МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Київський національний університет будівництва і архітектури

# ЦИФРОВА ОБРОБКА ЗОБРАЖЕНЬ

**Методичні вказівки**
**до виконання лабораторних робіт**
для студентів спеціальності
123 «Комп’ютерна інженерія»

**Київ 2022**

УДК 004.051
М54

Укладачі: А.М. Котенко, канд. техн. наук, доцент;
Ю.І. Хлапонін, д-р техн. наук, професор

Рецензент докт. техн. наук, професор Терентьєв О.О.

Відповідальний за випуск Ю.І. Хлапонін д-р техн. наук, професор

*Затверджено на засіданні кафедри кібербезпеки та комп'ютерної інженерії, протокол № 2 від 22 вересня 2022 р.*

В авторській редакції.

М54 **Методичні вказівки до виконання лабораторних робіт** для студентів спеціальності 123 «Комп'ютерна інженерія» з дисципліни «Цифрова обробка зображень»/ Уклад. А.М. Котенко, Ю.І. Хлапонін – Київ: КНУБА, 2022. – 44 с.

Містять зміст, порядок оформлення і вказівки до виконання лабораторних робіт.

Призначено для студентів спеціальності 123 «Комп'ютерна інженерія» галузі знань 12 «Інформаційні технології».

**ЗМІСТ**

# 1. ЗАГАЛЬНІ ПОЛОЖЕННЯ

1. **Лабораторні заняття** з дисципліні «Цифрова обробка зображень» у студентів спеціальності 123 «Комп'ютерна інженерія», займають 20 годин і охоплюють розділи курсу, що пов'язані з методами цифров ої обробки зображень.

2. **Мета лабораторних** занять полягає у ознайомленні студентів з існуючими методами, алгоритмами обробки та перетворень зображень та практичними навиками реалізації цих методів у програмному середовищі MATLAB.

3. Лабораторні роботи виконуються на лабораторних заняттях з дисципліни. Підготовка до лабораторних занять здійснюється студентами в часи самостійної роботи. Перелік і кількість задач для розв'язання визначається викладачем, який веде лабораторні заняття, відповідно до робочої програми дисципліни. Після виконання кожної роботи студенти складають звіт, який захищають. За результатами виконання та захисту роботи виставляються бали за спеціальною шкалою оцінювання, наведеною у робочій програмі. Бали, отримані за окремі роботи, формують загальну суму балів за дисципліну, яка враховується у підсумкову оцінку за модуль та семестр.

Практична реалізація методів цифрової обробки зображень здійснюється у програмному середовищі MATLAB.

Цифрова обробка зображень вивчає фізичні засади, методичні та алгоритмічні основи реєстрації, трансформації, обробки, перетворення та візуалізації цифрових зображень для різних областей застосувань.

Результатом вивчення навчальної дисципліни є набуття студентом наступних компетентностей пов'язаних із здатністю:

– вільного володіння професійними знаннями для аналізу і синтезу фізичної інформації (відповідно до профілю підготовки).

– самостійного здобування за допомогою інформаційних технологій і практичної діяльності нових знань та вмінь, розширювання і поглиблювання свого наукового світогляду;

– демонстрації поглиблених знань з математичних і природничих наук;

– використання вільного володіння професійно-профільованими знаннями в галузі інформаційних технологій, сучасних комп'ютерних мереж, програмних продуктів і ресурсів Інтернет для вирішення завдань професійної діяльності, у тому числі, що знаходяться за межами профільної підготовки;

## 2. ЗАВДАННЯ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ

### Лабораторна робота №1. Дискретне (цифрове) представлення відеоінформації.

**Мета:** навчитися розраховувати інформаційний об’єм графічних файлів.

ТЕОРЕТИЧНІ ВІДОМОСТІ

Інформація - це послідовність символів, знаків.

Кількість символів в повідомленні називається довжиною повідомлення.

Основою будь-якої мови є алфавіт.

**Алфавіт** - це набір знаків (символів), в якому визначено їх порядок.

**Потужність алфавіту** - це повне число символів алфавіту. Позначимо цю величину буквою М.

Наприклад, потужність алфавіту з російських букв дорівнює 33, потужність алфавіту з англійських букв дорівнює 26.

При алфавітному підході до вимірювання інформації кількість інформації від змісту не залежить. Кількість інформації залежить від обсягу тексту (тобто від числа знаків в тексті) і від потужності алфавіту. Тоді інформацію можна обробляти, передавати, зберігати. Кожен символ несе *X* біт інформації.

