Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет _інформаційних радіотехнологій та технічного захисту інформації_
(повна назва)

Кафедра __медіаінженерії та інформаційних радіоелектронних систем_____
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти _____другий (магістерський)_____

Програмні методи обробки фотографій
_____
(тема)

Виконав:
студент _2_ курсу, групи _СТМм-22-1_
_____Олександр_ ТКАЧЕНКО_____
(прізвище, ініціали)

Спеціальність 171 Електроніка_____
_____
(код і повна назва спеціальності)
Тип програми ___освітньо-професійна____
Освітня програма ____Електронні системи
і комп'ютерні засоби мультимедіа____
(повна назва освітньої програми)

Керівник __ст.викл. Тетяна ВАСИЛЕНКО
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ ___Володимир КАРТАШОВ__
(підпис)

2024 р.

Харківський національний університет радіоелектроніки

Факультет інформаційних радіотехнологій та технічного захисту інформації
Кафедра медіаінженерії та інформаційних радіоелектронних систем
Рівень вищої освіти другий (магістерський)
Спеціальність 171 Електроніка
(код і повна назва)
Тип програми освітньо-професійна
Освітня програма Електронні системи і комп'ютерні засоби мультимедіа
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«____»_____ 20 ___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студентові Ткаченко Олександру Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Програмні методи обробки фотографій.
затверджена наказом по університету від "_20_"_____11_____ 2023 р. № ___1371 Ст____

2. Термін подання студентом роботи до екзаменаційної комісії _____08.01.2024 р._____

3. Вихідні дані до роботи
1) Провести аналітичний огляд сучасних можливостей обробки зображення в автоматичному режимі. Приділити увагу видаленню цифрового шуму з зображення.
2) Оглянути базові, вбудовані можливості програмного пакету для обробки зображень. При наявності визначити столронні бібліотеки та функції для обробки зображень.
3)Провести обробку фотографій у програмнгому пакеті з описом використаних команд.

4. Перелік питань, що потрібно опрацювати в роботі
Вступ
1. Аналітичний огляд загальних відомостей про зображення.
2. Використання можливостей та базових функцій у програмному пакеті
3. Обробка фотографій у програмному пакеті
Висновки
Перелік посилань
Додатки
5. Перелік графічного матеріалу із зазначенням обов'язкових креслеників, схем, плакатів, комп'ютерних ілюстрацій 1)Результати видалення Гаусового шуму; 2)Результати видалення комбінованого шуму; 3)Результати видалення імпульсного шуму; 4)Зображення у процессі обробки медіанним фільтром; 5)Зображення у процессі обробки медіанним фільтром.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналітичний огляд загальних відомостей про зображення	21.11.23–27.11.23	
2	Використання можливостей та базових функцій у програмному пакеті	28.11.23–09.12.23	
3	Обробка фотографій у програмному пакеті	10.12.23–20.12.23	
4	Обробка результатів	21.12.23-23.12.23	
5	Графічна частина роботи	24.12.23-26.12.23	
6	Перевірка керівником	27.12.23-28.12.23	
7	Перевірка на академічний плагіат	29.12.23-02.01.24	
8	Перевірка завідувачем кафедри,	03.01.24-08.01.24	

Дата видачі завдання 20.11.2023 р.

Студент _____ Олександр ТКАЧЕНКО
(підпис)

Керівник роботи _____ Тетяна ВАСИЛЕНКО
(підпис)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи має: 64 с., 29 Рис., 1 табл., 2 додатків, 19 джерел.

ФОТОГРАФІЯ, ОБРОБКА ЗОБРАЖЕНЬ, MATLAB, ШУМ, ФІЛЬТРАЦІЯ, WAVELET-ФІЛЬТРАЦІЯ, АДАПТИВНИЙ ФІЛЬТР СЕРЕДНЬОГО, МЕДІАННИЙ ФІЛЬТР, ГАУСОВИЙ ШУМ, ЯКІСТЬ ЗОБРАЖЕННЯ.

Об'єкт дослідження даної кваліфікаційної роботи – програмні методи обробки фотографії з використанням пакету Matlab.

Предметом дослідження є розробка та оптимізація алгоритмів обробки фотографій за допомогою функціоналу, який надається у зазначеному програмному пакеті.

Кваліфікаційна робота, присвячена програмним методам обробки фотографій у середовищі Matlab, розглядає важливі аспекти аналізу та покращення зображень за допомогою різних методів, враховуючи шумові ефекти та використовуючи різні фільтраційні та обробники в пакеті Matlab.

Однією з ключових частин дослідження є аналіз типів шуму, які можуть виникнути на фотографії, такі як Гаусовий шум, чи шум з випадковим розподілом. Це важливий етап, оскільки різні типи шуму можуть вимагати різних методів обробки для їхнього видалення. Далі робота розглядає застосування фільтрів для обробки різних типів шуму.

Загальний підхід даної роботи включає в себе ретельний аналіз різних методів обробки фотографій в середовищі Matlab, а також їхній експериментальний порівняльний аналіз. Результати цього дослідження можуть мати практичне застосування у галузі обробки зображень, вдосконалюючи якість та візуальну привабливість фотографій, а також забезпечуючи інструменти для впровадження в реальні сценарії обробки даних.

ABSTRACT

The explanatory note of the qualification paper has: 64 pages, 29 figures, 1 tables, 2 appendices, 19 sources.

PHOTOGRAPHY, IMAGE PROCESSING, MATLAB, NOISE, FILTERING, WAVELET FILTERING, ADAPTIVE MEAN FILTER, MEDIAN FILTER, GAUSSIAN NOISE, IMAGE QUALITY.

The object of research of this qualification work is software methods of photo processing using the Matlab package.

The subject of the study is the development and optimization of photo processing algorithms using the functionality provided in the specified software package.

The qualification work, dedicated to the software methods of photo processing in the Matlab environment, considers important aspects of image analysis and enhancement using various methods, taking into account noise effects and using various filtering and processors in the Matlab package.

One of the key parts of the research is analyzing the types of noise that can occur in a photograph, such as Gaussian noise or random noise. This is an important step because different types of noise may require different processing methods to remove them. Next, the work considers the use of filters for processing different types of noise.

The general approach of this work includes a thorough analysis of various photo processing methods in the Matlab environment, as well as their experimental comparative analysis. The results of this research may have practical applications in the field of image processing, improving the quality and visual appeal of photographs, as well as providing tools for implementation in real-world data processing scenarios.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

RP – одноплатний комп'ютер Raspberry Pi;

ПС - повнозв'язковий шар нейронної мережі.

СНР - сверточна нейронна мережа;

СРЖ - система розпізнавання жестів;

СС – згортковий шар нейронної мережі;

ПЗ – програмне забезпечення;

ЗМІСТ

ВСТУП

Однією із фундаментальних проблем сучасності є проблема зорового сприйняття. Виникнувши дуже давно, вона є актуальною і в даний час, бо зображення є природним засобом спілкування людини та машини у будь-яких системах обробки, аналізу та контролю.

Комп'ютерна графіка поділяється на три основні напрямки: візуалізація, обробка зображень та розпізнавання образів.

Візуалізація - Це створення зображення на основі якогось опису (моделі). Основне завдання розпізнавання образів – отримання семантичного опису зображених об'єктів. Обробка зображень відповідає за перетворення (фільтрування) зображень. Розвиток сучасних засобів комп'ютерної техніки та інформаційних технологій сприяє широкому впровадженню у практику систем автоматичної обробки зображень.

Першорядним завданням такої системи є покращення якості зображення. Проблема шумоподавлення є однією з найактуальніших і найпоширеніших проблем у галузі обробки зображень. Найпоширенішими видами шумів є Гауссів та імпульсний шуми, а також їх комбінація. У роботі пропонується універсальний алгоритм, який пригнічує ці три види шумів краще за стандартні. Цей фільтр заснований на методі LPA (локально-поліноміальної апроксимації) та техніці ICI (перетин інтервалів довіри), що спочатку застосовувалися в статистиці для обробки даних, але також запропонованому в для обробки зображень. З їхньою допомогою для кожного пікселя на зображенні знаходиться вікно (апертура), тобто околиця, в якому пікселі не дуже відрізняються від нього. У разі Гауссова шуму центральному пікселю присвоюється виважене середнє пікселів, що потрапили в околицю. У разі імпульсного шуму невеликих розмір цього вікна допомагає ідентифікувати «биті» пікселі. Для прискорення роботи алгоритму використовується адаптивність високого рівня, тобто, алгоритм підлаштовується під наявні шуми і видаляє їх.

У магістерській кваліфікаційній роботі розглядається адаптивний метод видалення шуму, заснований на техніці LPA та методі ICI.

Метою роботи є модифікація методу LPA та ICI, запропонованого в для ширшого класу завдань, а саме:

- видалення імпульсного шуму;
- видалення комбінованого шуму;
- адаптивність методу.

1 АНАЛІТИЧНИЙ ОГЛЯД ЗАГАЛЬНИХ ВІДОМОСТЕЙ ПРО ЗОБРАЖЕННЯ

У галузі обробки візуальної інформації виділяють три основні науковінапрямки: науковий напрямок, що займається власне обробкою зображень; аналіз(Розуміння) зображень; синтез зображень (машинна графіка) Відповідно виділяють системи власне обробки зображень, системи аналізу (розуміння) зображень та системи синтезу зображень (машинної графіки). У першому випадку вирішуються завдання, в яких вхідні дані та результати обробки подаються у образотворчій формі. У другому випадку вхідні дані подаються у образотворчій формі, а результат обробки – у незображеній формі, наприклад у вигляді текстового опису сцени. У разі вирішуються завдання синтезу зображень у тому образотворчої формі за деяким їх опису чи алгоритму побудови.

Як основні операції, що реалізуються в системах обробки візуальної інформації, можна виділити: оцифрування, кодування та стиснення, декодування та відновлення стислих зображень, поліпшення якості та реставрацію зображень, сегментацію, аналіз зображень та складання їх описів, розуміння зображень, управління візуальною інформацією.

В узагальненій схемі системи обробки візуальної інформації на вхід пристрою введення зображень надходить безперервне зображення f(x, y), що задає розподіл яскравості елементів деякої сцени просторовими координатами x, y. У пристрої введення це зображення потім піддається дискретизації, тобто. формується кілька відліків функції f(x, y). Далі отримані відліки піддаються квантуванню. Таким чином, цифрове зображення є двовимірним масивом квантованих відліків функції f(x, y). Надалі елемент такого масиву називатимемо елементом цифрового зображення або пікселем, а саме цифрове зображення задаватимемо функцією f(i, j) від дискретних аргументів i і j – індексів елементів цифрового зображення, що визначають їх положення у двовимірному масиві квантованих відліків функції f(x, y).

Розглянуті вище функції зображення є вихідною формою уявлень зображень. Результати обробки таких зображень можуть бути представлені в різних формах, зокрема списком сегментів зображення (для сегментованого зображення), геометричною моделлю у вигляді сукупності геометричних фігур, текстовим описом сцени та ін.

Найбільш складними за своєю реалізацією та виконуваним функціям є системи розуміння зображень. У найпростішому випадку такі системи є системою розпізнавання образів.

У такій системі як ознаки, що використовуються для розпізнавання образів, можуть, наприклад, виступати параметри текстури, контуру об'єктів, ознаки, сформовані з використанням перетворення Фур'є, різні моменти виділених сегментів зображення та ін. Необхідно зазначити, що як ознаки можуть виступати і значення яскравостей самих елементів зображення.

1.2 Растрові та векторні зображення

Традиційно цифрові зображення поділяють на растрові та векторні. Для того, щоб краще розібратися в цих форматах, уявімо два різні завдання. Перше завдання – отримати цифровий еквівалент звичайного фотографічного відбитка – растрове зображення. Друге завдання – намалювати креслення певної деталі – векторне зображення.

