІКМ). Суть ДІКМ така. Організується передбачення значення яскравості кожного наступного елемента зображення на основі лінійної комбінації значень яскравості попередніх елементів. Оцінка отримана в результаті передбачення віднімається від істинного значення яскравості елемента зображення і різницевий сигнал квантується невеликим числом рівнів. За рахунок цього і досягається скорочення об'єму даних[1,3,7].

Методи кодування зображень з передбаченнями дозволяють стиснути зображення в 2-2,5 рази при простій технічній реалізації. Серед недоліків цих методів необхідно відзначити такі:

- похибки в місцях різких перепадів яскравості;
- низька завадостійкість.

Інтерполяційні методи основані на числових методах апроксимації, за допомогою яких послідовність або двовимірний масив відліків яскравості наближено подаються через неперервні функції [8,9]. При цьому кодуються лише окремі відліки зображення, а сусідні з ними отримують в результаті інтерполяції поліномами, звичайно, не більше чим

третього степеня. Коефіцієнт стиснення, який досягається при використанні цих методів, дорівнює 5-6.

Основним недоліком інтерполяційних методів є великий об'єм обчислень при інтерполяції поліномами високих степенів, а також необхідність зберігання координат базових відліків зображення.

Важливий клас методів кодування зображень - це методи кодування на основі перетворень. Кодування на основі перетворень - непрямий метод. Зображення піддаються унітарному математичному перетворенню, отримані в результаті коефіцієнти перетворення квантуються і кодуються для передачі по каналу зв'язку або запису в файл. Для більшості зображень значення багатьох коефіцієнтів перетворення порівняно мале. Такі коефіцієнти часто можна відкинути або виділити для їх кодування невелике число двійкових розрядів. Найчастіше використовують двовимірні ортогональні перетворення. Це такі як перетворення Карунена-Лоева, дискретне перетворення Фур'є (ДПФ), дискретне косинусне перетворення (ДКП), перетворення Уолша-Адамара [2,3,8,9,10] та інші. Коефіцієнти стиснення при застосуванні цих методів можуть досягати 6-8, завдяки чому деякі з цих методів(ДКП) знайшли широке застосування як основа промислових стандартів з кодування зображень. Спільним недоліком цих методів є досить висока обчислювальна складність, а також неможливість розв'язання ряду задач обробки зображень на скороченому об'ємі даних.

Серед статистичних методів найбільш широке застосування знаходять блочні методи кодування зображень. Блоки розміром MxN елементів кодуються у відповідності з імовірністю їх появи. Для найбільш імовірних конфігурацій використовуються короткі кодові слова, а для менш імовірних - довгі кодові слова(алгоритм Хаффмана), в результаті чого досягається стиснення даних [11]. Коефіцієнти стиснення при використанні цих методів можуть досягати 4-5.

Перспективним для кодування як рухомих так і нерухомих зображень є метод покомпонентного кодування [12-14]. Особливістю цього методу є формування декількох двовимірних сигналів, що несуть інформацію про деталі зображення різного розміру. Наявність декількох каналів окремої обробки деталей зображення різного розміру, дозволяє

ефективно кодувати зображення як внутрікадровими так і міжкадровими методами з урахуванням особливостей сприйняття інформації зоровим аналізатором людини. Важливою перевагою цього методу є те, що формати подання даних після стиснення є компонентами вихідного зображення, тобто кожна компонента візуально представляє зображення з тим чи іншим ступенем роздільної здатності, а це в свою чергу дозволяє розв'язувати ряд задач обробки зображень на скороченому об'ємі даних.

Хоча практично досягнутий коефіцієнт стиснення при застосуванні цього методу незначно менший в порівнянні з методами кодування з перетвореннями, його технічна реалізація значно простіша і відповідно швидкодія значно більша.

Для стиснення зображень у реальному часі за сукупністю таких параметрів як простота технічної реалізації, обчислювальна складність, коефіцієнт стиснення найкращі результати одержують при використанні Wavelet-кодування, що базується на пірамідальних схемах розкладу початкового зображення на компоненти (S-перетворення) [6]. Цей алгоритм орієнтований на стиснення кольорових і чорно-білих зображень з плавними переходами. Ідеальний для картинок типу рентгенівських фотографій. Коефіцієнт стиснення варіюється в межах 5-100. При великих коефіцієнтах стиснення на різких границях, особливо діагональних, можливі спотворення. Важливою перевагою S-перетворення є можливість показати “огрублене” зображення (низької роздільної здатності), використавши тільки початок файлу.

Найбільші коефіцієнти стиснення забезпечує метод фрактального стиснення зображень [5,15-19 та ін. (більш детальну інформацію можна отримати в Internet наприклад за адресою *http://www.dip.ee.uct.ac.za/imageproc/compression/fractal/*)], відкритий в 1988 році. Процес фрактального стиснення оснований на твердженні, що зображення реального світу мають афінну надлишковість. Коефіцієнти стиснення можуть досягати 50-60 раз. Основним недоліком цього методу є велика обчислювальна складність. Однак, враховуючи великий ступінь стиснення, який можна отримати цим методом, а також гігантський прогрес у збільшенні продуктивності мікропроцесорів та інших апаратних засобів, слід очікувати самого найширшого застосування даного методу в найближчі роки.

КОНТРОЛЬНІ ПИТАННЯ

1. Чому необхідне стиснення зображень?
2. Охарактеризуйте класи зображень.
3. Які основні джерела надлишковості зображень?
4. Охарактеризуйте основні класи програм, які використовують зображення.
5. Наведіть вимоги до алгоритмів кодування зображень та охарактеризуйте їх
6. Які етапи кодування зображень?
7. Охарактеризуйте критерії вірності кодування.
8. Чому основним критерієм якості відновлених зображень є метод експертних оцінок?
9. Наведіть недоліки і переваги інтерполяційних методів кодування зображень.
10. Які ортогональні перетворення застосовуються при кодуванні зображень?
11. Який вид надлишковості усувається при фрактальному стисненні?
12. Охарактеризуйте статистичні методи стиснення зображень.
13. Які види зображень найкраще стискуються за допомогою Wavelet-кодування.
14. Який недолік ДІКМ?

РОЗДІЛ 2 АЛГОРИТМИ СТИСНЕННЯ ЗОБРАЖЕНЬ БЕЗ ВТРАТ

2.1 Класифікація алгоритмів стиснення

Серед алгоритмів стиснення без втрат дві схеми стиснення - кодування Хаффмана (Huffman) і LZW - кодування (за початковими літерами прізвищ Лемпел (Lempel) і Зів (Ziv) - його автори і Уелч (Welch), який його суттєво модифікував), формують основу для багатьох систем стиснення. Ці схеми представляють два різних підходи до стиснення даних:

- статистичні методи стиснення;
- словарні (евристичні) методи стиснення.

Статистичні алгоритми (Хаффмана, Шеннона-Фано, арифметичні) потребують знання імовірностей появи символів в зображенні, оцінкою якої є частота появи символів у вхідних даних. Як правило, ці імовірності невідомі. З урахуванням цього статистичні алгоритми можна поділити на три класи:

1. Не адаптивні - використовують фіксовані, завчасно задані імовірності символів. Таблиця імовірностей символів не передається разом з файлом, оскільки вона відома завчасно. Недолік: невеликий перелік файлів, для яких досягається прийнятний коефіцієнт стиснення.
2. Напівадаптивні - для кожного файлу будується таблиця частот символів і за її допомогою стискають файл. Разом зі стисненим файлом передається таблиця символів. Такі алгоритми досить непогано стискають більшість файлів, але необхідна додаткова передача таблиці частот символів, а також два проходи початкового файлу для виконання кодування.
3. Адаптивні - починають працювати з фіксованою початковою таблицею частот символів (зазвичай всі символи спочатку рівноімовірні) і в процесі роботи ця таблиця змінюється в залежності від символів, що зустрічаються в файлі. Переваги: однопрохідність алгоритму, не потребує передачі таблиці частот символів, достатньо ефективно стискає широкий клас файлів.

Евристичні (словарні) алгоритми стиснення (типу LZ77, LZ78), як правило, шукають в файлі рядки, що повторюються і будують словник фраз, що вже зустрічались.

Зазвичай такі алгоритми мають цілий ряд специфічних параметрів (розмір буфера, максимальна довжина фрази і т.п.), підбір яких залежить від досвіду автора роботи, і ці параметри добираються таким чином, щоб досягти оптимального співвідношення часу роботи алгоритму, коефіцієнта стиснення та переліку файлів, що добре стискаються.

2.2 Алгоритм RLE

Даний алгоритм надзвичайно простий в реалізації. RLE (Run Length Encoding), або групове кодування, один з найбільш давніх алгоритмів кодування графіки. Його суть полягає в заміні символів, що повторюються, на один цей символ та лічильник повторень. Проблема є в тому, щоб архіватор при відновленні міг відрізнити в результувальному потоці таку кодовану серію від інших символів. Рішення очевидне - помістити у всі ланцюжки деякі заголовки (наприклад, використати перший біт як ознаку кодування серії). Метод достатньо ефективний для графічних зображень в форматі «байт на піксел» (наприклад, формат PCX використовує кодування RLE) .

Проте недоліки алгоритму очевидні - це, зокрема, мала пристосованість до багатьох типів файлів, що часто зустрічаються, наприклад, до текстових. Тому його можна ефективно використовувати тільки в поєднанні з вторинним кодуванням. Цей підхід знайшов своє відображення в алгоритмі кодування факсів: спочатку зображення розбивається на чорні та білі точки, які перетворюються алгоритмом RLE в потік довжин серій, а потім ці довжини серій кодуються методом Хаффмана зі спеціально підібраним (експериментально) деревом [6].

2.3 Кодування Хаффмана

Статистичне кодування Хаффмана співставляє вхідним символам, представленим ланцюжками бітів однакової довжини (наприклад, восьмибітовими байтами), ланцюжки бітів змінної довжини. Довжина коду для символу пропорційна двійковому логарифму його частоти, взятому із протилежним знаком. Це кодування є префіксним, що дозволяє

легко його декодувати однопрохідним алгоритмом. Префіксний код зручно відображати у вигляді двійкового дерева, в якого символи знаходяться в листях, а дуги відмічені 0 або 1. Тоді код символу можна задавати як шлях від кореня дерева до листка, який містить цей символ.

При використанні адаптивного кодування Хаффмана необхідне постійне коректування дерева у відповідності зі зміною статистики вхідного потоку. При реалізації це звичайно потребує значних витрат на перебалансування кодового дерева у відповідності з новими частотами символів на кожному кроці. Стиснення інформації виконується за два проходи:

1. Під час першого проходу оцінюються частоти появи символів в початковому файлі.
2. Під час другого проходу виконується представлення цих символів префіксними кодами змінної довжини.

Як це реалізується на практиці розглянемо на прикладі деякого файлу. Після першого етапу отримана така інформація про частоти появи символів в файлі:

Символ	U	A	M	H	N	F
Частота	43	19	7	23	27	61

Об'єм файла 180 байт (1440 біт). Він складається з шести різних символів, які зустрічаються в файлі від 7 до 61 раза.

На наступному етапі необхідно закодувати символи у відповідності з частотою їх повторення. Цей процес пояснює рис. 2.1. Виберемо із таблиці два символи з найменшою частотою появи у вхідному файлі і утворимо з них вузол. Вміст вузла - це сума обох частот символів (19+7=26). Далі знову знаходимо два символи з найменшою частотою. При цьому частоти 19 і 7 вже не враховуються, але в пошуку символів з найменшими частотами бере участь утворений вузол. Разом з символом "H" перший вузол створює наступний вузол (26+23=49).

Якщо проводити вибір символів за даною схемою, то колись структура повинна закінчитись. Для останнього вузла сума вже не записується,

оскільки порівняння закінчилось. Цей останній вузол називається коренем, через який можливий вхід в дерево кодування.

Символ	U	A	M	H	N	F
Частота	43	19	7	23	27	61

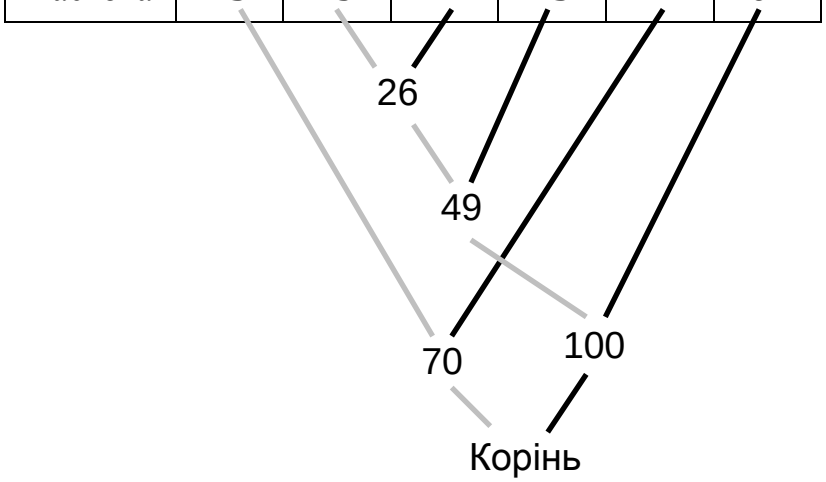


Рисунок 2.1 – Побудова двійкового дерева

Новий код для символів упакованого файлу одержимо із створеного двійкового дерева. Ця структура називається двійковим деревом тому, що кожний вузол утворюється із двох можливих розгалужень. Оскільки для біта існують теж тільки дві можливості (0 і 1), то утворюється новий код, який, починаючи від кореня, проходить від вузла до вузла, до відповідного символу. Для кожного вузла необхідно вирішити, куди потрібно повернути (ліворуч чи праворуч), щоб дійти до символу, який кодується. Будемо вважати, що ліва гілка відповідає "0", а права - "1". Перехід від кореня дерева до будь-якого символу дає такі коди:

U	Ліворуч-ліворуч	00
A	Праворуч-ліворуч-ліворуч-ліворуч	1000
M	Праворуч-ліворуч-ліворуч-праворуч	1001
H	Праворуч-ліворуч-праворуч	101
N	Ліворуч-праворуч	01
F	Праворуч-праворуч	11

Символи, які найбільш часто зустрічаються ("U" і "F") отримали короткі 2-бітові кодові комбінації, а ті, що найрідше ("M") - 4-бітові кодові комбінації. Ступінь стискання показаний в табл. 2.1: початкові 1440 біт файлу стиснулись до 435 біт, що відповідає ступеню стискання 31%.