Кількість інформації *X*, яке несе один символ в тексті, залежить від потужності алфавіту М, які пов'язані формулою $M = 2^x$.

Кількість інформації в тексті, що складається з К символів, так само К * Х.

У 2-символьному алфавіті кожен символ несе 1 біт інформації ($2^x = 2$, звідки х = 1 біт).

Якщо подивитися на екран працюючого монітору можна побачити безліч дрібних точок - пікселі. Кожен відеопіксель на кольоровому екрані складається з трьох точок (зерен) базових кольорів: червоного, зеленого і синього. Таким чином, сусідні різнокольорові точки зливаються, формуючи інші кольори. Інформація про стан кожного пікселя зберігається в закодованому вигляді в пам'яті ПК. З основної формули інформатики можна підрахувати об’єм пам'яті, необхідний для зберігання одного пікселя:

$$N = 2^i$$

де:

і - глибина кодування (кількість біт, які займає 1 піксель);

N – кількість кольорів (палитра).

Для отримання чорно-білого зображення один піксель може знаходитись у одному зі станів: світиться – ***білий (1)***, не світиться – ***чорний (0)***.

$$2 = 2^i, i = 1$$

Тобто, для його зберігання необхідно **1 біт.**

## Порядок виконання лабораторної роботи

**Завдання_1,** Розрахувати кількість інформації яку несе кожен символів якщо $M = 32$; $M = 100$.

**Завдання_2.** Країна "А" використовує 32-символьний алфавіт. Країна "Б" - 64-символьним алфавіт. Повідомлення країни "А" містить 90 символів, а повідомлення "Б" – 80 символов. В якому повідомлені більше інформації?

**Завдання 3.** Обчислення об'єму растрового зображення (табл. 1.1).

***(№ варіанту = № по списку)***

Таблиця 1.1

Варіанти до завдання 3

| № за списком | Розмір зображення (пікселі) | № за списком | Розмір зображення (пікселі) | № за списком | Розмір зображення (пікселі) | № за списком | Розмір зображення (пікселі) |
|---|---|---|---|---|---|---|---|
| 1 | 200*200 | 9 | 300*320 | 17 | 400*420 | 25 | 450*200 |
| 2 | 200*300 | 10 | 300*330 | 18 | 400*440 | 26 | 450*220 |
| 3 | 200*250 | 11 | 300*350 | 19 | 400*200 | 27 | 450*250 |
| 4 | 250*250 | 12 | 350*350 | 20 | 400*250 | 28 | 450*300 |
| 5 | 250*200 | 13 | 350*370 | 21 | 400*300 | 29 | 450*350 |
| 6 | 250*300 | 14 | 350*390 | 22 | 400*350 | 30 | 450*400 |
| 7 | 300*300 | 15 | 400*400 | 23 | 450*100 | 31 | 450*450 |
| 8 | 300*310 | 16 | 400*410 | 24 | 450*150 | 32 | 500*500 |

**Завдання 3.1.**

Розрахувати об'єм растрового чорно-білого зображення. Розмір зображення згідно з вашим варіантом по списку групи.

**Завдання 3.2.**

Розрахувати об'єм растрового зображення палітрою у 256 кольорів. Розмір зображення згідно з вашим варіантом.

**Завдання 3.3.**

Розрахувати об’єм растрового зображення палітрою у 16 кольорів. Розмір зображення згідно з вашим варіантом.

## Контрольні питання

1. Що називається потужністю алфавіту?
2. Як розрахувати кількість інформації яке несе один символ в тексті?
3. Від чого залежить кількість інформації у тексті?

## Лабораторна робота №2. Основи роботи з зображеннями в MATLAB

**Мета:** ознайомитись з принципами обробки зображень, які використовуються у програмному середовищі MATLAB; вивчити команди MATLAB для зчитування, виведення та запису зображень.

### Порядок виконання лабораторної роботи

**Завантаження зображення.** Щоб завантажити зображення у робочу область MATLAB застосовується функція imread:

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

де:

filename – назва файлу, який завантажуємо, з його розширенням. Наприклад, команда:

```
>> c=imread (apple.jpg');
```

присвоює змінній c, зображення з ім'ям apple формату jpg.

filename буде шукатися у поточній папці. Для пошуку в інших місцях потрібно вказувати повний шлях до 'filename' зображення. Це найпростіший спосіб зчитати зображення з конкретної папки.