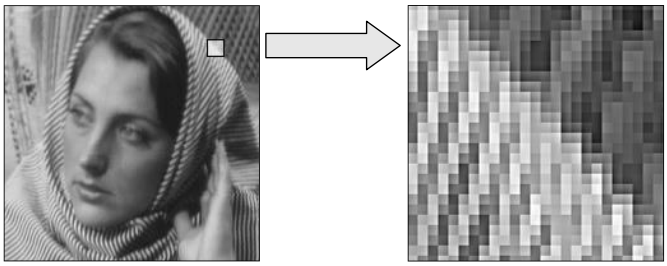
У першому випадку розумно розбити всю фотографію на дрібні впорядковані частинки (зазвичай зображення розбивають прямокутною сіткою), кожному такому осередку поставити у відповідність деяке значення, що кодує колір. Масив таких значень називається растр, а зображення називається растровим.

Кожен елемент зображення - комірка сітки, що розбиває, називається піксель (або піксел - англ. pixel, скор. від англ. picture's element, елемент зображення). Від кількості пікселів залежить детальність зображення. Максимальна деталізація растрового зображення задається при його створенні і не може бути збільшена.

Якщо збільшується масштаб зображення, пікселі перетворюються на великі зерна (рис. 1.1).

Для зображень у градаціях сірого яскравість пікселя найчастіше кодується значеннями від 0 (чорний) до 255 (білий). У разі кольорового зображення кількість кольорових компонентів може бути різною, однак для кожної колірної компоненти в межах пікселя визначається яскравість, так само як і для напівтонового зображення, тому колір кожного пікселя кодується вже набором значень.

Наприклад, знімки, отримані з цифрової камери, є кольоровими зображеннями, представленими трьома компонентами: червоною, зеленою та синьою. Таке зображення має встановлений камерою розмір растру, який вимірюється в пікселях. Кожен елемент растру містить інформацію про яскравість кожної компоненти кольору.



а) б)

Рисунок 1.1 - Растрове зображення: а) нормальний масштаб; б) збільшений фрагмент

Переваги растрової графіки:

– Простота і, як наслідок, технічна реалізованість автоматизації введення (оцифрування) образотворчої інформації. Існує розвинена система зовнішніх пристроїв для введення фотографій, слайдів, малюнків – сканерів, відеокамер, цифрових фотоапаратів.

– Фотореалістичність. Можна отримувати мальовничі ефекти, наприклад туман або серпанок, домагатися найтонших нюансів кольору, створювати глибину та нерізкість, розмитість, акварельність та багато іншого.

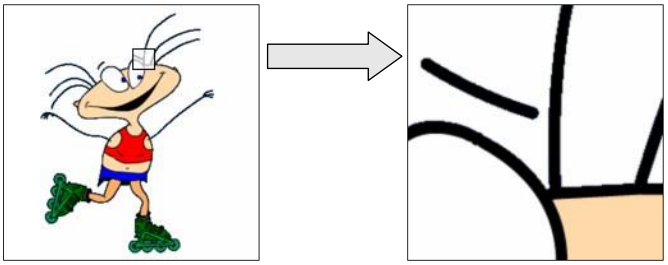
Недоліки растрової графіки:

– Великий розмір займаний файлами.

– Відсутність можливості для прямого визначення об'єкта, оскільки кожен об'єкт є безліччю елементів (пікселів) зображення.

– Спотворення, що виникають під час виконання будь-яких трансформацій (поворотів, масштабування, нахилів).

Векторна графіка значно відрізняється від растрової тим, що вона не є похідною від матеріального джерела, більше того, іноді її називають геометричне моделювання. Суть векторної графіки зводиться до використання геометричних примітивів, таких як точки, лінії, сплайни та полігони для представлення зображень. Інженер, що зображує деяку деталь, описує їх у термінах побудови примітивів з різними параметрами й у результаті отримує зображення саме собою, а деякий алгоритм побудови зображення з математично описаних складових. Кожен примітив є незалежним об'єктом, який можна переміщати, масштабувати і змінювати. Все це призводить до того, що векторне зображення можна редагувати, не побоюючись спотворень, які викликають процес редагування (рис. 1.2).



а) б)

Рисунок 1.2 - Векторне зображення: а) нормальний масштаб; б) збільшений фрагмент

У тих галузях графіки, де важливе значення має збереження ясних і чітких контурів, наприклад, у шрифтових композиціях, у створенні логотипів, векторна графіка незамінна. Вона використовує всі переваги роздільної здатності будь-якого

вивідного пристрою (зображення завжди буде виглядати настільки якісно, наскільки здатний даний пристрій).

Переваги векторної графіки:

- Векторні дані потребують менше пам'яті.
- Об'єкти векторної графіки легко трансформуються, і ними просто маніпулювати, що не має ніякого впливу на якість зображення.
- Векторна графіка може включати і елементи растрової графіки, фрагмент стає таким же об'єктом, як і всі інші (щоправда, зі значними обмеженнями в обробці).
- Важливою перевагою програм векторної графіки є розвинені засоби інтеграції зображень та тексту, єдиний підхід до них і, як наслідок, можливість створення кінцевого продукту.

Програми векторної графіки незамінні у сфері дизайну, технічного малювання, креслярсько-графічних та оформлювальних робіт.

Недоліки векторної графіки:

- Векторна графіка може здатися надмірно жорсткою, "фанерної". Вона справді обмежена у мальовничих засобах. З допомогою векторної графіки практичнонеможливо створювати фотореалістичні зображення
- Векторний принцип опису зображення не дозволяє автоматизувати введення графічної інформації, як це робить сканер для растрової графіки.

1.3 Яскрава та кольорова інформація

Світло як носій містить лише два види інформації – інформацію про яскравість і інформацію про колір. Висловлюючись спрощено, можна сказати, що загальна кількість всіх світлових хвиль у світловому промені, яке еквівалентно його загальної енергії, зумовлює інтенсивність або яскравість світла, тоді як

пропорції, в яких представлені різні світлові хвилі, впливають на його кольоровість.

Для більшості людей відчуття яскравості при сприйнятті кольорових зображень визначається на 59% зеленої складової (G), на 30% червоної складової (R) та на 11% синій складової (B). Якщо відомі зелена, червона і синя складові джерела світла, то яскравість цього джерела, що сприймається, не можна обчислити простим підсумовуванням трьох колірних складових. Необхідно взяти до уваги різну чутливість зору кожної з них. При цьому загальна яскравість обчислюється за такою формулою:

Яскравість = 0,59 · Зелений + 0,3 · Червоний + 0,11 · Синій.

Існує одна серйозна проблема – безперечна суб'єктивність нашого колірного сприйняття. Колір представляє індивідуальне відчуття, і ми не можемо скласти судження про спектральний склад світла. Тому принципово неможливо визначити, наскільки по-іншому сприймають кольори інші люди, тим більше, що навіть у однієї людини колірна чутливість зазнає змін.

У техніці, і особливо при обробці зображень, суб'єктивність небажана. Тільки за наявності об'єктивних вимірювальних систем, що дозволяють встановити однозначне визначення кольоровості, можна забезпечити однакове відтворення того самого кольору відеомоніторами і телевізорами, виготовленими різними фірмами. Саме для цієї мети були розроблені різні математичні методи точного опису кольору, кожен з яких більше підходить для певної сфери застосування. Найважливіші системи, які використовуються практично у всіх програмах обробки зображень, будуть розглянуті в наступних розділах.

1.3.1 Колірна система RGB

Людське око сприймає навколишній світ за допомогою трьох типів кольорочутливих елементів, які називають колбочками. Ці елементи чутливі до випромінювання трьох кольорів: червоного, синього та зеленого, і всі кольори, які може бачити людина, видаються як їх поєднання. Система координат, що базується

на цих кольорах, називається системою RGB. З метою стандартизації Міжнародна комісія з висвітлення CIE надала наступні фіксовані значення довжин хвиль, що відповідають координатним кольорам: Red (Червоний) = 700 нм, Green (Зелений) = 546,1 нм, Blue (Блакитний) = 435,8 нм. Математично найзручніше уявити колірну систему RGB як куба (рис. 1.3).

Для куба характерно, що кожна його просторова точка однозначно визначається координатами X, Y, Z. Якщо по осі X відкладати червону, по осі Y синю, а по осі Z зелену складові кольору, то кожному кольору можна поставити у відповідність певну точку всередині RGB- куб. Крапки, що відповідають червоному, зеленому та синьому кольорам, розташовані у трьох вершинах куба, що лежать на координатних осях. Блакитний, пурпуровий та жовтий кольори розташовані у трьох інших вершинах куба. Чорний колір знаходиться на початку координат, а білий – у найбільш віддаленій від початку координат вершині.

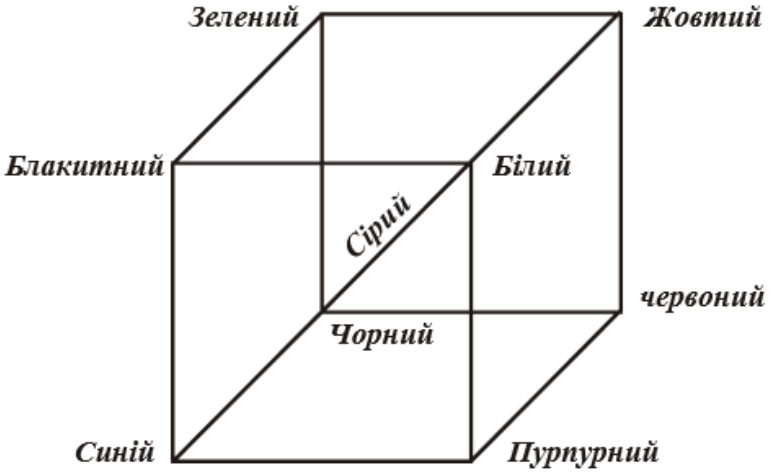


Рисунок 1.3. Колірний RGB-куб

Значення сірого, які відповідають точкам, де червона, зелена та синя складові в кожному випадку однакові, знаходяться на діагоналі між точками з координатами (RGB)=(0,0,0) та (RGB)=(Rmax,Gmax,Bmax). Таким чином, у RGB-системі кольору

видаються трьома чисельними значеннями, які задають червону, зелену та синю складові. Ці три чисельні значення часто називають RGB-тріадою.

В системі RGB кольори визначаються як результат змішування червоного, зеленого та синього кольорів. Тому вона особливо зручна для пристроїв, які самі випромінюють світлові хвилі. Типовими прикладами можуть бути кольоровий монітор і кольоровий телевізор.

Розглянемо RGB зображення, в якому кожна з компонентів – червона, зелена та синя – є 8-бітовою. У такому разі кажуть, що кожен кольоровий RGB піксель (тобто триплет значень (R, G, B)) має глибину 24 біти (три кольорові площини помножити на число бітів на кожну площину); для такого зображення часто використовується термін «повнокольорове зображення». Сумарна кількість різних кольорів у 24-бітовому RGB зображенні становить (28)3= 16777216.

Хоча високоякісні графічні адаптери та монітори забезпечують гарне відтворення кольорів 24-бітових RGB зображень, багато використовуваних в даний час системи обмежені кількістю кольорів, що дорівнює 256. Крім того, існує цілий ряд додатків, в яких має сенс використовувати не більше сотні, а то й менше кількість кольорів. Також бажано мати підмножину кольорів, які відтворювалися точно у всіх використовуваних графічних системах незалежно від їх особливостей. Таке підмножина кольорів називається палітрою фіксованих RGB кольорів (safe RGB colors) або набором кольорів, однаково відтворюваних усіма системами. Стосовно інтернет-додатків це підмножина кольорів називається палітрою фіксованих Web кольорів або набором кольорів, що однаково відтворюються всіма програмами перегляду Інтернет - сайтів.

Якщо виходити з припущення, що 256 кольорів - це мінімальний набір кольорів, які точно відтворюються будь-яким графічним пристроєм, то корисно мати загальноприйнятий стандарт запису цих кольорів.

Відомо, що сорок із цих 256 кольорів відтворюються різними операційними системами по-різному; при цьому залишається 216 кольорів, які є загальними для більшості систем. Ці 216 кольорів стали de facto стандартом фіксованих кольорів,

особливо для інтернет-додатків. Вони використовуються завжди, коли потрібно, щоб відтворювані кольори

виглядали однаково більшість користувачів.