Таблиця 2.1 – Оцінка ступеня стискання файлу

Символ	Частота	Кількість біт на символ	Загальна кількість
U	43	2	86
A	19	4	76
M	7	4	28
H	23	3	69
N	27	2	54
F	61	2	122
Всього	**180**		**435**

На останньому етапі програма оброблює файл і замінює кожний символ на його код. Також вона записує кодове дерево у файл, оскільки тільки за допомогою цього дерева можна знову декодувати файл. Ступінь стискання погіршується через те, що і кодове дерево потребує додаткової пам'яті.

Таким чином, алгоритм статистичного кодування способом Хаффмана буде складатись з таких етапів:

1. Визначення кількості повторювання кожного символу.
2. Побудова двійкового дерева.
3. Побудова кодової послідовності для кожного символу.
4. Заміна символів заданого файлу відповідними кодовими послідовностями і запис результату.

Недоліком кодування Хаффмана є залежність ступеня стискання від близькості ймовірностей символів до від'ємних степенів двійки, а також складна апаратурна реалізація, яка пов’язана з необхідністю запам’ятовування кадру зображення, тобто кодування виконується за два проходи.

2.4 Кодування за ступенем новизни

При невідомому розподілі частот використовують простий і досить ефективний метод стискання - кодування за ступенем новизни. Цей метод був так названий Елайєсом в 1987 році, але він був відкритий ще раніше Б.Я. Рябком в 1980 році і відомий під назвою стискання за допомогою купки книжок [20].

Ідея методу така: нехай алфавіт джерела складається з N символів з номерами 1,2,...,N. Кодер зберігає список символів, які являють собою деяку перестановку алфавіту. Коли на вхід поступає символ С, який має в списку номер “i”, кодер передає код номеру “i”. Потім кодер переставляє символ С на початок списку, збільшуючи на одиницю номери всіх символів, які стоять перед С. Таким чином, більш "популярні" символи будуть тяжіти до початку списку та будуть мати більш короткі коди.

В якості кода для кодування за ступенем новизни можна використовувати монотонний код - універсальний код джерела, для якого відомо лише впорядкованість імовірностей символів. Монотонний код побудований так, що більш близькі до початку списку позиції отримують більш короткі коди. В результаті літери, які часто з'являються та тяжіють до початку списку, будуть мати короткі кодові позначення. Розглянемо стискання інформації методом кодування за ступенем новизни на прикладі.

Нехай необхідно передати слово w=221312233 в алфавіті A={1,2,3}. Монотонним кодом позиції 1 є 0, позиції 2 - 10, позиції 3 - 11. Купка книжок при послідовній появі літер слова w змінюється таким чином:

(1,2,3) -- (2,1,3) -- (2,1,3) -- (1,2,3) -- (3,1,2) і т.д.

Кодом слова буде

100101110110…

Кодування за ступенем новизни ненабагато гірше найкращого кодування методом Хаффмана. Даний метод уступає кодуванню Хаффмана при стисканні текстових файлів, однак, для файлів з малим початковим алфавітом, можна досягти гарних результатів. Саме такими

файлами є файли, наприклад, контурних зображень. Кодування за ступенем новизни не потребує попереднього перегляду початкового масиву. Цей метод легко пристосовується до можливих коливань частот, тому він названий ще - локально адаптивним .

2.5 Алгоритм LZW

Власне вихідний Lempel/Ziv підхід до стиснення даних був вперше обнародуваний в 1977 р., а вдосконалений (Terry Welch) варіант був опублікований в 1984 р. [6]. Алгоритм дуже простий. Якщо коротко, то LZW - стиснення заміняє рядки символів деякими кодами. Це робиться без будь-якого аналізу вхідного тексту. Замість цього, при добавленні кожного нового рядка проглядається таблиця рядків. Стиснення відбувається, коли код заміняє рядок символів. Коди, генеровані LZW - алгоритмом, можуть бути будь-якої довжини, але вони повинні містити більше біт, ніж одиничний символ. Перші 256 кодів (коли використовуються 8-бітові символи) за замовчуванням відповідають стандартному набору символів. Решта кодів відповідають рядкам, що обробляються алгоритмом.

При стисненні та відновленні LZW маніпулює трьома об`єктами: потоком символів, потоком кодів і таблицею рядків. При стисненні потік символів є вхідним і потік кодів - вихідним. При відновленні вхідним є потік кодів, а потік символів - вихідним. Таблиця рядків породжується і при стисненні і при відновленні, однак вона ніколи не передається від стиснення до відновлення і навпаки.

Оскільки ми не знаємо наперед об`єм файлу, то важко визначити, яка розрядність кодів буде оптимальною. Тому доцільно використовувати коди різної довжини.

2.5.1 Стиснення інформації

Приведемо алгоритм в найпростішій формі. Кожний раз, коли генерується новий код, новий рядок добавляється в таблицю рядків. LZW постійно перевіряє, чи даний рядок вже відомий, і, якщо так, виводить код без генерування нового.

Процедура LZW - стиснення приведена на рис. 2.2 .

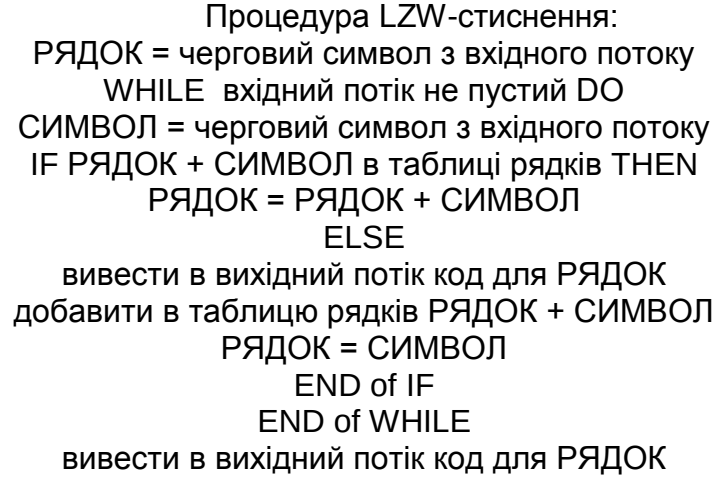


Рисунок 2.2 - Алгоритм стиснення

Простий рядок, використаний для демонстрації алгоритму, приведений на рис. 2.3. Вхідний рядок є коротким списком англійських слів, розділених символом «/». Як ви можете помітити, аналізуючи алгоритм, його робота починається з того, що на першому кроці циклу він виконує перевірку на наявність рядка «/W» в таблиці. Коли він не знаходить цей рядок, то генерує код для «/» і добавляє в таблицю рядок «/W». Оскільки 256 символів вже визначені для кодів 0 - 255, то першому визначеному рядку може бути поставлено у відповідність код 256. Після цього система читає наступну букву «E», добавляє другий підрядок «WE» в таблицю рядків і виводить код для букви «W». Коли знову буде прочитаний рядок, що складається з символів «/» і «W», то йому буде поставлено у відповідність рядок з таблиці рядків. В цьому випадку система виводить код 256 і добавляє трисимвольний рядок в таблицю (/WE). Цей процес продовжується до тих пір, поки не закінчиться вхідний потік і всі коди не будуть виведені.

Вихідний потік для заданого рядка (/WED/WE/WEE/WEB/WET) показаний на рис. 2.3, так як і отримана в результаті таблиця рядків. Ця таблиця швидко заповнюється, оскільки новий рядок добавляється в таблицю кожний раз, коли генерується код. В цьому прикладі було виведено п`ять закодованих підрядків і сім символів. Звичайно, цей приклад було вибрано тільки для демонстрації. В дійсності стиснення не

починається до тих пір, поки не буде збудована досить велика таблиця, зазвичай після читання порядка 100 вхідних байтів.

Вхідний рядок: /WED/WE/ WEE/ WEB/WET

Вхід (символи)	Вихід (коди)	Нові коди і відповідні рядки
/W	/	256 = /W
E	W	257 = WE
D	E	258 = ED
/	D	259 = D/
WE	256	260 = /WE
/	E	261 = E/
WEE	260	262 = /WEE
/W	261	263 = E/W
EB	257	264 = WEB
/	B	265 = B/
WET	260	266 = /WET
<EOF>	T	

Рисунок 2.3 - Процес стиснення

.

2.5.2 Відновлення інформації

Алгоритму стиснення відповідає свій алгоритм відновлення інформації. Він отримує вихідний потік кодів від алгоритму стиснення і використовує його для точного відновлення вхідного потоку. Однією з причин ефективності LZW - алгоритму є те, що він не потребує зберігання таблиці рядків, яка отримана при стисненні. Таблиця може бути точно відновлена на основі вихідного потоку алгоритму стиснення. Це можливо тому, що алгоритм стиснення виводить РЯДКОВУ та СИМВОЛЬНУ компоненти коду раніше, ніж він помістить цей код у вихідний потік. Це означає, що стисненні дані не обтяжені необхідністю тягнути за собою велику таблицю рядків.

Алгоритм декодування даних стиснутих методом LZW приведений на рис. 2.4. У відповідності з алгоритмом стиснення, він добавляє новий рядок в таблицю рядків кожний раз, коли читає з вхідного потоку новий код. Все, що йому необхідно зробити додатково - це перевести кожний

вхідний код в рядок і переслати його в вихідний потік. Важливо відзначити, що побудова таблиці рядків алгоритмом декодування закінчується тоді, коли побудована таблиця рядків алгоритмом стиснення.

Процедура LZW - декодування:
читати СТАРИЙ_КОД
вивести СТАРИЙ_КОД
WHILE вхідний потік не пустий DO
читати НОВИЙ_КОД
РЯДОК = перевести НОВИЙ_КОД
вивести РЯДОК
СИМВОЛ = перший символ РЯДКА
добавити в таблицю рядків СТАРИЙ_КОД + СИМВОЛ
СТАРИЙ_КОД = НОВИЙ_КОД
END of WHILE

Рисунок 2.4 - Алгоритм декодування

Роботу даного алгоритму на основі стиснених даних, отриманих в результаті стиснення вище розглянутого рядка (/WED/WE/WEE/WEB/WET), демонструє рис. 2.5.

Вихідний потік точно відповідає вхідному потоку алгоритму стиснення. Перші 256 кодів визначені для переводу одиничних символів, так само як і в алгоритмі стиснення.

Вхідні коди: / W E D 256 E 260 261 257 B 260 T

Вхід Новий код	Старий код	Рядок Вихід	Символ	Новий вхід Таблиці
/	/	/		
W	/	W	W	256 = /W
E	W	E	E	257 = WE
D	E	D	D	258 = ED
256	D	/W	/	259 = D/
E	256	E	E	260 = /WE
260	E	/WE	/	261 = E/
261	260	E/	E	262 = /WEE
257	261	WE	W	263 = E/W
B	257	B	B	264 = WEB
260	B	/WE	/	265 = B/
T	260	T	T	266 = /WET

Рисунок 2.5 - Процес декодування

2.6 Арифметичне кодування

Основні принципи арифметичного кодування були розроблені наприкінці 70-х років [21]. Арифметичне кодування, так само як і імовірностні методи, використовує як основу технології стиску імовірність появи символу у файлі, однак сам процес арифметичного кодування має принципові відмінності. У результаті арифметичного кодування символьна послідовність (рядок) замінюється дійсним числом більше нуля і менше одиниці.

Розглянемо процес арифметичного кодування слова "REDUNDANCE" (надмірність). Імовірність появи кожного символу в цьому слові дорівнює 0.1 за винятком букв E, D і N, що зустрічаються двічі, і, відповідно, імовірність їхньої появи дорівнює 0.2. Далі кожній букві присвоюється інтервал імовірності (range), довжина якого розраховується, виходячи з імовірності їхньої появи в слові (табл. 2.2).

