МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.
«____»_____ 2019 р.

МАГІСТЕРСЬКА РОБОТА

НА ТЕМУ:

Дослідження та програмна реалізація алгоритмів стиснення відеоданих

Освітньо-кваліфікаційний рівень "Магістр"
Спеціальність 122 «Комп'ютерні науки»

Науковий керівник роботи: _____ (підпис) В.М.Барбарук (ініціали, прізвище)

Консультант з охорони праці: _____ (підпис) Я.О. Критська (ініціали, прізвище)

Студент: _____ (підпис) Є. В. Діулін (ініціали, прізвище)

Група: КН-17дм

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерноих наук та інженерії
Освітньо-кваліфікаційний рівень "магістр"
Спеціальність 122 – "Комп'ютерні науки "
(шифр і назва)
Спеціалізація
(шифр і назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри КНІ
д.т.н., доц. *І.С. Скарга-Бандурова*
«_____» _____20____р.

З А В Д А Н Н Я
НА МАГІСТЕРСЬКУ РОБОТУ СТУДЕНТУ

Діуліну Євгену Вадимовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Дослідження та програмна реалізація алгоритмів стиснення відеоданих

керівник проекту (роботи) Барбарук В.М., к.т.н., доцент
(прізвище, ім.'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від «18» 10 2018 р. № 220/48
2. Строк подання студентом роботи 09.01.2019
3. Вихідні дані до роботи Матеріали науково-дослідної практики.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Огляд алгоритмів стиснення зображень.
2. Аналіз алгоритмів архівації без втрат.
3. Аналіз алгоритмів архівації з втратами.
3. Аналіз методів стиснення інформації.
4. Програмна реалізація та аналіз.
5.Охорона праці та безпека в надзвичайних ситуаціях.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
Електронні плакати

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	*Критська Я.О.*		

7. Дата видачі завдання *18.10.2018*

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Формування технічного завдання	*19.10.18-25.10.18*	
2	Збір необхідної інформації щодо задач стиснення зображень	*26.10.18-03.11.18*	
3	Аналіз алгоритмів архівації	*04.11.18-14.11.18*	
4	Розробка програмних додатків для експерименту	*15.11.18-30.11.18*	
5	Тестування програмних додатків	*01.12.18-19.12.18*	
	Проведення експериментальних досліджень	*20.12.18-29.12.18*	
6	Охорона праці	*30.12.18-02.01.19*	
7	Оформлення пояснювальної записки	*03.01.19-06.01.19*	
8	Підготовка презентації та доповіді	*07.01.19-09.01.19*	

Студент _____ *Діулін Є. В.*
(підпис) (прізвище та ініціали)

Науковий керівник _____ *Барбарук В.М.*
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Діулін Є. В. Дослідження та програмна реалізація алгоритмів стиснення відеоданих.

Метою магістерської роботи є дослідження алгоритмів стиснення інформації про зображення та розробка програми для проведення практичного аналізу. Результати досліджень дозволяють зробити висновок про доцільність застосування розроблених алгоритмів для стиснення зображень, а також для вирішення завдання зменшення кількості байтів необхідних для зберігання та подання зображення.

Ключові слова: метод, стиснення, зображення, алгоритм, архівація, коефіцієнт, втрати, дослідження.

АННОТАЦИЯ

Диулин Е. В. Исследование и программная реализация алгоритмов сжатия видеоданных.

Целью магистерской работы является исследование алгоритмов сжатия информации об изображении и разработка программы для проведения практического анализа. Результаты исследований позволяют сделать вывод о целесообразности применения разработанных алгоритмов для сжатия изображений, а также для решения задачи уменьшения количества байтов необходимых для хранения и представления изображения.

Ключевые слова: метод, сжатие, изображения, алгоритм, архивация, коэффициент, потери, исследования.

ABSTRACT

Diulin E. V. Research and software implementation of video data compression algorithms.

The aim of certification diploma is to study the image information compression algorithms and the development of a program for carrying out practical analysis. The results of researches let do a summary about a expediency to use developed algorithms to compression and decompression images and to resolve issues regarding reducing bytes to store and represent images.

Key words: method, compression, image, algorithm, archivation, coefficient, losses, research.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

RLE — run-length encoding.

JPEG — Joint Photographic Experts Group.

LZW — Lempel-Ziv-Welch.

MMX — Multimedia Extensions.

GIF — Graphics Interchange Format.

LZ — сімейство алгоритмів словникового стиснення даних.

IFS — Iterated Function System

ВСТУП

Растрові файли мають дуже великі розміри. Якщо знехтувати заголовками файла й іншими неграфічними даними, то його розмір пропорційний кількості пікселів у зображенні і кількості бітів, необхідних для відображення кожного пікселя. Повнокольорове зображення розміром 1024x768 пікселів займає більш двох мегабайт пам'яті, а одна секунда відео телевізійної якості в растровому виді вимагає біля тридцятьох мегабайт. Тому жорсткий диск можна заповнити миттєво. Навіть компакт-диск, що вміщає біля 700 мегабайт даних, не настільки великий, щоб помістити такий об'єм інформації [4].

Використовуючи метод, який називається стисненням зображень, можна різко зменшити в розмірі графічні файли. При стисненні графічної інформації використовуються спеціальні прийоми, що зменшують кількість байтів, необхідних для представлення зображення. Степінь стиснення залежить від методу стиснення і вмісту графічного файла. Як правило графічний файл стискується в п'ять і більш разів. Існують методи, що стискують ще сильніше, але з втратами якості. При відновленні зображення втрачається деяка частина колірної інформації. У підсумку, розпаковане зображення може стати злегка розмитим або знебарвленим [4, 5].

Методи стиснення растрової інформації діляться на дві великі групи: стиснення з втратами і стиснення без втрат. Методи стиснення без втрат дають більш низький коефіцієнт стиснення, але зате зберігають точне значення пікселів вихідного зображення. Методи з втратами дають більш високі коефіцієнти стиснення, але не дозволяють відтворити початкове зображення з точністю до пікселя. Для файлів, які формуються програмами автоматизованого проектування, дуже важливо зберегти всю інформацію, тому що втрата хоча б одного біта може змінити зміст усього файла. Зовсім інша справа з растровими даними. Людське око не сприймає всі відтінки кольору в звичайному растровому зображенні. Таким чином, деякі деталі можуть бути опущені без видимого порушення інформаційного змісту зображення.

Протягом останніх 10 років в рамках комп’ютерної графіки бурхливо розвивається напрям розробки алгоритмів архівації зображень. Це обумовлено тим, що зображення — це своєрідний тип даних, характеризується трьома особливостями [5 - 7]:

– Зображення (як і відео) займають набагато більше місця в пам'яті, ніж текст.

Так, неякісна ілюстрація на обкладинці книги розміром 500x800 займає 1.2 Мб — стільки ж, скільки художня книга з 400 сторінками (60 знаків у строчці, 42 строки). Можно

помістити тисячу сторінок тексту на CD-ROM, і так само менше якісних незжатих фотографій.

Ця особливість зображень визначає актуальність алгоритмів архівації графіки.

– Людський зір при аналізі зображення оперує контурами, загальним переходом кольорів і порівняно не відчутний до малих змін в зображенні.

Таким чином, можно створити ефективні алгоритми архівації зображень, в яких декомпресування зображення не буде збігатися з оригіналом, однак людина цього не помітить

Дана особливість людського зору дозволила створити спеціальні алгоритми стиснення, орієнтовані тільки на зображення. Ці алгоритми мають дуже високі характеристики.

– Зображення (легко помітити) на відміну, наприклад, від тексту володіє надмірністю в 2-х вимірах (тобто як правило, сусідні точки, як по горизонталі, так і по вертикалі в зображенні близькі за кольором. Крім того, можно використовувати подобу між колірними площинами R, G і B в наших алгоритмах, що дає можливість створити ще більш ефективні алгоритми).

Таким чином, при створенні алгоритму компресії графіки ми використовуємо особливості структури зображення.

В даний момент відомо близько трьох сімейств алгоритмів, які використовуються виключно для стиснення зображень, і методи, що застосовуються в них, практично неможливо застосувати до архівації яких-небудь ще видів даних [15].

Для того, щоб говорити про алгоритми стиснення зображень, потрібно визначитися з кількома важливими питаннями [15, 16]:

— Які критерії можно запропонувати для порівняння різних алгоритмів?

— Які класи зображень існують?

— Які класи додатків існують, і які вимоги вони висувають до алгоритмів компресії?

1 ЗАГАЛЬНІ ПОЛОЖЕННЯ АЛГОРИТМІВ СТИСНЕННЯ ЗОБРАЖЕНЬ

1.1 Класи зображень

Статичні растрові зображення представляють собою двовимірний масив чисел. Елементи цього масиву називають пікселями (Pixel — picture element).

Всі зображення можна поділити на дві групи — з палітрою і без неї.

У зображень з палітрою в пікселі зберігається число — індекс в деякому одновимірному векторі кольорів, званому палітрою. Найчастіше зустрічаються палітри з 16 і 256 кольорів.

Зображення без палітри бувають або в будь-якій системі кольорово-уявлення, або в градаціях сірого (grayscale). Для останніх значення кожного пікселя інтерпретується як яскравість відповідної точки. Зустрічаються зображення з 2, 16 і 256 рівнями сірого.

Одне з цікавих практичних завдань полягає у приведенні кольорового або чорно-білого зображення до двох градацій яскравості, наприклад, для друку на лазерному принтері.

При використанні певної системи кольорово-уявлення кожен піксель являє собою запис (структуру), полями якого є компоненти кольору.

Найпоширенішою є система RGB (колір представлений значеннями інтенсивності червоної (R), зеленої (G) і синьої (B) компоненти).

Існують і інші системи кольорового-уявлення, такі, як CMYK, CIE XYZccir60-1 і т.п. Нижче побачимо, як використовуються колірні моделі при стисненні зображень з втратами.

Для того, щоб коректніше оцінювати ступінь стиснення, потрібно ввести поняття класу зображень.

Під класом буде розумітися якась сукупність зображень, застосування до яких алгоритму архівації дає якісно однакові результати.

Наприклад, для одного класу алгоритм дає дуже високу ступінь стиснення, для іншого — майже не стискає, для третього — збільшує файл в розмірі. Відомо, що багато алгоритми в гіршому випадку збільшують файл.

Розглянемо наступні приклади неформального визначення класів зображень:

— 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.

— Клас 2. Зображення з плавними переходами кольорів, побудовані на комп'ютері. Приклади: графіка презентацій, ескізні моделі в САПР, зображення, побудовані по методу Гуро.

— Клас 3. Фотореалістичні зображення. Приклад: відсканованні фотографії.

— Клас 4. Фотореалістичні зображення з накладенням ділової графіки. Приклад: реклама.

Розвиваючи цю класифікацію, як окремі класи можуть бути запропоновані неякісно відсканованні в 256 градацій сірого кольору сторінки книг або растрові зображення топографічних карт (цей клас не тотожний класу 4).

Формально будучи 8- або 24-бітними, вони несуть навіть не растрову, а чисто векторну інформацію.

Окремі класи можуть утворювати і зовсім специфічні зображення: рентгенівські знімки або фотографії в профіль і фас з електронного досьє.

Досить складним і цікавим завданням є пошук найкращого алгоритму для конкретного класу зображень.

Немає сенсу говорити про те, що якийсь алгоритм стиснення кращий за інший, якщо не позначили класи зображень, на яких порівнюються алгоритми.

1.2 Класи додатків

Розглянемо наступну просту класифікацію додатків, що використовують алгоритми компресії:

Клас 1. Характеризується високими вимогами до часу архівації та розархівації.

Нерідко потрібен перегляд зменшеної копії зображення і пошук в базі даних зображень.

Є можливість оперувати зображеннями меншої якості і використовувати алгоритми стиснення з втратами. У подібних системах доводиться мати справу з кольоровими зображеннями самого різного розміру (від 640x480 — формат цифрового фотоапарату, до 3000x2000) і з великими двокольоровими зображеннями.

Клас 2. Характеризується високими вимогами до ступеня архівації та часу розархівації. Час архівації ролі не грає. Іноді подібні додатки також вимагають від алгоритму компресії легкості збільшувати або зменшувати зображення під конкретний дозвіл монітора у користувача. Приклад: Довідники та енциклопедії на CD-ROM. З появою великої кількості комп'ютерів, оснащених цим приводом (В США — у 50% машин) досить швидко

сформувався ринок програм, що випускаються на лазерних дисках. Незважаючи на те, що ємність одного диска досить велика (приблизно 650 Мб), її, як правило, не вистачає. При створенні енциклопедій та ігор більшу частину диска займають статичні зображення і відео. Таким чином, для цього класу додатків актуальності набувають істотно асиметричні за часом алгоритми (симетричність за часом — відношення часу компресії до часу декомпресії).

Клас 3. Характеризується дуже високими вимогами до ступеня архівації. Додаток клієнта отримує від сервера інформацію по мережі. Приклад: Нова система, що швидко розвивається, — "Всесвітня інформаційна павутина" (WWW).

Можна навести безліч більш вузьких класів додатків. Так своє застосування машинна графіка знаходить і в різних інформаційних системах. Наприклад, вже стає звичним досліджувати ультразвукові та рентгенівські знімки не на папері, а на екрані монітора. Поступово в електронний вигляд переводять і історії хвороб. Зрозуміло, що зберігати ці матеріали логічніше в єдиній картотеці. При цьому без використання спеціальних алгоритмів більшу частину архівів займуть фотографії. Тому при створенні ефективних алгоритмів вирішення цього завдання потрібно врахувати специфіку рентгенівських знімків — переважання розмитих ділянок.

У геоінформаційних системах — при зберіганні аерофотознімків місцевості — специфічними проблемами є великий розмір зображення і необхідність вибірки лише частини зображення на вимогу. Крім того, може знадобитися масштабування. Це неминуче накладає свої обмеження на алгоритм компресії.

Немає сенсу говорити про те, що якийсь конкретний алгоритм компресії кращий за інший, якщо не позначено клас додатків, для якого ці алгоритми збираємося використовувати.

1.3 Вимоги додатків до алгоритмів компресії

Зауважимо, що програма визначає характер використання зображень (зберігається, використовується, викачується по мережі; або воно велике за розмірами, а нам необхідна лише частина). Характер використання зображень задає ступінь важливості наведених нижче суперечливих вимог до алгоритму:

– Високий ступінь компресії. Зауважимо, що далеко не для всіх додатків актуальна висока ступінь компресії. Крім того, деякі алгоритми дають краще співвідношення якості до розміру файлу при високих ступенях компресії, однак програють іншим алгоритмам при низьких ступенях.

– Висока якість зображень. Виконання цієї вимоги безпосередньо суперечить виконанню попередньої.

– Висока швидкість компресії. Ця вимога для деяких алгоритмів з втратою інформації є взаємовиключною з першими двома. Інтуїтивно зрозуміло, що чим більше часу будемо аналізувати зображення, намагаючись отримати найвищу ступінь компресії, тим краще буде результат. І, відповідно, чим менше часу витратимо на компресію (аналіз), тим нижче буде якість зображення і більше його розмір.

– Висока швидкість декомпресії. Досить універсальна вимога, актуальна для багатьох додатків. Однак можна навести приклади додатків, де час декомпресії далеко не критичний.

– Масштабування зображень. Дана вимога має на увазі легкість зміни розмірів зображення до розмірів вікна активного застосування. Справа в тому, що одні алгоритми дозволяють легко масштабувати зображення прямо під час декомпресії, в той час як інші не тільки не дозволяють легко масштабувати, але і збільшують ймовірність появи неприємних артефактів після застосування стандартних алгоритмів масштабування до декомпресованого зображення. Наприклад, можна привести приклад "поганого" зображення для алгоритму JPEG — це зображення з досить дрібним регулярним малюнком (піджак в дрібну клітку). Характер внесених алгоритмом JPEG спотворень такий, що зменшення або збільшення зображення може дати неприємні ефекти.

– Можливість показати огрублене зображення (низького дозволу), використавши тільки початок файлу. Дана можливість актуальна для різного роду мережевих додатків, де перекачування зображень може зайняти досить великий час, і бажано, отримавши початок файлу, коректно показати preview. Зауважимо, що примітивна реалізація зазначеної вимоги шляхом записування в початок зображення його зменшеної копії помітно погіршить ступінь компресії.

– Стійкість до помилок. Дана вимога означає локальність порушень в зображенні при псуванні або втраті фрагмента переданого файлу. Дана можливість використовується при передачі за багатьма адресами (broadcasting) зображень по мережі, тобто в тих випадках, коли неможливо використовувати протокол передачі, повторно запитувати дані у сервера при помилках. Наприклад, якщо передається відеоряд, то було б неправильно використовувати алгоритм, у якого стався сбій, до повного припинення показу всіх наступних кадрів. Дана вимога суперечить високому ступеню архівації, оскільки інтуїтивно зрозуміло, що треба вводити в потік надлишкову інформацію. Однак для різних алгоритмів обсяг цієї надлишкової інформації може істотно відрізнятися.

– Врахування специфіки зображення. Більш висока ступінь архівації для класу зображень, які статистично частіше будуть застосовуватися в нашому додатку. У попередніх розділах цю вимогу вже обговорювалося.

– Можливість редагування. Багато алгоритмів з втратою інформації можуть істотно зіпсувати зображення за кілька ітерацій редагування.

– Невелика вартість апаратної реалізації. Ефективність програмної реалізації. Дані вимоги до алгоритму реально пред'являють не лише виробники ігрових приставок, але і виробники багатьох інформаційних систем. Так, декомпресор фрактального алгоритму дуже ефективно і коротко реалізується з використанням технології MMX і розпаралелювання обчислень, а стиснення за стандартом CCITT Group 3 легко реалізується апаратно.

Очевидно, що для конкретного завдання будуть дуже важливі одні вимоги і менш важливі (і навіть абсолютно байдужі) інші.

На практиці для кожного завдання можно сформулювати набір пріоритетів з вимог, викладених вище, який і визначить найбільш підходящий для вирішення в наших умовах алгоритм (або набір алгоритмів).

1.4 Критерії порівняння алгоритмів

Характеристики алгоритму щодо деяких вимог додатків, які сформульовані вище, залежать від конкретних умов, в які буде поставлено алгоритм.

Так, ступінь компресії залежить від того, на якому класі зображень алгоритм тестується.

Аналогічно, швидкість компресії нерідко залежить від того, на якій платформі реалізований алгоритм.

Перевагу одного алгоритму перед іншим може дати, наприклад, можливість використання в обчисленнях алгоритму технологій нижнього рівня, типу MMX, а це можливо далеко не для всіх алгоритмів. Так JPEG істотно виграє від застосування технології MMX, а LZW немає. Крім того, доведеться враховувати, що робити паралельність на деяких алгоритмах легко, а на деяких ні.

Таким чином, неможливо скласти універсальний порівняльний опис відомих алгоритмів. Це можна зробити тільки для типових класів додатків за умови використання типових алгоритмів на типових платформах. Однак такі дані надзвичайно швидко застарівають.

Так, наприклад, в 1994 інтерес до показу огрубіння зображення, використовуючи тільки початок файлу (вимога 6), був чисто абстрактним. Реально ця можливість практично ніде не була потрібна і клас додатків, що використовують дану технологію, був украй невеликий. З вибуховим розповсюдженням Internet, який характеризується передачею зображень по порівняно повільним каналам зв'язку, використання Interlaced GIF (алгоритм LZW) і Progressive JPEG (варіант алгоритму JPEG), що реалізують цю можливість, різко зросло. Те, що новий алгоритм (наприклад, wavelet) підтримує таку можливість, істотний плюс для нього сьогодні.

У той же час можно розглянути таку рідкісну на сьогодні вимогу, як стійкість до помилок. Можна припустити, що незабаром (через 5-10 років) з поширенням широкомовлення в мережі Internet для його забезпечення будуть використовуватися саме алгоритми стійкі до помилок, які навіть не розглядаються в сьогоднішніх статтях і оглядах.

З усіма зробленими вище застереженнями, виділимо кілька найбільш важливих для нас критеріїв порівняння алгоритмів компресії:

– Найгірший, середній і кращий коефіцієнти стиснення (частка, на яку зросте зображення, якщо вихідні дані будуть найгіршими; якийсь середньостатистичний коефіцієнт для того класу зображень, на який орієнтований алгоритм; і, нарешті, кращий коефіцієнт). Останній необхідний лише теоретично, оскільки показує ступінь стиснення найкращого (як правило, абсолютно чорного) зображення, іноді фіксованого розміру.