Звернемо увагу, що 216 = (63) , а значить, кожен з 216 варіантів кольору, що розглядаються, можна формувати як і раніше з трьох RGB компонент, але кожна з яких може приймати лише 6 можливих значень: 0, 51, 102, 153, 204 або 255.

Колірна система RGB може здатися дуже простою, але при її практичному застосуванні зустрічаються дві серйозні проблеми. Перша - це залежність від апаратури, а друга пов'язана з тим, що технічно неможливо отримувати всі кольори шляхом адитивного синтезу кольорів.

1.3.2 Колірна система HSV

Загальним недоліком систем RGB та CMYK є їх апаратна залежність. Вказівки лише значень RGB або CMYK недостатньо, щоб однозначно визначити змішаний колір, що виникає в результаті. Це стає можливим, якщо врахувати ще й характеристики апаратури, яка застосовується для отримання кольору. Для цього необхідно розрізняти дві основні властивості кольору, а саме: яскравість та кольоровість.

Як прототип всіх колірних систем, у яких розрізняють яскравість і кольоровість кольору, можна використовувати модель HSV. До інших подібних систем відносяться системи HSI, HLS та YUV. Спільним для них є те, що колір задається вже не як суміш трьох основних кольорів – червоного, зеленого та синього, а визначається шляхом вказівки колірного тону, насиченості та інтенсивності. Наприклад, HSV утворюється зі слів "hue", "saturation" і "value", які означають колірний тон (основна довжина хвилі, що описує колір), насиченість (скільки чистого білого кольору міститься в кольорі) і величину (міра яскравості кольору).

Як і в системі RGB для визначення колірної системи HSV скористаємося геометричним способом. У дещо спрощеному вигляді ця система подається у

вигляді кола (рис. 1.4). У цьому колі розташовуються кольори видимого спектра. Колірний тон («hue») кольору визначається кутом вектора, що виходить із центру кола.

Рисунок 1.4. Розташування кольорів у колі

Існує кілька форм перетворень кольорів з моделі RGB модель HSV. Одна з форм перетворення представлена нижче:

$$\begin{cases} H = \arccos \dfrac{2R - G - B}{\sqrt{6[(R-1/3)^2 + (G-1/3)^2 + (B-1/3)^2]}}, \\ S = 1 - 3\min(R, G, B), \\ V = R + G + B. \end{cases}$$

1.3.3 Колірний простір YCbCr

У цьому колірному просторі компонент Y означає освітленість, а різницеві компоненти Cb і Cr відображають колірну інформацію. Цей колірний простір використовується у цифровому відео. RGB-дані перекладаються YCbCr за допомогою наступної формули:

$$\begin{bmatrix} Y \\ C_b \\ C_r \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65,4810 & 128,553 & 24,966 \\ -37,797 & -74,203 & 112,000 \\ 112,000 & -93,786 & -18,214 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

1.4 Формати графічних файлів

Нині у світі відомо кілька десятків форматів графічних файлів. Найбільш популярними є представлені далі.

1.4.1 Формат BMP (Bit Map)

BMP – це стандартний формат файлів для комп'ютерів під керуванням Windows. Цей формат розроблений компанією Microsoft для зберігання растрових зображень у машинно-незалежному форматі, що дозволяє Windows відображати графічну інформацію на дисплеї. Термін "машинно-незалежний" означає, що колір пікселя зберігається у формі, незалежної від методу, яким кольори відображаються на дисплеї.

BMP завжди містить растрові дані. Файл може бути:

- 1 біт: 2 кольори (монохромне зображення);
- 4 біти: 16 кольорів;
- 8 біт: 256 кольорів;
- 16 біт: 65536 кольорів;
- 24 біти: 16,7 млн. кольорів.

1.4.2 Формат PNG (Portable Network Graphics)

Версія формату PNG вийшла в 1996 році і була рекомендована консорціумом W3C як повноправний мережевий формат. PNG використовує відкритий, не запатентований алгоритм стиснення Deflation, безкоштовні реалізації якого

доступні в Інтернеті та використовуються багатьма програмами компресії даних, у тому числі PKZIP та GNU GZIP.

Формат PNG зберігає графічну інформацію у стислому вигляді. Причому цей стиск здійснюється «без втрат», на відміну, наприклад, від стандартного JPEG (навіть із максимально високим рівнем якості). Він позиціонується насамперед для використання в мережі Інтернет та редагування графіки.

PNG має такі основні переваги перед GIF:

— кількість кольорів у зображенні не обмежується 256;

— можливість збереження альфа-каналу (напівпрозорі зображення);

— використання гамма-корекції (міжплатформне керування яскравістю зображення).

Формат PNG має більш сильний рівень стиснення для файлів з більшою кількістю кольорів, ніж GIF, але різниця становить близько 5-25%, що недостатньо для абсолютного переважання формату, так як маленькі 2-16 кольорові картинки GIF стискає як мінімум не гірше.

Існує одна особливість GIF, яку PNG не намагається відтворити - це підтримка множинного зображення, особливо анімації, PNG призначений лише для збереження одного зображення у файлі. Для передачі множинних зображень використовується розширений формат MNG, опублікований у середині 1999 року.

PNG є гарним форматом для редагування зображень, для зберігання проміжних стадій редагування, оскільки відновлення та перезбереження зображення відбуваються без втрат

1.4.3 Формат SVG (Scalable Vector Graphics)

Порівняно нова мова розмітки, призначена для опису двовимірної векторної графіки (як нерухомої, так і анімованої). Це відкритий стандарт, що базується на XML і розробляється консорціумом W3C. Специфікацію SVG 1.0 було випущено 1999 року. У ній визначалося, як у XML записується структура даних, створюється SVG анімація, SVG програми та документи. Перевагами цього формату є:

— текстовий формат – файли SVG можна читати та редагувати широким колом додатків, і вони зазвичай менші за розміром, ніж порівняні зображення у форматах JPEG та GIF;

— SVG надає можливість роботи з 16-мільйонною палітрою, підтримує ICC профілі, sRGB, градієнти та маски;

— текст у SVG можна виділяти та знаходити за допомогою пошукових машин. Можна шукати заданий рядок тексту – наприклад ім'я міста на SVG-карті;

— SVG надасть усі переваги XML:

- можливість роботи у різних середовищах;
- інтернаціоналізація (підтримка Unicode);
- широка доступність для різних програм;
- Легка модифікація через стандартні API.

Можливості використання формату SVG широкі, а завдяки трансформаціям XML у SVG ці можливості стають практично безмежними. Прикладом такої гнучкості може бути, наприклад, перетворення накопичених статистичних даних населення у кольорову карту залежно від вибраних районів і часового інтервалу. Зміни, внесені до XML-даних, негайно відображаються у SVG-графіці.

Розробка вдосконаленої специфікації SVG 1.1, як і раніше, триває. Також у розробці знаходяться специфікації SVG Basic та SVG Tiny, призначені для мобільних пристроїв, що дозволить у майбутньому мобільним телефонам та налагодженим комп'ютерам виводити на екран SVG-файли.

1.5. Формат RAW

Для початку необхідно встановити, що RAW як стандартизований формат не існує. Структура RAW-файлів змінюється від камери до камери і навіть виробів одного виробника може відрізнятися. Справа в тому, що RAW-файли являють собою ніяк не оброблені дані з матриці фотоапарата, які стискаються архіватором (а в деяких камерах зовсім не стискаються), і до них прикріплюється заголовок з параметрами зйомки (модель камери, дата зйомки, витримка, діафрагма і т.д.) і все

це разом записується у файл, званий RAW. Вся подальша обробка цих даних перекладається на плечі програми-конвертера.

Під час зйомки у форматах JPEG та TIFF інтерполяцією займається вбудований процесор фотокамери, і на носій записується вже перетворений файл.

За лічені частки секунди після захоплення матрицею кадру, вбудований у камеру софт повинен:

— проаналізувати «сирі» дані;

— вирішити, наскільки яскравим має бути в результаті кадр;

— Визначити колірну температуру знімка.

Зрозуміло, що заради швидкості доводиться багатьом жертвувати, і якщо з типовими сюжетами – не дуже контрастними, з нормальним світлом – автоматика камери справляється добре, то складніші кадри вона може передати неправильно, а то й зовсім зіпсувати. RAW дозволяє в досить широких межах коригувати помилки зйомки.

Фокус полягає в тому, що в RAW-файлах міститься набагато більше інформації, ніж у зображенні, що виходить в результаті. JPEG і навіть TIFF файли з камер мають глибину кольору

8 біт на канал. У RAW записується інформація з матриці, яка зазвичай має розрядність 10 або частіше 12 біт на піксель, що дає 1024 - 4096 градацій яскравості. Це дає великий запас для різноманітних перетворень картинки.

Ось основні коригування, які можна робити при обробці RAW-файлів:

— корекція експозиції;

— корекція світла (баланс білого);

— керування глибиною кольору та колірними профілями.

2 ВИКОРИСТАННЯ МОЖЛИВОСТЕЙ ТА БАЗОВИХ ФУНКЦІЙ У ПРОГРАМНОМУ ПАКЕТІ

Будь яке зображення можливо створити, створивши матрицю. Для прикладу, створимо матрицю 5 х 5.

```
>>A=[1 2 3 0 0; 4 5 6 0 0; 7 8 9 0 0; 0 0 0 0 0; 0 0 0 0 0]
A =
     1     2     3     0     0
     4     5     6     0     0
     7     8     9     0     0
     0     0     0     0     0
     0     0     0     0     0
```

Щоб її побачити (рис.2.1) як зображення використовуємо команду:

```
>>imshow(A); % малюнок зліва
```

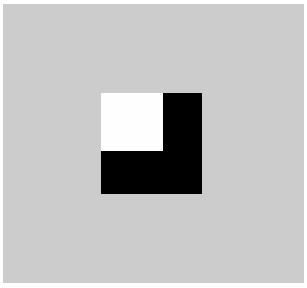


Рисунок 2.1 – Зображення матриці розмірністю 5х5

Також на (рис.2.2) показана невелику частину графічного вікна MATLAB. Функція mat2gray(A) перекладає елементи матриці А до діапазону [0,1].

```
>>B=mat2gray(A)
B =
    0.1111    0.2222    0.3333         0         0
    0.4444    0.5556    0.6667         0         0
    0.7778    0.8889    1.0000         0         0
         0         0         0         0         0
         0         0         0         0         0
```

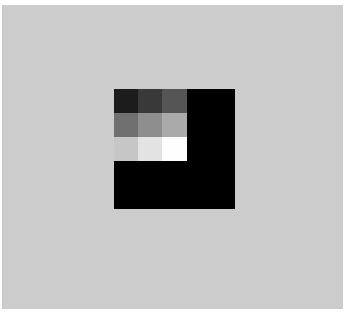


Рисунок 2.2 – Зображення матриці розмірністю 5x5

А функціяim2uint8(B) перетворює матрицю-аргумент до діапазону [0,255].

```
>>C=im2uint8(B)
C =
             28     57     85    0       0
            113    142    170    0       0
            198    227    255    0       0
              0      0      0    0       0
              0      0      0    0       0
```

>>imshow(C);% продемонстровано на (рис. 2.2).

Також у вікні Workspace можливо переглянути тип матриць:

```
A – double, B – double, C – uint8.
```

Є також функція im2uint16, яка перетворює величини в діапазон [065535].

Для матриці A, створеної вище,

```
>>A=[1 2 3 0 0; 4 5 6 0 0; 7 8 9 0 0; 0 0 0 0 0; 00000];
```

команда

```
>>image(A);
```

створює велике зображення (з величезними пікселями) у вікні. Для завантаження (читання) зображень у робочий простір програмного пакету використовується функція imread з наступним синтаксисом:

```
>>imread('filename')
```

Тут filename - це рядок символів, що утворюють повне ім'я файлу зображення, що завантажується (включаючи будь-яке розширення). Наприклад, командний рядок:

```
>> f=imread('football.jpg');
```

надає зображення формату JPEG з ім'ям «football.jpg» матричної змінної f. Зауважимо, що символ ' (апостроф) використовується як обмежувач символьного рядка. Точка з комою в кінці командного рядка означає інструкцію системі MATLAB не відображати виведення у її командне вікно. Якщо точка з комою відсутня, то MATLAB відображає результат (виведення) виконання операцій у командному рядку. Символ запрошення >> означає початок командного рядка у вікні команд MATLAB.