Перша буква слова – “R” - одержує інтервал з нижньою границею 0.8 і з верхньої - 0.9. Нижня границя інтервалу і стає першою значущою цифрою коду. Потім виконується розрахунок границь підінтервалів для кожної наступної букви за такими виразами:

$$\beta_n^l = \beta_{n-1}^l + (\beta_{n-1}^h - \beta_{n-1}^l) P_n^l$$
$$\beta_n^h = \beta_{n-1}^l + (\beta_{n-1}^h - \beta_{n-1}^l) P_n^h,$$

де β^l, β^h – нижня і верхня границя кодового інтервалу, P^l і P^h – нижня і верхня границі інтервалу імовірності для символа.

У результаті, послідовність символів 'REDUNDANCE' заміняється числом 0.8478570048, що представлено в табл. 2.3. Таким чином, замість 10 байт необхідних для збереження символьного рядка, буде потрібно всього 4 байти для запису числа.

Арифметичне кодування дозволяє забезпечити високий ступінь стиску даних, особливо у випадках, коли зустрічаються дані, де частота появи різних символів сильно відрізняється одна від одної. У той же час сама процедура арифметичного кодування вимагає потужних обчислювальних ресурсів, і донедавна цей метод мало застосовувався при кодуванні зображень через повільну роботу алгоритму і, відповідно, істотного часу затримки при передачі даних. Хоча, варто очікувати

високих коефіцієнтів стиску при його застосуванні для кодування зображень.

Таблиця 2.2 - Інтервали імовірності для символів в слові Redundance

Символ	Імовірність (p)	Інтервал
A	0.1	0.0-0.1
C	0.1	0.1-0.2
D	0.2	0.2-0.4
E	0.2	0.4-0.6
N	0.2	0.6-0.8
R	0.1	0.8-0.9
U	0.1	0.9-1.0

Таблиця 2.3 - Покрокове представлення рядка REDUNDANCE методом арифметичного кодування

Символ	Нижня границя (β^l)	Верхня границя (β^h)
R	0.8	0.9
E	0.84	0.86
D	0.844	0.848
U	0.8476	0.848
N	0.84784	0.84792
D	0.847856	0.847872
A	0.8478560	0.8478576
N	0.84785696	0.84785728
C	0.847856992	0.847857024
E	0.8478570048	0.8478570432

2.7 Шифрування і стиснення зображень

Шифрування і стиснення даних - надзвичайно пов’язані задачі. В обох випадках змінюється частотний розподіл символів у вихідних даних. Зменшення надлишковості (стиснення) початкових даних значно підвищує криптостійкість алгоритмів шифрування.

Наукова криптологія (cryptos – таємний, logos – слово) бере початок з роботи К.Шеннона “Теорія зв’язку в секретних системах” (1949р.), в якій було показано, що для деякого випадкового шифру кількість знаків шифротексту, отримавши який криптоаналітик при необхідних обчислювальних ресурсах зможе відновити ключ (тобто розкрити шифр), становить:

$$n = \frac{H(Z)}{r \log N}, \tag{2.1}$$

де H(Z) – ентропія ключа, r – надлишковість відкритого тексту, N-обсяг алфавіту.

З виразу (2.1) видно, що зниження надлишковості (стиснення даних) може значно збільшити криптостійкість навіть для коротких ключів [22].

2.7.1 Шифрування інформації

Найбільш простими в реалізації є методи прямої підстановки та перестановки. В прямих підстановках кожний знак початкового тексту замінюється одним або декількома іншими знаками. Розрізняють:

- моноалфавітну підстановку
- багатоалфавітну підстановку.

При моноалфавітній підстановці встановлюється відповідність між кожним знаком a_i алфавіта повідомлення А і відповідного знаку зашифрованого тексту. Всі методи моноалфавітної підстановки можливо представити як числові перетворення букв початкового тексту у відповідності з виразом:

$$C = (ap + S) \bmod k, \tag{2.2}$$

де a - десятковий коефіцієнт, S - коефіцієнт зсуву, k - розмір алфавіту, p - символ початкового тексту [23].

Недоліком моноалфавітної підстановки є низька криптостійкість, оскільки в шифрованому повідомленні зберігається частотний розподіл символів початкового тексту.

Ці недоліки можна зменшити за рахунок використання багатоалфавітної підстановки. При n-алфавітній підстановці знак m_1 з початкового повідомлення замінюється знаком з алфавіту B_1, m_2 – відповідно з алфавіту B_2, …, m_n – знаком з алфавіту B_n, m_{n+1} – знову з

алфавіту B_1 і т.д. Ефект використання багатоалфавітної підстановки полягає в тому, що забезпечується маскування природної частотної статистики початкової мови повідомлення, оскільки конкретний знак з алфавіту А може бути перетворений в декілька різних знаків шифрувального алфавіту В.

Знаки початкового тексту можливо також переставляти у відповідності з деяким правилом. Недолік перестановок той самий, що і моноалфавітної підстановки.

Ефективність шифрування можливо підвищити, використовуючи комбінацію перестановок і підстановок.

Значно більшу криптостійкість забезпечує шифрування з використанням генератора псевдовипадкових чисел (ПВЧ). Найбільш широке застосування знаходять лінійні конгруентні генератори ПВЧ. Цей генератор формує послідовність псевдовипадкових чисел T_1, T_2,…, T_m, використовуючи співвідношення:

$$T_{i+1} = (aT_i + c) \bmod m, \qquad (2.3)$$

де a і c - константи, T_0 – початкова величина, вибрана в якості породжувального числа.

Період повторення псевдовипадкових чисел залежить від вибраних значень a і c. Лінійний конгруентний генератор має максимальну довжину m тільки тоді, поки c - непарне, а $a \bmod 4 = 1$.

При шифруванні псевдовипадкові числа, отримані з виразу (2.3), об’єднуються деяким чином з відповідним обсягом тексту, але так, щоб можна було відновити початковий текст. Наприклад, з використанням додавання за модулем 2 (накладання гами-ключа на початковий текст). Якщо періодичність генератора більша довжини всіх посланих повідомлень початкового тексту і якщо криптоаналітику невідомий ніякий початковий текст, то шифр теоретично неможливо відкрити [23].

Такі відомі схеми кодування як DES, RSA, алгоритм криптографічного перетворення згідно з ГОСТ 28147-89 [24] повністю або частково є комбінацією методів, розглянутих вище.

2.7.2 Одночасне стиснення і шифрування даних

Основною метою кодування або стиснення даних є перетворення вхідного потоку символів в потік бітів мінімальної довжини. Це досягається за рахунок зменшення надлишковості вихідного потоку. При цьому символи, які найбільш часто зустрічаються, представляються короткими кодовими комбінаціями, за рахунок чого і досягається стиснення. Однак, при стисненні даних деякими методами існує можливість одночасного шифрування або без додаткових обчислювальних затрат, або з низькими затратами. Оскільки в більшості випадків, файли, які передаються по комп'ютерних мережах або зберігаються в пам'яті, представлені в стиснутому вигляді, то було б доцільно надати можливість користувачам виконувати при стисненні файлів і їх шифрування.

Розглянемо з цієї точки зору деякі методи стиснення інформації.

Достатньо простий і ефективний метод стиснення даних з невідомим розподілом частот, відомий як кодування за ступенем новизни (див. п. 2.4).

При стисненні інформації цим методом можливе одночасне виконання шифрування методом багатоалфавітної підстановки без додаткових затрат, або з застосуванням інших методів і їх комбінацій з незначними додатковими затратами. Нехай передається повідомлення:

ω**=ABCDDAAACAB**

в алфавіті **AL={A,B,C,D}**. При появі в слові ω чергової букви "a" по лінії зв'язку передається монотонний код номера позиції, яку займає в даний момент символ "a" в списку, а символ "a"

Таблиця 2.4 - Кодування за ступенем новизни

Вхід	Вихід1	Алфавіт1	Вихід2	Алфавіт2
A	0_2	{ABCD}	110_2	{CBAD}
B	10_2	{ABCD}	110_2	{ACBD}
C	110_2	{BACD}	110_2	{BACD}
D	111_2	{CBAD}	111_2	{CBAD}
D	0_2	{DCBA}	0_2	{DCBA}
A	111_2	{DCBA}	111_2	{DCBA}
A	0_2	{ADCB}	0_2	{ADCB}

переміщується на початок списку. Таким чином символи, які часто зустрічаються, завжди будуть знаходитись на початку списку і відповідно представлятись короткими кодовими комбінаціями. Порядок кодування за ступенем новизни демонструє табл. 2.4.

Якщо список алфавіту “AL” згенерувати, наприклад, використовуючи генератор ПВЧ, то вихідна послідовність стане іншою (табл. 2.4 – вихід2, алфавіт2). Не знаючи початковий стан алфавіту, неможливо дешифрувати повідомлення. При цьому реалізується шифрування методом багатоалфавітної підстановки і стиснення даних одночасно. Для одночасного стиснення і шифрування даних методом багатоалфавітної підстановки може використовуватись така схема:

- згідно з формулою (2.3) генерується алфавіт повідомлення. Оскільки символи в комп’ютерних системах представлені 8-и бітовими комбінаціями, то m=256, а константи a, c і породжувальне число T_0 можна вводити як ключ шифра.

- виконується стиснення згідно з приведеним методом

Додаткові затрати відсутні, оскільки генерація символів алфавіту виконується в будь-якому випадку.

При незначних додаткових затратах можливо застосувати комбінацію методів, яка включає два основні етапи:

- Етап1. Кодування за ступенем новизни для стиснення і шифрування методом багатоалфавітної підстановки.

- Етап2. Шифрування за допомогою додаткового генератора ПВЧ з m рівним довжині файлу, який отриманий в результаті виконання етапу 1. Цей або інший генератор ПВЧ може бути використаний також для виконання перестановок, оскільки він генерує всі числа в межах довжини початкового файлу. Якщо в стисненні даних немає потреби, то ця перестановка дозволить замаскувати стиснені дані в межах файлу такого ж розміру як і початковий, з попередньо записаною випадковою інформацією.

Аналіз таких відомих алгоритмів стиснення інформації як кодування Хаффмана та LZW показує також, що їх легко адаптувати до одночасного виконання стиснення і шифрування інформації. А при деяких додаткових обчисленнях згідно з формулою (2.3), ще і шифрування методом ПВЧ. Для цього достатньо лише сформувати таблицю перекодування кожного зчитаного з пам’яті символу. Ця таблиця може бути сформована, наприклад, за допомогою генератора ПВЧ (3) з m=256. При цьому зчитаний з початкового файлу символ даних задає позицію в таблиці перекодування, з якої вибирається код підстановки.

КОНТРОЛЬНІ ПИТАННЯ

1. Наведіть класифікацію алгоритмів стиснення без втрат.
2. Охарактеризуйте словарні методи стиснення зображень.
3. Охарактеризуйте статистичні методи стиснення зображень.
4. Дайте характеристику стиснення зображень методом RLE.
5. Які недоліки кодування Хаффмена?
6. Наведіть порядок побудови кодового дерева.
7. Що таке префіксні коди?
8. Наведіть алгоритм кодування зображень методом LZW.
9. Наведіть алгоритм декодування зображень методом LZW.
10. Поясніть основи арифметичного кодування.
11. Які недоліки алгоритму LZW?
12. Чи забезпечують алгоритми стиснення без втрат високий коефіцієнт стиснення зображень?
13. Який основний недолік арифметичних методів стиснення зображень?
14. На якому етапі кодування зображень застосовуються алгоритми стиснення без втрат?
15. Як пов’язані шифрування і стиснення даних?

РОЗДІЛ 3 АЛГОРИТМИ СТИСНЕННЯ ЗОБРАЖЕНЬ З ВТРАТАМИ

3.1 Кодування зображень відповідно стандартам JPEG та MPEG

Досягти великих коефіцієнтів стиснення, використовуючи один метод (крім фрактального), практично неможливо. Тому ефективне кодування зображень виконується із застосуванням декількох методів за декілька етапів. Ця концепція і покладена в основу стандартів, розроблених міжнародною організацією по стандартизації (ISO), які відомі як стандарти JPEG (Joint Photographic Expert Group) та MPEG (Motion Picture Experts Group). Стандарт JPEG призначений для стиснення нерухомих зображень, а MPEG - рухомих зображень[25-27].

Процес кодування за схемою JPEG ділиться на такі етапи (рис. 3.1):

1. Перетворення зображення в оптимальний кольоровий простір (тільки у випадку кодування кольорових зображень).
2. Субдискретизація компонент різницевих кольорових сигналів шляхом усереднення груп пікселів (тільки у випадку кодування кольорових зображень) .
3. Виконання ДКП для зменшення надлишковості даних зображення.
4. Квантування кожного блока коефіцієнтів ДКП із застосуванням вагових функцій оптимізованих з урахуванням сприйняття людиною.
5. Кодування результувальних коефіцієнтів із застосуванням статистичного кодування Хаффмана.