Приклад:

```
>> c=imread ('D:\Library\apple.jpg');
```

ця команда зчитає зображення з папки `Library` на диску `D:`, а команда

```
>> c=imread ('.\Library \apple.jpg');
```

завантажить зображення з підпапки `Library` поточної робочої папки. Поточна робоча папка MATLAB відображається в рядку інструментів робочого стола. Там її легко змінити вручну.

Команда `size(c)` покаже розмір зображення - число рядків і стовпчиків:

```
>> size(c)
   ans=
 2048 2048
```

Ця функція дозволяє автоматично визначити розмір зображення, що виконується операцією

```
>> [M,N]=size(c)
```

Змінній `M` присвоїться число рядків зображення, а змінній `N` – число стовпчиків.

Функція `whos` повідомляє додаткову інформацію про масив. Наприклад, рядок `>> whos c`

дає наступний
результат

```
Name            Size            Bytes          Class
c               2048×2048       4194304        uint8 array
Grand total is 4194304 elements using 4194304 bytes
```

**Виведення зображення на екран монітору.** Зображення на екран виводиться функцією `imshow`, наступним чином:

```
imshow(c,G)
```

де:

`c` – матриця завантаженого зображення, а `G` – кількість рівнів яскравості, які застосовується при відображенні цього зображення. Якщо величина `G` не використовується, то по замовчуванню застосовується 256 рівнів яскравості. Наприклад команда

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

означає, що всі пікселі зі значенням не більше числа `low` треба показувати чорними, а всі пікселі зі значенням не менше числа `high` – білими. Усі проміжні значення показуються з проміжною яскравістю з використанням кількості рівнів, які прийнято за замовчуванням.

Наступна команда:

```
imshow(c,[ ])
```

визначає для змінної `low` мінімальне значення масиву `c`, а змінній `high` його максимальне значення. Такий запис команди `imshow` може бути корисним якщо необхідно показати зображення у якого вузький динамічний діапазон значень пікселів.

Функція `pixval` може використовуватися при інтерактивному визначенні значень яскравості окремих пікселів. Ця функція відображає курсор розміщений поверх зображення. Курсор переміщується по зображенню разом з курсором мишки, і під вікном зображення показуються поточні координати курсору та значення інтенсивності в цій точці. У випадку роботи з кольоровим зображенням, разом з координатами будуть відображатися інтенсивність (яскравість) червоної, зеленої та синьої складової кольорового пікселя. Синтаксис цієї команди наступний:

```
pixval,
```

при цьому курсор встановлюється на останнє виведене на дисплеї зображення. Натисканням кнопки X у вікні курсора відключимо курсор на зображенні.

**Збереження зображень.** Зображення зберігаються на диск функцією `imwrite`, наступним чином:

```
imwrite(c,'filename')
```

При такому запису, `filename` повинен містити розширення, яке підтримується системою MATLAB. Або потрібний формат можна задати за допомогою самому. Наприклад, наступна команда, зберігає матрицю `c` в зображення з форматом TIFF з ім'ям `apple_1` .

```
imwrite(c,'apple_1','tif')
```

або

```
imwrite(c,'apple_1.tif')
```

Якщо `filename` не містить інформації про шлях, то функція `imwrite` запише файл в поточну робочу папку.

Більш загальний синтаксис функції `imwrite`, при застосуванні тільки для файлів JPEG, має вигляд:

```
imwrite(c,'filename.jpg', 'quality', q)
```

де:

`q` – ціле число від 0 до 100 (чим воно менше, ти буде більшим спотворення при стисненні файлу у форматі JPEG).

**Класи даних.** Хоча при обробці зображень робота проводиться з цілочисленними координатами, значення самих пікселів можуть не бути цілими числами. В табл. 1.1 перераховані різні класи даних, які підтримує система MATLAB. Перші вісім записів в таблиці належать до числових класів даних. Дев'ятий запис – це клас символів, а останній запис належить до логічного класу даних.