Якщо в ім'я файлу не включена інформація про шлях до цього файлу, файл filename шукається в поточній папці. А якщо його там немає, то виконується пошук файлу у всіх папках, шляхи до яких вказані в шляхах пошуку MATLAB.

Можемо припустити, що це графічні файли зібрані у одному каталозі, який у MATLAB встановлені шляхи пошуку.

Функція size(f) повертає розмір зображення,тобто число рядків і стовпців масиву, що представляє зображення:

```
size(f)% f містить зображення football.jpg  ans
= 256320 3
```

Ця функція буде особливо корисною для автоматичного визначення розміру зображення, яке робиться операцією:

```
[M, N, K] = size (f);
```

За такого запису змінної M буде присвоєночисло рядків зображення, а змінної N – число стовпців, K – число колірних площин.

Функція whosповідомляє додаткову інформацію про масиву.

Наприклад, рядок

```
whos f
```

видає наступний результат:

```
NameSizeBytesClass
 f256x320x3245760  uint8 array Grand total is 245760 elements using
245760 bytes
```

Функція numel(f) повертає кількість елементів масиву f, тобто. число пікселів зображення

```
numel(f) ans = 245760
```

Як ми говорили вище, завантажене зображення можна вивести на дисплей комп'ютера за допомогою функції imshow, яка має наступний синтаксис:

```
imshow(f);
```

або

```
imshow(f, G),
```

де f – це матриця зображення, а G – це кількість рівнів яскравості, що використовується при відображенні цього зображення.

Якщо аргумент G опущений, то за замовчуванням приймається 256 рівнів яскравості (то отримане з графічного файлу).

Команда

```
imshow(f,[low high])
```

означає, що всі пікселі зі значенням не більше за число low треба показувати чорними, а всі пікселі зі значеннями не менше числа high - білими. Всі значення, розташовані між low і high, показуються з проміжною яскравістю з використанням числа рівнів, прийнятого за умовчанням.

Зрештою, запис у командному рядку

```
imshow(f,[ ] )
```

ставить для змінноїlow мінімальне значення масиву f, а змінній high присвоюється його максимальне значення. Така форма функції imshow є корисною при показі зображень, що мають вузький динамічний діапазон значень пікселів, або коли серед них є позитивні та негативні значення.

Будь яке зображення після обробки має бути збережене у графічному файлі. Це можна виконати командою

```
imwrite(A,filename,fmt),
```

яка зберігає образ A у файл, заданий ім'ям filename у графічному форматі, визначеному аргументом fmt.

Наприклад, для зображення футбольного м'яча, що зберігається у файлі 'football.jpg', наступні команди створять графічний файл того самого зображення, але у форматі bmp.

```
f=imread('football.jpg'); % зазвичай слідує обробка зображення f
```

```
imwrite(f,'football.bmp','bmp');
```

Одна з двох форм графічних файлів у MATLAB має вигляд матриці з елементами, що зазвичай представляються цілими числами, які є індексами (номерами) кольорів у матриці званою картою або палітрою кольорів (colormap).

Розглянемо для прикладу вхідний рисунок(рис. 2.3), який знаходиться в графічному файлі Pogorelov32x32.bmp, розміром 32 на 32 пікселі.

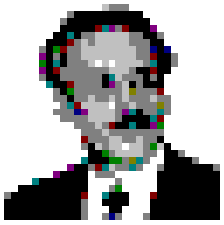


Рисунок 2.3 – Демонстраційне вхідне зображення

Команда

```
A=imread('Pogorelov32x32.bmp')
```

читає та друкує матрицю A розміром 32 x 32. Значення елементів матриці є номерами кольорів відповідних пікселів. Нижче наведено фрагмент цієї матриці

```
15  15  15  15   0   0 0    0   0   1   6 …
15   0   0   6   0   7 7    8   8   8   8 …
15  15   0   0   0   1 7    8   7   8   8 …
15   7   0   2   1   7 7    8   8   8   8 …
15   5   0   7   0   7 7    8   8   8   8 …
```

Команда

```
image(A);
```

по приведеній вище матриці малюємо картинку у графічному вікні (рис. 2.4).

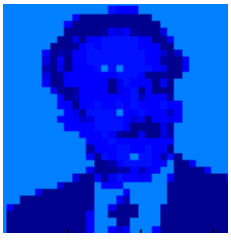


Рисунок 2.4 – Відновлене демонстраційне вхідне зображення

Відновлене зображення було відновлено з помилкою і для отримання прийнятного зображення необхідно змінити параметри його відображення.

Коли матриця містить цілі позитивні числа, її елементи інтерпретуються як індекси в карті кольорів. Функція image(A) будує її образ у поточній карті – кожному елементу матриці буде відповідати певний прямокутник на малюнку, зафарбований кольором, номер якого відповідає рядку матриці кольорів (палітра). Є також додаткові аргументи, які керують роботою функції image.

Отриманим чином у графічному вікні можна керувати, використовуючи звичайні функції MatLab, що керують графікою. Зокрема, у прикладі малюнок витягнутий по горизонталі.

Команда:

```
axis equal;
```

вирівнює горизонтальні та вертикальні розміри пікселів у графічному вікні.

Вказана команда встановлює коефіцієнт стиснення зображення однаковим по всіх осях (див. попередній рисунок праворуч). Той самий результат дає команда

```
axis image;
```

але охоплюючи зображення прямокутник у графічному вікні деформується під розмір зображення. Кольори точок беруться із поточної карти кольорів. Карта кольорів (палітра) - Це матриця, яка містить 3 стовпці і кілька (може бути багато) рядків.

Наприклад, команда

```
gray(5)
```

створює наступну матрицю, яка може бути використана як палітра кольорів

```
         0         0         0
    0.2500    0.2500    0.2500
    0.5000    0.5000    0.5000
    0.7500    0.7500    0.7500
    1.0000    1.0000    1.0000
```

Карта кольорів `colormap` є $m\times 3$ матрицею дійсних чисел діапазон змін яких від 0 до 1. Кожен її рядок є RGB вектор, який визначає один колір. Рядок картки

квітів з номером *k* визначає *k*- й колір у форматі [*r*(*k*)*g*(*k*)*b*(*k*)] з відповідними інтенсивностями червоногоr(k), зеленого g(k) та синього b(k) кольорів. Ми бачили, що матриця A, прочитана з файлу містить числа від 0 до 15. Для того, щоб переконатися в цьому, достатньо виконати команди

```
min(min(A)
) ans=0
max(max(A)
) ans=15
```

Команди max і min визначають найбільше та найменше значення матриці по одній з її розмірностей. Наприклад, одноразове застосування команди max(A) повертає вектор із 32 чисел, що є максимальними значеннями за відповідними стовпцями матриці A. Для визначення максимального значення всієї матриці потрібно ще раз застосувати функцію max до результату її першого застосування.

Оскільки матриця A містить числа від 0 до 15, розумно створити чорно-білу палітру з 16 відтінками сірого. Команда colormap(gray(16)); перетворює вхідне зображення до вигляду (рис. 2.5).

Рисунок 2.5 – Відновлене зображення у 16ти відтінках сірого

Зверніть увагу на тип елементів матриці «A» у вікні Workspace - uint8 (беззнакові 8 - бітові цілі можуть змінюватися в діапазоні від 0 до 255). Якщо виконати команду

```
B=A/15;
```

то операція поділу цілих чисел дасть тільки 0 або 1, а тип елементів матриці *B* не зміниться (розділити ціле на ціле дає ціле). Це дозволяє створити чорно-білий

малюнок (рис. 2.6). Для цього створимо палітру з двох кольорів – чорного (йому відповідає рядок 0, 0, 0) та білого (йому відповідає рядок 1, 1, 1).

```
cm=[0 0 0;1 11]; %     матриця палітри (два кольори – чорний та білий)
image(B);
axis image; colormap(cm);
```

```
>> cm
cm =
     0     0     0
     1     1     1
```

Рисунок 2.6 – Відновлене чорно-біле зображення

Команда

```
colormap('default')
```

встановлює карту кольорів за замовчуванням.

Графічний файл міг бути монохромним (рис. 2.7). Наприклад, якщо зображення створено у

`Paint-e` чорно-біле зображення 32 х 32 пікселів AuthorMono.bmp

Рисунок 2.7 – Відновлене чорно-біле зображення

Команда

```
P=imread('AuthorMono.bmp')
```

надрукувала матрицю32 х 32 пікселів, складених тільки з нулів або одиниць. Команда

```
image (P); axis image; colormap('default');
```

намалювала дуже темну картинку. Але команди

```
cm=[0 0 0;1 1 1];colormap(cm);
```

перетворять картинку до виду, що показано вище (рис. 2.8).

Щоб дізнатися формат графічного файлу, можна використовувати команду

```
info = imfinfo('Pogorelov32x32.bmp')
```

Вона повертає структуру з великою кількістю полів, що містить різноманітну інформацію про графічний файл. Тут наводяться деякі поля цієї структури

```
info =
Filename: 'Pogorelov32x32.bmp'
FileSize: 630
```

Формат: `'bmp'`

```
FormatVersion: 'Version 3 (Microsoft Windows 3.x)'
Width: 32
Height: 32
 BitDepth: 4 ColorType:
'indexed'
NumColormapEntries: 16
 Colormap: [16x3 double]
NumPlanes: 1
```

Необхідно звернути увагу на рядок із параметром ColorType, у якому вказується тип графічного файлу, оскільки розглядаються лише графічні файли типу "indexed".

Команда

```
[X,map] = imread(filename)
```

читає графічний файл у матрицю X, а його карту кольорів у матрицю map, елементи якої масштабуються до діапазону [0, 1].

```
[A, m] = imread('Pogorelov32x32.bmp');
image(A);
colormap(m);
```

Команда

```
[A,map,alpha] = imread(...)
```

у третьому параметрі повертає маску (матрицю з логічних нулів та одиниць), яка використовується для визначення інформації про прозорість точок образу (які

точки образу відображати на екрані, а які ні; не відображати ті, для яких елементи alpha рівні 1). Наприклад, цей параметр корисний для малювання зображення файлу іконки.

```
[A,B,C]=imread('Author.ico'); % A містить числа від 0 до 255.
image(A); axisimage;% лівий малюнок
colormap(B); середній малюнок
```

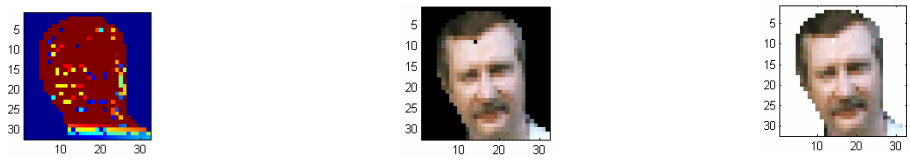


Рисунок 2.8 – Відновлене кольорове зображення

Тепер зробимо малюнок білим.

```
B2=B;% копіюємо палітру в B2
B2(256,:)=[1 1 1]; % останній колір палітри зробити білим
кольором D = ones(size(A)) * (length(B2)-1); %створити
матрицю з елементами 255 D(C == 0) = A(C == 0);
image(uint8(D)), colormap(B2); %правий малюнок
axisimage;
```

Пояснимо роботу рядка D(C==0)=A(C==0).

Якщо подивитися у вікні Workspace, то видно, що масив C є логічним. І при цьому він має такий самий розмір, як і масив A. Тут використовується логічне індексування у формі

```
ім'я_матриці (логічний масив)
```

де логічний масив повинен мати той самий розмір, як і матриця.

Результатом цієї операції логічного індексування є вектор, складений з елементів вихідної матриці, для яких у логічному масиві відповідні елементи дорівнюють логічній одиниці. Цей вектор може стояти як у правій частині оператора присвоєння, так і у лівій. Спочатку ми створили double масив D з чисел 255, а потім створили вектор однакової довжини D(C==0) і A(C==0). Другий вектор містить лише елементи матриці A, котрим у матриці C елементи дорівнюють нулю. Це означає, що відповідні точки образу мають бути

намальовані. Значення цього вектора A(C==0) надаються елементам вектора D(C==0), тобто. відповідним елементам матриці D.