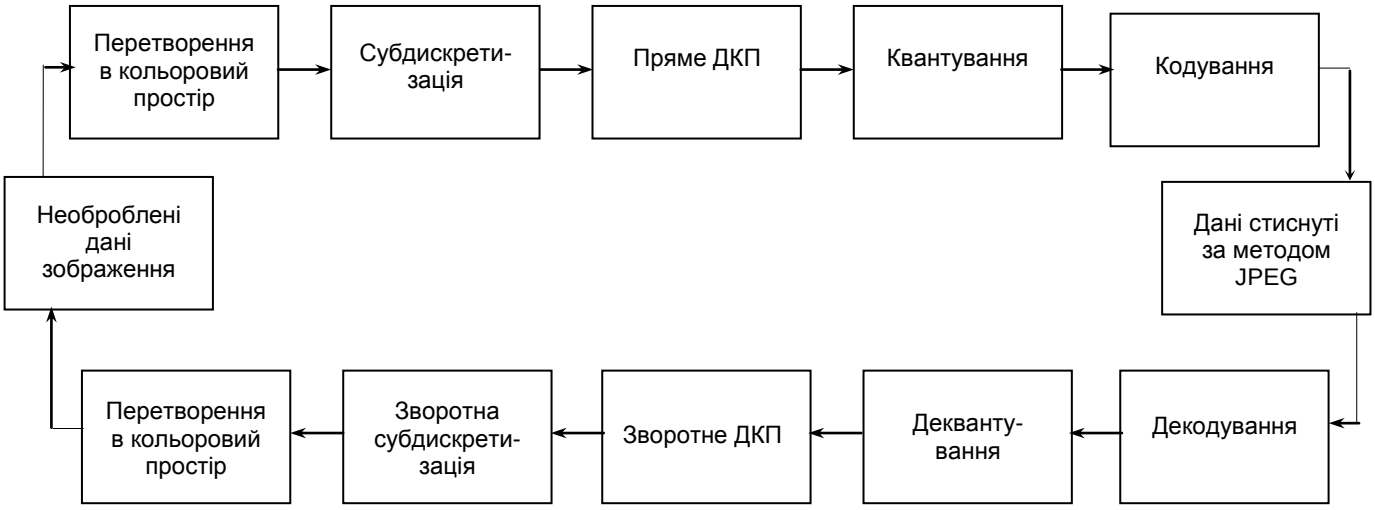


Рисунок 3.1-Послідовність операцій при стисненні зображень за методом JPEG

При виконанні першого етапу виконується перетворення кольорового зображення з системи представлення RGB в іншу систему, наприклад YUV, де Y – сигнал яскравості, U і V – різницеві кольорові сигнали.

Перетворення виконується від піксела до піксела за такими формулами:

$$Y = 0.3 \cdot R + 0.59 \cdot G + 0.11 \cdot B;$$
$$U = -0.15 \cdot R - 0.29 \cdot G + 0.44 \cdot B;$$
$$V = 0.62 \cdot R - 0.52 \cdot G - 0.1 \cdot B.$$

де *R, G, B* – кольорові сигнали зображення (червоний, зелений і синій відповідно).

Представлення кольорового зображення в системі YUV дає можливість використати особливості зорового сприйняття зображень людиною – низьку чутливість до точності представлення кольорів. Це виконується за рахунок субдискретизації компонент різницевих кольорових сигналів U і V шляхом усереднення груп пікселів (тільки у випадку кодування кольорових зображень). Наприклад, усереднення значень відліків в межах примикальних квадратів з розмірами сторін 2x2, зменшує розмір компонент U і V в чотири рази без помітного зменшення якості зображення. Враховуючи те, що на кожний піксел витрачається три байти (YUV), виграш значний.

Найбільш важким для реалізації є виконання ДКП. Однак, завдяки наявності швидких алгоритмів (ті ж самі, що для обчислення ДПФ) число арифметичних операцій може бути зменшено в десятки разів. Процес кодування із застосуванням ДКП пояснює рис.3.2.

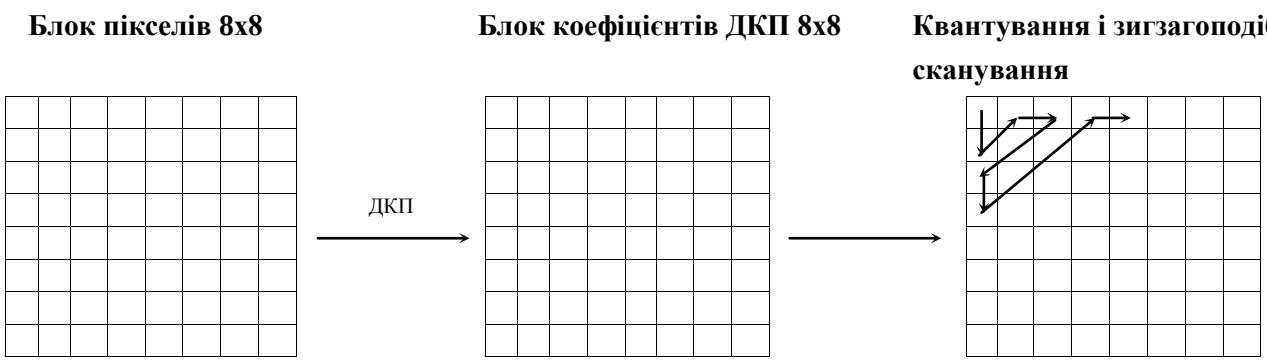


Рисунок 3.2 - Кодування з застосуванням ДКП

Зображення розбивається на примикальні один до одного блоки розміром 8x8 (при кодуванні кольорових зображень кожна компонента обробляється незалежно). В межах кожного блока виконується двовимірне ДКП у відповідності з виразом:

$$F(u,v) = \frac{1}{4} C(u)C(v) \sum_{I=0}^{7} \sum_{J=0}^{7} f(i,j) \cos(\frac{(2i+1)u\pi}{16}) \cos(\frac{(2j+1)v\pi}{16}), \quad (3.1)$$

де

$$C(x) = \begin{cases} \frac{1}{\sqrt{2}} & x = 0 \\ 1 & x <> 0, \end{cases}$$

u, v = 0,1,2 … 7.

При декодуванні обчислюється зворотне ДКП:

$$f(i,j) = \frac{1}{4} \sum_{U=0}^{7} \sum_{V=0}^{7} C(u)C(v)F(u,v) \cos(\frac{(2i+1)u\pi}{16}) \cos(\frac{(2j+1)v\pi}{16}), \quad (3.2)$$

i, j = 0,1,2 … 7.

Квантування виконується за рахунок ділення кожного коефіцієнта ДКП на свій "коефіцієнт квантування" з округленням результату до цілого. Терми більшого порядку квантуються з більшим "коефіцієнтом квантування". Крім того, для даних яскравості і кольору застосовуються різні таблиці квантування, оскільки око людини має різну чутливість до яскравості і кольору зображення.

На етапі статистичного кодування специфікація JPEG допускає застосування крім алгоритму Хаффмана і інших методів з метою зменшення об'єму інформації.

При кодуванні рухомих зображень відповідно стандарту MPEG застосовуються два типи стиснення: внутрікадрове кодування (подібно JPEG) і міжкадрове кодування. Міжкадрове кодування основано на кодуванні з передбаченням та інтерполяцією. Кадри рухомого зображення містять багато ідентичних даних. Якщо використати цей факт, то в результаті набагато збільшиться ступінь стиснення.

Для підтримки міжкадрового і внутрікадрового кодування потік даних MPEG містить три типи закодованих кадрів:

- I-кадри,
- P-кадри,
- B-кадри.

I-кадр містить один кадр відеоданих, який не пов'язаний з інформацією у будь-якому іншому кадрі. P-кадр містить відмінності між поточним та попереднім I- або P-кадром. B-кадр містить відмінності між поточним кадром і двома (попереднім і наступним) I- або P-кадрами. Типова послідовність кадрів в потоці MPEG виглядає приблизно так:

IBBPBBPBBPBBIBBP...(0123456789...).

Декодовані вони будуть в послідовності:

IPBBPBBPBB...(0312645978...).

А відображені в послідовності:

IBBPBBPBBP...(0123456789...).

I-, P-, B-кадри стискаються з використанням ДКП і статистичного кодування Хаффмана подібно JPEG.

Перша модифікація стандарту MPEG (MPEG-1) орієнтована на застосування в комп'ютерних системах для відтворення кольорових рухомих зображень в форматі 352x240 з частотою 30 кадрів за секунду. Наступні модифікації стандарту MPEG (MPEG-2, MPEG-4) розраховані на більшу роздільну здатність і можуть знайти застосування в цифровому телебаченні. Серед недоліків JPEG і MPEG кодування необхідно відзначити такі як:

1. Недостатні коефіцієнти стиснення складних зображень.
2. Неможливість зміни роздільної здатності.
3. Деяка втрата достовірності відновлених зображень.

Однак, незважаючи на це, ці методи поки що мають найкращі реально досягнуті характеристики по сукупності таких параметрів як коефіцієнт стиснення, якість, швидкодія і підтримуються основними виробниками комп'ютерної техніки і техніки зв'язку. Додаткову інформацію можна знайти в Internet, наприклад, за адресою *http://www.mpeg.org.*

3.2 Фрактальне стиснення зображень

З відомих методів кодування зображень фрактальний метод дозволяє отримувати найбільші коефіцієнти стиснення. З фізичної точки зору фрактальне кодування грунтується на твердженні, що зображення містить афінну надлишковість [5]. Математична модель, яка використовується при фрактальному стисненні зображень, називається системою ітерованих функцій (Iterated Function System – IFS). Система ітерованих функцій містить набір стискальних перетворень w_i, які можливо задати так:

$$\mathbf{W}(\cdot) = \bigcup_{i=1}^{n} w_i(\cdot) \qquad (3.3)$$

Хатчинсон [2] показав, що для вхідного зображення f_0 результат перетворення

$$f_\infty = \lim_{n\to\infty} \mathbf{W}^{0n}(f_0) \qquad (3.4)$$

не залежить від вибору f_0. Зображення f_∞ називається нерухомою точкою перетворення **W**. В якості перетворень w_i використовуються афінні перетворення:

$$w_i\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{bmatrix} a_i & b_i & 0 \\ c_i & d_i & 0 \\ 0 & 0 & S_i \end{bmatrix}\begin{bmatrix} x \\ y \\ z \end{bmatrix} + \begin{bmatrix} dx \\ dy \\ O_i \end{bmatrix} \quad (3.5)$$

де a_i, b_i, c_i, d_i – афінні коефіцієнти деформації, стиснення, обертання, dx, dy – коефіцієнти переміщення; x,y – координати точки, що перетворюється; z – її інтенсивність. Параметр S_i керує контрастністю, а O_i – яскравістю зображення. Знаючи коефіцієнти цих перетворень, ми можемо відновити початкове зображення.

3.2.1 Алгоритм кодування-декодування зображень фрактальним методом

Одна з можливих схем кодування зображень фрактальним методом запропонована Джеквіном (Jacquin) [15,18], і містить такі етапи:

- Зображення розділяється на області, що примикають одна до одної розміром nxn (рангові області).
- Задається набір доменних областей. Доменні області можуть перекриватись, вони не повинні обов'язково закривати всю поверхню зображення. Розміри доменних областей звичайно вибирають 2nx2n.
- Для кожної рангової області підбирається доменна область, яка після афінних перетворень найбільш точно апроксимує рангову область. На практиці застосовується вісім варіантів відображення одного квадрата в інший з використанням афінних перетворень, які приведені на рис. 3.3 [15-19]. Це повороти зображення на кути 90,180, 270 (-90) градусів відносно його центра і перетворення симетрії відносно ортогональних осей, які проходять через центр фрагмента перпендикулярно його сторонам.
- Точність апроксимації визначається за допомогою середньоквадратичного критерію:

$$F=\sum_{i,j}(Sd_{ij} + O_{ij} - r_{ij})^2 \quad (3.6)$$

де d_{ij} – значення, отримані в результаті усереднення по фрагментах з розмірами 2×2 елементів доменної області, що приводить її розмір до розміру рангової області; r_{ij} – значення елементів рангової області. Зміщення O_{ij} може бути як константою (як в даній роботі), так і описуватись поліномами першого, другого, третього порядків [3].