– Клас зображень, на який орієнтований алгоритм. Іноді зазначено також, чому на інших класах зображень виходять гірші результати.

– Симетричність. Ставлення характеристики алгоритму кодування до аналогічної характеристики при декодуванні. Характеризує ресурсомісткість процесів кодування і декодування (найбільш важливим є симетричність за часом: ставлення часу кодування до часу декодування). Іноді потрібна симетричність по пам'яті.

– Втрати якості. Якщо вони є, то за рахунок чого змінюється коефіцієнт архівації? Справа в тому, що у більшості алгоритмів стиснення з втратою інформації існує можливість зміни коефіцієнта стиснення.

– Характерні особливості алгоритму і зображень, до яких його застосовують. Тут можуть зазначатися найбільш важливі для алгоритму властивості, які можуть стати визначальними при виборі алгоритму.

Використовуючи дані критерії, приступимо до розгляду алгоритмів архівації зображень.

Один і той же алгоритм часто можна реалізувати різними способами.

Багато відомих алгоритмів, таких як RLE, LZW або JPEG мають десятки різноманітних реалізацій. Крім того, у алгоритмів буває кілька явних параметрів, варіюючи які, можна змінювати характеристики процесів архівації та розархівації (приклади в розділі про формати). При конкретній реалізації ці параметри фіксуються, виходячи з найбільш ймовірних характеристик вхідних зображень, вимог на економію пам'яті, вимог на час архівації і т.д. Тому у алгоритмів одного сімейства кращий і гірший коефіцієнти можуть відрізнятися, але якісно картина не зміниться.

1.5 Постановка завдання

Незважаючи на досягнуті успіхи, задачі стиснення інформації про зображення викликають багато питань стосовно яким алгоритмом слід робити стиснення та у яких випадках.

Метою даної дипломної роботи є аналіз алгоритмів стиснення інформації про зображення і проведення практичного дослідження. Для цього необхідно виконати наступні завдання:

– позначити критерії, які можно запропонувати для порівняння різних алгоритмів;

– виділити всі класи зображень, які існують;

– позначити всі існуючі класи додатків і вимоги, які вони висувають до алгоритмів компресії;

– розробити й програмно реалізувати два типу алгоритмів стиснення з втратами та без втрат;

– провести порівняльний аналіз існуючих алгоритмів стиснення зображень.

Об’єкт дослідження: методи стиснення зображень.

Предмет дослідження: використання методів стиснення інформації про зображення в задачах обміну та зберігання зображень.

2 АЛГОРИТМИ АРХІВАЦІЇ БЕЗ ВТРАТ

2.1 Кодування довжин серій RLE

2.1.1 Простий метод алгоритму

Кодування довжин серій (RLE - Run-Length Encoding) - це один з найпростіших і розповсюджених алгоритмів стиснення даних. У цьому алгоритмі послідовність символів, що повторюються замінюється символом і кількістю його повторів.

Наприклад, рядок «AAAAA», що вимагає для зберігання 5 байт (за умови, що на зберігання одного символу відводиться байт), можна замінити на «5A», що складається з двох байт. Очевидно, що цей алгоритм тим ефективніше, чим довше серія повторів.

Найпростішим методом є наступна модифікація: байт, що кодує кількість повторів, повинен зберігати інформацію не тільки про кількість повторів, а й про їх наявність. Алгоритм декомпресії для нього виглядає так:

```
Initialization (...);
do {
 byte = ImageFile.ReadNextByte ();
 if (є лічильником (byte)) {
 counter = Low6bits (byte) +1;
 value = ImageFile.ReadNextByte ();
 for (i = 1 to counter)
 DecompressedFile.WriteByte (value)
 }
 else {
 DecompressedFile.WriteByte (byte)
} While (ImageFile.EOF ());
```

Алгоритм розрахований на ділову графіку — зображення з великими областями повторюваного кольору.

Ситуація, коли файл збільшується, для цього простого алгоритму не так вже рідкісна. Її можна легко отримати, застосовуючи групове кодування до оброблених кольорових фотографій. Для того, щоб збільшити зображення в два рази, його треба застосувати до зображення, в якому значення всіх пікселів більше двійкового 11000000 і поспіль попарно не повторюються.

Даний алгоритм реалізований у форматі PCX.

2.1.2 Метод підвищення ефективності алгоритму

Другим методом підвищення ефективності алгоритму RLE є використання алгоритмів перетворення інформації, які безпосередньо не стискають дані, але приводять їх до виду, більш зручному для стиснення. Як приклад такого алгоритму розглянемо BWT-перестановку, названу за прізвищами винахідників Burrows-Wheeler transform. Ця перестановка не змінює самі символи, а змінює лише їх порядок в рядку, при цьому підстроки, що повторюються, після застосування перестановки збираються в щільні групи, які набагато краще стискаються за допомогою алгоритму RLE. Пряме BWT перетворення зводиться до послідовності наступних кроків:

– Додавання до вихідного рядку спеціального символу кінця рядка, який ніде більше не зустрічається.

– Отримання всіх циклічних перестановок початкового рядка.

– Сортування отриманих рядків в лексикографічному порядку.

– Повернення останнього стовпчика отриманої матриці.

Алгоритм декомпресії для нього виглядає так:

```
        Initialization (...);
        do {
         byte = ImageFile.ReadNextByte ();
         counter = Low7bits (byte) +1;
         if (якщо ознака повтору (byte)) {
          value = ImageFile.ReadNextByte ();
         for (i = 1 to counter)
         CompressedFile.WriteByte (value)
        }
         else {
         for (i = 1 to counter) {
         value = ImageFile.ReadNextByte ();
         CompressedFile.WriteByte (value)
        }
         CompressedFile.WriteByte (byte)
} While (ImageFile.EOF ());
```

Ознакою повтору в даному алгоритмі є одиниця в старшому розряді відповідного байта.

Як можна легко підрахувати, в кращому випадку цей алгоритм стискає файл в 64 рази (а не в 32 рази, як у попередньому варіанті), в гіршому збільшує на 1/128. Середні показники ступеня компресії даного алгоритму знаходяться на рівні показників першого варіанту.

Схожі схеми компресії використані в якості одного з алгоритмів, підтримуваних форматом TIFF, а також в форматі TGA.

Коефіцієнти компресії:

– Перший варіант: 32, 2, 0,5.

– Другий варіант: 64, 3, 128/129 (кращий, середній, гірший коефіцієнти).

– Клас зображень: орієнтований алгоритм на зображення з невеликою кількістю кольорів: ділову і наукову графіку.

– Симетричність: приблизно одиниця.

– Характерні особливості: до позитивних сторін алгоритму, мабуть, можна віднести тільки те, що він не вимагає додаткової пам'яті при архівації та розархівації, а також швидко працює. Цікава особливість групового кодування полягає в тому, що ступінь архівації для деяких зображень може бути істотно підвищена всього лише за рахунок зміни порядку кольорів в палітрі зображення.

2.2 Алгоритм Лемпеля — Зіва — Велча LZW

Алгоритм LZW при стисненні даних (кодуванні джерела) динамічно створює таблицю перетворення рядків: певним послідовностям символів ставляться у відповідність групи бітів фіксованої довжини (зазвичай 12-бітні). Таблиця ініціюється всіма 1-символьними рядками. По мірі кодування алгоритм переглядає текст символ за символом і зберігає кожен новий унікальний 2-символьний рядок в таблицю у вигляді пари код/символ, де код посилається на відповідний перший символ. Після того, як новий 2-символьний рядок збережений в таблиці, на вихід передається код першого символу. Коли на вході читається черговий символ, для нього по таблиці знаходиться рядок, що вже зустрічався, максимальної довжини, після чого в таблиці зберігається код цього рядка з наступним символом на вході; на вихід видається код цього рядка, а наступний символ використовується як початок наступного рядка.

Розглянутий нижче варіант алгоритму буде використовувати дерево для подання та зберігання ланцюжків. Очевидно, що це досить сильне обмеження на вид ланцюжків, і далеко не всі однакові подцепочкі в нашому зображенні будуть використані при стисненні.

Однак в пропонованому алгоритмі вигідно стискати навіть ланцюжки, що складаються з 2 байт.

Процес стиснення виглядає досить просто. Зчитуємо послідовно символи вхідного потоку і перевіряємо, чи є у створеній нами таблиці рядків такий рядок. Якщо рядок є, то зчитуємо наступний символ, а якщо рядку немає, то заносимо в потік код для попереднього знайденого рядка, заносимо рядок в таблицю і починаємо пошук знову.Функція InitTable () очищає таблицю і поміщає в неї все рядки одиничної довжини.

```
InitTable ();
CompressedFile.WriteCode (ClearCode);
CurStr = порожній рядок;
while (НЕ ImageFile.EOF ()) {// Ще не кінець файлу
C = ImageFile.ReadNextByte ();
if (CurStr + C є в таблиці)
CurStr = CurStr + C; // Приклеїти символ до рядка
else {
code = CodeForString (CurStr); // code-ні байт!
CompressedFile.WriteCode (code);
AddStringToTable (CurStr + C);
CurStr = C; // Рядок з одного символу
}
}
code = CodeForString (CurStr);
```

Як говорилося вище, функція InitTable () ініціалізує таблицю рядків так, щоб вона містила всі можливі рядки, що складаються з одного символу. Наприклад, якщо стискаються байтові дані, то таких рядків в таблиці буде 256 ("0", "1", ..., "255"). Для коду очищення (ClearCode) і коду кінця інформації (CodeEndOfInformation) зарезервовані значення 256 і 257. В даному варіанті алгоритму використовується 12-бітний код і, відповідно, під коди для рядків залишаються значення від 258 до 4095. Рядки, що додаються, записуються в таблицю послідовно, при цьому індекс рядка в таблиці стає її кодом.

Функція ReadNextByte () читає символ з файлу. Функція WriteCode () записує код (не дорівнює за розміром байту) в вихідний файл. Функція AddStringToTable () додає новий рядок в таблицю, приписуючи їй код. Крім того, в даній функції відбувається обробка ситуації переповнення таблиці. В цьому випадку в потік записується код попереднього

знайденого рядка і код очищення, після чого таблиця очищається функцією InitTable (). Функція CodeForString () знаходить рядок в таблиці і видає код цього рядка.

Розглянемо приклад стиснення і декодування зображення. Спочатку створимо початковий словник одиничних символів. У стандартному кодуванні ASCII є 256 різних символів, тому, для того, щоб всі вони були коректно закодовані (якщо невідомо, які символи будуть присутні у вихідному файлі, а які - ні), початковий розмір коду буде дорівнює 8 бітам. Якщо заздалегідь відомо, що в початковому файлі буде менша кількість різних символів, то цілком розумно зменшити кількість біт. Щоб форматувати таблицю, встановимо відповідність коду 0 відповідному символу з двійкового кодом 00000000, тоді 1 відповідає символу з кодом 00000001, і т.д., до коду 255. Насправді ми вказали, що кожен код від 0 до 255 є кореневим.

Більше в таблиці не буде інших кодів, що володіють цією властивістю. У міру зростання словника, розмір груп повинен зростати, з тим, щоб врахувати нові елементи. 8-бітові групи дають 256 можливих комбінації біт, тому, коли в словнику з'явиться 256-е слово, алгоритм повинен перейти до 9-бітових груп. При появі 512-ого слова станеться перехід до 10-бітових груп, що дає можливість запам'ятовувати вже 1024 слова і т.д. У даному прикладі алгоритму заздалегідь відомо про те, що буде використовуватися лише 5 різних символів, отже, для їх зберігання буде використовуватися мінімальна кількість біт, що дозволяє їх запам'ятати, тобто 3 (8 різних комбінацій).

Особливість LZW полягає в тому, що для декомпресії не треба зберігати таблицю рядків в файл для розпакування. Алгоритм побудований таким чином, що можливо відновити таблицю рядків, користуючись тільки потоком кодів.

Ми знаємо, що для кожного коду треба додавати в таблицю рядок, що складається з уже наявного там рядку і символу, в якого починається.

Рисунок 2.4 – Коди строк йдуть до вихідного потоку

Коди, що записуються в потік, поступово зростають. До тих пір, поки в таблиці не з'явиться, наприклад, в перший раз код 512, всі коди будуть менше 512. Крім того, при компресії і при декомпресії коди в таблиці додаються при компресії одного і того ж символу, тобто це відбувається "синхронно". Можемо скористатися цією властивістю алгоритму для того, щоб підвищити ступінь компресії. Поки в таблицю не додано 512 символ, будемо

писати в вихідний потік бітів коди з 9 біт, а відразу при додаванні 512 — коди з 10 біт. Відповідно декомпресор також повинен буде сприймати всі коди вхідного потоку 9-бітними до моменту додавання в таблицю коду 512, після чого буде сприймати все вхідні коди як 10-бітові. Аналогічно будемо поступати при додаванні в таблицю кодів 1024 і 2048. Даний прийом дозволяє приблизно на 15% підняти ступінь компресії:

0	512	1024	2048
від 0 до 512	від 0 до 1024	від 0 до 2048	від 0 до 4095
9 біт	10 біт	11 біт	12 біт

При стисненні зображення важливо забезпечити швидкість пошуку рядків в таблиці. Можно скористатися тим, що кожен наступний підрядок на один символ довше попереднього, крім того, попередній рядок вже був нами знайдений в таблиці. Отже, досить створити список посилань на рядки, що починаються з даного підрядка, як весь процес пошуку в таблиці зведеться до пошуку в рядках, що містяться в списку для попереднього рядка. Зрозуміло, що така операція може бути проведена дуже швидко.

Зауважимо також, що реально досить зберігати в таблиці тільки пару <код попереднього підрядка, доданий символ>. Цієї інформації цілком достатньо для роботи алгоритму. Таким чином, масив від 0 до 4095 з елементами <код попереднього підрядка; доданий символ; список посилань на рядки, що починаються з цього рядка> вирішує поставлене завдання пошуку, хоча і дуже повільно.

На практиці для зберігання таблиці використовується таке же швидке як у випадку списків, але більш компактне по пам'яті рішення — хеш-таблиця. Таблиця складається з 8192 (2^{13}) елементів. Кожен елемент містить <код попереднього підрядка; доданий символ; код цього рядка>. Ключ для пошуку довжиною в 20 біт формується з використанням двох перших елементів, збережених в таблиці як одне число (key). Молодші 12 біт цього числа віддані під код, а наступні 8 біт під значення символу.

Як хеш-функції при цьому використовується:

$$Index\ (key) = ((key >> 12)\ \wedge key)\ \&\ 8191, \qquad (2.1)$$

де >> — побітовий зсув вправо (*key >> 12* —отримуємо значення символу), ^ — логічна операція побітового виключного АБО, & логічне побітове І.

Таким чином, за лічену кількість порівнянь отримуємо шуканий код або повідомлення, що такого коду в таблиці немає.

Підрахуємо кращий і гірший коефіцієнти компресії для даного алгоритму. Кращий коефіцієнт, очевидно, буде отримано для ланцюжка однакових байт великої довжини (тобто для 8-бітного зображення, всі крапки якого мають, для визначеності, колір 0). При цьому в 258 рядок таблиці запишемо рядок "0, 0", в 259 — "0, 0, 0", ... в 4095 — рядок з 3809 (= 4095-256) нулів. При цьому в потік потрапить 3810 кодів, включаючи код очищення. Отже, порахувавши суму арифметичної прогресії від 1 до 3809 (тобто довжину стислого ланцюжка) і поділивши її на 3810 * 12/8 (в потік записуються 12-бітові коди), отримаємо кращий коефіцієнт компресії.

Найгірший коефіцієнт буде отримано, якщо жодного разу не зустрінемо підрядок, який вже є в таблиці (в ній не повинно зустрітися жодної однакової пари символів).

У разі, якщо постійно будемо зустрічати новий підрядок, запишемо в вихідний потік 3810 кодів, яким буде відповідати рядок з 3808 символів. Без урахування зауваження 1 це складе збільшення файлу майже в 1.5 рази.

LZW реалізований в форматах GIF і TIFF.

Характеристики алгоритму LZW:

– Коефіцієнти компресії: приблизно 1000, 4, 5/7 (кращий, середній, гірший коефіцієнти). Стиснення в 1000 разів досягається тільки на одноколірних зображеннях розміром кратним приблизно 7 Мб.

– Клас зображень: орієнтований LZW на 8-бітові зображення, побудовані на комп'ютері. Стискає за рахунок однакових підланцюжків в потоці.

– Симетричність: майже симетричний, за умови оптимальної реалізації операції пошуку рядка в таблиці.

Характерні особливості: ситуація, коли алгоритм збільшує зображення, зустрічається вкрай рідко.

2.3 Алгоритми Хаффмана

2.3.1 Класичний алгоритм Хаффмана

Один з класичних алгоритмів, відомих з 60-х років. Використовує тільки частоту появи однакових байт в зображенні.

Зіставляє символам вхід вхідного потоку, які зустрічаються більше число раз, ланцюжок біт меншої довжини. І, навпаки, якщо зустрічаються рідко — ланцюжок більшої довжини.

Для збору статистики вимагає двох проходів по зображенню.

Для початку введемо декілька визначень.

Визначення. Нехай заданий алфавіт $\Psi=\{a_1, ..., a_r\}$, що складається з кінцевого числа букв. Кінцеву послідовність символів з Ψ

$$A=a_{i_1} a_{i_2} ... a_{i_n}\ , \tag{2.2}$$

де Ψ — слово в алфавіті Ψ, а число n — *довжина слова* A. Довжина слова позначається як $l(A)$.

Нехай заданий алфавіт Ω, $\Omega=\{b_1, ..., b_q\}$. Через B позначимо слово в алфавіті Ω і через $S(\Omega)$ — безліч всіх непустих слів в алфавіті Ω.

Нехай $S=S(\Psi)$ — безліч всіх непустих слів в алфавіті Ψ, і S' — деяка підмножина множини S. Нехай, також поставлено відображення F, яке кожному слову A, $A \in S(\Psi)$, ставить у відповідність слово

$$B=F(A),\ B \in S(\Omega), \tag{2.3}$$

де, B — код повідомлення A, а перехід від слова A до його коду — *кодування*.

Визначення. Розглянемо відповідність між буквами алфавіту Ψ і деякими словами алфавіту Ω:

$$\begin{aligned} &a_1 — B_1, \\ &a_2 — B_2, \\ &\ldots \\ &a_r — B_r. \end{aligned} \tag{2.4}$$

Цю відповідність називають схемою і позначають через Σ. Вона визначає кодування наступним чином: кожному слову $A=a_{i_1} a_{i_2} ... a_{i_n}$ з $S'(\Omega)=S(\Omega)$ ставиться у відповідність слово $B=B_{i_1} B_{i_2} ... B_{i_n}$, що називається кодом слова A. Слова B_1 ... B_r називаються елементарними кодами. Даний вид кодування називають алфавітним кодуванням.

Визначення. Нехай слово B має вигляд

$$B=B'B''. \tag{2.5}$$

Тоді слово B' називається початком або префіксом слова B, а B'' — кінцем слова B. При цьому пусте слово Λ і саме слово B вважаються началами і кінцями слова B.

Визначення. Схема Σ має властивість префікса, якщо для будь-яких i і j ($1 \le i, j \le r, i \ne j$) слово B_i не є префіксом слова B_j.

Теорема 1. Якщо схема Σ має властивість префікса, то алфавітне кодування буде взаємно однозначним.

Припустимо, що задано алфавіт $\Psi=\{a_1,..., a_r\}$ ($r>1$) і набір ймовірностей $p_1, \ldots, p_r$ $\left(\sum_{i=1}^{r} p_i = 1\right)$ появи символів $a_1,..., a_r$. Нехай, далі, заданий алфавіт Ω, $\Omega=\{b_1, ..., b_q\}$ ($q>1$). Тоді можна побудувати цілий ряд схем Σ алфавітного кодування

$$\begin{aligned} a_1 &— B_1, \\ &\ldots \\ a_r &— B_{r.,} \end{aligned} \qquad (2.6)$$

що володіють властивістю взаємної однозначності.

Для кожної схеми можна ввести середню довжину $l_{ср}$, яка визначається як математичної очікування довжини елементарного коду.

Довжина $l_{ср}$ показує, у скільки разів збільшується середня довжина слова при кодуванні зі схемою Σ.