Якщо матриці C елемент був 1, то відповідний елемент матриці D змінюється, тобто. залишається рівним 255. В результаті елементи матриці A, що відповідають точкам, що відображаються, скопіювалися в матрицю D, а відповідні точкам, що не відображаються, залишилися в матриці D рівними 255.

Крім того, знадобилось замінити останній (256 - й) рядок матриці палітри, щоб вона відповідала білому кольору. Оскільки функція image відображає лише матриці з типом елементів uint8 (діапазон зміни чисел від 0 до 255), нам також знадобилося перетворення типу речовинної матриці D в тип uint8.

Команда *image(x,y,C)*, де x та y є двоелементними векторами, які визначають діапазон зміни координат x та y, малює такий самий образ, як і команда image(C). Це корисно, наприклад, у разі, коли необхідно, щоб мітки осей відповідали реальним фізичним координатам графічного образу (рис. 2.9).

```
A = magic(5);
x = [-2.5 2.5]; y = [0.5 5.5];
image(x,y,A), axis image, colormap(jet(25))
```

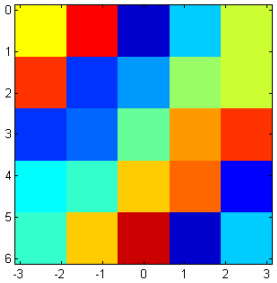


Рисунок 2.9 –Зображення з фізичним координатам графічного образу

Функція imshow(M) теж будує графічний образ напівтонової (тобто одноколірної з різними градаціями яскравості) матриці M елементи якої мають тип double (або uint8, uint16). При цьому кожній точці матриці відповідає один піксель у вікні графічному. Варіантів використання цієї функції дуже багато.

Вона є базовою графічною функцією Image Processing Toolbox. З різними варіантами її використання слід ознайомитись за довідковою системою.

```
I = zeros(4,4) %генерування нульової матриці розміром 4 x 4
I=0000
0000
0000
0000
I(2:3, 2:3) =1%        зміна значень матриці
I=0000
0110
0110
0000
imshow(I);%            побудова образу матриці
```

Тут одиниці у матриці I відповідають білим точкам малюнка, а нулі – чорним. Але точок мало, тому рисунок маленький. Якби використовувалась функція image, то треба було б виконати перетворення типу, наприклад, так image(uint8(I)), і при цьому образ матриці був би побудований за розміром графічного вікна.

Збільшимо розмір малюнка, збільшивши розмір матриці

```
I = zeros (100,100);
I(25:75, 25:75) = 1;
colormap('default');
imshow(I);%       наступний (рис.2.10)
colormap(hot); % наступний (рис.2.11)
```

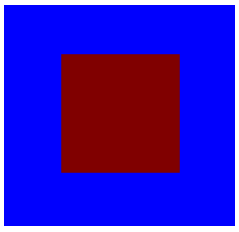


Рисунок 2.10 – Зображення побудоване функцією imshow

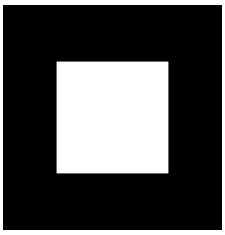


Рисунок 2.11 – Зображення побудоване функцією colormap

Слово hot є ім'ям зумовленої матриці палітри. Імена готових матриць палітр можна знайти у довідковій системі. Нижче ще приклад використання функції imshow (рис.2.12).

```
moon = imread('moon.tif');
imshow(moon);
```

Рисунок 2.12 – Зображення побудоване функцією imshow

або можна відразу використовувати ім'я файлу

```
imshow('moon.tif');
```

Команда

```
imshow(I,[low high])
```

синтезує образ матриці I у сірих відтінкахз діапазоном зміни значень I дорівнює [low high].

Значення low та менші малюються чорним кольором, значення high та великі малюються білим. Значення між low і high малюються проміжними відтінками сірого, використовуючи рівні сірого, що визначаються за умовчанням. Якщо використовується порожня матриця [] замість [low high], то

imshow використовує [min(I(:))max(I(:))], тобто. мінімальне значення I відображається чорним, а максимальне – білим кольорами.

Вгадувати палітру чи використовувати палітру графічного файлу не завжди зручно. Тому в MatLab є функція imagesc, яка масштабує матрицю під поточну палітру, а потім малює її образ.

Команда imagesc(A) відображає матрицю як графічний образ. Кожен елемент матриці відповідає прямокутній ділянці образу. Значення елементів є індексами рядків поточної матриці палітри, де трійка чисел кожного рядка у форматі RGB визначає колір.

Команда imagesc(x,y,A) відображає матрицю A у вигляді графічного образу на координатній сітці з діапазоном, що визначається векторами x і y. Наприклад

```
A=[1 2 3; 4 5 6; 7 8 9];
imagesc(A); colormap('default'); % наступний (рис.2.13)
imagesc([0 10],[020], A); % наступний (рис.2.14)
```

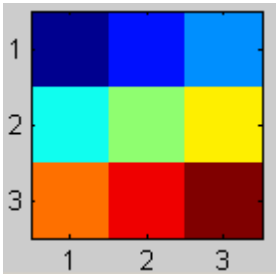


Рисунок 2.13 – Зображення побудоване функцією imshow

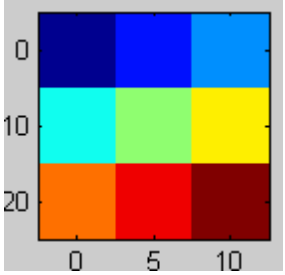


Рисунок 2.14 – Зображення побудоване функцією imshow

Іноді бажано відобразити матрицю як поверхню над прямокутною областю. Це можна зробити за допомогою функції surface(Z), де матриця Z містить речові числа, які інтерпретуватимуться як висота поверхні над площиною XY. Але матриці, прочитані із графічного файлу, містили лише цілі числа. Тут нам може знадобитися перетворення типу. Наприклад, команда

```
A = imread('Pogorelov32x32.bmp');
C = double(A)/16;
```

створить речову матрицю double(A), кожен елемент якої буде речовим числом, які після поділу на 16 створять матрицю C з дійсними елементами в діапазоні від 0 до 1 (рис.2.15).

```
C=double(A)/16;
surface(C);%      малює поверхню матриці з double елементами,
surface(C*10); colormap(hot);
```

Аналогічно працює команди surf(Z) та surfc(Z)які створюють 3 - х мірний графік поверхні з координатам точок, заданим у матриці Z використовуючи x=1:n і y=1:m, де [m,n]=size(Z). Висоти точок поверхні Z є речовими числами, а колір поверхні пропорційний висоті крапок (рис.2.16).

```
A = imread('Pogorelov32x32.bmp')
C=double(A)/16;
surfc(C); colormap(gray(17));
```

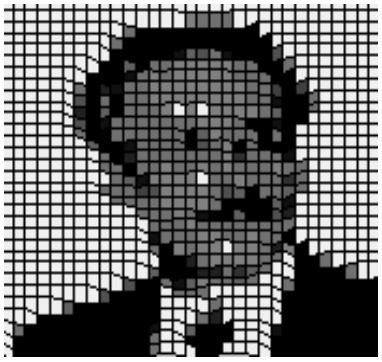


Рисунок 2.15 – Секторізоване зображення

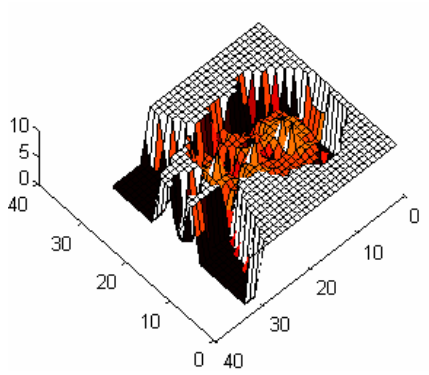


Рисунок 2.16 – Тривимірна модель вхідного зображення

Корисною може бути функція побудови каркасного зображення поверхні. Один з синтаксисів, що найчастіше використовуються, наступний

```
mesh (X, Y, Z);
```

де матриці X,Y,Z повинні бути однаковими розміру. Графік поверхні будується над прямокутною областю площини XY, яка розбивається сіткою розміром таким же, як у матриць. У вузлах сітки значення функції дорівнюють відповідним значенням елементів матриці Z. Матриці X та Y містять x та y координати вузлів сітки.

Побудуємо поодиноку на півсферу за рівнянням

$$z = \begin{cases} \sqrt{1 - x^2 - y^2}, & x^2 + y^2 \le 1 \\ 0 \quad , & x^2 + y^2 > 1 \end{cases},$$

де $1\ x \le\le -1, 1\ y \le\le -1$

```
[X Y]=meshgrid(-1:0.1:1);
XY=X.^2+Y.^2;
C=XY<=1;
XYZ=zeros(size(XY)); % нульова матриця
XYZ(C)=1-XY(C);%          логічне індексування
```

```
Z = sqrt
(XYZ); mesh
(X, Y, Z);
```

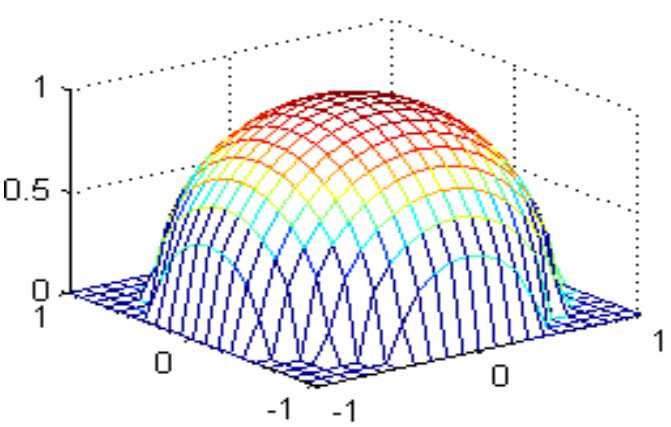


Рисунок 2.16 – Тривимірна модель вхідного зображення

Всі приведені приклади використання вбудованих функцій це лише логічне індексування так.

Висновки за розділом: перерахуємо функції та їх значення, які розглянуті у цьому розділі:

`imread('файл',...)`	`читання графічного файлу`
`imfinfo('файл',...)`	`інформація про графічний файл`
`image(матриця,...)`	графічний образ матриці
`colormap(матриця(mx 3))`	завдання палітри
`imagesc(матриця,...)`	відображає матрицю, кожен елемент матриці відповідає прямокутній ділянці образу.
`surface(матриця,...)`	відобразити матрицю як поверхню над прямокутною областю
`surf(матриця,...)`	побудова графіка поверхні
`mesh(матриця,...)`	побудови каркасного зображення

3 ОБРОБКА ФОТОГРАФІЙ У ПРОГРАМНОМУ ПАКЕТІ

Під адаптивністю методу мається на увазі наступне: заздалегідь може бути невідомо, яким шумом спотворено зображення. Оскільки фільтр обробляє зображення поточечно, одночасно з обробкою проводиться аналіз зображення на присутніх шуми, з метою подальшого прискорення роботи фільтра.

3.1 Метод LPA

Серед різних підходів до видалення шуму, LPA (локально-поліноміальна апроксимація) може розглядатися як одна з найбільш теоретично обґрунтованих та досліджених. Спочатку LPA використовувалася в статистиці для обробки скалярних та багатовимірних зашумлених даних. Це техніка, яка використовується для непараметричної оцінки з використанням поліноміальних даних, вписаних у ковзне вікно. Ідея LPA методу така: передбачається, що функція інтенсивності y(x) добре наближається поліномом в деякій околиці даної точки x. Коефіцієнти перебувають виваженим шляхом найменших квадратів. Отримане наближення використовують, щоб обчислити оцінку для точки, що розглядається, яку називають "центром" LPA. Для інших точок обчислення проводяться окремо.

Для вибору розміру адаптивного вікна використовуються дві основні ідеї. Перша заснована на наближенні зсуву та дисперсії оцінки сигналу з відповідним обчисленням оптимального розміру вікна, заснованому на теоретичних формулах. Однак для цього знання похідних даного сигналу. Таким чином, стає необхідним оцінювати ці похідні за допомогою допоміжних вікон. Такі методи є досить складними для реалізації.