Всі числові операції в MATLAB здійснюються з подвійною точністю у класі даних *double*, який часто використовується в додатках для обробки зображень. Клас *uint8* також зустрічається дуже часто. Особливо часто використовується при зчитуванні даних із запам'ятовуючих пристроїв, наприклад, найбільш поширених на практиці 8-бітних зображень. Клас *logical* і в меншій мірі клас *uint16*, утворюють першостепеневі класи даних. Клас даних *double* потребує 8 байт для представлення одного числа, клас *uint8* використовує один байт на кожний елемент. Типам *uint16* та *int16* потрібно по 2 байти, а типи *uint32*, *int32* та *single* займають в пам'яті по 4 байти на кожен елемент. Клас даних *char* є символьним та зберігає букви та символи в кодуванні *Unicode.* В ньому використовується два байти на елемент. Символьний рядок представляє собою масив 1×n символів класу *char*. Масив *logical* складається з елементів, що приймають тільки два значення 0 та 1, які зберігаються в пам'яті, займаючи по одному байту кожне.

Таблиця 2.1.

Класи даних. Перші вісім класів називаються числовими, дев'ятий символьний клас, останній клас – логічний

| Ім'я | Опис |
|---|---|
| double | Числа з плаваючою комою подвійної точності в діапазоні, приблизно, від -10308 до 10308 (8 байт на число). |
| uint8 | Цілі без знаку в інтервалі [0, 255] (1 байт на число) |
| uint16 | Цілі без знаку в інтервалі [0, 65535] (2 байти на число) |
| uint32 | Цілі без знаку в інтервалі [0, 4294967295] (4 байти на число) |
| int8 | Цілі зі знаком в інтервалі [-128, 127] (1 байт на число) |
| int16 | Цілі зі знаком в інтервалі [-32768, 32767] (2 байти на число) |
| int32 | Цілі зі знаком в інтервалі [-2147483648, 2147483647] (4 байти на число) |
| single | Числа з плаваючою комою, звичайної точності в діапазоні, приблизно, від -1038 до 1038 (4 байти на число). |
| char | Символи (букви та знаки) (2 байти на число) |
| logical | Значення 0 чи 1 (1 байт на число) |

**Хід виконання роботи.**

Завантажити з інтернету яку-небудь кольорову картинку-зображення у форматі JPG.

1. Зчитати це своє зображення до робочої області MATLAB з використанням команди `imread`:

```
w=imread('filename.ext');
```

де filename – ім'я Вашого файлу; ext – його розширення.

2. Застосовуючи команду `size` вивести розмір зображення, присвоївши змінній `M` значення, що відповідає кількості рядків зображення, а змінній `N` - значення, що відповідає кількості стовпчиків:

```
[M,N]=size(w)
```

3. Підрахувати об'єм зображення у байтах.

4. Перевірити правильність розрахунку за допомогою команди whos.

5. Вивести це зображення на екран за допомогою команди imshow:

```
imshow(w)
```

6. Перетворити ваш кольорове зображення у чорно-біле. Використовується команда `rgb2gray`. Приклад перетворення:

```
h = rgb2gray(w);
figure
```

Вивести це зображення на екран

7. Зберегти на жорсткий диск до поточного каталогу ваше зображення у форматі JPEG з налаштуваннями якості $q$ = 100 потім 10 і 0. Для цього застосувати команду `imwrite`:

```
imwrite(w,'filename.jpg','quality',q)
```

Переглянути збережені зображення за допомогою стандартних засобів операційної системи. Зробити висновки про розмір зображення.

## Контрольні запитання

1. Якою командою можна визначити розмір зображення?
2. Яка команда використовується для зчитування зображення до робочої області MATLAB?
3. Яка команда використовується для виведення зображення на екран?
4. Які класи даних використовуються у системі MATLAB?

## Лабораторна робота № 3. Дослідження способі кольорового кодування

**Мета:** ознайомитись зі способами кодування кольорів, конвертуванням колірних систем, а також командами програмного продукту MatLab для роботи з кольоровими зображеннями.

### Теоретичні відомості

Персональні комп'ютери при відображенні використовують 8, 16 або 24 біт на піксель. Це визначає глибину кольору зображення. Система MATLAB може обробляти зображення з різним числом біт на піксель: 224 кольорів для RGB-зображень у форматі uint8, 248 кольорів для RGB-зображень у форматі uint16 та 2159 кольорів для RGB-зображень у форматі точності. Зображення найкраще відображаються у 24-бітний системі.