Можна показати, що $l_{ср}$ досягає величини свого мінімуму l_* на деякій Σ і визначена як

$$l_* = \min_{\Sigma} l_{cp}^{\Sigma}. \qquad (2.7)$$

Коди, що визначаються схемою Σ з $l_{ср}= l_{*,}$ називаються кодами з мінімальною надмірністю або кодами Хаффмана.

Коди з мінімальною надмірністю дають в середньому мінімальне збільшення довжин слів при відповідному кодуванні.

У нашому випадку, алфавіт $\Psi=\{a_1,..., a_r\}$ задає символи вхідного потоку, а алфавіт $\Omega=\{0,1\}$, тобто складається всього з нуля і одиниці.

Алгоритм побудови схеми Σ можна представити таким чином:

Крок 1. Упорядковуємо всі букви вхідного алфавіту в порядку убування ймовірності. Вважаємо всі відповідні слова B_i, з алфавіту $\Omega=\{0,1\}$ порожніми.

Крок 2. Об'єднуємо два символи a_{ir-1} і a_{ir} з найменшими ймовірностями $p_{i\ r-1}$ і $p_{i\ r}$ в псевдосімвол $a'\{a_{ir-1}\ a_{ir}\}$ с ймовірністю $p_{ir-1}+p_{ir}$. Дописуємо 0 в початок слова B_{ir-1} ($B_{ir-1}=0B_{ir-1}$), і 1 в початок слова і B_{ir} ($B_{ir}=1B_{ir}$).

Крок 3. Видаляємо зі списку упорядкованих символів a_{ir-1} і a_{ir}, заносимо туди псевдосімвол $a'\{a_{ir-1}a_{ir}\}$. Проводимо крок 2, додаючи при необхідності 1 або нуль для всіх слів *Bi*, відповідних псевдосімволам, до тих пір, поки в списку не залишиться 1 псевдосимвол.

Приклад: Нехай у нас є 4 букви в алфавіті $\Psi=\{a_1,..., a_4\}$ (r=4), p_1=0.5, p_2=0.24, p_3=0.15, p_4=0.11 $\left(\sum_{i=1}^{4} p_i = 1\right)$. Тоді процес побудови схеми можна уявити так:

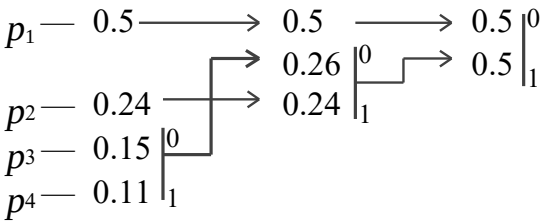


Рисунок 2.5 – Процес побудови схеми

Виробляючи дії, відповідні 2-му кроку отримуємо псевдосимвол з ймовірністю 0.26 (і приписуємо 0 і 1 відповідним словам). Повторюючи ж ці дії для зміненого списку, отримуємо псевдосімвол з ймовірністю 0.5. І, нарешті, на останньому етапі отримуємо сумарну ймовірність 1.

Для того, щоб відновити слова, що кодують, треба пройти по стрілках від початкових символів до кінця отриманого бінарного дерева. Так, для символу з ймовірністю p_4, отримаємо *B4 = 101*, для p_3 отримаємо B3 = 100, для p_2 отримаємо *B2* = 11, для p_1 отримаємо *B1* = 0. Що означає схему:

a_1 — 0,

a_2 — 11,

a_3 — 100,

a_4 — 101.

Ця схема являє собою префіксний код, який є кодом Хаффмана. Символ a_1, що зустрічається найбільш часто в потоці, будемо кодувати самим коротким словом 0, а a_4 рідко зустрічається - довгим словом 101.

Для послідовності з 100 символів, в якій символ a_1 зустрінеться 50 раз, символ a_2 — 24 рази, символ a_3 — 15 разів, а символ a_4 — 11 разів, даний код дозволить отримати послідовність з 176 біт ($100 \cdot l_{cp} = \sum_{i=1}^{4} p_i l_i$). Тобто в середньому витратимо 1.76 біта на символ потоку.

Доведення теореми, а також того, що побудована схема дійсно задає код Хаффмана розглянуто в [10].

Як стало зрозуміло з викладеного вище, класичний алгоритм Хаффмана вимагає записи в файл таблиці відповідності кодованих символів і ланцюжків, що кодують.

На практиці використовуються його різновиди. Так, в деяких випадках резонно або використовувати постійну таблицю, або будувати її "адаптивно", тобто в процесі архівації/розархівації. Ці прийоми позбавляють від двох проходів по зображенню і необхідності зберігання таблиці разом з файлом. Кодування з фіксованою таблицею застосовується в якості останнього етапу архівації в JPEG і в алгоритмі CCITT Group 3.

2.3.2 Алгоритм Хаффмана з фіксованою таблицею CCITT Group 3

Близька модифікація алгоритму використовується при стисненні чорно-білих зображень (один біт на піксель). Повна назва цього алгоритму CCITT Group 3. Це означає, що даний алгоритм був запропонований третьою групою по стандартизації Міжнародного Консультаційного Комітету з телеграфного й телефонного зв'язку (Consultative Committee International Telegraph and Telephone). Послідовності поспіль чорних і білих точок в ньому замінюються числом, рівним їх кількості. А цей ряд вже, в свою чергу, стискається по Хаффману з фіксованою таблицею.

Визначення: набір, що складажться з точок зображення одного кольору називається серією. Довжина цього набору точок називається довжиною серії.

У таблиці, наведеної нижче, задані два види кодів:

– Коди завершення серій ― задані з 0 до 63 з кроком 1.
– Складові (додаткові) коди ― задані з 64 до 2560 з кроком 64.

Кожен рядок зображення стискується незалежно. Вважаємо, що в нашому зображенні істотно переважає білий колір, і всі рядки зображення починаються з білої точки. Якщо рядок починається з чорної точки, то вважаємо, що рядок починається білою серією довжини 0. Наприклад, послідовність довжин серій 0, 3, 556, 10, ... означає, що в цьому рядку зображення йдуть спочатку 3 чорних точки, потім 556 білих, потім 10 чорних і т.д.

На практиці в тих випадках, коли в зображенні переважає чорний колір, інвертуємо зображення перед компресією і записуємо інформацію про це в заголовок файлу.

Оскільки чорні і білі серії чергуються, то реально код для білої та код для чорної серії будуть працювати по черзі.

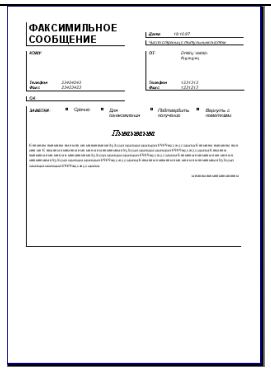
У термінах регулярних виразів отримаємо для кожного рядка нашого зображення (досить довгого, що починається з білої точки) вихідний потік бітів виду:

$$((<Б\text{-}2560>)^*[<Б\text{-}сст.>]<Б\text{-}зв.>(<Ч\text{-}2560>)^*[<Ч\text{-}сст.>]<Ч\text{-}зв.>)^+$$

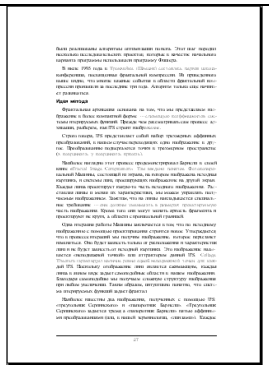
$$[(<Б\text{-}2560>)^*[<Б\text{-}сст.>]<Б\text{-}зв.>],$$

де $()^*$ — повтор 0 або більше разів, $()^+$.— повтор 1 або більше разів, $[]$— включення 1 або 0 раз.

Для наведеного раніше прикладу: 0, 3, 556, 10 ... алгоритм сформує наступний код: *<Б-0> <Ч-3> <Б-512> <Б-44> <Ч-10>*, або, згідно з таблицею, 00110101100110010100101101000010 0 (різні коди в потоці виділені для зручності). Цей код має властивість префіксних кодів і легко може бути згорнутий назад в послідовність довжин серій. Легко підрахувати, що для наведеного рядки в 569 біт отримали код, довжиною в 33 біта, тобто коефіцієнт стиснення складає приблизно 17 разів.



Зображення, для якого дуже вигідно застосування алгоритму CCITT-3 (Великі області заповнені одним кольором.)



Зображення, для якого менш вигідно застосування алгоритму CCITT-3. (Менше областей, заповнених одним кольором. Багато коротких "чорних" і "білих" серій.)

Рисунок 2.6 – Порівняння зображень для CCITT-3

Таблиця 2.1 побудована за допомогою класичного алгоритму Гоффмана (окремо для чорних і білих довжин серій). Значення ймовірностей появи конкретних довжин серій були отримані шляхом аналізу великої кількості факсимільних зображень.

Таблиця 2.1 – Таблиця кодів завершення.

Довжина серії	код білої підстроки	код чорної підстроки
0	00110101	0000110111
1	00111	010
2	0111	11
3	1000	10
4	1011	011
5	1100	0011
6	1110	0010
7	1111	00011
8	10011	000101
9	10100	000100
10	00111	0000100
11	01000	0000101
12	001000	0000111
13	000011	00000100
14	110100	00000111
15	110101	000011000
16	101010	0000010111
17	101011	0000011000
18	0100111	0000001000
19	0001100	00001100111
20	0001000	00001101000
21	0010111	00001101100
22	0000011	00000110111
23	0000100	00000101000
24	0101000	00000010111
25	0101011	00000011000
26	0010011	000011001010
27	0100100	000011001011
28	0011000	000011001100
29	00000010	000011001101
30	00000011	000001101000
31	00011010	000001101001

довжина серії	код білої підстроки	код чорної підстроки
32	00011011	000001101010
33	00010010	000001101011
34	00010011	000011010010
35	00010100	000011010011
36	00010101	000011010100
37	00010110	000011010101
38	00010111	000011010110
39	00101000	000011010111
40	00101001	000001101100
41	00101010	000001101101
42	00101011	000011011010
43	00101100	000011011011
44	00101101	000001010100
45	00000100	000001010101
46	00000101	000001010110
47	00001010	000001010111
48	00001011	000001100100
49	01010010	000001100101
50	01010011	000001010010
51	01010100	000001010011
52	01010101	000000100100
53	00100100	000000110111
54	00100101	000000111000
55	01011000	000000100111
56	01011001	000000101000
57	01011010	000001011000
58	01011011	000001011001
59	01001010	000000101011
60	01001011	000000101100
61	00110010	000001011010
62	00110011	000001100110
63	00110100	000001100111

2.4 JBIG

Алгоритм розроблений групою експертів ISO (Joint Bi-level Experts Group) спеціально для стиснення однобітних чорно-білих зображень [5]. Наприклад, факсів або відсканованих документів. Цей алгоритм може застосовуватися і до 2-х, і до 4-х бітових зображень. При цьому алгоритм розбиває їх на окремі бітові площини. JBIG дозволяє управляти такими параметрами, як порядок розбиття зображення на бітові площини, ширина смуг в зображенні, рівні масштабування. Остання можливість дозволяє легко орієнтуватися в базі великих за розмірами зображень, переглядаючи спочатку їх зменшені копії. Налаштовуючи ці параметри, можна використовувати описаний вище ефект "огрубіння зображення" при отриманні зображення по мережі або по будь-якому іншому каналу, пропускна здатність якого мала в порівнянні з можливостями процесора. Розпаковуватися зображення на екрані буде поступово, як би повільно "проявляючись". При цьому людина починає аналізувати картинку задовго до кінця процесу розархівації.

Алгоритм, побудований на базі Q-кодувальника [6], патентом на який володіє IBM. Q-кодер так само, як і алгоритм Хаффмана, використовує для символів, що частіше з'являються, короткі ланцюжки, а для тих, що рідше з'являються — довгі. Однак, на відміну від нього, в алгоритмі використовуються і послідовності символів.

2.5 Lossless JPEG

Цей алгоритм, розроблений групою експертів в області фотографії (Joint Photographic Expert Group). У відміну від JBIG, Lossless JPEG орієнтований на повнокольорові 24-бітові або 8-бітові в градаціях сірого зображення без палітри. Він являє собою спеціальну реалізацію JPEG без втрат. Коефіцієнти стиснення: 20, 2, 1. Lossless JPEG рекомендується застосовувати в тих додатках, де необхідна побітова відповідність вихідного і декомпресованого зображень.

3 АЛГОРИТМИ АРХІВАЦІЇ З ВТРАТАМИ

3.1 Проблеми алгоритмів архівації з втратами

Першими для архівації зображень стали застосовуватися звичні алгоритми. Ті, що використовувалися і використовуються в системах резервного копіювання, при створенні дистрибутивів і т.п. Ці алгоритми архівували інформацію без змін. Однак основною тенденцією останнім часом стало використання нових класів зображень. Старі алгоритми перестали задовольняти вимогам, що пред'являються до архівації. Багато зображень практично не стискалися, хоча "на погляд" володіли явною надмірністю. Це призвело до створення нового типу алгоритмів ― алгоритмів, що стискають з втратою інформації. Як правило, коефіцієнт архівації та, отже, ступінь втрат якості в них можна задавати. При цьому досягається компроміс між розміром і якістю зображень.

Одна з серйозних проблем машинної графіки полягає в тому, що до цих пір не знайдений адекватний критерій оцінки втрат якості зображення. А втрачається воно постійно — при оцифруванні, при перекладі в обмежену палітру кольорів, при перекладі в іншу систему представлення кольорів для друку, і, що для нас особливо важливо, при архівації з втратами. Можна навести приклад простого критерію: середньоквадратичне відхилення значень пікселів (L_2 міра, або root mean square — RMS):

$$d(x,y) = \sqrt{\frac{\sum_{i=1,j=1}^{n,n} \left(x_{ij} - y_{ij}\right)^2}{n^2}}. \qquad (3.1)$$

По ньому зображення буде сильно зіпсовано при зниженні яскравості всього на 5% (це може бути не помітно на перший погляд — залежно від виробу настройки яскравості). У той же час зображення зі "снігом" — різкою зміною кольору окремих точок, слабкими смугами або "мусором" будуть визнані "майже не змінені". Свої неприємні сторони є і у інших критеріїв.

Розглянемо, наприклад, максимальне відхилення:

$$d(x,y) = \max_{i,j} \left| x_{ij} - y_{ij} \right|. \qquad (3.2)$$

Цей підхід, як можна здогадатися, вкрай чутливий до биття окремих пікселів. Тобто у всьому зображенні може істотно змінитися тільки значення одного пікселя, що практично непомітно для очей, проте згідно цього підходу зображення буде сильно зіпсовано.

Міра, яку зараз використовують на практиці, називається мірою відносини сигналу до шуму (peak-to-peak signal-to-noise ratio — PSNR).

$$d(x,y) = 10 \cdot \log_{10} \frac{255^2 \cdot n^2}{\sum_{i=1,j=1}^{n,n} \left(x_{ij} - y_{ij} \right)^2}. \quad (3.3)$$

Дана міра, по суті, аналогічна середньоквадратичному відхиленню, проте користуватися нею кілька зручніше за рахунок логарифмічного масштабу шкали. Їй притаманні ті ж недоліки, що і середньоквадратичному відхиленню.

Найкраще втрати якості зображень оцінюють наші очі. Відмінною вважається архівація, при якій неможливо на очі розрізнити початкове і стиснене зображення. Непомітна архівація — можливість визначити, яке з зображень піддавалося архівації, можна тільки порівнюючи дві картинки, що знаходяться поруч. При подальшому збільшенні ступеня стиснення, як правило, стають помітні побічні ефекти, характерні для даного алгоритму. На практиці, навіть при відмінному збереженні якості, в зображення можуть бути внесені регулярні специфічні зміни. Тому алгоритми архівації з втратами не рекомендується використовувати при стисненні зображень, які в подальшому збираються або друкувати з високою якістю, або обробляти програмами розпізнавання образів. Неприємні ефекти з такими зображеннями, можуть виникнути навіть при простому масштабуванні зображення.

3.2 Алгоритм JPEG

JPEG — один з найновіших і досить потужних алгоритмів. Практично він є стандартом де-факто для повнокольорових зображень [1]. Оперує алгоритм областями 8x8, на яких яскравість і колір змінюються плавно. Внаслідок цього, при розкладанні матриці такої області в подвійний ряд по косинусам (див. формули нижче) значущими виявляються тільки перші коефіцієнти. Таким чином, стиснення в JPEG здійснюється за рахунок плавності зміни кольорів у зображенні.

Алгоритм розроблений групою експертів в області фотографії спеціально для стиснення 24-бітових зображень. JPEG — Joint Photographic Expert Group — підрозділ в

рамках ISO — Міжнародної організації зі стандартизації. Алгоритм заснований на дискретному косинусоїдальному перетворенні (ДКП), що застосовується до матриці зображення для отримання деякої нової матриці коефіцієнтів. Для отримання вихідного зображення застосовується зворотне перетворення.

ДКП розкладає зображення по амплітудам деяких частот, таким чином, при перетворенні отримуємо матрицю, в якій багато коефіцієнтів або близькі, або дорівнюють нулю. Крім того, людська система колірного сприйняття слабо розпізнає певні частоти. Тому можна апроксимувати деякі коефіцієнти більш грубо без помітної втрати якості зображення.

Для цього використовується квантування коефіцієнтів. У найпростішому випадку — це арифметичне побітове зрушення вправо. При цьому перетворенні втрачається частина інформації, але можуть досягатися великі коефіцієнти стиснення.

Розглянемо алгоритм докладніше. Нехай стискаємо 24-бітове зображення.

Крок 1.

Переводимо зображення з колірного простору RGB з компонентами, що відповідають за червону (Red), зелену (Green) і синю (Blue) складові кольору точки, в колірний простір YCrCb (іноді називають YUV).

У ньому Y — яркостная складова, а Cr, Cb — компоненти, що відповідають за колір (хроматичний червоний і хроматичний синій). За рахунок того, що людське око менш чутливе до кольору, ніж до яскравості, з'являється можливість архівувати масиви для Cr і Cb компонент з великими втратами і, відповідно, великими коефіцієнтами стиснення.

Подібне перетворення вже давно використовується в телебаченні. На сигнали, що відповідають за колір, там виділяється більш вузька смуга частот.

Спрощено переклад з колірного простору RGB в колірний простір YCrCb можна представити так:

$$\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} = \begin{vmatrix} 0.299 & 0.587 & 0.114 \\ 0.5 & -0.4187 & -0.0813 \\ 0.1687 & -0.3313 & 0.5 \end{vmatrix} * \begin{vmatrix} R \\ G \\ B \end{vmatrix} + \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix}. \quad (3.4)$$

Зворотне перетворення здійснюється множенням вектора YUV на зворотну матрицю.

$$\begin{vmatrix} R \\ G \\ B \end{vmatrix} = \begin{vmatrix} 1 & 0 & 1.402 \\ 1 & -0.34414 & -0.71414 \\ 1 & 1.772 & 0 \end{vmatrix} * \left(\begin{vmatrix} Y \\ Cb \\ Cr \end{vmatrix} - \begin{vmatrix} 0 \\ 128 \\ 128 \end{vmatrix} \right). \quad (3.5)$$

Крок 2.

Розбиваємо вихідне зображення на матриці 8x8. Формуємо з кожної три робочі матриці ДКП — по 8 біт окремо для кожної компоненти. При великих коефіцієнтах стиснення, цей крок може виконуваться трохи складніше. Зображення ділиться по компоненті Y — як і в першому випадку, а для компонент Cr і Cb матриці набираються через рядок і через стовпець. Тобто з вихідної матриці розміром 16x16 виходить тільки одна робоча матриця ДКП. При цьому, як неважко помітити, втрачаємо 3/4 корисної інформації про колірних складових зображення і отримуємо відразу стиснення в два рази. Можемо чинити так завдяки роботі в просторі YCrCb. На результуючому RGB зображенні, як показала практика, це позначається не сильно.

Крок 3.

Застосовуємо ДКП до кожної робочої матриці. При цьому отримуємо матрицю, в якій коефіцієнти в лівому верхньому кутку відповідають низькочастотній складовій зображення, а в правому нижньому — високочастотній.