Друга альтернативна ідея не використовує оцінки усунення. Ця група методів ґрунтується на статистиці якості фільтрації, яка використовується для прямої оптимізації точності.

ICI rule (правило перетину інтервалів довіри) – критерій, що використовується для адаптивного вибору розміру вікна. Суть алгоритму полягає у пошуку найбільшої околиці точки оцінки, де припущення локальної поліноміальної апроксимації добре вписуються в дані. Оцінки обчислюються для різних параметрів і порівнюються. Адаптивний параметр визначається як найбільший, для якого оцінка не відрізняється від оцінок для менших параметрів.

Розглянемо зашумлені спостереженняz такого виду:

$$z(x) = y(x) + \varepsilon(x),\ x \in X,$$

$y : X \to R$ функція інтенсивності вихідного чорно-білого зображення

$x(s)=(x_1(s), x_2(s))$ – двовимірний вектор з компонентами $x_1(s)$ і $x_2(s)$ де параметр s відповідає s тому пікселю на зображенні. ε -незалежні однаково розподілені випадкові Гауссові помилки, їх розподіл має вигляд N(0, σ2), а X - таблиця. Передбачається, що функція y(x) - кусково-безперервна і належить класу Fr :

$$F_r = \left\{ \left| D^r_{r_1,r_2} y(x) \right| \le L_r(x) \le \bar{L}_r, \forall r_1 + r_2 = r \right\}, \quad (2)$$

де $D^{\kappa_1+\kappa_2}_{r_1,r_2} = \dfrac{\partial^{\kappa_1+\kappa_2}}{\partial^{r_1} x_1 \partial^{r_2} x_2}$ - оператор диференціювання, а L_r - константи

) за цією функцією z(x) з найменшим значенням СКО. Локально-поліноміальна апроксимація функції y(x) виглядає так. Спочатку поруч Тейлора наближається інтенсивність y(x), потім це розкладання використовується локально відносно маленької околиці.

Власне, локальне розкладання використовується обчислення оцінки інтенсивності лише центрального пікселя. Для наступного пікселя обчислення повторюються. Така поточкова процедура визначає непараметричний характер оцінювання.

Для LPA застосовується наступна функція:

$$J_h(x) = \sum_s w_h(x(s) - x)(z(x(s)) - C^T \phi(x(s) - x))^2,$$

$$\phi(x) = (\phi_1(x), \phi_2(x), ..., \phi_M(x))^T,$$

$$C = (C_1, C_2, ..., C_M)^T, x(s) = (x_1(s), x_2(s)),$$

$\phi(x)$е Набір сімейство лінійно-незалежних поліномів ступеня від 0 до m, та $\phi_1 = 1$

$$M = C_{2+m}^2 = \frac{(2+m)(1+m)}{2}.$$

. Загальна кількість поліномів

Вікно $w_h(x) = w(x/h)/h$ — е вагова функція, а параметр $h>0$ визначає розмір.

Для m=2 та M=6 повне сімейство лінійно-незалежних багаточленів має вигляд:

$$\phi_1 = 1, for\ m = 0;$$
$$\phi_2 = x_1, \phi_3 = x_2, for\ m = 1;$$
$$\phi_4 = x_1 x_2, \phi_5 = x_1^2 / 2, \phi_6 = x_2^2 / 2, for\ m = 2.$$

Можна перевірити, що у випадку, коли *y(x)* є багаточленом ступеня *m*, така оцінка точна. Зокрема, для багаточленів що $\phi_1, ..., \phi_M$ це означає що

$$\sum_s g_1(x, x(s), h)\phi_k(x(s)) = \phi_k(x), для\ k = 1, ..., M, \forall x.$$

Це показує, що перетворення з вагою g1має точну репродуктивність для двовимірних поліноміальних компонентів інтенсивності.

3.2 Точність LPA

Відомо, що вибір розміру вікна є ключовим моментом ефективності локальної оцінки. Коли *h* мало, LPA дає хороше наближення y(x), але тоді використовується менше даних та оцінка більш чутлива до шуму. Якщо ж *h* велике, то даних використовується багато і зображення буде розмито. Нев'язка може бути представлена таким чином:

$$e(x,h) = y(x) - \hat{y}(x,h) = y(x) - \sum_s g_1(x, x(s), h) z(x(s)) = E(e(x,h)) + e^0(x,h),$$

де:

$$E(e(x,h)) = y(x) - \sum_s g_1(x, x(s), h) y(x(s)) = \sum_s g_1(x, x(s), h)[y(x) - y(x(s))]$$

и

$$e^0(x,h) = -\sum_s g_1(x, x(s), h)\varepsilon(x(s))$$

- це зсув та випадкова компонента нев'язки.
-

Для обчислення різниці y(x)-y(x(s)) запишемо залишковий член ряду Тейлора у формі Лагранжа:

$$y(x) - y(x(s)) = S_1(s) + S_2(s);$$

$$S_1(s) = \sum_{k_1+k_2=1}^{r-1} \frac{1}{k_1!k_2!}[x_1 - x_1(s)]^{k_1}[x_1 - x_1(s)]^{k_2} D_{k_1,k_2}^{k_1+k_2} y(x)$$

$$S_2(s) = \sum_{k_1+k_2=r} \frac{1}{k_1!k_2!}[x_1 - x_1(s)]^{k_1}[x_1 - x_1(s)]^{k_2} D_{k_1,k_2}^{r} y(\lambda_s x(s) + (1-\lambda_s)x), \quad 0 \le \lambda_s \le 1.$$

Дисперсія випадкової компоненти задається рівнянням:

$$\mathrm{var}(x,h) = E\left(e^0(x,h)^2\right) = \sigma^2 \sum_s g_1(x, x(s), h)^2.$$

Щоб отримати формули, що характеризують точність в залежності від параметра *h*, припустимо, що інтервал дискретизації Δ малий та $\Delta \to 0$.

Тоді всі формули, наведені вище, можна перетворити на інтеграли і отримати такі вирази:

$$\hat{y}(x,h) = \iint g_1(u) z(x+hu) du_1 du_2,$$
$$g = \Phi^{-1} w(u) \phi(u),$$
$$\Phi = \iint w(u) \phi(u) \phi^T(u) du_1 du_2,$$
$$g = (g_1, \ldots, g_M)^T, \quad u = (u_1, u_2).$$

Тоді формули зміщення та дисперсії задаються у явному аналітичному вигляді:

$$|E(e(x,h))| \le h^{m+1} L_{m+1}(x) A,$$
$$A = \sum_{k_1+k_2=m+1} \frac{1}{k_1! k_2!} \iint |g_1(u)| |u_1|^{k_1} |u_2|^{k_2} du_1 du_2,$$
$$\mathrm{var}(x,h) \approx \frac{\Delta^2 \sigma^2}{h^2} B, \quad B = \iint |g_1(u)|^2 du_1 du_2.$$

Ці формули (16) показують, що ідеальний розмір вікна *h*(x)* залежить від *(m+1)*-ої похідної *y(x)* та ідеальне поєднання зсуву та ди $\overline{\omega}^*(x) / std^*(x)$ ться, коли відношення зміщення та стандартного відхилення

дорівнює γ . Можна помітити, що

$$\overline{\omega}(x,h) \begin{cases} < \gamma \cdot std(x,h), \text{ когда } h < h^* \\ > \gamma \cdot std(x,h), \text{ когда } h > h^* \end{cases}$$

Ця нерівність використовується для перевірки гіпотези $h< > \eta$ *.

3.3Розширення алгоритму

Наведені вище алгоритми дозволяють ефективно видаляти Гаусов шум із зображення. Але якщо застосувати його до зображення, спотвореного імпульсним шумом, зображення залишиться тим самим (рис. 3.1). При цьому техніка ICI дозволяє для кожного пікселя знаходити околицю, в якій пікселі «не занадто сильно» відрізняються від центру околиці.

Рисунок 3.1 - Приклад адаптивного вікна.

Для "битих" пікселів (тобто пікселів, які є імпульсним шумом) така околиця зазвичай складається з одного пікселя (його самого). Якщо битих пікселів на зображенні багато, можливо сусідство таких пікселів, і тоді околиця буде трохи більше. Таким чином, оцінивши розмір околиці, ми можемо ідентифікувати биті пікселі та згладити їх.

3.4Адаптивність алгоритму

Так як алгоритм повинен бути застосований для будь-якого виду шуму, було б дивно наполегливо видаляти Гауссов шум із зображення, спотвореного лише

імпульсним шумом. Тому алгоритм вбудовані елементи адаптивності високого рівня, які прискорюють його роботу. При початковому обчисленні дисперсії шуму наявність битих пікселів вплине на цю величину. І навіть якщо Гауссова шуму на зображенні не буде, дисперсія буде відмінна від нуля. При видаленні Гауссова шуму підраховується дисперсія віддаленого шуму, якщо вона виявляється значно меншою за обчислену раніше (а так виявиться, якщо Гауссова шуму насправді немає, тому що імпульсний шум первинна обробка не видаляє), то дисперсія коригується, і алгоритм вже працює з більш адекватними умовами.

3.5 Реалізація

Для написання бібліотеки було обрано Matlab. По-перше, у Matlab є вбудовані аналогічні бібліотеки та написаний алгоритм легко порівняти із вбудованим за швидкістю та якістю. По-друге, з Matlab код можна транслювати C, C++ і C#.

На вхід до програми надходить зображення у форматі PNG. Цей формат дозволяє зберігати графічну інформацію за допомогою стиснення без втрат. Вбудованими функціями зображення перетворюється на двовимірний масив чисел від 0 до 1. Далі, залежно від налаштованих параметрів, додається той чи інший вид шуму.

Задається масив H, у якому зберігаються можливі значення розміру вікна. Задається вид вікна: симетричне, що складається з 4 різних квадрантів або 8 секторів.

Спочатку над отриманими даними виконується процедура LPA. Вона складається з кількох частин:

– Знаходження дисперсії шуму

function_std_dev

– Знаходження матриці ступенів для ядер

function_LPA_Kernel_Matrix

– Створення ядер

function_Create_LPA_Kernels

Після створення ядер функція function_LPA_Estimate кожної точки будує ряд оцінок. Кожна оцінка відповідає елементу з масиву H. Серед цих оцінок потрібно вибрати оптимальну у кожному напрямку (їх може бути 1, 4 або 8 залежно від заданого виду вікна). Це реалізує функція function_ICI, яка будує інтервали довіри та знаходить оптимальний розмір вікна. Паралельно функція function_adaptive_dev перераховує дисперсію. Після того, як розміри вікна по всіх напрямках знайдено, можна оцінити розмір околиці, ідентифікувати биті пікселі та застосувати до них медіанний фільтр.

Після фільтрації можна підрахувати СКО відфільтрованого зображення та оригінального. Також зображення виводиться на екран для суб'єктивної оцінки користувачем.

Проведено порівняння якості роботи вбудованих у Matlab функцій обробки зображень та даної бібліотеки. В якості тестового зображення було взято чорно-біле зображення Cameraman 256x256. Воно було спотворене трьома способами:

- Гаусовим шумом із дисперсією 0.01
- Імпульсним шумом із щільністю 0.05
- Сумою цих двох шумів

У таблиці 3.1 представлені результати порівняння запропонованого алгоритму та стандартних.

Таблиця 3.1 - Значення СКО

	wiener2	medfilt2	new LPA+ICI
Гаусов	0,0024	0,0038	0,0016
Імпульсний	0,0087	0,0028	0,0022
Комбінований	0,0094	0,0043	0,0032

Як видно з таблиці, запропонований алгоритм дає більш високу якість фільтрації у всіх трьох випадках.

Нижче наведено результати роботи програми для трьох видів шумів (рис. 3.2).