Для визначення глибини кольору, що може відображати система MATLAB використовує код:

```
get(0,'ScreenDepth')
ans =
```

Так MATLAB возвращает число бит на пиксель:

Таблиця 3.1

Опис бітового представлення кольорової інформації

| Значення | Опис |
|---|---|
| 8 | 8 - бітне уявлення відображається 256 кольорами.<br>8 - бітні напівтонові зображення є складовою 24-бітного представлення графічної інформації |
| 16 | 16 - бітове представлення використовує 5 - біт на кожну колірну компоненту. Це дорівнює 32 градаціям на червону, зелену та синю складові. В результаті підтримується 32768 різних кольорів. |
| 24 | 24 - бітна представлення використовує 8 біт на кожну із трьох колірних складових, тобто. 256 градацій на червону, зелену та синю компоненти. Результатом є 16777216 різних кольорів |
| 32 | 32 - бітна представлення використовує 24 біти для запам’ятовування кольорової інформації. Ще 8 біт використовується для запам’ятовування насиченості даних. |

Метод зменшення числа кольорів на зображенні.

У програмному середовищі MATLAB для зменшення кольорів у зображенні використовуються наступні команди:

imapprox - створює нове палітрове зображення з вихідного палітрового, при цьому зменшуючи кількість кольорів у ньому;
rgb2ind - створює палітрове зображення з полноколірного зображення RGB.

Команда rgb2ind виконує перетворення RGB - зображення в індексне зображення, зменшуючи кількість кольорів. Для обробки вихідного зображення команда використовує наступні методи зменшення кольорів:

- квантування:
  - рівномірне квантування;
  - квантування з найменшою дисперсією;
- відображення палітри.

Квантування веде до зменшення кількості кольорів на зображенні. Функція *rgb2ind* застосовує квантування як частину алгоритму зменшення кольорів. В цьому питанні використовується термін куб RGB кольорів. Куб RGB кольорів є тривимірним масивом всіх кольорів, що визначені для цього типу даних. Оскільки зображення в MATLAB можуть бути представлені у різних форматах (uint8, uint16, double), то це впливає на дискретизацію кольорів у кубі RGB.

При рівномірному квантуванні застосовується *rgb2ind* з відповідними параметрами. У прикладі другий аргумент впливає на дискретність квантування:

```
RGB = imread('peppers.png');
[x,map] = rgb2ind(RGB, 0.1);
```

При застосуванні квантування з найменшою дисперсією команда rgb2ind використовується із зазначенням максимальної кількості кольорів на результуючому зображенні. Це число визначає кількість ячеєк, на яке буде розбитий колірний куб RGB. Наведено приклад застосування методу квантування для створення індексного зображення з використанням 190 кольорів.

```
RGB = imread('peppers.png');
[X,map] = rgb2ind(RGB,190);
```

В основу методу квантування з найменшою дисперсією покладено об'єднання пікселів у групи на підставі відхилень між їх значеннями. Тобто. обраний піксель повинен мати найменше відхилення від усіх пікселів групи.

Команда *imapprox* для зменшення кольорів використовує методи апроксимації. Команда imapprox спочатку за допомогою команди *ind2rgb* здійснює перетворення зображення у формат RGB, а потім використовує команду *rgb2ind* для перетворення на індексне зображення зі зміненою кількістю кольорів.

Нижче приведено приклад формування зображень, які містять 128 та 16 кольорів з застосуванням команд *rgb2ind* та *imapprox* .

```
L=imread('peppers.png');
figure,imshow(L);
L=im2double(L);
[x,map] = rgb2ind(L, 128);
figure,imshow(x,map); %Изображение со 128 цветами
[Y,newmap] = imapprox(x,map,16);
figure,imshow(Y, newmap); %Изображение с 16 цветами
```

**Згладжування колірних переходів**. Метод дифузійного псевдозмішування кольорів. При застосуванні команд *rgb2ind* або *imapprox* для зменшення кількості кольорів, якість результуючого зображення трохи нижче. Це пов'язано зі зменшенням кількості кольорів, за допомогою яких відображається зображення. Обидві команди – *rgb2ind* та *imapprox* – використовують метод дифузійного псевдозмішування кольорів. Це призводить до візуального збільшення кількості відображуваних кольорів. Метод *dithering* змінює кольори околиць пікселів так, що середній колір околиці апроксимує начальний RGB-колір.