У спрощеному вигляді це перетворення можна представити так:

$$Y[u,v] = \frac{1}{4}\sum_{i=0}^{n-1}\sum_{j=0}^{n-1} C(i,u)\times C(j,v)\times \quad y[i,j], \tag{3.6}$$

де $C(i,u) = A(u)\times \quad \cos\left(\frac{(2\times i+1)\times u\times \pi}{2\cdot n}\right)$,

$$A(u) = \begin{cases} \frac{1}{\sqrt{2}}, & \text{for} \quad u \equiv 0 \\ 1, & \text{for} \quad u \neq 0 \end{cases}. \tag{3.7}$$

Крок 4.

Виробляємо квантування. В принципі це просто розподіл робочої матриці на матрицю квантування поелементно. Для кожної компоненти (Y, U і V), в загальному випадку, задається своя матриця квантування (МК) q [u, v].

$$Yq[u,v] = IntegerRound \quad \left(\frac{Y[u,v]}{q[u,v]}\right). \tag{3.7}$$

На цьому кроці здійснюється управління ступенем стиснення і відбуваються найбільші втрати. Зрозуміло, що, задаючи МК з великими коефіцієнтами, отримаємо більше нулів і, отже, більшу ступінь стиснення.

У стандарт JPEG включені рекомендовані МК, побудовані дослідним шляхом. Матриці для більшого або меншого коефіцієнтів стиснення отримують шляхом множення вихідної матриці на деяке число gamma.

З квантуванням пов'язані і специфічні ефекти алгоритму. При великих значеннях коефіцієнта gamma втрати в низьких частотах можуть бути настільки великі, що зображення розпадеться на квадрати 8x8. Втрати в високих частотах можуть проявитися в так званий "ефект Гіббса", коли навколо контурів з різким переходом кольору утворюється своєрідний "німб".

Крок 5.

Переводимо матрицю 8x8 в 64-елементний вектор за допомогою "зігзаг"-сканування, тобто беремо елементи з індексами (0,0), (0,1), (1,0), (2,0)...

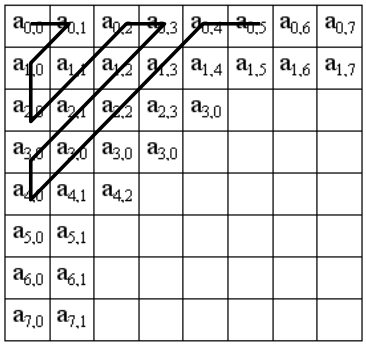


Рисунок 3.1 — Процес відновлення зображення

Таким чином, на початку вектора отримуємо коефіцієнти матриці, відповідні низьким частотам, а в кінці — високим.

Крок 6.

Згортаємо вектор за допомогою алгоритму групового кодування. При цьому отримуємо пари типу (пропустити, число), де "пропустити" є лічильником нулів, що пропускаються, а "число" — значення, яке необхідно поставити в наступну комірку. Так, вектор 42 3 0 0 0 -2 0 0 0 0 1 ... буде згорнуто в пари (0,42) (0,3) (3, -2) (4,1) ....

Крок 7.

Згортаємо отримані пари кодуванням по Хаффману з фіксованою таблицею.

Процес відновлення зображення в цьому алгоритмі повністю симетричний. Метод дозволяє стискати деякі зображення в 10-15 разів без серйозних втрат.

3.3 Фрактальный алгоритм

Фрактальна архівація заснована на тому, що представляємо зображення в більш компактній формі — за допомогою коефіцієнтів системи ітеріруємих функцій IFS. Перш ніж розглядати сам процес архівації, розберемо, як IFS будує зображення, тобто процес декомпресії.

Строго кажучи, IFS являє собою набір тривимірних афінних перетворень, в нашому випадку переводять одне зображення в інше. Перетворенню піддаються точки в тривимірному просторі (x-координата, y-координата, яскравість).

Найбільш наочно цей процес продемонстрував Барнслі в своїй книзі "Fractal Image Compression". Там введено поняття фотокопіювальних машин, що складається з екрану, на якому зображена вихідна картинка, і системи лінз, що проектують зображення на інший екран:

— Лінзи можуть проектувати частину зображення довільної форми в будь-яке інше місце нового зображення.

— Області, в які проектується зображення, не перетинаються.

— Лінза може змінювати яскравість і зменшувати контрастність.

— Лінза може дзеркально відображати і повертати свій фрагмент зображення.

— Лінза повинна зменшувати (масштабувати) свій фрагмент зображення.

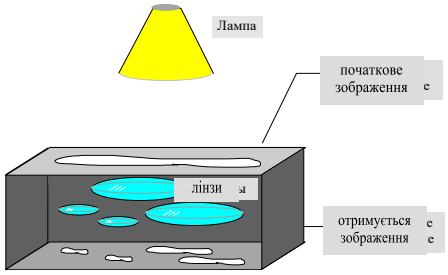


Рисунок 3.3 — Машина Барнслі

Розставляючи лінзи і змінюючи їх характеристики, можемо керувати одержуваним зображенням. Одна ітерація роботи Машини полягає в тому, що з вихідного зображення за допомогою проектування будується нове, після чого нове береться в якості вихідного. Стверджується, що в процесі ітерацій отримаємо зображення, яке перестане змінюватися. Воно буде залежати тільки від розташування і характеристик лінз і не буде залежати від вихідної картинки. Це зображення називається "нерухомою точкою" або аттрактором даної IFS. Відповідна теорія гарантує наявність рівно однієї нерухомої точки для кожної IFS.

Оскільки відображення лінз є стисканим, кожна лінза в явному вигляді задає самоподібні області в нашому зображенні. Завдяки самоподібності отримуємо складну структуру зображення при будь-якому збільшенні. Таким чином, інтуїтивно зрозуміло, що система ітеріруемих функцій задає фрактал (нестрого — самоподібний математичний об'єкт).

Найбільш відомі два зображення, отриманих за допомогою IFS: "трикутник Серпінського" і "папороть Барнслі" (рис.3.4). "Трикутник Серпінського" задається трьома, а "папороть Барнслі" чотирма аффіннимі перетвореннями (або, в нашій термінології, "лінзами"). Кожне перетворення кодується буквально ліченими байтами, в той час як зображення, побудоване з їх допомогою, може займати і кілька мегабайт.

Рисунок 3.4 — Папороть Барнслі. Задається 4 перетвореннями.

З вищесказаного стає зрозуміло як працює архіватор і чому йому потрібно так багато часу. Фактично фрактальна компресія — це пошук самоподібних областей в зображенні і визначення для них параметрів афінних перетворень.

У гіршому випадку, якщо не буде застосовуватися оптимізації алгоритму, потрібно перебрати і порівняти всі можливі фрагменти зображення різного розміру. Навіть для невеликих зображень при обліку дискретності отримаємо астрономічне число перебору варіантів. Причому, навіть різке звуження класів перетворень, наприклад, за рахунок масштабування, тільки в певну кількість разів, не дає помітного виграшу в часі. Крім того, при цьому втрачається якість зображення.

Переважна більшість досліджень в області фрактальної компресії зараз спрямовані на зменшення часу архівації, необхідного для отримання якісного зображення.

Далі наводяться основні визначення та теореми, на яких базується фрактальна компресія. Цей матеріал більш детально і з доказами розглядається в [3] і в [4].

Визначення. Перетворення $w: R^2 \to R^2$, яке представляється у вигляді

$$w(\bar{x}) = w\begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} a & b \\ c & d \end{pmatrix} \bullet \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} e \\ f \end{pmatrix}, \tag{3.8}$$

де *a, b, c, d, e, f* дійсні числа і $\begin{pmatrix} x & y \end{pmatrix} \in R^2$ називається двовимірним аффінним перетворенням.

Визначення. Перетворення $w: R^3 \to R^3$, яке представляється у вигляді

$$w(\bar{x}) = w\begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & t \\ c & d & u \\ r & s & p \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix}, \tag{3.9}$$

де *a, b, c, d, e, f, p, q, r, s, t, u* дійсні числа і $\begin{pmatrix} x & y & z \end{pmatrix} \in R^3$ називається тривимірним аффінним перетворенням.

Визначення. Нехай $f: \mathrm{X} \to \mathrm{X}$ — перетворення в просторі *X*. Точка $x_f \in \mathrm{X}$ така, що $f(x_f) = x_f$ називається нерухомою точкою (аттрактором) перетворення.

Визначення. Перетворення $f: \mathrm{X} \to \mathrm{X}$ в метричному просторі *(X, d)* називається стискає, якщо існує число s: $0 \le s < 1$, таке, що

$$d(f(x), f(x)) \le s \cdot d(x, y) \qquad \forall x, y \in \mathrm{X} \quad . \tag{3.10}$$

Зауваження: Формально можемо використовувати будь-який стискає відображення при фрактальної компресії, але реально використовуються лише тривимірні аффінні перетворення з досить сильними обмеженнями на коефіцієнти.

Теорема. (Про перетворення, що стискає)

Нехай $f: \mathrm{X} \to \mathrm{X}$ в повному метричному просторі *(X, d)*. Тоді існує в точності одна нерухома точка $x_f \in \mathrm{X}$ цього перетворення і для будь-якої точки $x \in \mathrm{X}$, послідовність $\left\{ f^n(x): \quad n = 0,1,2... \quad \right\}$ сходиться к x_f.

Більш загальне формулювання цієї теореми гарантує нам збіжність.

Визначення. Зображенням називається функція *S*, певна на одиничному квадраті і приймаюча значення від 0 до 1 або $S(x, y) \in [0...1] \qquad \forall x, y \in [0...1]$

Нехай тривимірне Афінне перетворення $w_i : R^3 \to R^3$, записано у вигляді

$$w_i(\bar{x}) = w_i \begin{pmatrix} x \\ y \\ z \end{pmatrix} = \begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & p \end{pmatrix} \bullet \begin{pmatrix} x \\ y \\ z \end{pmatrix} + \begin{pmatrix} e \\ f \\ q \end{pmatrix}, \qquad (3.11)$$

і визначено на компактній підмножині R_i декартова квадрата [0..1]*x*[0..1]. Тоді воно переведе частину поверхні *S* в область, розташовану із зсувом *(e, f)* і поворотом, заданим матрицею

$$\begin{pmatrix} a & b & 0 \\ c & d & 0 \\ 0 & 0 & 0 \end{pmatrix}.$$

При цьому, якщо інтерпретувати значення *S* як яскравість відповідних точок, то вона зменшиться в *p* раз (перетворення має бути стискає) і зміниться на зрушення *q*.

Визначення. Кінцева сукупність *W* стискає тривимірні афінні перетворення w_i, визначені на областях R_i, таких, що $w_i(R_i) = D_i$ і $D_i \cap D_j = \Theta \quad \forall i \neq j$ називається системою ітерованих функцій (IFS).

У системі ітерованих функцій однозначно зіставляється нерухома точка — зображення. Таким чином, процес компресії полягає в пошуку коефіцієнтів системи, а процес декомпресії — в проведенні ітерацій системи для отримання нерухомої точки. Області R_i в подальшому будуть називатися ранговими, а області D_i — доменними.

3.4 Рекурсивний (хвильовий) алгоритм

Англійська назва рекурсивного стиснення — wavelet. Воно також перекладається як хвильове стиснення, і як стиснення з використанням сплесків. Цей вид архівації відомий досить давно і безпосередньо виходить з ідеї використання когерентності областей. Орієнтований алгоритм на кольорові і чорно-білі зображення з плавними переходами. Ідеальний для картинок типу рентгенівських знімків. Коефіцієнт стиснення задається і варіюється в межах 5-100 разів. При спробі задати більший коефіцієнт на різких межах, які особливо проходять по діагоналі, проявляється "сходовий ефект" — сходинки різної яскравості, розміром в декілька пікселів.

Ідея алгоритму полягає в тому, що зберігаємо в файл різницю число між середніми значеннями сусідніх блоків в зображенні, яка зазвичай приймає значення близькі до 0.

Так два числа a_{2i} і a_{2i+1} завжди можна представити у вигляді $b^1_i=(a_{2i}+a_{2i+1})/2$ і $b^2_i=(a_{2i}-a_{2i+1})/2$. Аналогічно послідовність a_i може бути попарно переведена в послідовність $b^{1,2}_i$.

Розберемо конкретний приклад: нехай стискаємо рядок з 8 значень яскравості пікселів (a_i): (220, 211, 212, 218, 217, 214, 210, 202). Отримаємо такі послідовності b^1_i, і b^2_i: (215.5, 215, 215.5, 206) і (4.5, -3, 0.5, 4). Зауважимо, що значення b^2_i досить близькі до 0. Повторимо операцію, розглядаючи b^1_i як a_i. Дана дія виконується як би рекурсивно, звідки і назва алгоритму. Отримаємо з (215.5, 215, 215.5, 206): (215.25, 210.75) (0.25, 4.75). Отримані коефіцієнти, округливши до цілих і стиснувши, наприклад, за допомогою алгоритму Хаффмана з фіксованими таблицями, можемо помістити в файл.

Зауважимо, що застосовували наше перетворення до ланцюжку тільки два рази. Реально можемо дозволити собі застосування wavelet- перетворення 4-6 разів. Більш того, додаткове стиснення можна отримати, використовуючи таблиці алгоритму Хаффмана з нерівномірним кроком (тобто доведеться зберігати код Хаффмана для найближчого в таблиці значення). Ці прийоми дозволяють досягти помітних коефіцієнтів стиснення.

Алгоритм для двовимірних даних реалізується аналогічно. Якщо є квадрат з 4 точок з яркостями $a_{2i,2j}$, $a_{2i+1,\,2j}$, $a_{2i,\,2j+1}$, і $a_{2i+1,\,2j+1}$, то

$$
\begin{aligned}
b^1_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} + a_{2i,2j+1} + a_{2i+1,2j+1})/4,\\
b^2_{i,j} &= (a_{2i,2j} + a_{2i+1,2j} - a_{2i,2j+1} - a_{2i+1,2j+1})/4,\\
b^3_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} + a_{2i,2j+1} - a_{2i+1,2j+1})/4,\\
b^4_{i,j} &= (a_{2i,2j} - a_{2i+1,2j} - a_{2i,2j+1} + a_{2i+1,2j+1})/4.
\end{aligned}
\qquad (3.12)
$$

Використовуючи ці формули для зображення 512x512 пікселів отримаємо після першого перетворення 4 матриці розміром 256x256 елементів (рис.3.5).

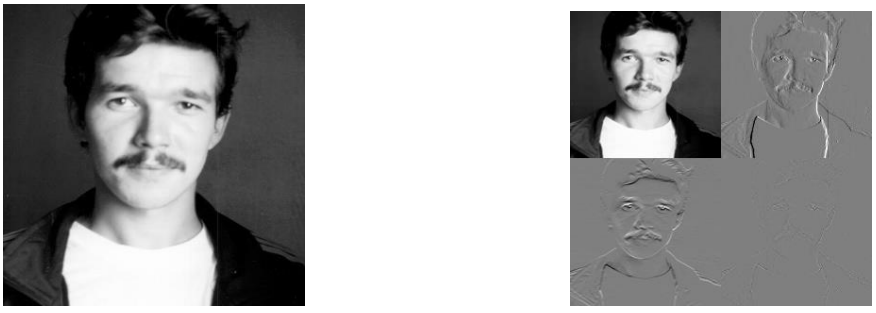


Рисунок 3.5 — Перетворення зображення

У першій, як легко здогадатися, буде зберігатися зменшена копія зображення. У 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, можемо повторити перетворення і отримати замість першої матриці 4 матриці розміром 128x128. Повторивши перетворення в третій раз, отримаємо в результаті: 4 матриці 64x64, 3 матриці 128x128 і 3 матриці 256x256. На практиці при записі в файл, значеннями, отриманими в останньому рядку ($b^4_{i,j}$), зазвичай нехтують (відразу отримуючи виграш приблизно на третину розміру файлу — 1 1/4 - 1/16 - 1/64 ...).

До переваг цього алгоритму можна віднести те, що він дуже легко дозволяє реалізувати можливість поступової "прояви" зображення при передачі зображення по мережі. Крім того, оскільки на початку зображення фактично зберігаємо його зменшену копію, що спрощує показ "огрубіння" зображення по заголовку.

На відміну від JPEG та фрактального алгоритму даний метод не оперує блоками, наприклад, 8x8 пікселів. Точніше оперуємо блоками 2x2, 4x4, 8x8 і т.д. Однак за рахунок того, що коефіцієнти для цих блоків зберігаємо незалежно, можемо досить легко уникнути дроблення зображення на "мозаїчні" квадрати.

4 АНАЛІЗ МЕТОДІВ СТИСНЕННЯ ІНФОРМАЦІЇ

4.1 Загальний підсумок аналізу алгоритмів стиснення зображень

Існує досить багато алгоритмів стиснення зображень, але роздивимось найбільш універсальні.

Алгоритм RLE розрахований на ділову графіку - зображення з великими областями повторюваного кольору. Орієнтований алгоритм на зображення з невеликою кількістю кольорів: ділову і наукову графіку.

До позитивних сторін алгоритму відноситься тільки те, що він не вимагає додаткової пам'яті при архівації та розархівації, а також швидко працює. Особливість групового кодування полягає в тому, що ступінь архівації для деяких зображень може бути істотно підвищена всього лише за рахунок зміни порядку кольорів в палітрі зображення.

Алгоритм LZW реалізований в форматах GIF та TIFF. Коефіцієнти компресії приблизно 1000, 4, 5/7 (кращий, середній, гірший коефіцієнти). Стиснення в 1000 разів досягається тільки на одноколірних зображеннях розміром кратним приблизно 7 Мб (6.918 ...). Орієнтований LZW на 8-бітові зображення, побудовані на комп'ютері. Стискає за рахунок однакових підланцюжків в потоці.

Алгоритм Хаффмана практично не застосовується до зображень в чистому вигляді, а використовується як один з етапів компресії в більш складних схемах. Єдиний алгоритм, який не збільшує розміру вихідних даних в гіршому випадку, якщо не брати до уваги необхідність зберігати таблицю перекодування разом з файлом.

JPEG ― один з найновіших і потужних алгоритмів. Практично є стандартом для повнокольорових зображень. Оперує алгоритм областями 8x8, на яких яскравість і колір змінюються порівняно плавно. Внаслідок цього, при розкладанні матриці такої області в подвійний ряд по косинусам значущими виявляються тільки перші коефіцієнти.

Таким чином, стиснення в JPEG здійснюється за рахунок плавності зміни кольорів у зображенні. Як стандарт ISO JPEG широко використовується при обміні зображеннями в комп'ютерних мережах. Підтримується алгоритм JPEG в форматах Quick Time, PostScript Level 2, Tiff 6.0. Один з найпоширеніших алгоритмів в системах мультимедіа. . Найбільшого поширення JPEG отримав в цифровій фотографії і для зберігання і передачі зображень з використанням мережі Інтернет.

Фрактальний алгоритм. Основним завданням при компресії фрактальним алгоритмом є знаходження відповідних афінних перетворень. У найзагальнішому випадку можемо перекладати будь-які за розміром і формою області зображення, проте в цьому випадку виходить дуже велика кількість варіантів, які перебираються, різних фрагментів, яку, на поточний момент, неможливо обробити навіть на найпотужнішому комп'ютері.

Виходячи з усіх розглянутих алгоритмів найбільш універсальними ялвяються алгоритм LZW, який реалізує стиснення без втрат, і алгоритм JPEG, який реалізує стиснення з втратами. Ще одним важливим фактором є активне використання інтернету, в якому передача зображень виконується за порівняно повільним каналам зв'язку, використання Interlaced GIF (алгоритм LZW) і Progressive JPEG (варіант алгоритму JPEG), що реалізують можливість показу зображень низької якості та preview, різко зросла.

4.2 Програмна реалізація

У ході дослідження було виявлено, що неможливо говорити про якийсь алгоритм як найкращий, тому що ефективність кожного алгоритму залежить від класу зображення. Тому з усіх розглянутих алгоритмів були відібрані два LZW та JPEG, як найбільш універсальні та ефективні в першу чергу для використання в інтернеті.

Для порівнення результатів стиснення алгоритмами JPEG та LZW було реалізовано програму, яка визначає відмінності між оригінальним зображенням та зображенням, яке було стиснуто. За допомогою реалізованої програми був проведен практичний аналіз.

Для дослідження було обрано технологію C#. Була розроблена форма, яка дозволяє завантажувати два зображення та будувати гістограму для порівнення двох зображень - оригіналу та стисненого.

Візьмемо 2 зображення формату TIFF (рис.4.1).