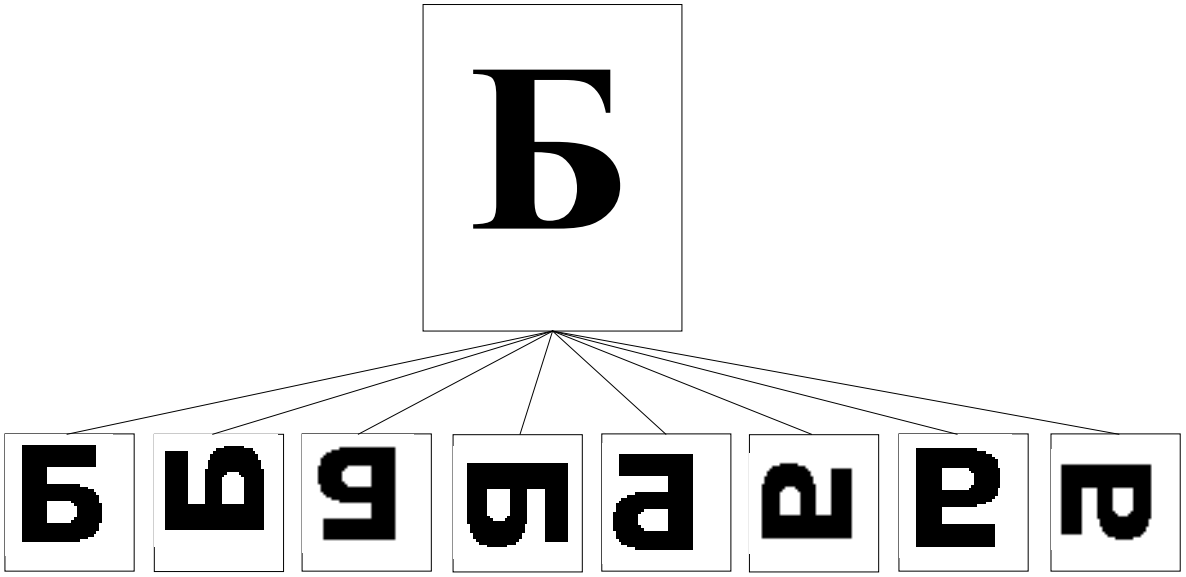


Рисунок 3.3 - Можливі варіанти відображення одного квадрата в інший з використанням афінних перетворень

Прирівнявши до нуля часткові похідні від виразу (3.6) по S і O:

$$\frac{\partial F}{\partial S} = 0, \qquad \frac{\partial F}{\partial O} = 0,$$

знайдемо значення S і O, при яких досягається мінімум виразу (3.6):

$$O = \frac{1}{n^2}\left(\sum_{i,j}^{n} r_{ij} - S\sum_{i,j}^{n} d_{ij}\right) \tag{3.7}$$

$$S = \frac{n^2\sum_{i,j}^{n} r_{ij}d_{ij} - \sum_{i,j}^{n} r_{ij}\sum_{i,j}^{n} d_{ij}}{n^2\sum_{i,j}^{n} d_{ij}^{\,2} - (\sum_{i,j}^{n} d_{ij})^2} \tag{3.8}$$

Домені блоки звичайно вибирають з кроком n/2 при n=4. У вихідний файл записуються такі параметри:

- координати доменної області з найменшим значенням Fmin;

- значення для O і S, отримані згідно з формулами (3.7, 3.8);
- номер афінного перетворення.

Процес фрактального перетворення асиметричний. Тобто, процес декодування не є простою інверсією процедур стиснення і вимагає значно менше часу для його виконання, що дає можливість уже тепер використовувати цей метод для зберігання зображень на компакт-дисках та інших носіях інформації. Декодування стисненого зображення носить ітераційний характер і складається з таких етапів:

- Створюються два зображення однакового розміру А і Б. Розмір цих зображень не обов'язково рівний розміру початкового зображення, початковий рисунок областей А і В будь-який.
- Зображення Б розбивається на рангові області так, як на першій стадії процесу стиснення. Для кожної рангової області зображення Б виконується афінне перетворення відповідної доменної області зображення А і результат поміщається в Б.
- Виконуються операції ідентичні попередньому пункту, тільки зображення А і Б міняються місцями.
- Багатократно повторюються другий і третій кроки до тих пір поки зображення А і Б не стануть нерозрізненими.

Основним недоліком фрактального методу є низька швидкість кодування, яка пов'язана з тим, що для отримання високої якості зображення для кожного рангового блока необхідно виконати перебір всіх доменних блоків, і для кожного доменного блока необхідно виконати не менше восьми афінних перетворень.

В цифровій обробці сигналів швидкодія методу оцінюється кількістю арифметичних операцій, необхідних для виконання перетворення. Оцінимо, наприклад, число операцій множення при кодуванні зображень фрактальним методом (вважаючи, що S=1), оскільки для їх виконання витрачається найбільше часу. Нехай MxN розміри зображення і $M=N=n^{к}$, де n - розмір сторін рангового блока. Нехай крок вибору доменних блоків n/2. Для порівняння даного рангового блока з доменним блоком необхідно виконати $8n^2$ операцій множення. Тоді

загальна кількість операцій множення для пошуку відповідного доменного блока така:

$$M = 8n^2 * \text{(кількість доменних блоків)},$$

Кількість доменних блоків при кроці вибору n/2 складе:

$$(\frac{n^k - 2n}{n/2} + 1) * (\frac{n^k - 2n}{n/2} + 1),$$

Після підстановки кінцева формула має такий вигляд:

$$M = 8(4n^{k+1}(n^{k-1} - 3) + 9n^2).$$

Для порівняння, виконання ДКП для блока такого ж розміру, навіть без використання швидких алгоритмів, вимагає лише n^4 операцій множення.

Незважаючи на те, що виконані в останні роки роботи дозволили зменшити час кодування в десятки раз [15,19], актуальним залишається проведення досліджень з підвищення швидкості стиснення зображень цим методом в таких напрямках:

- пошук критеріїв, які дозволяли б швидко виконувати відбір доменних блоків без перебору всіх афінних перетворень;
- зменшення числа афінних перетворень.

3.2.2 Підвищення швидкодії фрактального стиснення

Найпростіший і найповільніший спосіб фрактального кодування є перевірка кожного доменного блока і виконання обчислень згідно з формулами (3.6-3.8). Такий спосіб називається **повним пошуком або повним перебором.** При кодуванні зображень природного походження можна підвищити швидкодію кодування, прийнявши S=1, оскільки враховуючи статистику зображень завжди знайдеться доменний блок, який апроксимує заданий ранговий блок з необхідною точністю. Тоді з виразів (3.6, 3.7) одержимо:

$$F = \sum_{i,j} (d_{ij} + O_{ij} - r_{ij})^2 \qquad (3.9)$$

$$O = \frac{1}{n^2} \left(\sum_{i,j}^{n} r_{ij} - \sum_{i,j}^{n} d_{ij} \right) \qquad (3.10)$$

Контрастність декодованого зображення може бути відновлена після декодування іншими методами. Таке спрощення дозволяє знизити кількість арифметичних операцій на 60% і відповідно підвищити швидкість кодування.

Однак, такий спосіб підвищення швидкодії не є коректним з тієї точки зору, що вже в самому алгоритмі передбачена візуально недостовірна передача деяких графічних зображень штучного походження, тому необхідні додаткові заходи для забезпечення хоча би візуальної достовірності.

Серед відомих способів підвищення швидкодії кодування зображень фрактальним методом можна виділити такі [15, 18, 19]:

- пошук доменних блоків, для яких F не перевищує заданого значення;
- локальний та сублокальний пошук;
- ізометричне передбачення;
- класифікація доменних і рангових блоків. Ранговий блок порівнюється з доменними блоками того ж самого класу.

Кодування відповідно першого підходу пояснює рис. 3.4.

```
* Вибрати прийнятний рівень Ec і установити розмір рангових
  блоків Ri  n=nmax.
* Помітити блоки Ri як непокриті;
* Поки є непокриті Ri {
    * Знайти Di і відповідне перетворення wi яке дає
    мінімум F;
    * Якщо F< Ec або розмір Ri n ≤ nmax, то
        * Відмітити Ri як покритий і записати параметри wi ;
    * Інакше
        * Розділити Ri на менші блоки і помітити їх як непокриті
}
```

Рисунок 3.4 - Пошук доменних блоків з заданим F

Тобто, передбачається перемінний розмір доменних і рангових блоків. При постійному розмірі доменних і рангових блоків n = const пошук продовжується до тих пір поки не буде знайдений доменний блок, який забезпечує $F< E_c$ [18].

Локальний пошук (Local Search) передбачає пошук доменного блока тільки на ділянках зображення сусідніх з даним ранговим блоком.

Збільшити швидкість кодування можливо також, збільшивши крок вибору доменних блоків, що зменшує простір пошуку. Однак, дуже великий крок веде до зниження якості зображення. Тому спочатку пошук виконується з великим кроком, і коли знаходиться блок з мінімальним значенням F_{min}, то пошук продовжується навколо цього блока з меншим кроком (рис.3.5). Такий спосіб носить назву **локальний субпошук** (Local Sub Search – LSS).

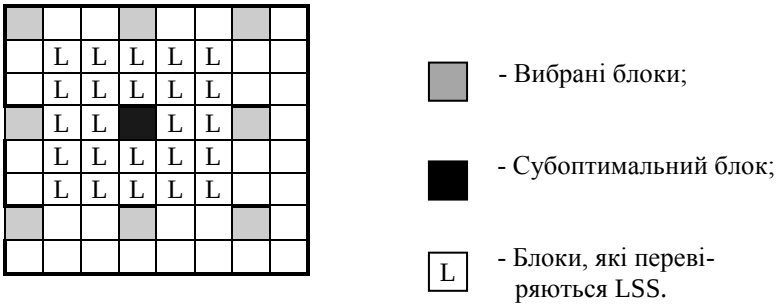


Рисунок 3.5 - Локальний субпошук

Кількість афінних перетворень зменшується за рахунок **ізометричного передбачення.** Кожний доменний (ранговий) блок розділяється на чотири субблоки, обчислюється середнє значення яскравості в межах кожного субблока і виконується сортування цих субблоків в порядку зростання. Порівнюючи сортований і несортований списки, обчислюється лексикографічний порядок перестановок [19]:

$$\text{Perm \#} = \text{inv}(x_1)\cdot 3! + \text{inv}(x_2)\cdot 2! + \text{inv}(x_3)\cdot 1! + \text{inv}(x_4)\cdot 0!, \qquad (3.11)$$

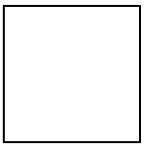
де $\text{inv}(x_i)$ - кількість перестановок, яка відноситься до і-ої позиції (рис. 3.6).

Тепер кожний блок має свій лексикографічний порядок, що дає можливість визначити найкраще покриття рангового блока.

Серед методів з класифікацією необхідно відзначити класифікацію, запропоновану Джеквіном [19], яка грунтується на топології блоків і передбачає блоки трьох типів:

- блоки без контурів (shade blocks);
- блоки, інваріантні до орієнтації (текстурні блоки);
- контурні блоки (edge blocks).

Shade-блоки можуть бути апроксимовані за допомогою поліномів третього або другого порядку, і пошук для них непотрібен, пошук текстурних блоків виконується без перевірки ізометрії (без виконання афінних перетворень). Тільки для контурних блоків виконується повний перебір.



2	3
0	1

[0,1,2,3]

L (#) - середня яскравість субблоку

L (2) > L (0) > L (1) > L (3)

[0,1,2,3] → [2,0,1,3]

Perm # = 2· 3! + 1· 2! + 1· 1! + 0· 0! = 15.

Рисунок 3.6 - Приклад обчислення лексикографічного порядку

3.2.3 Табличний метод підвищення швидкодії

Об'єм пам`яті та швидкодія комп'ютерів кожний рік подвоюються. Це дозволяє очікувати, що стане можливим застосування табличних методів пошуку доменних блоків без попередньої обробки зображень і пошуку критеріїв класифікації доменних блоків.

Доменний блок можна представити як вектор з n^2 координатами

$$\vec{D}(d_1, d_2, ..., d_{n^2}),$$

де d_i - значення яскравості елементів блока, n - розмір сторін рангового блока. Після виконання афінних перетворень утворюються ще 7 векторів, координати яких є перестановками координат початкового вектора. Якщо для представлення елемента d_i витрачається 8 біт, то розрядність числа $d_1 d_2 ... d_{n^2}$ буде рівною 2^{8n^2}.

Формуються таблиці, представлені на рис. 3.7. Як видно з рисунка, для значення координати рангового блока «k» вибираються доменні

блоки з номерами m та m1, для яких необхідно обчислити тільки значення F, S та O згідно з виразів (3.6 - 3.8). Зрозуміло, що розміри таблиці 1 на рис. 3.7 не дозволяють поки що реалізувати такий підхід на практиці без попередньої обробки доменних блоків для скорочення розмірності представлення елементів цих блоків.

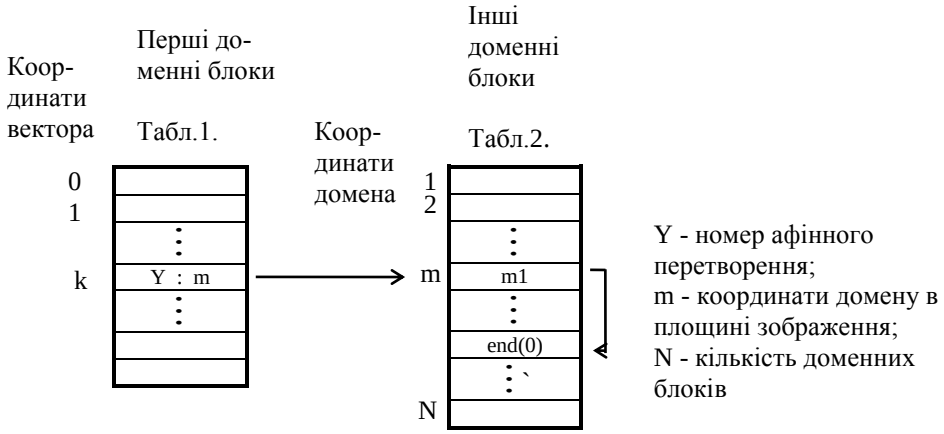


Рисунок 3.7-Табличний метод пошуку

Підвищення швидкодії фрактального стиснення грунтується на попередній обробці кожного доменного і рангового блока одним з операторів виділення контурів зображення і оцінці детальності кожного з цих блоків [5, 6]. Найбільші переваги має використання фільтра Лапласа, оскільки він точно ідентифікує контури і в порівнянні з градієнтними операторами забезпечує менші розміри таблиць пошуку:

$$G(i,j)= 4 f (i, j) - f (i-1, j) - f (i+1, j) - f (i, j-1) - f (i, j+1). \quad (3.12)$$

Де f(i,j) – значення відліків вхідного зображення. Обробка блоків згідно з виразом (3.12) виключає з порівняння постійну складову зображення і тому дозволяє попередньо відібрати близькі до даного рангового блока доменні блоки.