Нижче наведено приклад методу дифузійного псевдозмішування кольорів:

1. Зчитування та вивід на дісплей вихідного зображення:

```
rgb=imread('onion.png');
imshow(rgb);
```

2. Створення індексного зображення с восьма кольорами без застосування методу дифузійного псевдозмішування кольорів:

```
[X_no_dither,map]=rgb2ind(rgb,8,'nodither');
figure, imshow(X_no_dither,map);
```

3. Створення індексного зображення з восьма кольорами із застосуванням методу дифузійного псевдозмішування кольорів:

```
[X_dither,map]=rgb2ind(rgb,8,'dither');
figure, imshow(X_dither,map);
```

**Перетворення колірних просторів.**

Перетворення колірних даних між колірними просторами. Найбільш часто у MATLAB застосовується система RGB.

Взагалі відомі два підходи до відображення кольорів з більшого колірного простору у менше. Суть одного з них у тому, що кольори, які знаходяться за межами колірного поля, перетворюються на найбільш близькі за тоном кольори всередині колірного поля. Цей підхід має назву відсікання (Clipping). Другий – це метод стиснення (Compression). Його сутність, що кожен колір на вході незалежно від того, потрапляє він у колірне поле вивідного пристрою чи ні, приводиться до іншого кольору з діапазону кольорів вихідного пристрою. Існуючі методи перетворення колірних просторів відрізняються один від одного трьома головними особливостями: стисненням кольорової гами, тональною компресією (приведення динамічного діапазону вступного пристрою до вивідного) та відображенням точки білого кольору.

Приклад конвертування з RGB у HSV:

```
HSV=rgb2hsv(RGB)
hsvmap=rgb2hsv(rgbmap)
```

Команда *HSV = rgb2hsv (RGB)* створює повнокольорове зображення, значення пікселів якого представлені в колірній системі HSV (hue - колірний тон, saturation - насиченість, value - яскравість), з повного кольорового зображення в колірній системі RGB. Результуюче зображення має формат представлення даних double. Команда *hsvmap=rgb2hsv(rgbmap)* створює палітру *hsvmap*, значення кольору в якої задаються в колірній системі HSV, з вихідної палітри *rgbmap*, що зберігає кольори в колірній системі RGB.

Приклад конвертування з RGB в YIQ:

```
YIQ=rgb2ntsc(RGB)
yiqmap=rgb2ntsc(rgbmap)
```

Команда *YIQ=rgb2ntsc(RGB)* створює повнокольорове зображення, значення пікселів якого визначені у системі кольору YIQ, з початкового

повнокольорового зображення у системі RGB. Результуюче зображення має формат уявлення даних double. Команда *yiqmap=rgb2ntsc(rgbmap)* робить палітру *yiqmap*, кольори якої задаються в колірній системі YIQ, з початкової *rgbmap*, що зберігає кольори в системі RGB.
Приклад конвертування з системи RGB у систему YCbCr.

```
YCbCr=rgb2ycbcr(RGB)
ycbcrmap=rgb2ycbcr(rgbmap)
```

Команда *YCbCr=rgb2ycbcr(RGB)* робить повнокольорове зображення, значення пікселів якого представлені в системі YCbCr, з початкового повнокольорового зображення в системі RGB. Формат представлення даних вихідного та результуючого зображень збігаються.

Команда *ycbcrmap=rgb2ycbcr(rgbmap)* створює палітру *ycbcrmap*, значення кольору якої задаються у системі YCbCr, з початкової палітри *rgbmap*, яка зберігає кольори в системі RGB.

**Представлення зображень у різних колірних просторах.** У робочій простір MATLAB зчитаємо зображення у форматі RGB та перетворимо у колірний простір XYZ:

1. Зчитування початкових даних

```
I_rgb = imread('peppers.png');
figure, imshow(I_rgb);
```

2. Створення структури перетворення кольору. У цій структурі визначено перетворення між двома колірними просторами. Для формування цієї структури використовується команди *makecform*.

3. У прикладі розробляється структура перетворення колірних даних з RGB XYZ.

```
C = makecform('srgb2xyz');
```

4. Процес виконання перетворень. Для виконання перетворень застосовується функція *applycform*, яка як аргумент використовує початкові дані та структуру перетворення кольору. Результатом роботи функції *applycform* є перетворені дані.

```
I_xyz = applycform(I_rgb,C);
figure, imshow(I_rgb);
```

```
Name            Size              Bytes          Class
C               1x1               7744           struct array
I_xyz           384x512x3         1179648        uint16 array
I_rgb           384x512x3         589824         uint8 array
```

Утиліта *colormapeditor*.