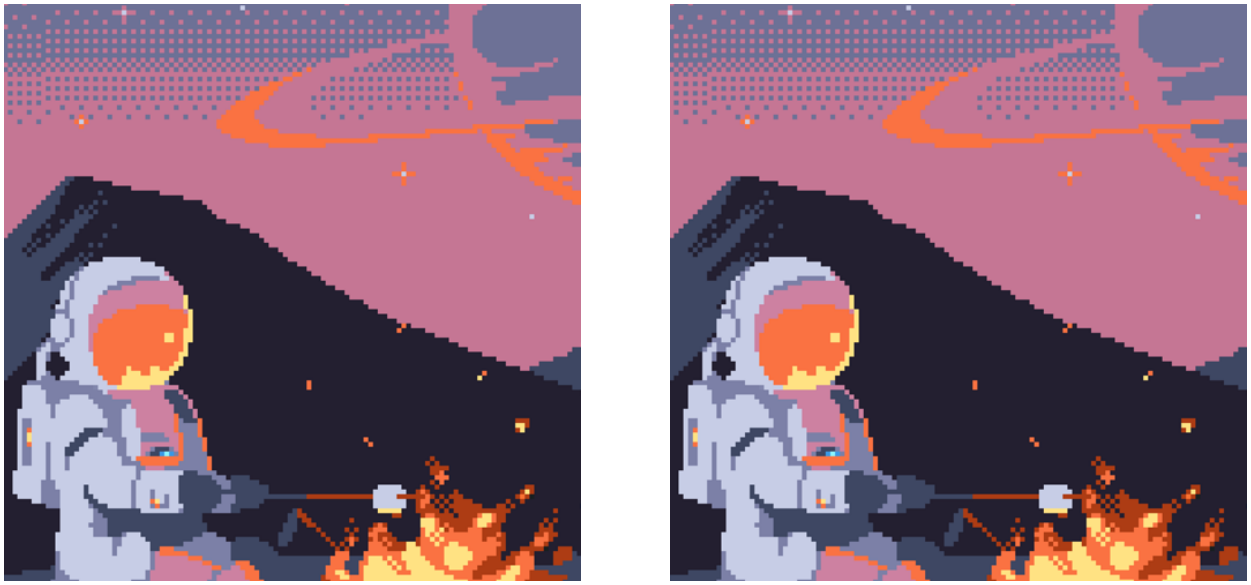


Рисунок 4.1 — Зліва зображення у форматі TIFF без стиснення, справа зображення у форматі TIFF стиснуто алгоритмом LZW

Зображення зліва, у якому не використовувались алгоритми стиснення, важить 755кб, зображення справа стиснуте алгоритмом LZW важить 125кб. Явних відмінностей не помічено. Проаналізуємо ці зображення у програмі.

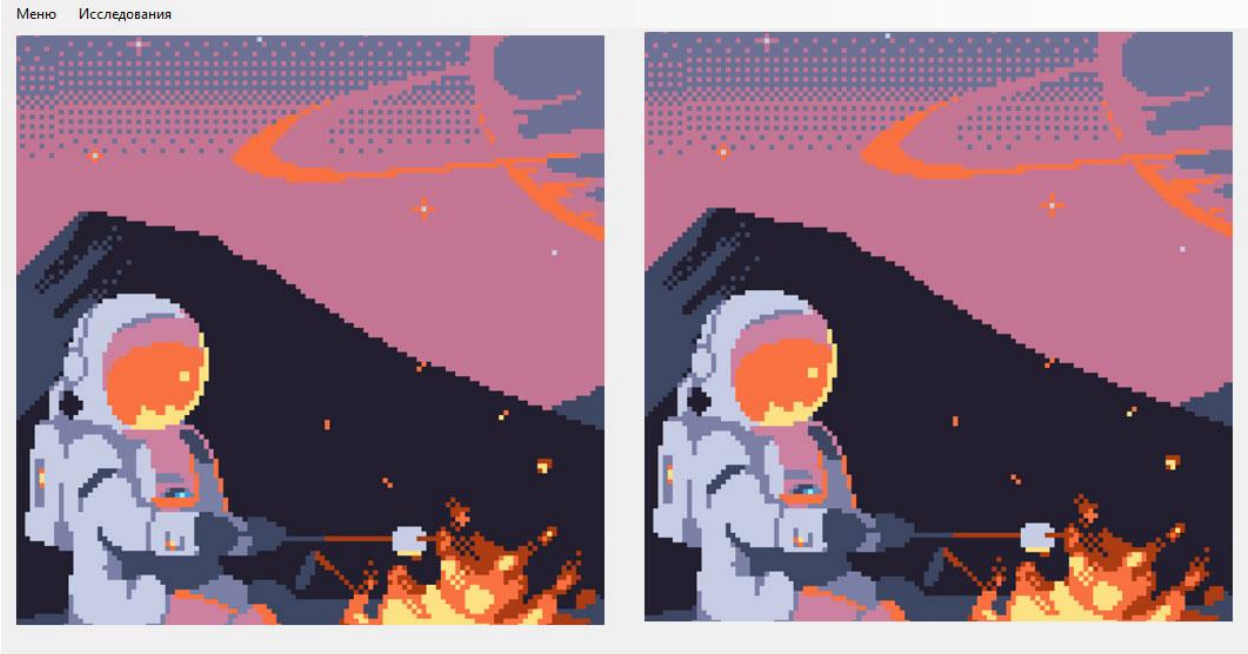


Рисунок 4.2 — Завантаження двух зображень у програмі для порівняння відмінностей

Тепер виконаємо порівнення цих зображень та побудову гістограми .

На рисунку 4.3 зображено зліва чорне зображення, що вказує на відсутність відміностей, також на рисунку 4.4 зображено гістограму, на якій також бачимо відсутність відмінних пікселів.

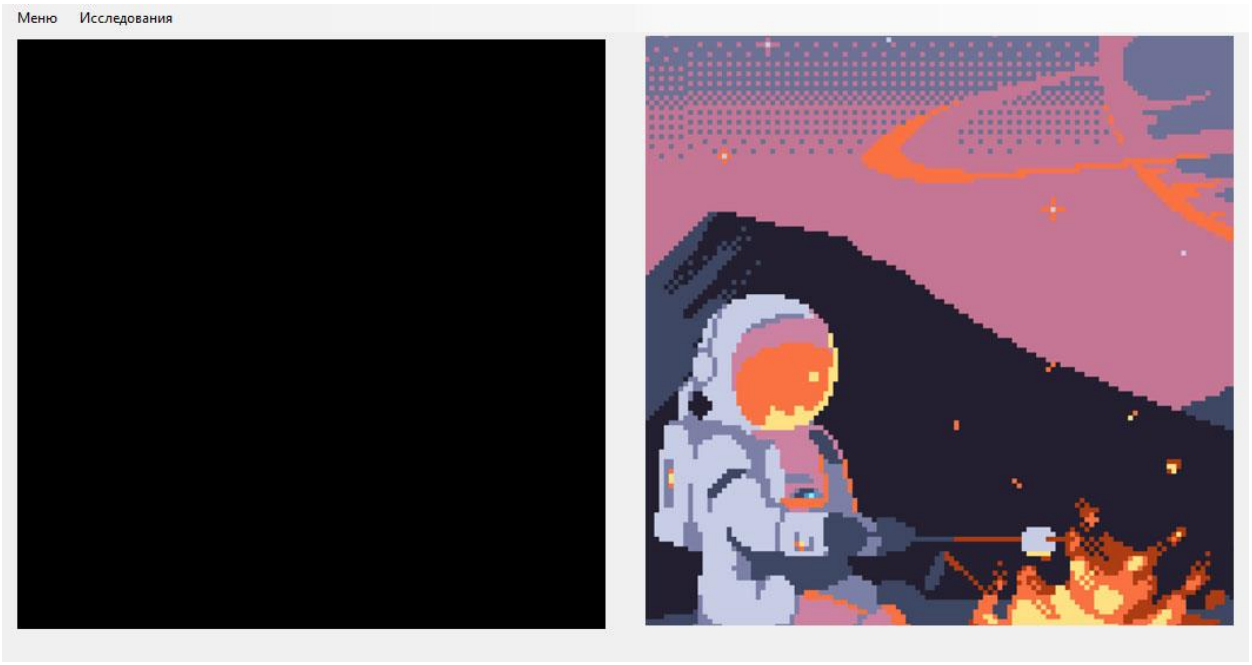


Рисунок 4.3 — Зліва зображено чорний рисунок, що вказує на відсутність відмінностей зображень

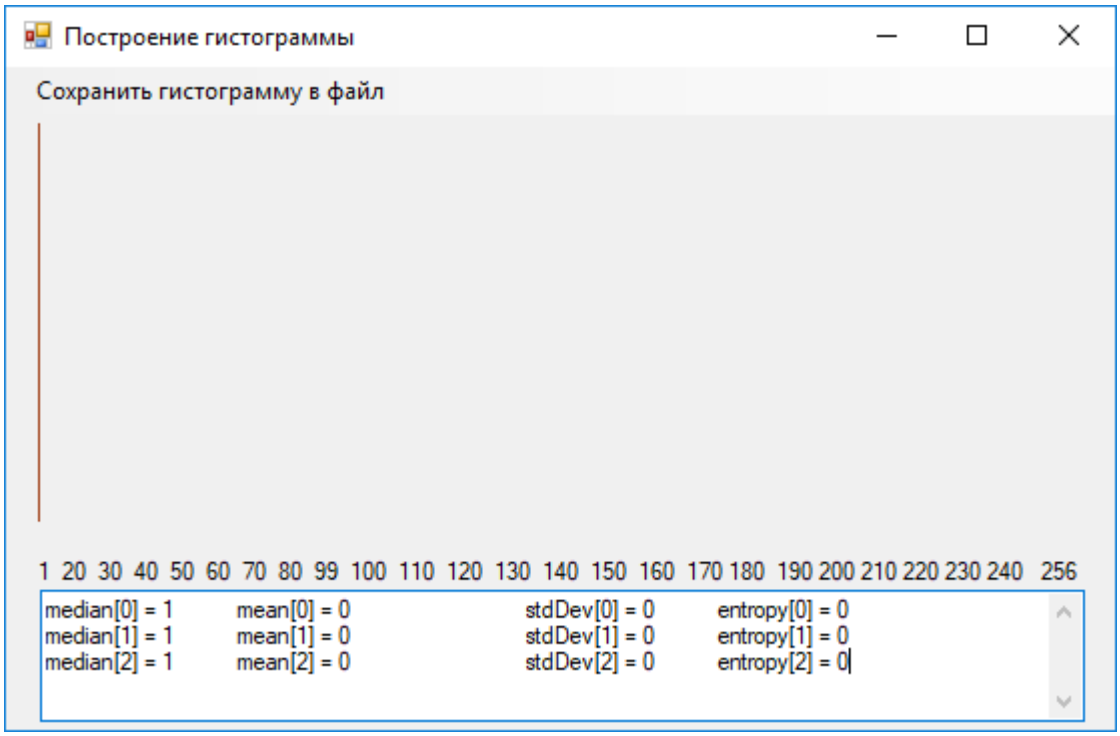


Рисунок 4.4 — Гістограма відмінностей

Стиснемо оригінальне зображення алгоритмом JPEG.

На рисунку 4.5 зображення стиснуте алгоритмом JPEG, яке важить 62.6кб, що відмінно від LZW стиснення на 60кб менше. На перший погляд це зображення не має явних відмінностей від оригіналу, тому перевіремо відмінності у програмі.

Рисунок 4.5 — Зображення стиснуте алгоритмом JPEG

На рисунку 4.6 зліва зображено відміності стиснутого зображення та оригіналу, а на рисунку 4.7 гістограму, яка вказує на наявність відмінних пікселів. Результат показує, що ступінь стиснення зображення алгоритмом JPEG набагато більший ніж у LZW, але зображення не зберігає оригінальної якості та стискається з втратами.

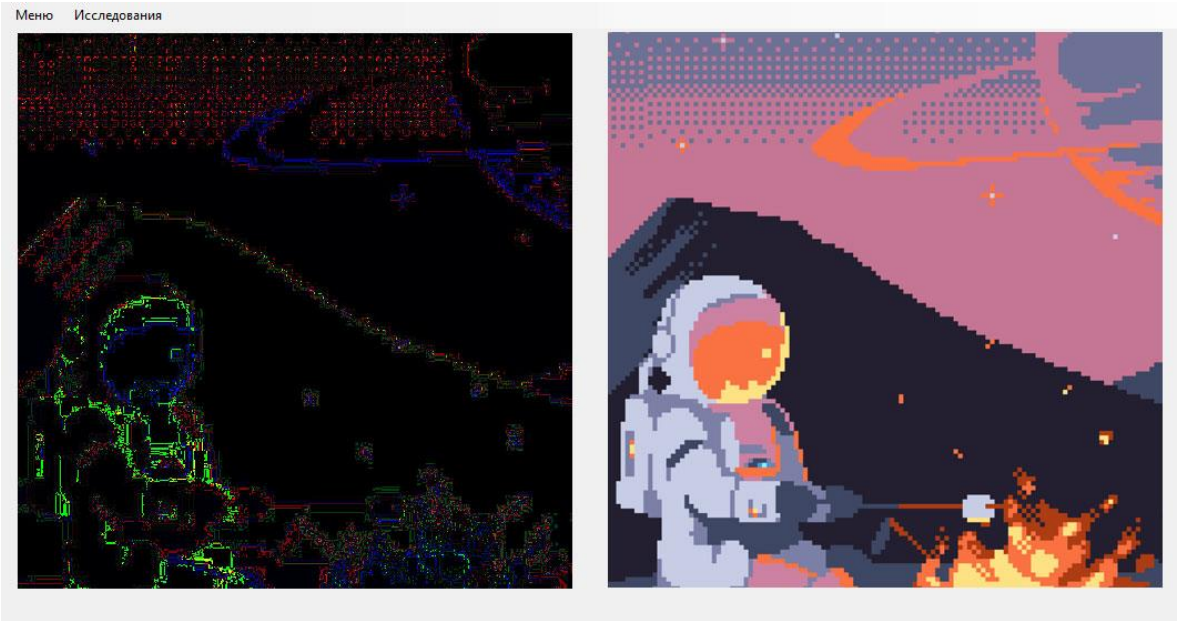


Рисунок 4.6 — Відмінності оригінального зображення та зображення стиснутого алгоритмом JPEG

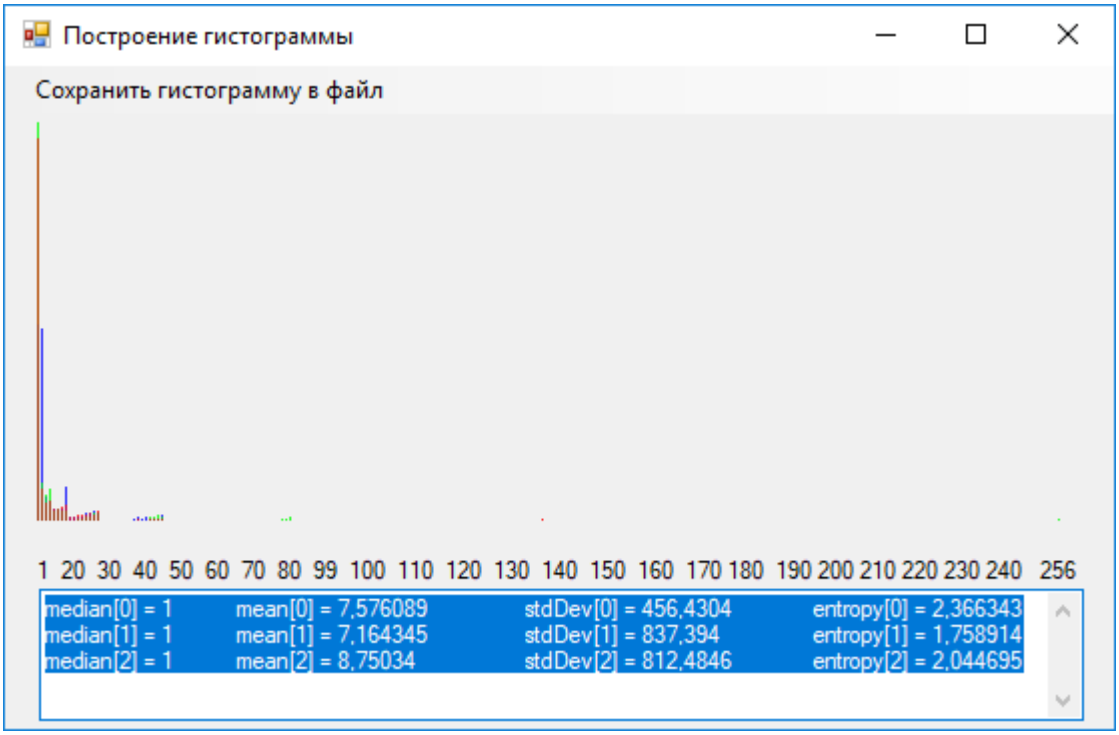


Рисунок 4.7 — Гістограма відмінностей

Тепер проведемо такий же аналіз, але з використанням більш багатобарвного зображення з плавністю зміни кольорів.

Рисунок 4.8 — Оригінал багатобарвного зображення у форматі TIFF без використання алгоритмів стиснення

Для наявності видимих відмінностей було застосовано максимальний ступінь стиснення JPEG. На рисунку 4.9 помітно відмінності у втратах якості від оригінального зображення 4.8. Оригінальне зображення важить 756кб, а стиснене 41.2кб. Це говорить про найсильнішу сторону алгоритма JPEG, він стиснув складне зображення майже у 18 разів. Проаналізуємо відмінності цих зображень у програмі та побудуємо гістограму.

Рисунок 4.9 — Зображення стиснуте алгоритмом JPEG

На рисунку 4.10 зліва зображено відмінності стиснутого зображення та оригіналу. Дуже велика наявність відмінностей по всьому зображенню вказує на те, що стиснення в JPEG здійснюється за рахунок плавності зміни кольорів у зображенні. Тому у цьому прикладі їх набагато більше ніж у попередньому (рисунок 4.6). Також зображення гістограми на рисунку 4.11 вказує багато відмінних областей на зображенні.

.

Рисунок 4.10 — Відмінності оригінального зображення
та стиснутого

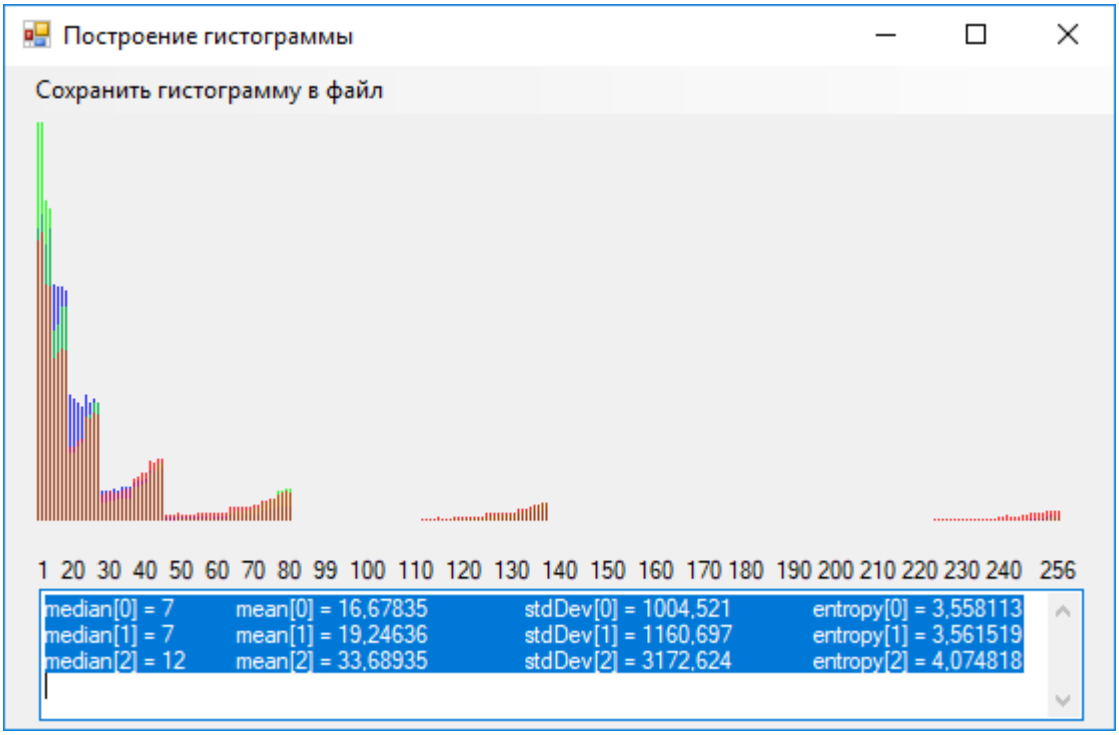


Рисунок 4.11 — Гістограма відмінностей

Тепер використаємо це зображення для стиснення LZW.