Під час першого проходу оцінюється детальність доменних і рангових блоків. Такою оцінкою може бути сума модулів значень відліків контурної компоненти:

$$S_{k_D} = \frac{1}{n^2}\left(\sum_{i=0}^{n}\sum_{j=0}^{n}|G_D(i,j)|\right)$$

$$S_{k_R} = \frac{1}{n^2}\left(\sum_{i=0}^{n}\sum_{j=0}^{n}|G_R(i,j)|\right)$$

де k_R – номер рангового блока, k_D - номер доменного блока, S_{k_D}, S_{k_R} - детальність відповідно доменного та рангового блока. Суми S_{k_R} заносяться в масив, в порядку слідування рангових блоків, а для доменних блоків формуються таблиці, приведені на рис. 3.8.

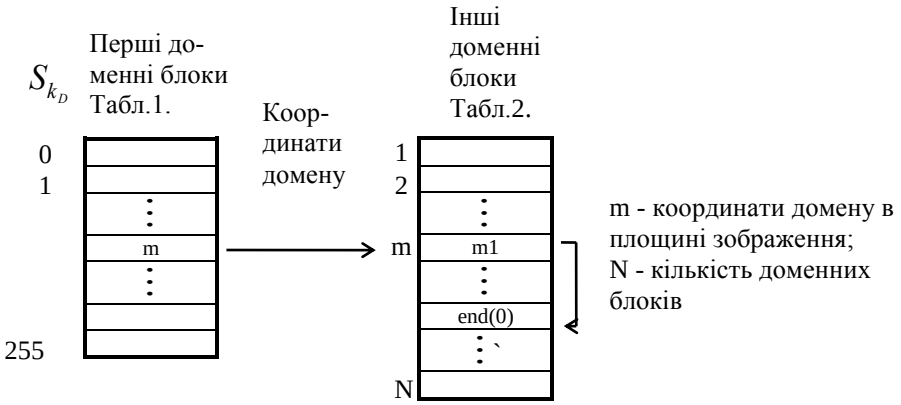


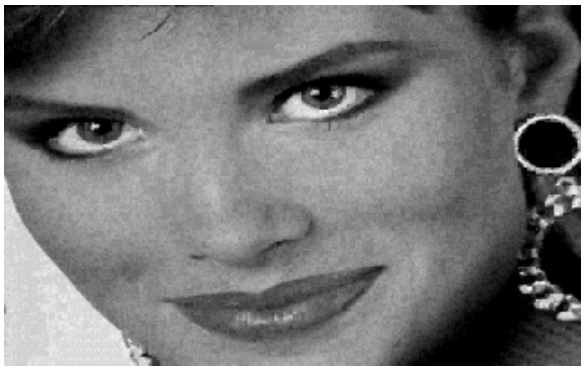
Рисунок 3.8 - Пошук доменного блока згідно з критерієм детальності

Під час другого проходу для кожного значення детальності рангового блоку S_{k_R} з таблиць 1, 2 (рис. 3.8) відбираються відповідні доменні блоки, для яких виконується обробка, характерна для фрактального стиснення та обчислення згідно з виразами (3.6-3.8). Оскільки вибраних блоків значно менше загальної кількості доменних блоків, то слід очікувати значного виграшу в швидкодії.

Приклад декодування зображення фрактальним методом приведений на рис. 3.9.

Початкове зображення

Декодоване зображення - три ітерації



Декодоване зображення - 8 ітерацій

Декодоване зображення - 16 ітерацій

Рисунок 3.9 - Декодування зображення фрактальним методом

3.3 Рекурсивний (хвильовий) алгоритм стиснення зображень

Англійська назва рекурсивного стиснення – wavelet. Цей алгоритм орієнтований на стиснення кольорових і чорно-білих зображень з плавними переходами, ідеальний для картинок типу рентгенівських фотографій. Коефіцієнт стиснення варіюється в межах 5-100. При великих коефіцієнтах стиснення на різких границях, особливо діагональних, можливі спотворення [6].

Основна ідея алгоритму полягає в тому, що зберігаються різниці між середніми значеннями сусідніх блоків зображення, які звичайно приймають значення близькі до нуля.

Рекурсивне стиснення базується на пірамідальному S-перетворенні, яке може використовуватись для стиснення фотореалістичних зображень як майже без втрат, так і з втратами.

Стиснення виконується за два кроки: перший – S - перетворення початкового зображення; другий - перетворені дані кодуються одним з статистичних методів. Обидві операції зворотні, що дозволяє відновити початкове зображення. Однак для отримання великих коефіцієнтів стиснення необхідно зниження точності представлення компонент зображення, отриманих в результаті виконання S-перетворення, але так щоб спотворення не були візуально помітні.

Структура кодування приведена на рис. 3.10. Вхідні відеодані обробляються фільтрами S-перетворення, які формують компоненти зображення. Адаптивний квантувач призначений для квантування значень відліків компонент з урахуванням особливостей зорового сприйняття. Процес виконання квантування пов'язаний з втратами інформації, однак ці втрати повинні бути непомітними, тобто візуальна якість відновленого зображення не повинна відрізнятися від вхідного.

Власне стиснення може бути виконано на основі одного з методів статистичного кодування (наприклад, кодування Хаффмана, LZW-кодування, арифметичного кодування).

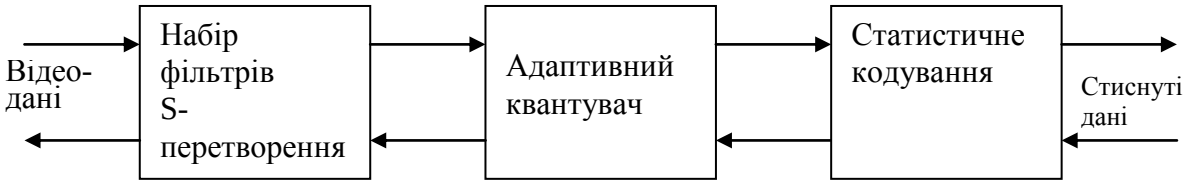


Рисунок 3.10 - Схема хвильового кодування

Основна схема пірамідального S-перетворення приведена на рис. 3.11, а зворотного перетворення на рис. 3.12. Відліки низько- (фільтр F1) і високочастотної (фільтр F2) складової одержують відповідно до таких виразів:

$$\begin{aligned} L(n) &= \lfloor C(2*n) + C(2*n+1) \rfloor / 2 \\ H(n) &= \lceil C(2*n) - C(2*n+1) \rceil / 2 \end{aligned}. \qquad (3.13)$$

Де n= 0, 1... (N/2 – 1), C [2*n], C [2*n + 1] – відліки вхідного цифрового сигналу, а L[n], H[n] – низькочастотні і високочастотні коефіцієнти S-перетворення (рис. 3.11), L[n] – відліки на виході низькочастотного фільтра, H[n] – відліки на виході високочастотного фільтра. Ділення на два може бути виконано зсувом вправо на один біт, що дає результат з округленням до меншого.

До кожного вихідного компонента фільтрів знову застосовуються дані формули, утворюючи в такий спосіб піраміду. Ясно, що чим більша кількість компонентів, тим більший очікуваний коефіцієнт стиску.

Перевагою даного представлення вихідного зображення є те, що дисперсія H[n] менша ніж для C [n].

Інверсне S-перетворення обчислюється в такий спосіб:

$$\begin{aligned} C(2*n) &= L(n) + H(n), \\ C(2*n+1) &= L(n) - H(n) \end{aligned}. \qquad (3.14)$$

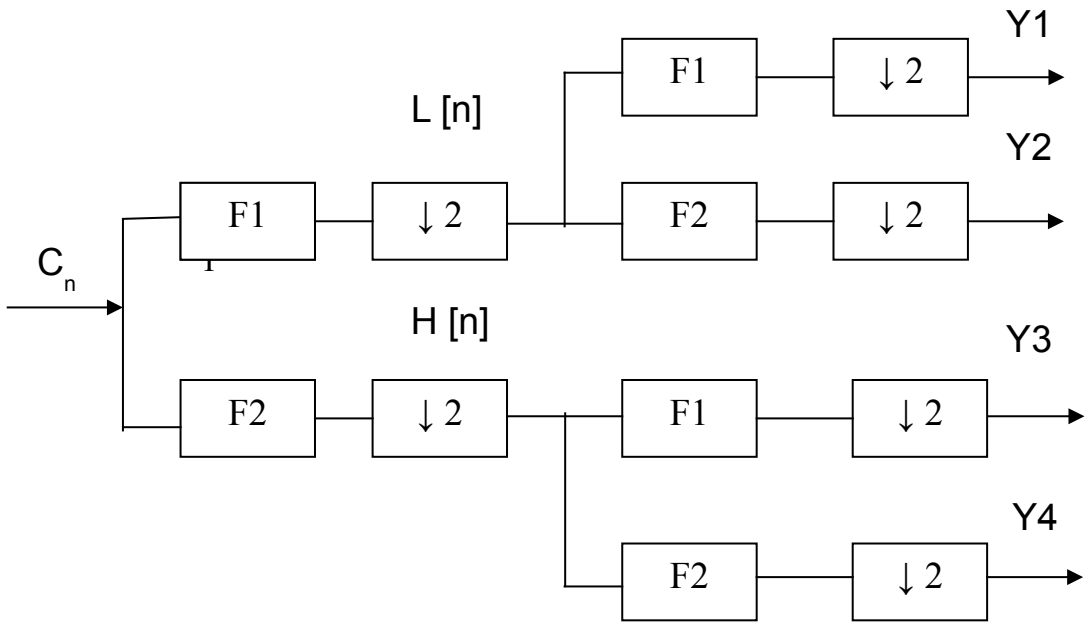


Рисунок 3.11- Пряме S - перетворення

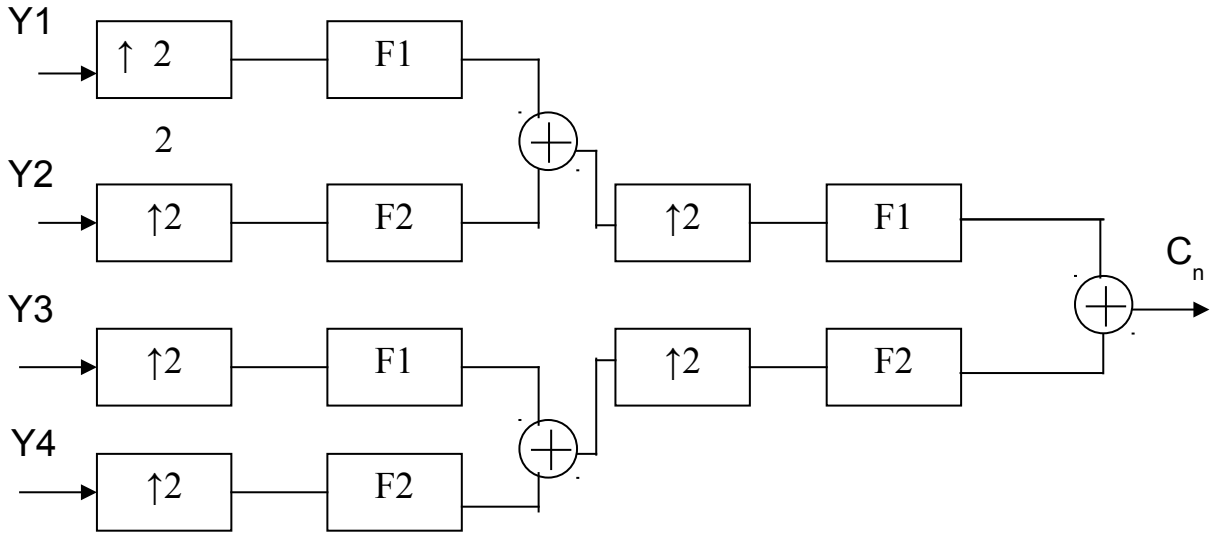
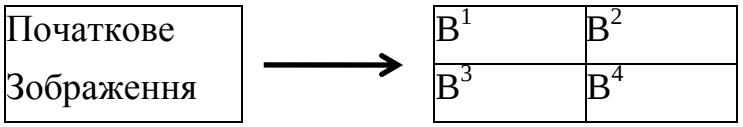


Рисунок 3.12 - Зворотне S - перетворення

Алгоритм для двовимірних даних реалізується аналогічно. Для квадрата з розмірами 2x2 і значеннями елементів $a_{2i,2j}$, $a_{2i+1,\,2j}$, $a_{2i,\,2j+1}$, $a_{2i+1,\,2j+1}$ отримаємо

$$\begin{aligned} b^1_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4 \\ b^2_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4 \\ b^3_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4 \\ b^4_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4 \end{aligned} \qquad (3.15)$$

Використовуючи ці формули, наприклад, для зображення з розмірами 512x512 пікселів після першого перетворення одержимо 4 матриці розміром 256x256 елементів:



У першій, як легко догадатися, буде зберігатися зменшена копія зображення. В другій — усереднені різниці пар значень пікселів по горизонталі. У третій — усереднені різниці пар значень пікселів по вертикалі. У четвертій — усереднені різниці значень пікселів по діагоналі.