Використання фільтра Гауса:

```
% Завантаження зображення
img = imread('your_image.jpg');
% Додавання Гаусового шуму
noisy_img = imnoise(img, 'gaussian', 0, 0.01);
% Видалення шуму за допомогою фільтра Гауса
filtered_img = imgaussfilt(noisy_img, 1); % Змініть параметр sigma за потребою
% Відображення результату
figure;
subplot(1, 2, 1), imshow(noisy_img), title('Зашумлене зображення');
subplot(1, 2, 2), imshow(filtered_img), title('Видалено Гаусовий шум');
```

Використання фільтра медіани:

```
% Завантаження зображення
img = imread('your_image.jpg');
% Додавання Гаусового шуму
noisy_img = imnoise(img, 'gaussian', 0, 0.01);
% Видалення шуму за допомогою фільтра медіани
filtered_img = medfilt2(noisy_img, [3, 3]); % Змініть розмір фільтра за потребою
% Відображення результату
figure;
subplot(1, 2, 1), imshow(noisy_img), title('Зашумлене зображення');
subplot(1, 2, 2), imshow(filtered_img), title('Видалено Гаусовий шум');
```

Рисунок 3.2 - Результати видалення Гаусового шуму.

```
% Завантаження зображення
img = imread('your_image.jpg');
% Додавання комбінованого шуму (Гаусовий шум і сіль-перець)
noisy_img = imnoise(img, 'gaussian', 0, 0.01);
noisy_img = imnoise(noisy_img, 'salt & pepper', 0.02);
% Видалення Гаусового шуму за допомогою фільтра Гауса
gaussian_filtered_img = imgaussfilt(noisy_img, 1); % Змініть параметр sigma за потребою
% Видалення сілі і перцю за допомогою медіанного фільтра
median_filtered_img = medfilt2(gaussian_filtered_img, [3, 3]); % Змініть розмір ядра за потребою
% Відображення результатів
```

```
figure;
subplot(1, 3, 1), imshow(noisy_img), title('Зашумлене зображення');
subplot(1, 3, 2), imshow(gaussian_filtered_img), title('Видалено Гаусовий шум');
subplot(1, 3, 3), imshow(median_filtered_img), title();
```

Рисунок 3.3 - Результати видалення комбінованого шуму

Медіанний фільтр ефективно видаляє імпульсний шум, який представляє собою випадкові високі або низькі значення пікселів.

Використовуйте функцію medfilt2 для застосування медіанного фільтра на зображенні.

```
% Завантаження зображення
img = imread('your_image.jpg');
% Додавання імпульсного шуму (сіль і перець)
```

```
noisy_img = imnoise(img, 'salt & pepper', 0.02);
% Видалення імпульсного шуму за допомогою медіанного фільтра
denoised_img = medfilt2(noisy_img, [3, 3]); % Змініть розмір ядра за потребою
% Відображення результатів
figure;
subplot(1, 2, 1), imshow(noisy_img), title('Зашумлене зображення');
subplot(1, 2, 2), imshow(denoised_img), title('Видалено імпульсний шум');
```

Адаптивний медіанний фільтр:

Використовуйте адаптивний медіанний фільтр (adfilt2), який автоматично адаптується до розміру шуму у різних частинах зображення (рис.3.4).

```
% Завантаження зображення
img = imread('your_image.jpg');
% Додавання імпульсного шуму (сіль і перець)
noisy_img = imnoise(img, 'salt & pepper', 0.02);
% Видалення імпульсного шуму за допомогою адаптивного медіанного фільтра
denoised_img = adfilt2(noisy_img, [3, 3]); % Змініть розмір ядра за потребою
% Відображення результатів
figure;
subplot(1, 2, 1), imshow(noisy_img), title('Зашумлене зображення');
subplot(1, 2, 2), imshow(denoised_img), title('Видалено імпульсний шум');
```

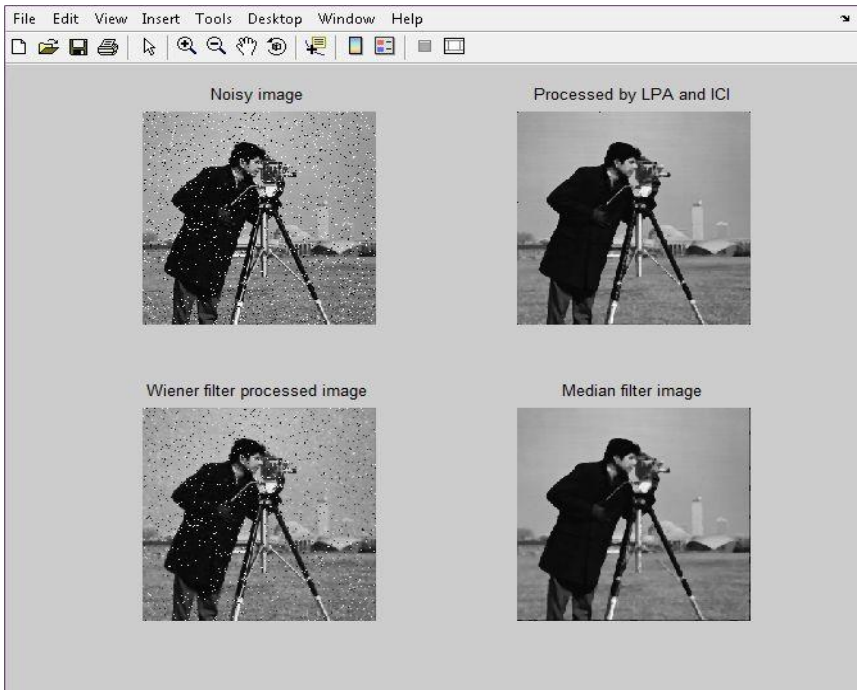


Рисунок 3.4 - Результати видалення імпульсного шуму.

3.6 Реалізація фільтра Лапласа

Розглянемо покращення зображення за допомогою фільтра Лапласа. На наступному рисунку, (рис. 3.5) зліва дано кілька розпливчастий знімок Місяця.

Поліпшення в цьому випадку полягає у підкресленні перепадів рівнів яскравості на зображенні при збереженні, наскільки це можливо, областей постійної тональності. Спочатку ми генеруємо та відображаємо фільтр Лапласа:

w = fspecial('laplacian',0)

w =

0	1	0
1	-4	1
0	1	0-

Зауважимо, що фільтр належить класу duoble, параметр форми alpha = 0 призводить до описаного вище фільтру. Таку форму можна легко задати вручну командою w = [010; 1 -4 1; 0 1 0];

Тепер застосуємофільтр w до зображення f, яке має клас *uint8:*

f=imread('Moon2.tif');imshow(f);% малюнок зліва

g1 = imfilter(f,w,'replicate');imshow(g1,[]); % Мал. Справа

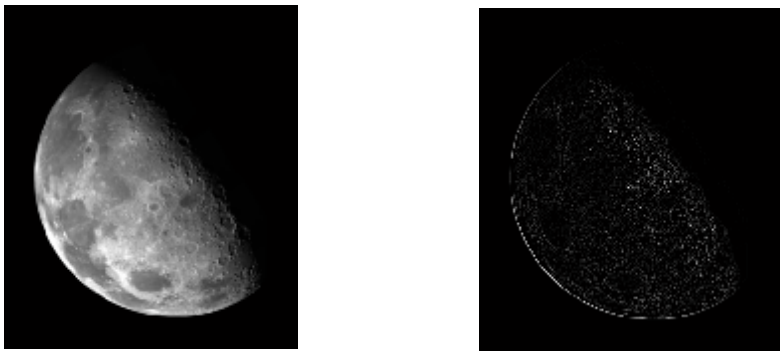


Рисунок 3.5 – Зображення у процессі обробки фільтром Лапласа

Тут наведено результат цих команд. Він є правдоподібним, але є одна проблема. Через негативність центрального коефіцієнта фільтра очікується появи фільтрованого зображення з негативними значеннями пікселів. Однак у нашому випадку вихідне зображення було класу uint8, а, як говорилося раніше, фільтрація функцією *imfilter* створює на виході зображення того ж класу, що було на вході, тому від'ємні величини будуть обрізані. Щоб обійти цю проблему, слід перетворити зображення f на клас double перед фільтрацією: f2=im2double(f);g2=imfilter(f2,w,'replicate');imshow(g2,[ ]);

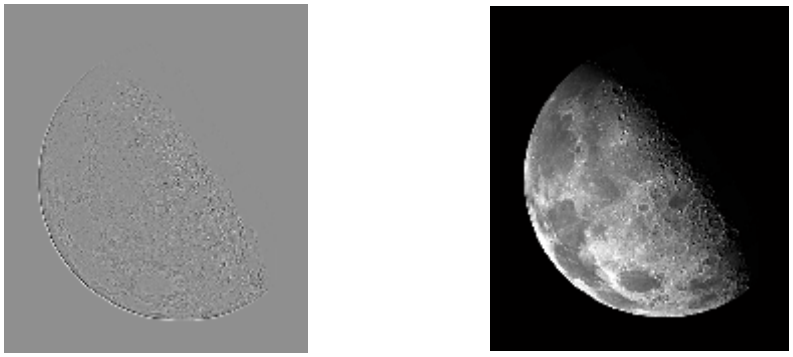


Рисунок 3.6 – Зображення у процессі обробки фільтром Лапласа

Результат, показаний на попередньому малюнку зліва, є правильно обробленим зображенням під час фільтрації за допомогою Лапласіана.

Нарешті, для відновлення тонів областей, втрачених при фільтрації лапласіаном, слід відняти (нагадаємо, що центральний коефіцієнт фільтра від'ємний) відфільтроване зображення з вихідного:

g = f2 – g2; imshow(g);

Остаточний результат показано на попередньому малюнку праворуч. Видно, наскільки покращене зображення є більш різким у порівнянні з вихідним

Завдання покращення зображень часто потребує вибору параметрів фільтрів з наявного набору. Лапласіан є добрим прикладом. У пакеті є фільтр-лапласіан 3 х 3 із числом – 4 у центрі. Як правило, ще більшу різкість зображення можна отримати з лапласіаном, в центрі якого стоїть число - 8, оточене з усіх боків 1.

Метою наступного прикладу є ручна реалізація цього фільтра для порівняння. Виконаємо наступний ланцюжок команд:

```
f=imread('Moon2.tif');imshow(f);% вихідний знімок зліва
w4 = fspecial( 'laplacian', 0);
   fd=im2double(f); gf4=imfilter(fd,w4,'replicate');
   g4 = fd - gf4; figure,imshow(g4);%  знімок у центрі
   w8=[1 1 1; 1 -8 1; 1 1 1];
   g8 = fd - imfilter(fd, w8, 'replicate');
   figure,imshow(g8);% знімок праворуч
```

Рисунок 3.7 – Зображення у процессі обробки медіанним фільтром

На малюнку справа показано результуюче дуже різке зображення.

Нелінійна просторова фільтрація також заснована на околицьких операціях, причому механізм визначення околиці розміру mxn і ковзання її центру зображення є таким же, як і в лінійній фільтрації. Однак там, де лінійна фільтрація використовує суму творів (тобто лінійну операцію), нелінійна фільтрація заснована на нелінійних операціях, що здійснюються над пікселями поточної околиці. Наприклад, якщо покласти, що відгук фільтра у кожній центральній точці дорівнює максимальному значенню її околиці, це визначить нелінійну операцію фільтрації. Інша фундаментальна відмінність у тому, що концепція маски не превалює у нелінійних процесах. Ідея фільтрації залишається, але сам «фільтр» слід уявляти у вигляді нелінійної функції, яка застосовується до пікселів околиці.

Найбільш уживані нелінійні фільтри породжуються в пакеті IPT функцією ordfilt2, яка будує фільтри порядкових статистик (їх також називають рангові фільтри).