Якщо засобами Matlab було активізовано деяке зображення, тоді за командою *colormapeditor* на екрані з'являється вікно редактора карти кольору. Цей редактор можна запустити з меню вікна, в якому розміщено зображення (Edit > Colormap).

У верхній частині вікна розміщено колірну карту у вигляді стрічки, складеної з кольорових прямокутників. Під цією стрічкою розміщені кольорові покажчики в тих місцях, де змінюється швидкість зміни колірних складових R, G, B. Ці покажчики можна переміщати за допомогою миші, при цьому їх колір залишається незмінним, а змінюються області колірної карти, що примикають до покажчика за рахунок лінійної інтерполяції RGB-значень у цих областях. Подвійним клацанням мишки за вказівником можна увімкнути режим редагування кольору покажчика. Також можна видаляти існуючі покажчики та додавати нові.

## Порядок виконання лабораторної роботи

1. Пропрацювати всі описані вище приклади.

2. Припустимо, що початкове повнокольорове зображення занадто темне і має низький контраст. Дане зображення перетворюється на колірну систему HSV, де V є складовою яскравості. Для цієї складової підвищується контраст. Потім зображення перетворюється назад у систему RGB.

```
%Приклад демонструє підхід до фільтрації та
% контрастуванню кольорових зображень.
%Зчитування початкового зображення та вивід на екран.
RGB=imread('flowers.tif'); % назва робочого файлу flowers.tif
imshow(RGB);
%Перетворення у кольорову систему HSV.
HSV=rgb2hsv(RGB);
%Контрастування складової H – яскравості зображення.
HSV(:,:,3)=imadjust(HSV(:,:,3), [0.02 0.68], [], 0.7);
%Перетворення у кольорову систему RGB.
RGB=hsv2rgb(HSV);
```

```
%Вывід результатів на экран.
figure, imshow(RGB);
```

Обробіть будь-яке малоконтрастне зображення за допомогою наведеної програми. Підберіть параметри контрастування так, щоб контрастність результуючого зображення підвищилася.

3.У командній строці MATLAB наберіть команду *Landsatdemo* – у результаті на моніторі відобразиться демонстраційний приклад роботи зі складнокольоровими зображеннями. Самостійно розберіться з можливостями та призначенням цього прикладу.

## Контрольні питання

1. Що називається колірною моделлю зображення. Які знаєте.
2. Охарактеризувати адитивний і субтрактивний методи опису кольорів.
3. Охарактеризувати колірну модель RGB
4. Охарактеризувати колірну модель CMYK
5. Охарактеризувати колірну модель HSB
6. Охарактеризувати колірну модель LAB
7. Охарактеризувати колірну модель YUV
8. Охарактеризувати колірну модель YCbCr
9. Охарактеризувати колірну модель YIQ

## Лабораторна робота 4. Просторовий метод покращення зображення з використанням гістограм.

**Мета**: ознайомитись з принципом просторового методу покращення зображення, на прикладі застосування гістограм у системі моделювання MATLAB використовуючи інструментарій Image Processing Toolbox.

### Теоретичні відомості

Термін «просторова область» відноситься до площини, в якій задане зображення. При просторовій обробці всі методи маніпулюють безпосередньо величинами яскравостів пікселів зображення. Їх можна описати загальним виразом:

$$L2[n, m] = T(L1[n, m])$$

де:

$T$ – деяке правило, яке ставить у відповідність кожному пікселю початкового зображення $L1[n, m]$ відповідний піксель результуючого зображення $L2[n, m]$, який отриманий в результаті обробки.

Для отримання результату $L2[n, m]$, може використовуватися не тільки значення початкового пікселя $L1[n, m]$, але також і значення інших пікселів у деякому околі . Як правило, це прямокутний або квадратний окіл.

Під час обробки зображення, центр цього околу зміщують від пікселя до пікселя і здійснюють розрахунки для кожного центрального пікселя. Якщо для того, щоб отримати яскравість пікселя обробленого зображення використовується яскравість лише одного пікселя початкового зображення, то мають місце градаційні перетворення. Тобто, обробка проводиться в околі розміром 1 х 1 піксель.

На практиці часто використовується наступне степеневе перетворення значення початкового пікселя у результуючий:

$$L2_{n, m} = cL1^{\gamma}{}_{n, m},$$

де:

$c$, $\gamma$ – деякі додатні константи.