За аналогією з двовимірним випадком, ми можемо повторити наше перетворення й одержати замість першої матриці 4 матриці розміром 128x128. Повторивши наше перетворення втретє, ми одержимо в підсумку: 4 матриці 64x64, 3 матриці 128x128 і 3 матриці 256x256. На практиці, при записі у файл, значення $b^4_{i,j}$ звичайно не враховують, відразу одержуючи виграш приблизно на третину розміру файлу.

Можливі варіанти розкладу зображення на компоненти приведені на рис. 3.13 – 3.20.

До достоїнств цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступового “прояву” зображення при передачі зображення по мережі. Крім того, оскільки на початку зображення ми фактично зберігаємо його зменшену копію, спрощується показ “огрубленого” зображення по заголовку.

Двовимірне 2-D перетворення може бути також виконане за допомогою застосування формул (3.13) для одновимірного перетворення послідовно до рядків і стовпців зображення. Це зменшує обчислювальні затрати необхідні для виконання цього перетворення. Сигнальний граф швидкого двовимірного Sперетворення для фрагмента 2x2, приведений на рис. 3.21.

Зображення, які ілюструють етапи Wavelet-кодування приведені на рис. 3.22-3.25.

На відміну від JPEG і фрактального алгоритму, S-перетворення не оперує з блоками, наприклад, 8x8 пікселів, а оперує блоками 2x2, 4x4, 8x8 і т.д. Однак за рахунок того, що коефіцієнти для цих блоків ми зберігаємо незалежно, ми можемо досить легко уникнути дроблення зображення на “мозаїчні” квадрати.

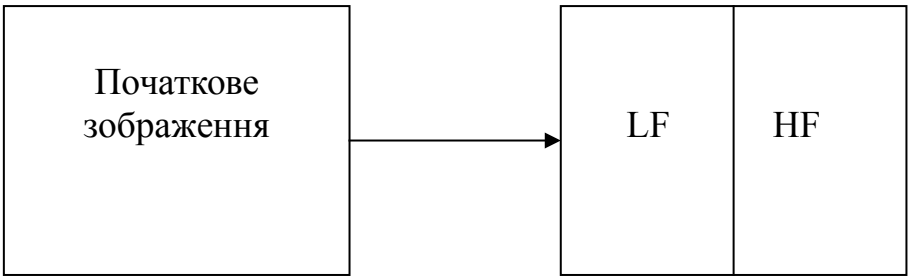


Рисунок 3.13 - Горизонтальне двокомпонентне кодування. LF – низькочастотні коефіцієнти S-перетворення, HF – високочастотні коефіцієнти

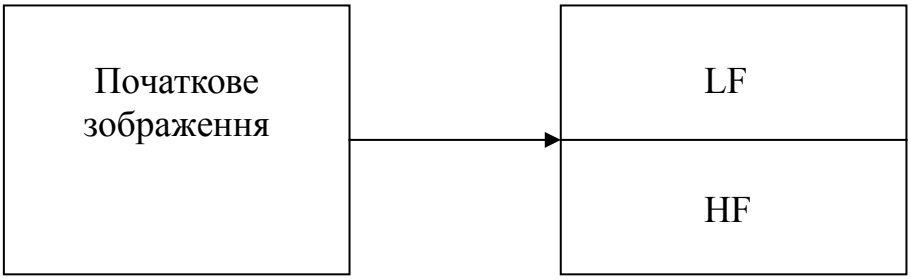


Рисунок 3.14 - Вертикальне двокомпонентне кодування. LF – низькочастотні коефіцієнти S-перетворення, HF – високочастотні коефіцієнти

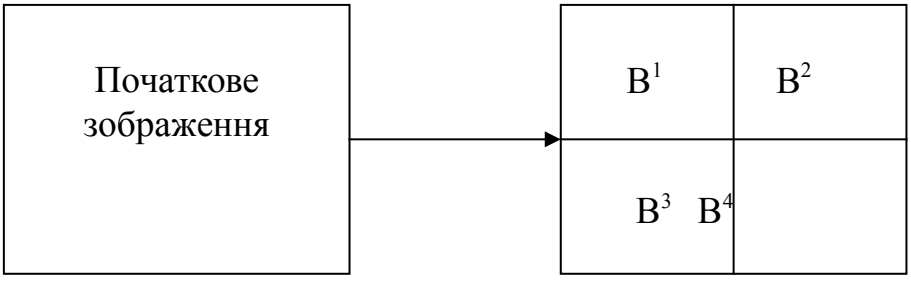


Рисунок 3.15 - Двовимірне чотирикомпонентне кодування

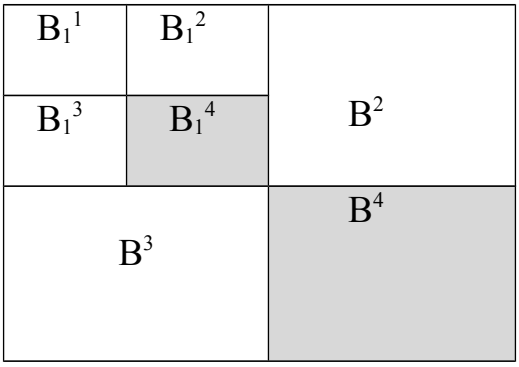


Рисунок 3.16 - Двовимірне семикомпонентне кодування

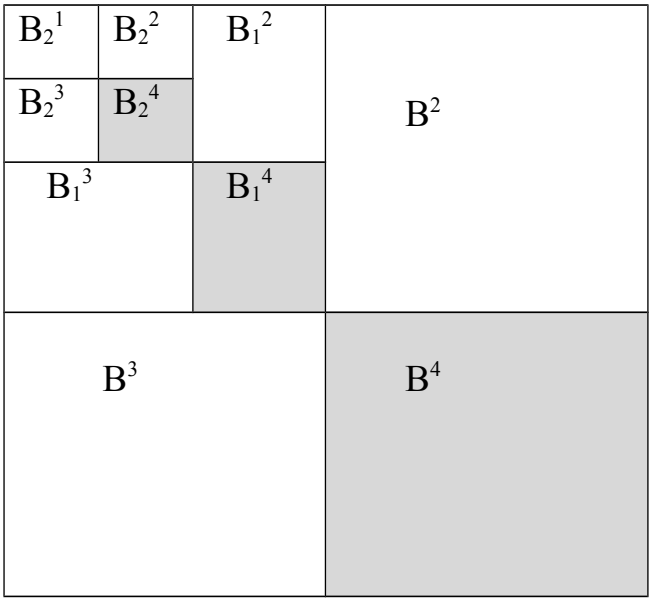


Рисунок 3.17 - Двовимірне десятикомпонентне кодування

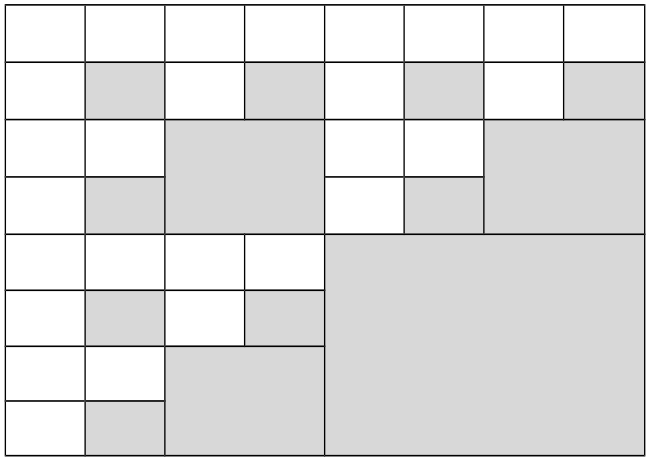
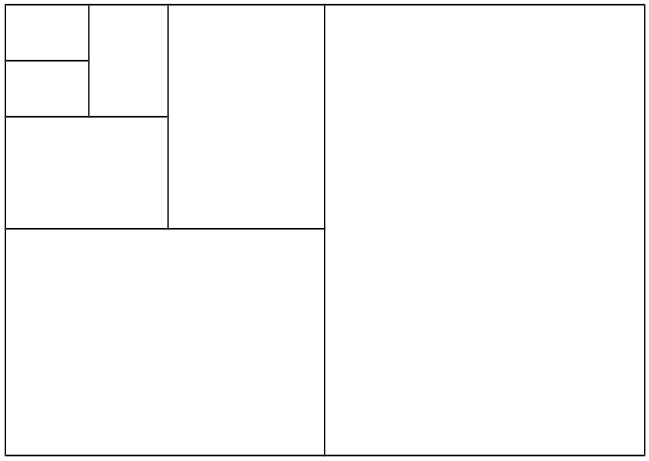
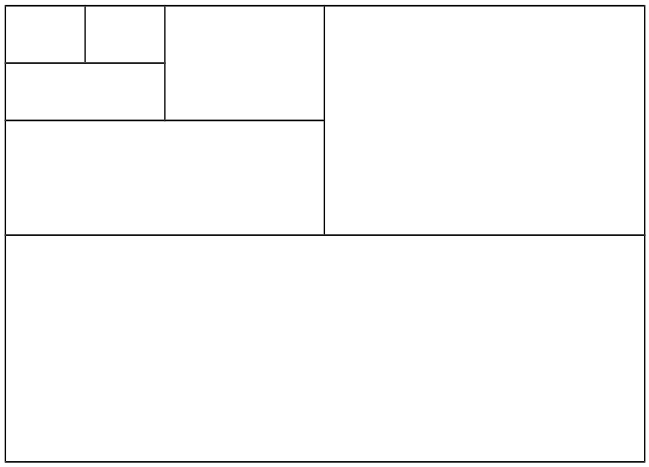


Рисунок 3.18 - Двовимірне 40-компонентне кодування



а



б

Рисунок 3.19 - Альтернативні варіанти виконання S-перетворення

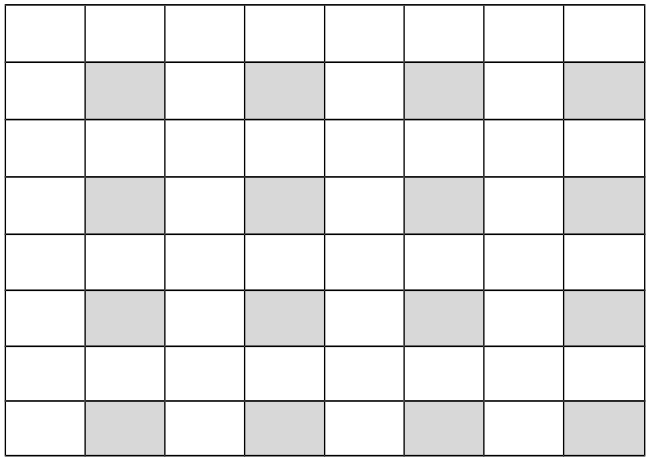


Рисунок 3.20 – Повний 64-компонентний розклад

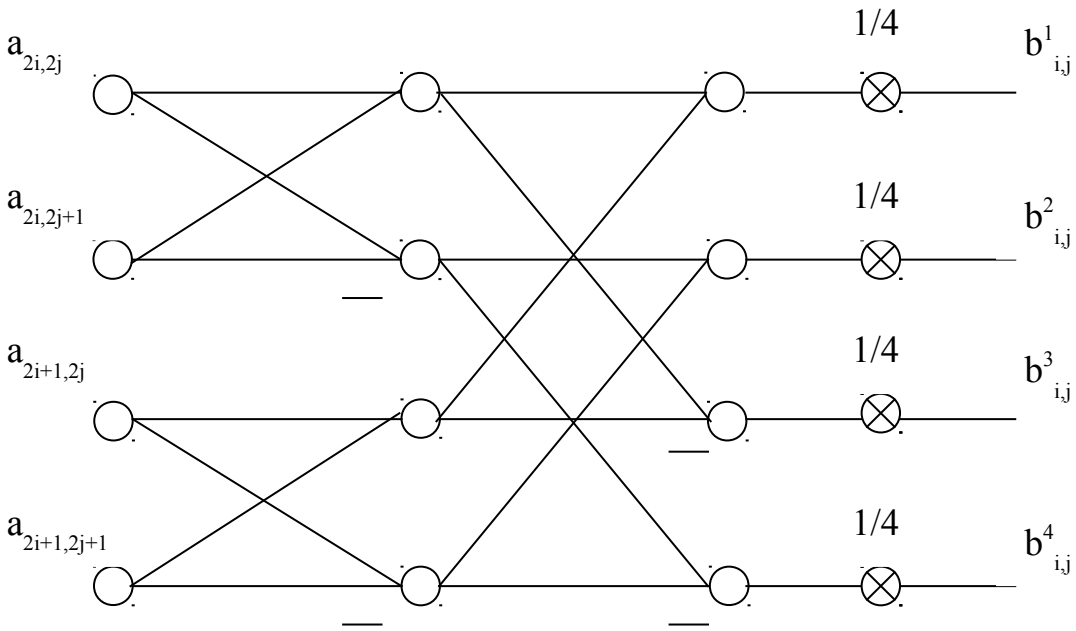


Рисунок 3.21 - Сигнальний граф швидкого двовимірного S-перетворення для фрагмента 2x2