На рисунку 4.12 зображення стиснуте алгоритмом LZW немає жодних відмінностей від оригіналу. У цьому можна переконатися за допомогою програми (дод. А).

Рисунок 4.12 — Зображення стиснуте алгоритмом LZW

На рисунку 4.13 зображено відсутність відмінностей. Це підтверджує й гістограма (дод. Б). Але стиснуте зображення важе 560кб на відміну від зображення стиснутого алгоритмом JPEG, яке важить 41.2кб.

Рисунок 4.13 — Відсутність відмінностей оригінального зображення та стиснутого

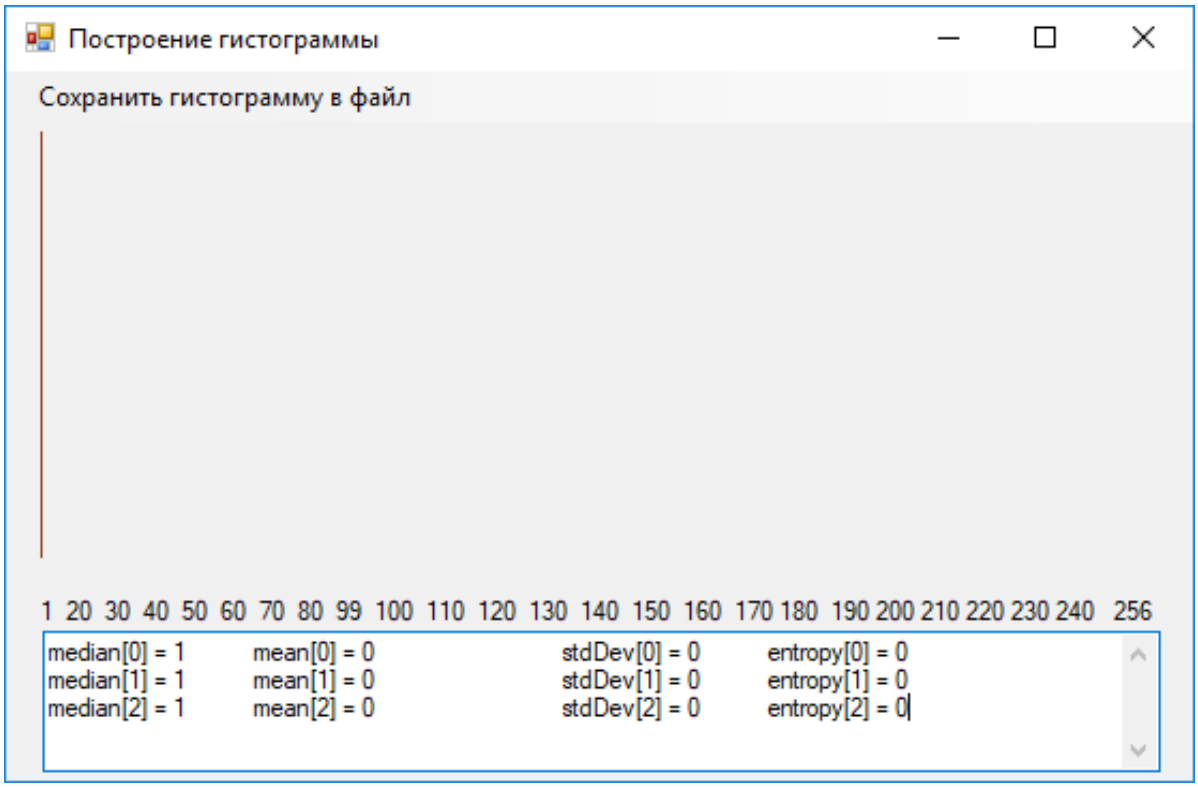


Рисунок 4.14 — Дані гістограми вказують на відсутність відмінних пікселів

Як показує дослідження, алгоритм LZW гарантує високий ступінь стиснення без втрат у зображеннях без наявності плавного переходу кольорів, а також дає дуже низкий сутпінь стиснення у багатокольорових зображеннях з плавністю зміни кольорів.

JPEG завжди гарантує високий ступінь стиснення, вищий ніж у LZW, але стиснення завжди відбувається з втратами. Алгоритм JPEG найкраще підходе для реалістичних зображень з плавними переходами яскравості і кольору.

5 ОХОРОНА ПРАЦІ ТА БЕЗПЕКА В НАДЗВИЧАЙНИХ СИТУАЦІЯХ. ЕКОЛОГІЯ

В даному розділі проведено аналіз потенційних небезпечних та шкідливих виробничих факторів. Розглянуті умови, що дозволяють забезпечити гігієну праці, виробничу санітарію. Розроблено заходи з техніки безпеки та рекомендації з пожежної профілактики. Аналіз потенційно небезпечних і шкідливих виробничих чинників виконується для робочих умов, з використанням персонального комп'ютера, на якому буде йти розробка.

5.1 Загальні питання з охорони праці

Умови праці на робочому місці, безпека технологічних процесів, машин, устаткування та інших засобів виробництва, що використовуються працівником, а також санітарно-побутові умови повинні відповідати вимогам нормативних актів про охорону праці. В законі України «Про охорону праці» визначається, що охорона праці - це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності.

5.2 Аналіз стану умов праці

Робота над проектом проходитиме в приміщенні багатоквартирного будинку. Для даної роботи достатньо однієї людини, для якої надано робоче місце зі стаціонарним комп’ютером.

5.2.1 Вимоги до приміщень

Геометричні розміри приміщення зазначені в табл. 5.1.

Таблиця 5.1 – Розміри приміщення

Найменування	**Значення**
Довжина, м	5
Ширина, м	5
Висота, м	3
Площа, $м^2$	25
Об’єм, $м^3$	75

Згідно з [30] розмір площі для одного робочого місця оператора персонального

комп'ютера має бути не менше 6 кв. м, а об'єм — не менше 20 куб. м. Отже, дане приміщення цілком відповідає зазначеним нормам.

5.2.2 Вимоги до організації місця праці

При порівнянні відповідності характеристик робочого місця нормативним основні вимоги до організації робочого місця за [28] (табл. 5.2) і відповідними фактичними значеннями для робочого місця, констатуємо повну відповідність.

Таблиця 5.2 - Характеристики робочого місця

Найменування параметра	Фактичне значення	Нормативне значення
Висота робочої поверхні, мм	750	680 ÷800
Висота простору для ніг, мм	730	не менше 600
Ширина простору для ніг, мм	660	не менше 500
Глибина простору для ніг, мм	700	не менше 650
Висота поверхні сидіння, мм	470	400 ÷500
Ширина сидіння, мм	400	не менше 400
Глибина сидіння, мм	400	не менше 400
Висота поверхні спинки, мм	600	не менше 300
Ширина опорної поверхні спинки, мм	500	не менше 380
Радіус кривини спинки в горизонтальній площині, мм	400	400
Відстань від очей до екрану дисплея, мм	800	700 ÷800

Температура в приміщенні протягом року коливається у межах 18–24°С, відносна вологість — близько 50%. Швидкість руху повітря не перевищує 0,2 м/с. Шум в лабораторії знаходиться на рівні 50 дБА. Система вентилювання приміщення — природна неорганізована, а опалення — централізоване.

5.2.3 Навантаження та напруженість процесу праці

За фізичним навантаженням робота відноситься до категорії легкі роботи (Іа), її виконують сидячи з періодичним ходінням. Щодо характеру організування виконання дипломної роботи, то він підпадає під нав'язаний режим, оскільки певні розділи роботи необхідно виконати у встановлені конкретні терміни.

Роботу за дипломним проектом визнано, такою, що займає 50% часу робочого дня та за восьмигодинної робочої зміни рекомендовано встановити додаткові регламентовані перерви - для розробників програм тривалістю 15 хв. через кожну годину роботи;

5.3 Виробнича санітарія

На підставі аналізу небезпечних та шкідливих факторів при виробництві (експлуатації), пожежної безпеки можуть бути надалі вирішені питання необхідності забезпечення працюючих достатньою кількістю освітлення, вентиляції повітря, організації заземлення, тощо.

5.3.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Аналіз небезпечних та шкідливих виробничих факторів виконується у табличній формі (табл. 5.3). Роботу, пов'язану з ЕОП з ВДТ, у тому числі на тих, які мають робочі місця, обладнані ЕОМ з ВДТ і ПП, виконують із забезпеченням виконання НПАОП 0.00.-1.28-10 [36], які встановлюють вимоги безпеки до обладнання робочих місць, до роботи із застосуванням ЕОМ з ВДТ і ПП. Основними робочими характеристиками персонального комп'ютера є:

- робоча напруга U=+220В +-5%;
- робочий струм I=2А;
- споживана потужність Р=350 Вт.

Таблиця 5.3 – Аналіз небезпечних і шкідливих виробничих факторів

Небезпечні і шкідливі виробничі фактори	Джерела факторів (види робіт)	Кількісна оцінка	Нормативні документи
1	2	3	4
Фізичні			
- підвищена температура поверхонь обладнання	експлуатація ЕОМ, принтерів, сканерів чи/або серверного обладнання для роботи	2	[30]
- підвищений рівень шуму на робочому місці	-//-	2	[29]
- підвищений рівень вібрації	-//-	2	[30] [43]
- підвищена або знижена вологість повітря	-//-	2	[30]
- підвищена або знижена рухливість повітря	-//-	1	[30]
- підвищений рівень іонізуючого випромінення в робочій зоні	-//-	2	[30] [40]

Продовження таблиці 5.3

1	2	3	4
- підвищений рівень електромагнітного випромінення	-//-	2	[40]
- підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини	-//-	4	[41] [30]
- підвищений рівень статичної електрики	-//-	2	[41]
- підвищена напруженість електричного поля	-//-	2	[40]
- підвищена напруженість магнітного поля	-//-	2	[40]
- недостатність природного світла	порушення умов праці (вимог до приміщень)	2	[27]
- недостатнє освітлення робочої зони	порушення гігієнічних параметрів виробничого середовища	3	[27]
- підвищена яскравість світла	порушення умов праці (організації місця праці-налагодження моніторів)	1	[28]
- понижена контрастність	-//-	1	[28]
психофізіологічні:			
- нервово-психічна перевантаження (розумове, перенапруження аналізаторів-зорових)	- пошук інформації для постановки теми; - пошук та аналіз аналогів і літератури; - пошук наявних технологій, моделювання та аналіз алгоритмів; - виконання роботи за темою диплома, тестування; - оформлення роботи	4	[42] [28]
- фізичні (статичне сидіння)	порушення умов праці (організації місця праці - сидіння користувача) та організації робочого часу - безпервна робота)	2	[42] [28]

Робочі місця мають відповідати вимогам Державних санітарних правил і норм роботи з візуальними дисплейними терміналами електронно-обчислювальних машин, затверджених постановою Головного державного санітарного лікаря України від 10.12.98 N 7 [28].

5.3.2 Пожежна безпека

Небезпека розвитку пожежі на обчислювальному центрі обумовлюється застосуванням розгалужених систем електроживлення ЕОМ, вентиляції і кондиціювання.

Запобігти утворенню горючого середовища (замінити горючі речовини і матеріали на негорючі і важкогорючі) не надається технічно можливим. Тому проектом передбачаються способи і засоби запобігання утворення (або внесення) в горюче середовище джерел запалювання, таких як:

1) застосування електроустаткування, відповідної пожежонебезпечної і вибухонебезпечної зонам відповідно до ПУЕ;

2) застосування в конструкції швидкодійних засобів захисного відключення можливих джерел запалення;

3) виключення можливості появи іскрового розряду в горючому середовищі з енергією, рівної і вище мінімальної енергії запалення.

Згідно [34] таке приміщення, площею 25 $м^2$, відноситься до категорії "В" (пожежонебезпечної) та для протипожежного захисту в ньому проектом передбачено устаткування автоматичною пожежною сигналізацією із застосуванням датчиків-сповіщувачів РІД-1 (сповіщувач димовий ізоляційний) в кількості 1 шт., і застосуванням первинних засобів пожежогасіння.

Простори усередині приміщень в межах, яких можуть утворюватися або знаходиться пожежонебезпечні речовини і матеріали відповідно до [34] відносяться до пожежонебезпечної зони класу П-IIа. Це обумовлено тим, що в приміщенні знаходяться тверді горючі та важкозаймисті речовини та матеріали. Приміщенню, у якому розташоване робоче місце, присвоюється II ступень вогнестійкості.

5.3.3 Електробезпека

На робочому місці виконуються наступні вимоги електробезпеки: ПК, периферійні пристрої та устаткування для обслуговування, електропроводи і кабелі за виконанням та ступенем захисту відповідають класу зони за ПУЕ (правила улаштування електроустановок), мають апаратуру захисту від струму короткого замикання та інших аварійних режимів. Лінія електромережі для живлення ПК, периферійних пристроїв і устаткування для обслуговування, виконана як окрема групова три- провідна мережа, шляхом прокладання фазового, нульового робочого та нульового захисного провідників. Нульовий захисний провідник використовується для заземлення (занулення) електроприймачів. Штепсельні

з'єднання та електророзетки крім контактів фазового та нульового робочого провідників мають спеціальні контакти для підключення нульового захисного провідника. Електромережа штепсельних розеток для живлення персональних ПК укладено по підлозі поруч зі стінами відповідно до затвердженого плану розміщення обладнання та технічних характеристик обладнання. Металеві труби та гнучкі металеві рукави заземлені. Захисне заземлення включає в себе заземлюючих пристроїв і провідник, який з'єднує заземлюючий пристрій з обладнанням, яке заземлюється - заземлюючий провідник.

5.4 Гігієнічні вимоги до параметрів виробничого середовища

5.4.1 Мікроклімат

Мікроклімат робочих приміщень – це клімат внутрішнього середовища цих приміщень, що визначається діючої на організм людини з'єднанням температури, вологості, швидкості переміщення повітря.

Дане приміщення обладнане системами опалення, кондиціонування повітря або припливно-витяжною вентиляцією. У приміщенні на робочому місці забезпечуються оптимальні значення параметрів мікроклімату: температури, відносної вологості й рухливості повітря у відповідності до [29]. Рівні позитивних і негативних іонів у повітрі мають відповідати [29].

Контроль параметрів мікроклімату в холодний і теплий період року здійснюється не менше 3-х разів на зміну (на початку, середині, в кінці).

5.4.2 Освітлення

Світло є природною умовою існування людини. Воно впливає на стан вищих психічних функцій і фізіологічні процеси в організмі. Хороше освітлення діє тонізуюче, створює гарний настрій, покращує протікання основних процесів вищої нервової діяльності.

У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення[27]. Джерелом природного освітлення є сонячне світло. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН і для даного приміщення в світлий час доби достатньо природного освітлення.

Розрахунок освітлення.

Для виробничих та адміністративних приміщень світловий коефіцієнт приймається не менше -1/8, в побутових – 1/10:

$$S_b = \left(\frac{1}{5} \div \frac{1}{10}\right) \cdot S_n, \quad (5.1)$$

де S_b – площа віконних прорізів, м^2;

S_n – площа підлоги, м^2.

$S_n = a \cdot b = 5 \cdot 5 = 25$ м^2,

$S = 1/8 \cdot 25 = 3{,}125$ м^2.

Приймаємо 2 вікна площею S=1,6 м^2 кожне.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні.

Розрахунок кількості світильників n виробляється по формулі (5.2):

$$n = \frac{E \cdot S \cdot Z \cdot K}{F \cdot U \cdot M}, \quad (5.2)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300 лк;

S – освітлювана площа, м^2; S = 25 м^2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (А.2), отримуємо:

$$n = \frac{300 \cdot 25 \cdot 1{,}1 \cdot 1{,}5}{5400 \cdot 0{,}575 \cdot 2} \approx 2{,}0$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

5.4.3 Шум та вібрація, електромагнітне випромінювання

Рівень шуму, що супроводжує роботу користувачів персональних комп'ютерів, коливається у межах 50–65 дБА [29]. У залах комп'ютерного набору рівні шуму не повинні перевищувати 65 дБА.

Віброізоляція можливо здійснювати за допомогою спеціальної прокладки під системний блок, який послаблює передачу вібрацій робочому столу. Вібрація на робочому місці, що розглядається, відповідає нормам [29].

5.4.4 Вентилювання

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в СНіП.

Також має здійснюватися провітрювання приміщення, в залежності від погодних умов, тривалість повинна бути не менше 10 хв. Найкращий обмін повітря здійснюється при наскрізному провітрюванні.

5.5 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Відповідно до санітарно-гігієнічних нормативів та правил експлуатації обладнання наводимо приклади деяких заходів безпеки.

Заходи безпеки під час експлуатації персонального комп'ютера та периферійних пристроїв передбачають:

- правильне організування місця праці та дотримання оптимальних режимів праці та відпочинку під час роботи з ПК;
- експлуатацію сертифікованого обладнання;
- дотримання заходів електробезпеки;
- забезпечення оптимальних параметрів мікроклімату;
- забезпечення раціонального освітлення місця праці;
- облаштовуючи приміщення для роботи з ПК, потрібно передбачити припливно-витяжну вентиляцію або кондиціювання повітря.

Вимоги безпеки при надзвичайних ситуаціях:

1) При раптовому припиненні подачі електричної енергії вимкнути ПК. Витягнути вилки з розеток. При наявності ознак горіння (дим, запах горілого) необхідно вимкнути ПК, знайти місце загоряння і виконати всі можливі заходи для його ліквідації.

2) При замиканні, перевантаженні електричного струму на електричному обладнанні, внаслідок ураження грозової блискавки та ймовірної небезпеки ураженням електричним струмом, приймають наступне:

- попередження замикання здійснюється правильним вибором, монтажем експлуатації мереж;

- застосування захисту схем у вигляді швидкодіючих реле, а також вимикачів, плавких запобіжників.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Згідно з класифікацією приміщень за ступенем небезпеки ураження електричним струмом [35], приміщення в якому проводяться всі роботи відноситься до першого класу (без підвищеної небезпеки). Коефіцієнт використання вертикальних заземлювачів $\eta_в$ в залежності від розміщення заземлювачів та їх кількості знаходиться в межах 0,4…0,99. Взаємну екрануючу дію горизонтального заземлювача (з'єднувальної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача η_c.

Послідовність розрахунку.

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з.} = \frac{R_д \cdot R_{пр.з.}}{R_{пр.з.} - R_д}, \qquad (5.3)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з.}=R_д$.

Підставивши числові значення у формулу (А.3), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом·м. Приблизне значення питомого опору глини приймаємо ρ=40 Ом•м (табличне значення).

3) Розрахунковий питомий опір ґрунту, $\rho_{розр}$, Ом·м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом·м за формулою:

$$\rho_{розр..} = \psi \cdot \rho, \qquad (5.4)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів I кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом·м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом•м}$$

$$\rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом•м}$$

4) Розраховується опір розтікання струму вертикального заземлювача $R_в$, Ом, за (5.5).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (5.5)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2–3 м; $l_в$=3 м);

$d_{ст}$ – діаметр стержня (для труб - 0,03–0,05 м; $d_{ст}$=0,05 м);

t – відстань від поверхні землі до середини заземлювача, яка визначається за ф. (5.6):

$$t = h_в + \frac{l_в}{2}, \quad (5.6)$$

де $h_в$ – глибина закладання вертикальних заземлювачів (0,8 м); тоді $t = 0,8 + \frac{3}{2} = 2,3$ м

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

5) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25 \quad (5.7)$$

I визначається коефіцієнт використання вертикальних електродів групового заземлювача без врахування впливу з'єднувальної стрічки $\eta_в$ =0,57 (табличне значення).

6) Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16 \quad (5.8)$$

7) Визначається довжина з'єднувальної стрічки горизонтального заземлювача $l_с$, м:

$$l_c = 1,05 \cdot L_в \cdot (n_в - 1), \quad (5.9)$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти за $L_в$ = 3м);

$n_в$ – необхідна кількість вертикальних заземлювачів.

$$l_c = 1,05 \cdot 3 \cdot (16 - 1) \approx 48 \text{ м}$$

8) Визначається опір розтіканню струму горизонтального заземлювача (з'єднувальної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_c} \cdot \ln \frac{2 \cdot l_c^2}{d_{см} \cdot h_г}, \quad (5.10)$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ =0,95b, b = 0,15 м;

$h_г$ – глибина закладання горизантальних заземлювачів (0,5 м);

l_c - довжина з'єднувальної стрічки горизонтального заземлювача l_c, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1 \text{ Ом}$$

9) Визначається коефіцієнт використання горизонтального заземлювача $\eta_{c.}$ відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання з'єднувальної смуги η_c=0,3 (табличне значення).

10) Розраховується результуючий опір заземлювального електроду з урахуванням з'єднувальної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_c + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (5.11)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпеку будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

3) При виникненню пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як:

- іскри і дуги коротких замикань;
- перегрів провідників, резисторів та інших радіодеталей ПЕОМ, від тривалої перевантаження та наявність перехідного опору;
- іскри при розмиканні і розмиканні ланцюгів;
- розряди статичної електрики;

- необережному поводженню з вогнем, а також вибухи газо-повітряних і паро-повітряних сумішей.

-

5.6 Охорона навколишнього природного середовища

5.6.1 Загальні дані з охорони навколишнього природного середовища

Діяльність за темою магістерської роботи, процес виконання якої впливає на навколишнє природнє середовище, регламентується нормами діючого законодавства: Законом України «Про охорону навколишнього природного середовища», Законом України «Про забезпечення санітарного та епідемічного благополуччя населення», Законом України «Про відходи», Законом України «Про охорону атмосферного повітря», Законом України «Про захист населення і територій від надзвичайних ситуацій техногенного та природного характеру», Водний кодекс України.

Основним екологічним аспектом в процесі діяльності за даними спеціальностями є процеси впливу на атмосферне повітря та процеси поводження з відходами, які утворюються, збираються, розміщуються, передаються на знешкодження, утилізацію, тощо в ІТ галузі.

Вплив на атмосферне повітря при нормальних умовах праці відсутній, бо немає в приміщенні сканерів, принтерів та інших джерел викиду забруднюючих речовин в повітря робочої зони.

5.6.2 Вимоги до збору, пакування та розміщення відходів ІТ галузі

Наводяться вимоги зберігання виявлених за своєю роботою відходів відповідно до вимог Державних санітарних правил і норм ДСанПіН 2.2.7.029.

Відходи в міру їх накопичення збирають у тару, відповідну класу небезпеки, з дотриманням правил безпеки, після чого доставляють до місця тимчасового зберігання відходів відповідно до затвердженої схеми їх розміщення. Зазначені для зберігання відходів місця чи об'єкти повинні використовуватися лише для заявлених відходів.

Не допускається зберігання відходів у невстановлених схемою місцях, а також перевищення норм тимчасового зберігання відходів.

Не допускається змішування відходів різних видів і класів небезпеки з будівельними і побутовими відходами, відходами дерев'яної, металевої, синтетичної тари, відходами текстильних матеріалів (старий спецодяг, ганчірки) і ін.

Проведення заготовки, здачі, переробки та реалізації металобрухту встановлені окремо Законом України «Про металобрухт».

Особливий контроль наділяється збору і зберіганню відпрацьованих енергоощадних ламп як відходам I класу небезпеки, що збираються і обов’язково передаються на утилізацію підприємствам, що мають ліцензію на поводження з такими небезпечними відходами.

Всі відходи, що утворюються в процесі діяльності/роботи, підлягають обліку.

Під час роботи з відходами (прибирання виробничих приміщень, збір і сортування, навантаження, транспортування, розвантаження та ін.) працівники та обслуговуючий персонал підприємства повинні бути забезпечені засобами індивідуального захисту та дотримуватися вимог інструкцій з охорони праці, що діють на підприємстві.

Побутові та будівельні відходи вивозяться на полігон твердих побутових відходів міста, також відповідно до договору з комунальним дорожньо-експлуатаційним управлінням.

Особи, винні в порушенні встановленого порядку поводження з відходами (порушення правил обліку відходів, самовільне складування і видалення відходів, передача відходів в інші підприємства/організації з порушенням встановлених правил), згідно законодавства несуть дисциплінарну, адміністративну або кримінальну відповідальність.

5.6.3 Визначення впливу та заходів щодо поводження з відходами IT галузі

З метою визначення та прогнозування впливу відходів на навколишнє середовище, своєчасного виявлення негативних наслідків, їх запобігання відповідно до Закону України «Про відходи» повинен здійснюватися моніторинг місць утворення, зберігання, і видалення відходів. Відомості про місце утворення та місце розташування відходів зазначаються на «План схемі місці розміщення відходів організації/виробництва» та наводяться у таблиці В.2 в додатку В, а Відомості про склад і властивості відходів, що утворюються, а також ступінь їх небезпечності для навколишнього природного середовища та здоров'я людини у табл. В.3 додатку В.

Висновки до розділу 5

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над запропонованим проектом, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника. Приведені рекомендації щодо організації робочого місця, а також важлива інформацію щодо пожежної та електробезпеки. Були наведені розміри приміщення та наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

У результаті виконання магістерської роботи були досліджені алгоритми стиснення зображень з втратами та без втрат, а також була реалізована програма, за допомогою якої був проведен практичний аналіз. У ході дослідження було виявлено, що неможливо говорити про якийсь алгоритм як найкращий, тому що ефективність кожного алгоритму залежить від класу зображення. Тому з усіх розглянутих алгоритмів були відібрані два LZW та JPEG, як найбільш універсальні та ефективні в першу чергу для використання в Інтернеті.

З алгоритму LZW було виявлено наступні переваги:

– Не потребує обчислення ймовірностей народження символів або кодів.

– Для декомпресії не треба зберігати таблицю рядків в файл для розпакування. Алгоритм побудований таким чином, що можливо відновити таблицю рядків, користуючись тільки потоком кодів.

– Даний тип компресії не вносить спотворень в вихідний графічний файл і підходить для стиснення растрових даних будь-якого типу.

Головним недоліком є те, що цей алгоритм ефективний тільки для зображень, переважно створенних за допомогою комп'ютера, на яких немає плавності переходу кольорів. Основними зображеннями для цього алгоритму є 8-бітові зображення, креслення, текстова та знакова графіка.

Алгоритм JPEG ефективний для стиснення фотографій і картин, що містять реалістичні сцени з плавними переходами яскравості і кольору. Одним з найвижливіших факторів також є можливість вказувати ступінь стиснення зображення. Найбільшого поширення JPEG отримав в цифровій фотографії і для зберігання і передачі зображень з використанням мережі Інтернет.

Перелік посилань

1. Путятин, Е.П. Обработка изображений в робототехнике [Текст] / Е.П. Путятин, С.И. Аверин. – Москва: Машиностроение, 1990. – 320 с.

2. Гороховатский, В.А. Распознавание изображений в условиях неполной информации [Текст] / В.А. Гороховатский. – Харьков, 2003. – 112с.

3. Машталир, В.П. Точечно-множественные методы обработки информации [Текст] / В.П. Машталир. – Харьков: Бизнес Информ. – 2001. – 199 с.

4. Гонсалес Р., Цифрова обробка зображень [Текст] / Гонсалес Р., Р. Вудс - Москва, 2002, 813 с.:іл. – 3000 екз. – ISBN 978-5-94836-331-8

5. Хоггар Г., Математика цифрових зображень. Навчальний посібник [Текст] / Хоггар Г. – 2006, 853 с.:іл. – 1000 екз. - ISBN 978-5-904509-13-2

6. Максименко О., Алгоритм обробки цифрових зображень. Навчальний посібник [Текст] / Л. Франкевич, О. Сахарук, О. Максименко - 2013, 756 с.:іл. - ISBN 0201-18075-8

7. Fisher Y., Fractal image compression [Text] / Fisher Y., SigGraph 92 . – 1995, pp. 111-122.

8. Chamzas C. C., Progressive Bi-level Image Compression [Text] / C. C. Chamzas, I. Sebestyen, Revision 4.1 - September 16, 1991, pp. 10-23.

9. Блаттер К., Вейвлет-анализ. Основы теории. [Текст] / К. Блаттер – М. – Техносфера. - 2006, - 279 с.

10. Брейсуэлл Р., Преобразование Хартли. Теория и приложения. [Текст] / Брейсуэлл Р. - М. - Мир, - 1990. – 192 с.

11. Ватолин, Д.С. Методы сжатия данных. Устройство архиваторов, сжатие изображений и видео [Текст] / Ватолин Д.С., Ратушняк А., Смирнов М. Юкин В -М. – Диалог- МИФИ. -2003. - 384 с.

12. Тропченко А.А., Особенности сжатия цветных изображений JPEG-подобными алгритмами [Текст] / Тропченко А.А., Молчанов В.А. – Научно-технический вестник СПбГУ ИТМО/ - Вып. 32. - СПб. - СПбГУ ИТМО. – 2006. - с.22-26.

13. Pennebaker W.B., An overview of the basic principles of the Q-coder adaptive binary arithmetic coder [Text] / W.B. Pennebaker, J.L. Mitchell, G.G. Langdon, R.B. Arps, IBM Journal of research and development, Vol.32, No.6 – November 1988, pp. 771-726.

14. Huffman D.A., A method for the construction of minimum redundancy codes. [Text] / D.A. Huffman – In processing. IRE val.40 – 1962 pp. 1098-1101.

15. Ватолин, Д.С. Сжатие статических изображений [Текст] / Д.С. Ватолин. – Открытые системы, Номер 2, 1995. – 80 с.

16. Ватолин, Д.С. MPEG - стандарт ISO на видео в системах мультимедиа [Текст] / Д.С. Ватолин. – Открытые системы, Номер 2, 1995. – 76 с.

17. Вендров, А.М. CASE-технологии. Современные методы и средства проектирования информационных систем/ А.М. Вендров. - М.: Финансы и статистика, 1998.– 176 с.

18. Маклаков, С.В. BPWin и ERWin. Case-средства разработки информационных систем/ С.В.Маклаков-М.: ДИАЛОГ–МИФИ, 1999.–256с.

19. Орлов, С.А. Технологии разработки программного обеспечения/ С.А. Орлов– СПб.: Питер, 2002.–464 с.

20. Гарсиа-Молина Г., Ульман Дж., Уидом Дж. Системы баз данных. Полный курс / Гарсиа-Молина Г, Ульман Дж, Уидом Дж. — М.: "Вильямс", 2003. – 229 с.

21. Дейт. К. Дж. Введение в системы баз данных / К. Дж. Дейт. — "Вильямс", 2001. – 426 с.

22. Харрингтон Д. Л Проектирование реляционных баз данных. Просто и доступно / Д. Л. Харрингтон. – М.: ЛОРИ, 2000. – 277 с.

23. Коннолли Т. М, Бегг К. Базы данных. Проектирование, реализация и сопровождение. Теория и практика / Т. М. Коннолли, К. Бегг. – М.: Издательский дом "Вильямс", 2003. – 261 с.

24. Калянов Г. Н. CASE. Структурный системный анализ (автоматизация и применение) / Г. Н. Калянов. – М.: "Лори", 2006. – 175 с.

25. Черемных, С.В. Структурный анализ систем: IDEF-технологии. / С.В. Черемных, И.О.Семенов, В.С. Ручкин-М.: Финансы и статистика, 2003.–208 с.

26. ГОСТ 12.1.044-89 ССБТ. Пожежовибухонебезпека речовин і матеріалів. Номенклатура показників і методи їх визначенняя

27. ДБН В.2.5-28:2015 Природнє і штучне освітлення

28. ДСанПіН 3.3.2.007-98 Гігієнічні вимоги до організації роботи з візуальними дисплейними терміналами електронно-обчислювальних машин

29. ДСН 3.3.6.037-99 Санітарні норми виробничого шуму, ультразвуку та інфразвуку

30. ДСН 3.3.6.042-99 Санітарні норми мікроклімату виробничих

31. НПАОП 0.00-4.12-05 Типове положення про порядок проведення навчання і перевірки знань з питань охорони праці

32. НПАОП 0.00-4.15-98 Про розробку інструкцій з охорони праці

33. НПАОП 0.00-6.03-93 Порядок опрацювання та затвердження власником

нормативних актів про охорону праці

34. НАПБ Б.03.002-2007 Норми визначення категорій приміщень, будинків та зовнішніх установок за вибухопожежною та пожежною небезпекою

35. НПАОП 40.1-1.01-97 Правила безопасной эксплуатации электроустановок

36. НПАОП 40.1-1.32-01 Правила устройства электроустановок. Электрооборудование специальных установок

37. ДСН 3.3.6.039-99 Санітарні норми виробничої загальної та локальної вібрації

38. ДСТУ ГОСТ 12.1.012-90 ССБТ. Вібраційна безпека. Загальні вимоги

39. ДБН В.2.5-67:2013 Опалення, вентиляція та кондиціонування

40. ГОСТ 12.1.006-84 ССБТ. Електромагнітні поля радіочастот. Загальні вимоги безпеки. Допустимі рівні на робочих місцях і вимоги до проведення контролю

41. ГОСТ 12.1.030-81 ССБТ. Електробезпечність. Захисне заземлення. Занулення

42. НПАОП 0.00-1.28-10 Правила охорони праці під час експлуатації електронно-обчислювальних машин

ДОДАТОК А

Лістинг коду MainForm.cs

```
using System;
using System.Drawing;
using System.Windows.Forms;
//using System.Drawing.Imaging;//добавим BitmapData
namespace TIC_LPC
{
    public partial class MainForm : Form
    {
        public MainForm()
        {
            InitializeComponent();
        }
        float koef = 1.0f;
        string filename;
        Bitmap picture;
        public static byte[, ,] picture1;
        private void сжатьИзображениеToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            Contour newform = new Contour(picture);
            newform.Text = "Сжатие изображения";
            this.AddOwnedForm(newform);
            newform.Show();
        }
        private void разжатьИзображениеToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            int k = 6;
            Contour newform = new Contour(picture, k);
            newform.Text = "Разжатие изображения";
            this.AddOwnedForm(newform);
            newform.Show();
        }
        private void построениеГистограммыToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            // Преобразование изображения в массив байт
            Form gistoForm = new
HistoForm(Synthesis.GetMatrixIzo(picture));//median,mean,stdDev,entropy
            this.AddOwnedForm(gistoForm);
            gistoForm.Show();
        }
        private void загрузитьИзображениеToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            OpenFileDialog ofd = new OpenFileDialog();
            ofd.Filter = "All image formats (*.bmp; *.jpg)|*.bmp;*.jpg|"
+
                                            "Windows Bitmap
(*.bmp)|*.bmp|" +
                                            "Graphics Interchange Format
(*.gif)|*.gif|" +
```

```
                                                "Joint Photographic Experts
Group (*.jpeg)|*.jpg;*.jpeg|" +
                                                "Windows Media
Format(*.wmf)|*.wmf" +
                                                "|All files (*.*)|*.*";
            if (ofd.ShowDialog() == DialogResult.OK)
            {
                filename = ofd.FileName;
                picture = new Bitmap(filename);
                //предусмотрим возможность масштабирования изображения
                int newW = (int)Math.Round(koef * picture.Width);
                int newH = (int)Math.Round(koef * picture.Height);
                // формируем изображение
                pictureBox1.Image = new Bitmap(picture, newW, newH);
                pictureBox1.Size = new Size(newW, newH);
            }
        }
        private void сохранитьИзображениеToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            SaveFileDialog sfDialog = new SaveFileDialog();
           // sfDialog.Filter = "*.bmp|*.bmp";
            sfDialog.Filter = "All image formats (*.bmp;
*.jpg)|*.bmp;*.jpg|" +
               "Windows Bitmap (*.bmp)|*.bmp|" +
               "Graphics Interchange Format (*.gif)|*.gif|" +
               "Joint Photographic Experts Group (*.jpeg)|*.jpg;*.jpeg|"
+
               "Windows Media Format(*.wmf)|*.wmf" +
               "|All files (*.*)|*.*";
            if (sfDialog.ShowDialog() == DialogResult.OK)
                pictureBox1.Image.Save(sfDialog.FileName);
        }
        Bitmap temp;
        Bitmap temp1;
        // string filename;
        // public static byte[, ,] picture;
        private void загрузитьИзображение1ToolStripMenuItem_Click(object
sender, EventArgs e)
        {
            OpenFileDialog of = new OpenFileDialog();
            of.Filter = "All image formats (*.bmp; *.jpg)|*.bmp;*.jpg|"
+
                "Windows Bitmap (*.bmp)|*.bmp|" +
                "Graphics Interchange Format (*.gif)|*.gif|" +
                "Joint Photographic Experts Group
(*.jpeg)|*.jpg;*.jpeg|" +
                "Windows Media Format(*.wmf)|*.wmf" +
                "|All files (*.*)|*.*";
            if (of.ShowDialog() == DialogResult.OK)
            {
                //pictureLoaded = true;
                {
                    Bitmap a = new Bitmap(of.FileName);
                    temp = new Bitmap(a);
                    a.Dispose();
                }
                filename = of.FileName;
```

```
111                 //предусмотрим возможность масштабирования изображения
112                 int newW = (int)temp.Width;
113                 int newH = (int)temp.Height;
114                 // формируем изображение
115                 pictureBox1.Image = new Bitmap(temp, newW, newH);
116                 pictureBox1.Size = new Size(newW, newH);
117                 //используется для отображения имени файла и пути на
118 форме
119                 Text = filename;
120                 // ta = null;
121             }
122         }
123 
124         private void загрузитьИзображение2ToolStripMenuItem_Click(object
125 sender, EventArgs e)
126         {
127             OpenFileDialog of = new OpenFileDialog();
128             of.Filter = "All image formats (*.bmp; *.jpg)|*.bmp;*.jpg|"
129 +
130                 "Windows Bitmap (*.bmp)|*.bmp|" +
131                 "Graphics Interchange Format (*.gif)|*.gif|" +
132                 "Joint Photographic Experts Group
133 (*.jpeg)|*.jpg;*.jpeg|" +
134                 "Windows Media Format(*.wmf)|*.wmf" +
135                 "|All files (*.*)|*.*";
136             if (of.ShowDialog() == DialogResult.OK)
137             {
138                 //pictureLoaded = true;
139                 {
140                     Bitmap a = new Bitmap(of.FileName);
141                     temp1 = new Bitmap(a);
142                     a.Dispose();
143                 }
144                 filename = of.FileName;
145                 //предусмотрим возможность масштабирования изображения
146                 int newW = (int)temp1.Width;
147                 int newH = (int)temp1.Height;
148                 // формируем изображение
149                 pictureBox2.Image = new Bitmap(temp1, newW, newH);
150                 pictureBox2.Size = new Size(newW, newH);
151                 //используется для отображения имени файла и пути на
152 форме
153                 // Text = filename;
154                 // ta = null;
155             }
156         }
157 
158         private void композициягистограммаToolStripMenuItem_Click(object
159 sender, EventArgs e)
160         {
161             picture1 = TIC_LPC.Synthesis.GetMatrixIzo(temp, temp1);////
162             //   Histogram(picture);//picturehistogram=
163            // Form ownedForm = new TIC_LPC.HistoForm(histogram);//
164             Form ownedForm = new TIC_LPC.HistoForm(picture1);
165             pictureBox1.Image =
166 TIC_LPC.Synthesis.SetMatrix(picture1);////////////////////////////////
167 ////////
```

```
168
169 pictureBox1.Refresh();////////////////////////////////////////////////
170 //////
171  ////////////
172       //      Form gistoForm = new
173 HistoForm(Synthesis.GetMatrixIzo(picture1));//median,mean,stdDev,entropy
174     //        this.AddOwnedForm(gistoForm);
175       //      gistoForm.Show();
176             /////////////////
177
178             // Добавление новой формы
179          //   this.AddOwnedForm(ownedForm);//bbbbb
180             // Show the owned form.
181            ownedForm.Show();///bbbbbb
182         }
183
184         private void композициягистграммаToolStripMenuItem_Click(object
185 sender, EventArgs e)
186         {
187             picture1 = TIC_LPC.Synthesis.GetMatrixIzo(temp, temp1);////
188             //   Histogram(picture);//picturehistogram=
189             // Form ownedForm = new TIC_LPC.HistoForm(histogram);//
190             Form ownedForm = new TIC_LPC.HistoForm(picture1);
191             pictureBox1.Image =
192 TIC_LPC.Synthesis.SetMatrix(picture1);////////////////////////////////
193 ////////
194
195 pictureBox1.Refresh();////////////////////////////////////////////////
196 //////
197             ////////////
198             //      Form gistoForm = new
199 HistoForm(Synthesis.GetMatrixIzo(picture1));//median,mean,stdDev,entropy
200             //        this.AddOwnedForm(gistoForm);
201             //      gistoForm.Show();
202             /////////////////
203
204             // Добавление новой формы
205             //   this.AddOwnedForm(ownedForm);//bbbbb
206             // Show the owned form.
207             ownedForm.Show();///bbbbbb
208         }
209     }
210 }
211 //}////////////
```