Відгуки цих нелінійних просторових фільтрів ґрунтуються на попередньому впорядкуванні (ранжуванні) пікселів зображення з поточної околиці, після чого центральному пікселю надається значення, визначене в результаті цього впорядкування. Синтаксис функції ordfilt2 має такий вигляд:

g = ordfilt2(f, order, domain);

Ця функція створює вихідне зображенняg, замінюючи кожен елемент f на елемент з номером order у впорядкованій послідовності ненульових елементів з околиці, заданої параметром domain. Тут domain - це матриця mxn, що складається з нулів та одиниць, які позначають позиції пікселів, що беруть участь у обчисленнях. У цьому сенсі матриця domain діє як маска. Пікселі околиці, яким відповідають нулі в domain, не беруть участь у обчисленнях. Наприклад, щоб реалізувати фільтр мінімуму розміру mxn, застосовується команда

g = ordfilt2(f, 1, ones(m, n));

У такому записі order=1 означає перший елемент упорядкованої послідовності з mn пікселів, а ones(m,n) — це матриця mxn, з одних одиниць,

що вказує на те, що всі елементи околиці беруть участь у обчисленні відгуку фільтра. Аналогічно, фільтр максимуму реалізується командою

*g = ordfilt2(f, m*n, ones(m, n));*

Найвідомішим фільтром порядкових статистик цифрової обробки зображень є медіанний фільтр, який відповідає середньому номеру елементів маски. Для цього використовують функцію median в ordfilt2 наступним чином

g = ordfiIt2(f, median(l:m*n), ones(m, n));

Тут median(l:m*n) – це медіана впорядкованої послідовності чисел 1,2,..., m·n. Функція median має наступний загальний синтаксис:

v = median(A, dim),

де v це вектор, елементи якого утворюють медіану A вздовж розмірності dim. Наприклад, якщо dim = 1, кожен елемент v — це вектор – рядок середніх значень елементів уздовж відповідних стовпців матриці A.

В силу практичної важливості медіанного фільтра в пакеті IPT передбачена спеціальна реалізація цього фільтра:

g = medfilt2(f, [m, n], padopt),

де пара [m,n] задає околицю розмірymxn, за якою обчислюється медіана. Параметр padopt визначає три можливі опції розширення меж зображення.

Вона може набувати значення за промовчанням 'zeros' з нульовим розширенням.

Друге значення 'symmetric' розширює зображення f шляхом його дзеркального відображення через кордони. Третє значення 'indexed', у якому f розширюється значенням 1, якщо f має клас double і значенням 0 інакше.

У найпростішому вигляді ця функція викликається командою

g = medfilt2(f);

При цьому використовується околиця 3x3 для обчислення медіани і вхідне зображення розширюється нульовими значеннями пікселів.

Медіанна фільтрація дуже ефективна при видаленні з зображень імпульсних шумів. Боротьбу з різними шумами ми обговорюватимемо пізніше, проте тут доречно проілюструвати ці дії за допомогою медіанного фільтра.

На наступному малюнку зліва представлений рентгенівський знімок (рис. 3.8) друкованої плати, отриманий на стадії автоматичного контролю продукції виробництва, який сильно зіпсований шумом. Виконаємо наступні команди

```
f=imread('ChkBoardNoise.tif');imshow(f);% фото зліва gm =
medfilt2(f);imshow(gm);% фото в центрі
gms = medfilt2(f, 'symmetric'); imshow(gms); % фото праворуч
```

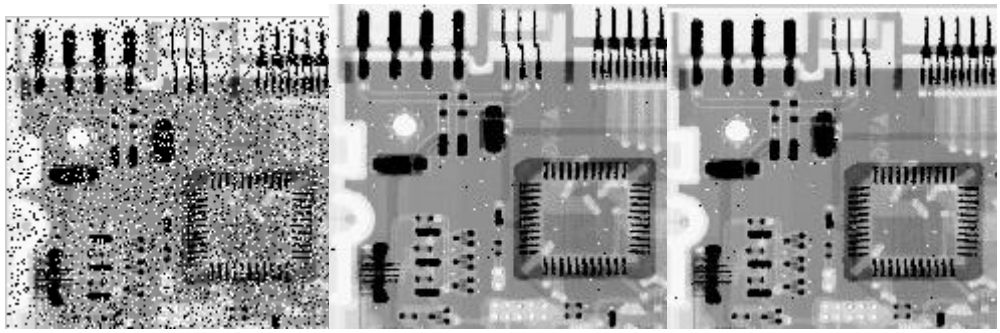


Рисунок 3.8 – Зображення у процессі обробки медіанним фільтром

Праве зображення схоже на середнє, але ефект почорніння кордонів (Чорні точки на кордонах) на ньому не проявляється.

Іншим важливим застосуванням фільтрів є розфокусування зображення, що дозволяє створити грубий образ об'єктів, які можуть становити інтерес. При цьому дрібні об'єкти поєднуються з фоном, у той час як великі об'єкти залишаються у вигляді плям і можуть бути легко виявлені. Розміри об'єктів, які будуть змішуватися з фоном, приблизно збігаються з розмірами фільтра, що згладжує.

ВИСНОВОК

Під час написання кваліфікаційної роботи було досліджено різні підходи до видалення трьох основних видів шуму: Гаусов шум, імпульсний шум, комбінований шум. Було розроблено адаптивний метод видалення цих шумів. На його основі було розроблено алгоритм, який був реалізований у середовищі Matlab.

Було створено систему, що видаляє перелічені шуми ефективніше існуючих систем. Крім того, ця система універсальна і сама підлаштовується за наявний шум.

Було проведено порівняльний аналіз роботи стандартних систем та нової системи.

Було встановлено, що у всіх випадках нова система працює ефективніше за стандартні. На це вказують як об'єктивні дані, такі як підрахунок СКО, так і суб'єктивні, такі як візуальна оцінка результатів роботи користувачем.

Надалі планується адаптувати ці підходи для обробки не тільки чорно-білих, а й кольорових зображень. Планується створення гібридних алгоритмів, що поєднують ефективні складові нових та стандартних алгоритмів

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. V. M. Kartashov, V. N. Oleynikov, S. A. Sheyko, I. V. Koryttsev, S. I. Babkin, O. V. Zubkov, "Peculiarities of small unmanned aerial vehicles detection and recognition," Telecommunications and Radio Engineering, 2019, V. 78, Iss. 9, pp. 771–781. DOI: 10.1615/TelecomRadEng.v78.i9.30.

2. Андреев, Г.А. Активные системы ориентации по геофизическим полям / Г.А. Андреев, А.А. Потапов // Зарубежная радиоэлектроника. – 1988. – №9. – С. 62-85.

3. Kartashov, V.M., Sidorov, G.I., Sheiko, S.A., Kolendovska, M.M., Sergienko O.Yu. Principles of Construction and Assessment of technical Characteristics of multi-Frequency atmospheric Sodar in the Humidity Measurement Mode / Telecommunications and Radio Engineering.- New York. - 2020.- Vol. 79, №4.- P.323-333. (стаття). DOI: 10.1615/TelecomRadEng.v79.i4.50.

4. Kartashov, V.M., Oleynikov V.N, Zubkov, O.V., Korytsev I.V., Babkin, S. I., Sheiko, S.A.,. Kolendovskaya, M.M. Spatial-temporal Processing of acoustic Signals of Unmanned Aerial Vehicles; Telecommunications and Radio Engineering, 2020. Vol. 79, Iss, 9, pp.769-780.

5. V.M. Semenets, V.M. Kartashov, V.I. Leonidov. Features of Acoustic Noise of Small Unmanned Aerial Vehicles // Telecommunications and Radio Engineering.- New York. - 2020.- Vol. 79, №11.- P. 985-995. DOI: 10.1615/TelecomRadEng.v79.i11.80 (стаття).

6. Oleynikov V.N., Kartashov, V.M., Babkin, S. I., Zubkov, O.V., Korytsev I.V., Sheiko, S.A.,. Seleznov I.S. Structure and Parameter Unmanned Aerial Vehicles Sound Fields/ Telecommunications and Radio Engineering.- New York. - 2020.- Vol. 79, №17.- P.1539-1550. DOI: 10.1615/TelecomRadEng.v79.i17.50 (стаття).

7. В.А. Тихонов, В.М. Карташов, В.М. Олейніков, В.И. Леонидов, Л.П. Тимошенко, И.С. Селезнеов, Н.В. Рыбников. Обнаружение-распознавание беспилотных летательных аппаратов с использованием составной модели

авторегрессии их акустического излучения// Вісник НТУУ «КПІ». Радіотехніка. Радіоапаратобудування. - 2020.- Вип. №81. – С.38-46. (Web of Science)

8. Карташов В.М., Корытцев И.В., Олейников В.Н., Зубков О.В., Шейко С.А., Бабкин С.И., Левский Н.А., Селезнев И.С. Алгоритмы пеленгации беспилотных летательных аппаратов по их акустическому излучению// Радиотехника. (Харьков). — 2019. — Вып. 196. — С. 22-31.

9. Карташов В.М., Харченко О.И., Чумаков В.И. Использование эффекта стохастического резонанса для анализа спектров акустического излучения малых беспилотных летательных аппаратов // Радиотехника. (Харьков). — 2019. — Вып. 197. — С. 100-106.

а. Потапов, А.А. Новейшие методы обработки изображений / А.А. Потапов, Ю.В. Гуляев, С.А. Никитов, А.А. Пахомов, В.А. Герман ; под ред. А.А. Потапова. – М.: Физматлит, 2008.

10. Карташов В.М., Посошенко В.А., Цехмистро Р.И., Тимошенко Л.П., Колендовская М.М. Методы ориентации, навигации и контроля мобильных робототехнических платформ// Радиотехника. (Харьков). — 2019. — Вып. 199. – С. 38-44.

11. Потапов, А.А. Разработка и структура ансамбля фрактальных

12. признаков классов целей для задач линейной и нелинейной радиолокации // Нелинейный мир. – 2006. – Т.4. – №7-9. – С. 361-386.

13. Карташов В.М., Олейников В.Н., Колендовская М.М., Тимошенко Л.П., Капуста А.И., Рыбников Н.В. Комплексирование изображений при обнаружении беспилотных летательных аппаратов // Радиотехника. (Харьков). — 2020. — Вып. 201. — С. -120-129.

14. Карташов В.М., Олейников В.Н., Воронин В.В., Рябуха В.П., Капуста А.И., Рыбников Н.В., Селезнев И.С. Методы комплексной обработки и интерпретации радиолокационных, акустических, оптических и инфракрасных сигналов беспилотных летательных аппаратов // Радиотехника. (Харьков). — 2020. — Вып. 202. — С. 173-182-. DOI:10.30837/rt.2020.3.202.19

15. Карташов В.М., Корытцев И.В., Олейников В.Н., Зубков О.В., Шейко С.А., Бабкин С.И., Обработка сигналов при пеленгации и определении дальности до малоразмерных БПЛА в оптическом и инфракрасном диапазонАХ // Радиотехника. (Харьков). — 2020. — Вып. 202. — С. 125-135. DOI:10.30837/rt.2020.3.202.13

16. Карташов В.М., Корытцев И.В., Олейников В.Н., Зубков О.В., Шейко С.А., Бабкин С.И. ОПТИКО-ЭЛЕКТРОННЫЕ МЕТОДЫ ОБНАРУЖЕНИЯ ВОЗДУШНЫХ ОБЪЕКТОВ И ИЗМЕРЕНИЯ ИХ КООРДИНАТ// Радиотехника. (Харьков). — 2020. — Вып. 202. — С. 153-59. DOI:10.30837/rt.2020.3.202.16

17. Карташов В.М., Корытцев И.В., Олейников В.Н., Зубков О.В., Шейко С.А., Бабкин С.И.. Эффективность детектирования и распознавания изображений дронов по видеопотоку стационарной видеокамеры // Радиотехника. (Харьков). — 2020. — Вып. 202. — С. 136-146. DOI:10.30837/rt.2020.3.202.14

18. Рябуха В.П., Карташов В.М. Методы обнаружения-распознавания радиолокационных, акустических, оптических и инфракрасных сигналов беспилотных летательных аппаратов / В.П. Рябуха, В.М. Карташов// Известия высших учебных заведений. Радиоэлектроника. — 2020. — Т. 63, № 11. — С. 1–35.

19. В.М. Карташов, В.Н. Олейников, В.И. Леонидов, канд. Техн. Наук, В.В. Воронин, А.И. Капуста, И.С. Селезнев, Е.В. Першин/ Комплексная обработка сигналов интегрированной системы наблюдения беспилотных летательных аппаратов с использованием целеуказания//Радиотехника. (Харьков). - 2020. - Вып. 203. - С. 1-13.