Рисунок 3.22 - Компоненти зображення

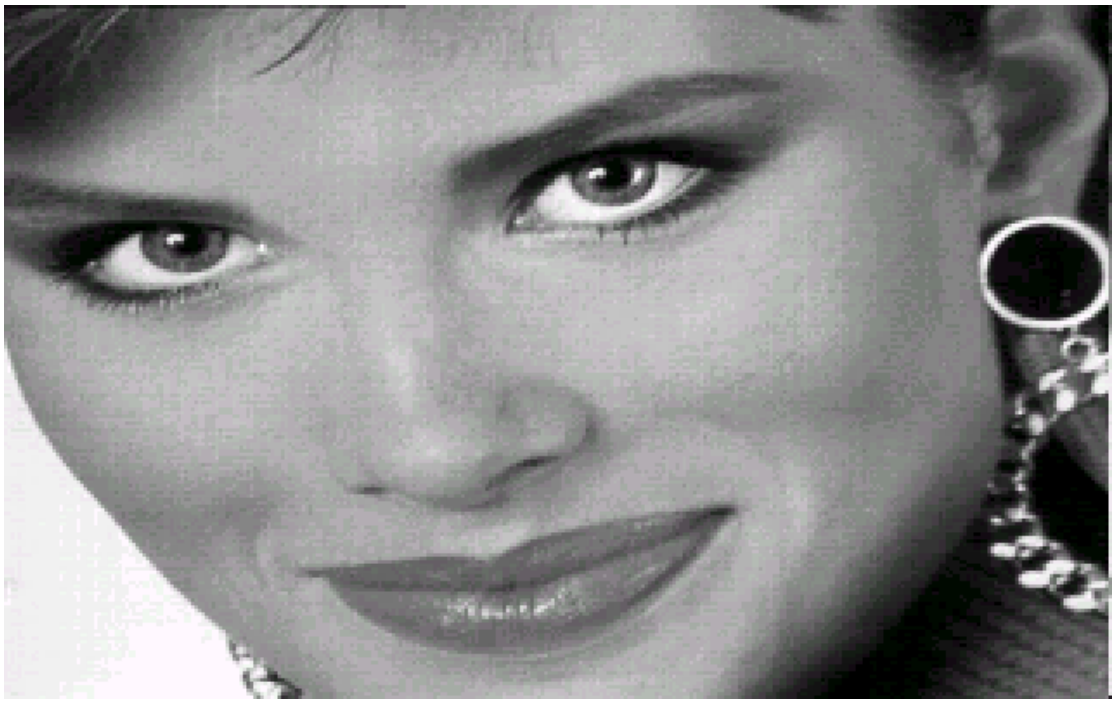


Рисунок 3.23 – Відновлене зображення майже без втрат

Рисунок 3.24 - Розклад без компоненти B^4

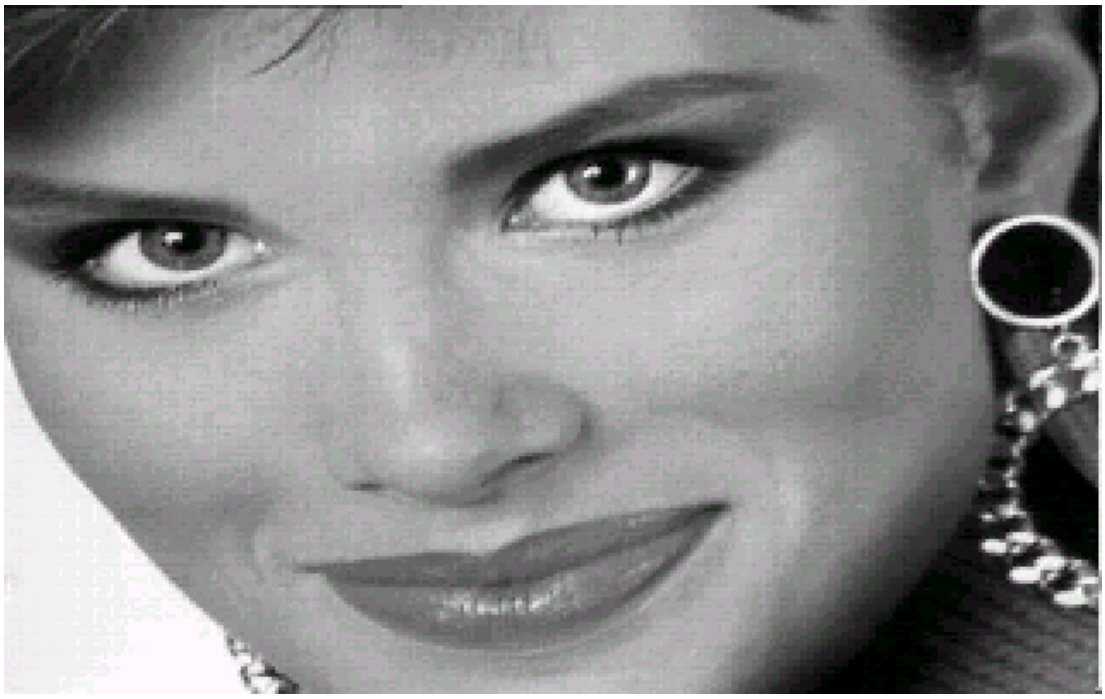


Рисунок 3.25 - Відновлене зображення без компоненти B^4

3.4 Перспективи розвитку кодування зображень

Розвиток кодування зображень тісно пов'язаний з двома основними моментами:

1. Ростом швидкодії та ступеня інтеграції електронної елементної бази. Сучасні мікропроцесори з тактовими частотами 200 МГц (і вище) і спеціальними апаратними засобами для підвищення швидкодії обробки зображень(наприклад, процесор сімейства Pentium з технологією MMX - Pentium II [28]) дозволяють реалізувати навіть програмним способом з прийнятною швидкодією ті методи кодування зображень, які раніше вважалися складними для промислового застосування. Прикладом може бути дискретне косинусне перетворення, яке стало основою стандартів JPEG та MPEG. Але це не означає, що буде припинено пошук та дослідження більш простих методів кодування, оскільки нові можливості формують нові більш складні задачі, тому зменшення обчислювальної складності залишиться актуальним.
2. Зниженням собівартості електронної елементної бази. Проведення досліджень методів кодування рухомих і нерухомих зображень завжди вимагало складної і дорогої обчислювальної техніки, доступ до якої мала незначна кількість спеціалістів. Поява недорогих персональних комп'ютерів з розвинутими засобами відображення графічної інформації створила передумови збільшення інтенсивності досліджень, в яких може брати участь більш широке коло дослідників, що дає надію на нові значні результати в майбутньому.

З врахуванням цього можна зробити такі висновки:

1. В найближчі роки стандарти MPEG знайдуть найширше застосування в цифровому телебаченні.
2. Великі зусилля будуть спрямовані на дослідження фрактального методу та його модифікацій, що можливо сприятиме появі нового стандарту ISO, основою якого буде фрактальний метод кодування зображень.
3. Будуть проводитись пошуки та дослідження нових методів кодування, які можуть забезпечити високий коефіцієнт стиснення і високу якість відновленого зображення, незважаючи на їх обчислювальну складність. З іншого боку будуть продовжені пошуки простих методів кодування, будуть досліджуватись комбінації цих методів з урахуванням нових технічних можливостей.

КОНТРОЛЬНІ ПИТАННЯ

1. Наведіть етапи кодування зображень згідно з стандартом JPEG.
2. Чому виконується перетворення із системи RGB в YUV?
3. Наведіть математичні вирази для прямого і зворотного ДКП.
4. Чому зображення розбивається на квадрати 8x8?
5. Які відмінності між MPEG і JPEG?
6. Які зображення стискає метод JPEG?
7. Які зображення стискає метод MPEG?
8. Які методи застосовуються при стисненні рухомих зображень?
9. Які недоліки MPEG і JPEG?
10. Наведіть алгоритм стиснення зображень фрактальним методом?
11. Який основний недолік фрактального методу?
12. Наведіть алгоритм декодування зображень фрактальним методом.
13. Охарактеризуйте основні методи підвищення швидкодії фрактального стиснення зображень.
14. Охарактеризуйте Wavelet-кодування.
15. Наведіть алгоритм кодування зображень методом Wavelet.
16. Які недоліки Wavelet-кодування?
17. Охарактеризуйте перспективи розвитку кодування зображень.

ЛІТЕРАТУРА

1. Цифровое телевидение /Кривошеев М.И., Виленчик С.Л., Красносельский И.Н. и др.; Под ред. М.И. Кривошеева. - М.: Связь, 1980. - 264 с.
2. Цифровое кодирование телевизионных изображений /Под ред. И.И.Цуккермана. - М.: Радио и связь, 1981. -240 с.
3. Птачек М. Цифровое телевидение. Теория и техника: пер. с чешск. - М.: Радио и связь, 1990. - 528 с.
4. МККР. Цифровое кодирование телевизионных изображений в студиях. Рекомендация 601. 14-я Пленарная Ассамблея (Дубровник, 1986).
5. Барнсли М., Ансон Л. Фрактальное сжатие изображений//Мир ПК.1992. - № 10. - С. 52-58.
6. Ватолин Д.С. Алгоритмы сжатия изображений. – М.: Издательский отдел факультета Вычислительной Математики и Кибернетики МГУ им. М.В.Ломоносова (лицензия ЛР № 040777 от 23.07.96), 1999 г. - 76 с.
7. Харатишвили Н.Г. Цифровое кодирование с предсказанием непрерывных сигналов. - М.: Радио и связь, 1986. - С. 3-25.
8. Прэтт У. Цифровая обработка изображений: Пер. с англ. - М.: Мир, 1982. - Т.2. - 480 с.
9. Бабак та ін. Обробка сигналів /В.П.Бабак, В.С. Хандецький, Е. Шрюфер. - К.: Либідь, 1996. - 392 с.
10. Быстрые алгоритмы в цифровой обработке изображений/Т.С. Хуанг и др.; Под ред. Т.С. Хуанга: Пер. с англ. - М.: Радио и связь, 1984. - 224 с.
11. Кунт М., Джонсон О. Блочное кодирование графических материалов. Обзор//ТИИЭР. - 1980. - Т.68 N 7. - С. 21-40.
12. Брауде-Золотарев Ю.М., Кожемяко В.П., Майданюк В.П. Особенности кодирования цифровых телевизионных сигналов вещательного телевидения//1-я Всесоюзная конференция "Распознавание образов и анализ изображений: новые информационные технологии": Тез. докл. - Минск, 1991. - Ч.1, С. 31-36.
13. Майданюк В.П. Разработка алгоритмов и аппаратурных средств систем сжатия телевизионных изображений.: Дис. канд. техн. наук. Винница, 1993. - 223 с.

14. Кожем’яко В.П., Майданюк В.П., Жуков К.М. Аналіз та перспективи розвитку кодування зображень// Вісник ВПІ, 1999, № 3. – С. 42-48.
15. Behnam Bani-Egbal. Speeding-Up fractal Image Compression. Internet, email: behnam @ cs.man.ac.uk., 1994.
16. Збарянский С. Фрактальное сжатие изображений//Компьютеры + программы. - 1997. - № 6. - С. 16-22.
17. Edward R. Vrscay. A Generalized Class of Fractal-Wavelet Transforms for Image Representation and Compression. Internet, e-mail: evrscay@links. uwaterloo.ca, 1998.
18. Fisher Y. Fractal Image Compression. SIGGRAPH 92 Course Notes. e-mail: *yfisher@ucsd.edu*.
19. Pulcini G., Verrando V., Rossi R. Fractal Image Compression through Iterated Function Systems. *www: http://www.webcom.com.*
20. Кричевский Р.Е. Сжатие и поиск информации. - М.: Радио и связь, 1989. - 168 с.: ил.
21. Ведев Д. Компрессия данных при организации удаленного доступа к компьютерным сетям. *http://www.chat.ru/~escoman/arc_main.htm.*
22. Новосельский А. Алгоритмы шифрования // Компьютеры + программы 1996 - № 5.- С. 70-77.
23. Хоффман Л. Современные методы защиты информации: Пер. с англ. - М.: Сов. радио, 1980. - 264 с.
24. Защита информации в персональных ЭВМ/ Спесивцев А.В., Вегнер В.А., Крутяков А.Ю. и др.- М.: Радио и связь, НП «Весна», 1992.
25. Мюррей Дж.Д., Райпер У. Энциклопедия форматов графических файлов/Пер. с англ. - К.: ВНУ, 1997. - 672 с.
26. Международный стандарт JPEG ISO/IEC 10918.
27. Международный стандарт MPEG ISO CD 11172.
28. Pentium Processor Family. Developer's Manual. Copyright Intel Corporation 1996, 1997.

Навчальне видання

Майданюк В.П.

Методи і засоби комп'ютерних інформаційних технологій. Кодування зображень

Навчальний посібник

Оригінал-макет підготовлено автором

Редактор В.О. Дружиніна

Підписано до друку
Формат 29,7х42 1/4 Гарнітура Times New Roman
Друк різографічний Ум. друк. арк.
Тираж __ прим.
Зам. №

Віддруковано в комп'ютерному інформаційно-видавничому центрі Вінницького державного технічного університету
21021, м. Вінниця, Хмельницьке шосе, 95, ВДТУ, ГНК, 9-й поверх
Тел. **(0432) 44-01-59**