ДОДАТОК Б

Лістинг коду HistoForm.cs

```
1
2   using System;
3   using System.Drawing;
4   using System.Windows.Forms;
5   using System.Drawing.Imaging;//для ImageAttributes
6
7   namespace TIC_LPC
8   {
9       public partial class HistoForm : Form
10      {
11          public int[,] histogram;
12          public byte[] median;          //среднее значение искажения
13          public byte[] background;  //оценка частоты текущего цвета фона
14          public float[] mean;           //математическое ожидание
15          public float[] stdDev;         //квадратические отклонение
16          public float[] entropy;        //этропия
17          private int width;               //ширина; горизонтальный размер
18          private int height;             //высота
19          private int size;
20          public bool colored;             //ложь, если изображение ч/б,
21  иначе - правда
22
23          public HistoForm()
24          {
25              InitializeComponent();
26          }
27
28         public HistoForm(byte[, ,] image)
29            {
30                InitializeComponent();
31
32                width = image.GetLength(1);
33                height = image.GetLength(2);
34                int colors = image.GetLength(0);
35                int half = (width * height) / 2;
36                int max = 0;
37                float M = 0;
38                float D = 0;
39                float P = 0;
40                int summa = 0;
41                float temp;
42
43                median = new byte[3];
44                mean = new float[3];
45                stdDev = new float[3];
46                background = new byte[3];
47                entropy = new float[3];
48                histogram = new int[1, 256];
49                size = width * height;
50
51                if (image.GetLength(0) ==1)
52                {
53                    // Подсчёт чёрнобелой гистограммы
54                    histogram = new int[1, 256];
```

```
55
56                      D = M = summa = max = 0;
57                      for (int x = 0; x < width; x++)
58                          for (int y = 0; y < height; y++)
59                              histogram[0, image[0, x, y]]++;
60                      for (int i = 0; i < 256; i++)
61                      {
62                          if (max < histogram[0, i]) { max = histogram[0,
63  i]; background[0] = (byte)i; }
64                          if (summa < half) summa += histogram[0, i];
65                          else if (median[0] == 0) median[0] = (byte)(i -
66  1);
67                          P = (float)histogram[0, i] / size;
68                          temp = i * P;
69                          M += temp;
70                          if (P != 0) entropy[0] += (float)(-P *
71  Math.Log(P));
72                      }
73                      mean[0] = M;
74
75                      for (int i = 0; i < 256; i++)
76                          D += (i - M) * (i - M) * (float)histogram[0, i];
77                      D /= size;
78                      stdDev[0] = D;
79
80                      textBox1.Text += "median[0] = " + median[0].ToString()
81  + "\t" +
82                                                      "mean[0] = " +
83  mean[0].ToString() + "\t\t" +
84                                                      "stdDev[0] = " +
85  stdDev[0].ToString() + "\t" +
86                                                      "entropy[0]  = " +
87  entropy[0].ToString() ;
88                      BuildHistoGram();
89                  }
90                  else
91                  {
92                      // Подсчёт цветной гистограммы
93                      histogram = new int[3, 256];
94                      for (int c = 0; c < 3; c++)
95                      {
96                          D = M = summa = max = 0;
97                          for (int x = 0; x < width; x++)
98                              for (int y = 0; y < height; y++)
99                                  histogram[c, image[c, x, y]]++;
100                         for (int i = 0; i < 256; i++)
101                         {
102                             if (max < histogram[c, i]) { max =
103 histogram[c, i]; background[c] = (byte)i; }
104                             if (summa < half) summa += histogram[c, i];
105                             else if (median[c] == 0) median[c] = (byte)(i
106 - 1);
107                             P = (float)histogram[c, i] / size;
108                             temp = i * P;
109                             M += temp;
110                             if (P != 0) entropy[c] += (float)(-P *
111 Math.Log(P));
112                         }
```

```
                        mean[c] = M;

                        for (int i = 0; i < 256; i++)
                            D += (i - M) * (i - M) * (float)histogram[c,
i];
                        D /= size;
                        stdDev[c] = D;
                    }
                      BuildHistoGramRGB();
                      for (int i = 0; i < 3;i++ )
                      {
                          textBox1.Text +="median["+i.ToString()+"] = "+
median[i].ToString() + "\t" +
                                                          "mean[" +
i.ToString() + "] = " + mean[i].ToString() + "\t\t" +
                                                          "stdDev[" +
i.ToString() + "] = " + stdDev[i].ToString() + "\t" +
                                                          "entropy[" +
i.ToString() + "] = " + entropy[i].ToString() + Environment.NewLine;
                      }
                }
            }

        public void BuildHistoGram()
        {
            int max = 0, edinica;
            for (int i = 0; i < 256; i++)
                if (histogram[0, i] > max) max = (int)histogram[0, i];
            if (max < 200) edinica = 0;
            else edinica = max / 200;
            edinica++;
            Bitmap h = new Bitmap(512, 200);
            Graphics g = Graphics.FromImage(h);
            Pen p = new Pen(Color.White);
            g.DrawRectangle(p, 0, 0, 512, 200);
            p = new Pen(Color.Black);
            for (int i = 0; i < 256; i++)
            {
                g.DrawLine(p, 2 * i, 200, 2 * i, 200 - histogram[0, i] /
edinica);
            }
            Rectangle rc = new Rectangle(0, 0, 512, 200);
            pictureBox1.Image = h;
            g.DrawImage(pictureBox1.Image, rc, 0, 0, 512, 200,
GraphicsUnit.Pixel);
            g.Dispose();
            pictureBox1.Refresh();
        }

        public void BuildHistoGramRGB()
        {
            int max = 0, edinica;
            Bitmap h = new Bitmap(512, 200);
            Pen p;
            Graphics g = null;
            Color[] c = new Color[3];
            Rectangle rc = new Rectangle(0, 0, 512, 200);
            ImageAttributes imatr = new ImageAttributes();
```

```
171
172             for (int i = 0; i < 256; i++)
173                 for (int j = 0; j < 3; j++)
174                     if (histogram[j, i] > max) max = (int)histogram[j,
175 i];
176             if (max < 200) edinica = 0;
177             else edinica = max / 200;
178             edinica++;
179             pictureBox1.Image = h;
180             c[0] = Color.FromArgb(150, 0, 0, 255);
181             c[1] = Color.FromArgb(150, 0, 255, 0);
182             c[2] = Color.FromArgb(150, 255, 0, 0);
183             for (int j = 0; j < 3; j++)
184             {
185                 g = Graphics.FromImage(pictureBox1.Image);
186                 p = new Pen(c[j]);
187                 for (int i = 0; i < 256; i++)
188                 {
189                     g.DrawLine(p, 2 * i, 200, 2 * i, 200 - histogram[j,
190 i] / edinica);
191                 }
192             }
193             g.DrawImage(pictureBox1.Image, rc, 0, 0, 512, 200,
194 GraphicsUnit.Pixel, imatr);
195             pictureBox1.Refresh();
196         }
197
198         private void
199 сохранитьГистограммуВФайлToolStripMenuItem_Click(object sender,
200 EventArgs e)
201         {
202             SaveFileDialog sfDialog = new SaveFileDialog();
203             sfDialog.Filter = "*.bmp|*.bmp";
204             if (sfDialog.ShowDialog() == DialogResult.OK)
205                 pictureBox1.Image.Save(sfDialog.FileName);
206         }
207     }
208 }
209
```

ДОДАТОК В

Комп'ютерна презентація

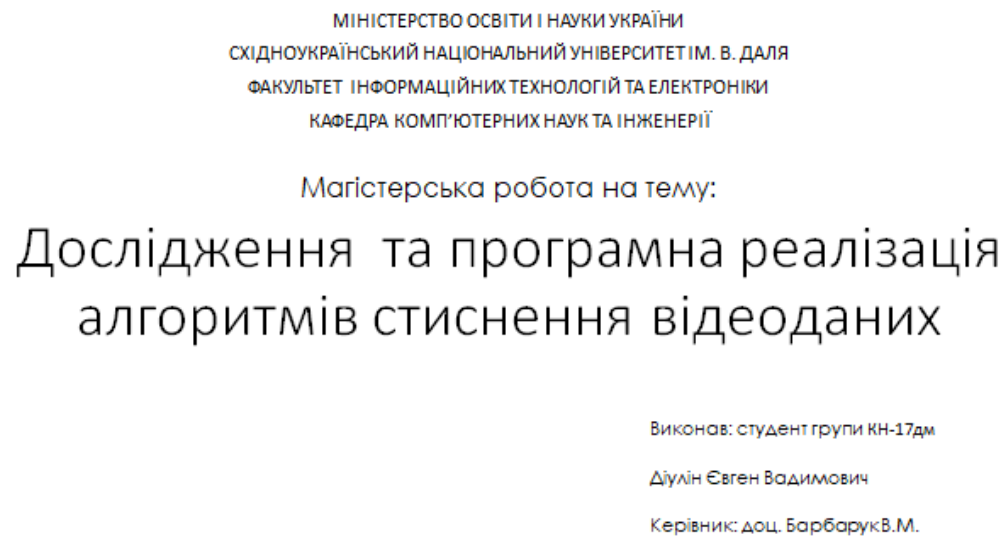


Рисунок В.1- Слайд №1

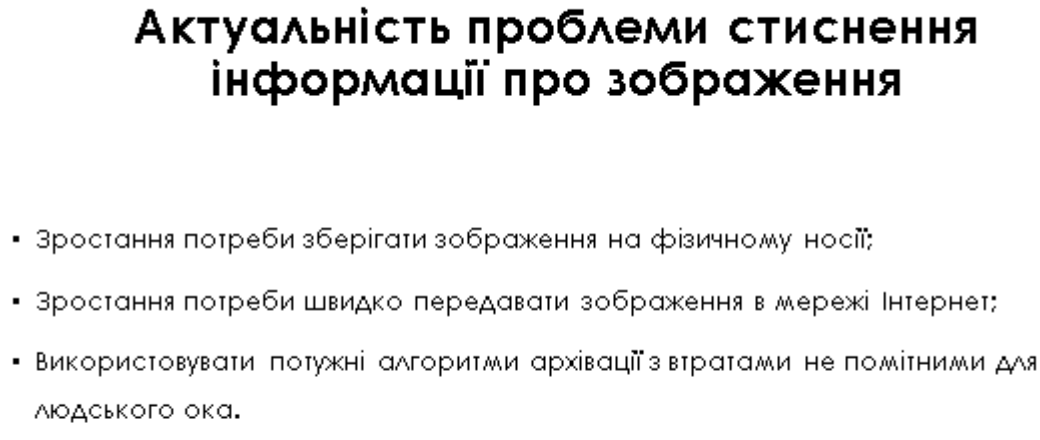


Рисунок В.2- Слайд №2

Актуальність проблеми стиснення інформації про зображення

- Зростання потреби зберігати зображення на фізичному носії;
- Зростання потреби швидко передавати зображення в мережі Інтернет;
- Використовувати потужні алгоритми архівації з втратами не помітними для людського ока.

Рисунок В.3- Слайд №3

Постановка задачі

- **Об'єктом** даної роботи є дослідження методів стиснення зображень;
- **Предметом** дослідження є використання методів стиснення інформації про зображення в задачах обміну та зберігання зображень;
- **Метою** роботи є дослідження впливу методів стиснення на якість та розмір зображення, та пошуку універсальних методів для використання в мережах

Рисунок В.4- Слайд №4

Методи стиснення зображень без втрат

- Кодування довжин серій RLE;
- Алгоритм Лемпеля — Зіва — Велча LZW;
- Алгоритм Хаффмана;
- JBIG;
- Lossless JPEG;

Рисунок В.5- Слайд №5

Області використання алгоритмів стиснення без втрат

- Ділова і наукова графіка;
- Зображення з невеликою кількістю кольорів;
- Зображення з великими областями повторюваного кольору;
- Зображення, що вимагають оригінальної якості;

Рисунок В.6- Слайд №6

Методи стиснення зображень з втратами

- Алгоритм JPEG;
- Фрактальный алгоритм;
- Рекурсивний алгоритм;

Рисунок В.7- Слайд №7

Проблеми алгоритмів архівації з втратами

Одна з серйозних проблем машинної графіки полягає в тому, що до цих пір не знайдений адекватний критерій оцінки втрат якості зображення.

Міра, яку зараз використовують на практиці, називається мірою відносини сигналу до шуму (peak-to-peak signal-to-noise ratio — PSNR).

Рисунок В.8- Слайд №8

PSNR - Пікове співвідношення сигналу до шуму

PSNR найчастіше використовується для вимірювання рівня спотворень при стисканні зображень. Найпростіше його визначити через середньоквадратичне відхилення (СКВ або MSE (англ. mean square error)), яке для двох монохромних зображень I та K розміру m×n, одне з яких вважається зашумленими наближенням іншого, обчислюється наступним чином:

$$MSE = \frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}[I(i,j) - K(i,j)]^2$$

PSNR визначається наступним чином:

$$PSNR = 10\log_{10}\left(\frac{MAX_I^2}{MSE}\right) = 20\log_{10}\left(\frac{MAX_I}{\sqrt{MSE}}\right)$$

де MAXI — це максимальне значення, яке приймається пікселем зображення. Коли пікселі мають розрядність 8 біт, MAXI = 255. В загальному випадку, коли значення сигналу представлені лінійно (PCM) з B битами на кожне значення, максимально можливе значення MAXI буде 2B−1.

Рисунок В.9- Слайд №9

Загальний підсумок алгоритмів стиснення зображень без втрат

Алгоритм	Коефіцієнт компресії	Клас зображень	Особливості
RLE	Перший варіант: 32, 2, 0,5. Другий варіант: 64, 3, 128/129.	зображення з невеликою кількістю кольорів: ділову і наукову графіком.	не вимагає додаткової пам'яті при архівації та розархівації, а також швидко працює
LZW	1000, 4, 5/7. В 1000 разів досягається тільки на одноколірних зображеннях розміром кратним приблизно 7 Мб	Орієнтований на 8-бітові зображення, побудовані на комп'ютері.	У деяких ситуаціях може збільшити розмір зображення
Алгоритм Хаффмана	8, 1.5, 1	Зазвичай використовується як один з етапів компресії в більш складних схемах.	Єдиний алгоритм, який не збільшує розміру вихідних даних в гіршому випадку
Lossless JPEG	20, 2, 1	орієнтований на повнокольорові 24-бітові або 8-бітові в градаціях сірого зображення без палітри.	рекомендується застосовувати в тих додатках, де необхідно побітове відповідність вихідного і декомпресіорват зображення

Рисунок В.10- Слайд №10

Загальний підсумок алгоритмів стиснення зображень з втратами

Алгоритм	Коефіцієнт компресії	Клас зображень	Особливості
JPEG	2-200 (Здається користувачем).	Повнокольорові 24 бітові зображення, або зображення в градаціях сірого без різких переходів кольорів (фотографії).	при високому ступені стиснення зображення розпадається на блоки 8x8 пікселів.
Фрактальний алгоритм	2-2000	Орієнтований на 24-бітні сірі.	вимагає алгоритми оптимізації перебору
Рекурсивне стиснення	2-200	в основному використовується для рентгенівських знімків	дозволяє реалізувати можливість поступового "прояви" зображення при передачі зображення по мережі.

Рисунок В.11- Слайд №11

Аналіз алгоритмів

Виходячи з усіх розглянутих алгоритмів найбільш універсальними являються алгоритм LZW, який реалізує стиснення без втрат, і алгоритм JPEG, який реалізує стиснення з втратами. Ще одним важливим фактором є активне використання інтернету, в якому передача зображень виконується за порівняно повільним каналам зв'язку, використання Interlaced GIF (алгоритм LZW) і Progressive JPEG (варіант алгоритму JPEG), що реалізують можливість показу зображень низької якості та preview

Рисунок В.12- Слайд №12

Рисунок В.13- Слайд №13

Рисунок В.14- Слайд №14

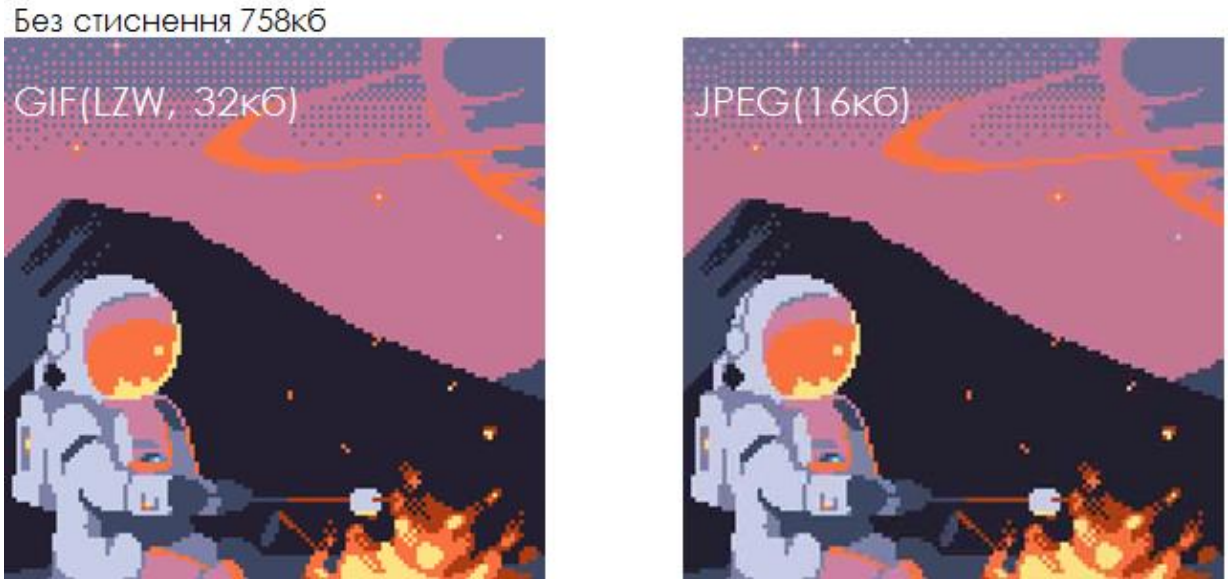


Рисунок В.15- Слайд №15

Рисунок В.16- Слайд №16

ВИСНОВКИ

Були досліджені алгоритми стиснення зображень з втратами та без втрат та реалізована програма, за допомогою якої був проведен практичний аналіз. У ході дослідження було виявлено, що ефективність кожного алгоритму залежить від класу зображення. З усіх розглянутих алгоритмів були відібрані LZW та JPEG, як найбільш універсальні та ефективні для використання в інтернеті.

З алгоритму LZW було виявлено наступні переваги:

- Не потребує обчислення ймовірностей народження символів або кодів.
- Для декомпресії не треба зберігати таблицю рядків в файл для розпакування.
- Даний тип компресії не вносить спотворень в вихідний графічний файл і підходить для стиснення растрових даних будь-якого типу.

Головним недоліком є те, що цей алгоритм ефективний тільки для зображень, переважно створенних за допомогою комп'ютера, на яких немає плавності переходу кольорів. Основними зображеннями для цього алгоритму є 8-бітові зображення, креслення, текстова та знакова графіка.

Рисунок В.17- Слайд №17

ВИСНОВКИ

Алгоритм JPEG ефективний для стиснення фотографій і картин, що містять реалістичні сцени з плавними переходами яскравості і кольору. Одним з найвижливіших факторів також є можливість вказувати ступінь стиснення зображення. Найбільшого поширення JPEG отримав в цифровій фотографії і для зберігання і передачі зображень з використанням мережі Інтернет.

Рисунок В.18- Слайд №18