Графіки залежностей при різних значеннях $c$, $\gamma$ показані на рис. 4.1.

Проаналізувавши графіки можна зробити висновок, що при значеннях менше одиниці, криві таких залежностей відображають вузький діапазон малих значень яскравостів пікселів вхідного зображення у широкий діапазон яскравостів пікселів результуючого зображення. Якщо значення показника степеню більше за одиницю, то маємо протилежний ефект: вузький діапазон великих яскравостей відображається у широкий діапазон яскравостів пікселів результуючого зображення. Ця процедура носить назву - гамма-корекція.

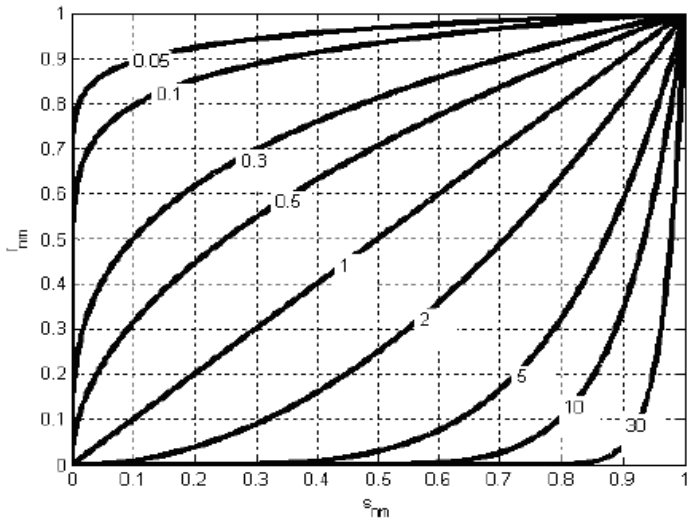



Рис. 4.1. Залежності між яскравостями пікселів вхідного та вихідного зображення при гамма-корецїї при різних значеннях γ

**Еквалізація гістограми.**

Гістограмою дискретного зображення називається дискретна функція $H(b_k)=\frac{N_k}{N}$, де $b$ є $k$ - им рівнем яскравості пікселя, $N_k$ – кількість пікселів, які мають яскравість $b_k$, а $N$ - кількість пікселів у всьому зображенні. Значення $H(b_k)$ є оцінкою імовірності появи пікселя яскравості $b_k$ у зображенні.

*Еквалізацію (лінеаризацію) гістограми* проводять у тому випадку, коли у зображенні присутні багато пікселів зі схожими яскравостями, і мало пікселів з іншими яскравостями. На гістограмі можливо побачити, що на деяких проміжках яскравостей зосереджено багато пікселів, в той час як деякі проміжки яскравостей майже не зайняті. З цього слідує, що деталі зображення, які зображені цими кольорами, складно розрізнити. Проте існують такі участки яскравості, пікселів з якими взагалі немає на зображенні. Ці вільні участки

яскравості можна «задіяти» для покращення якості зображення. Це і є метою процедури еквалізації гістограм.

В результаті проведення еквалізації гістограми, яскравості пікселів на ній будуть розподілені рівномірно по всій шкалі яскравостей.

Перевагою еквалізації гістограм є те, що цей метод легко автоматизується і не вимагає задавання ніяких додаткових параметрів для отримання покращеного зображення. Розрахунки для еквалізації гістограм такою достатньо нескладні.

## Порядок виконання роботи.

1. Завантажити своє зображення з файлу (бажано з поганою якістю у вигляді нерівномірною, контрастною яскравістю), використовуючи команду *imread,* та вивести його на екран за допомогою команди *imshow*:

```
H=imread     ;
   imshow(H)
```

2. Побудувати гістограму заданого зображення за допомогою команди *imhist:*

```
imhist(H)
```

3. Виконати покращення зображення за допомогою наступних команд:

*histeq* – еквалізація (вирівнювання) гістограми `H1=histeq(H);`

Зробити висновок про вплив операції еквалізації на початкове зображення.

Функція imadjust - корекція контрасності та яскравості зображення:

```
H2=imadjust(H, [low_in high_in],[low_out high_out],gamma);
```

Побудувати гістограми зображення для випадків:

а) $\gamma = 1$, $\gamma = 0{,}5$, $\gamma = 3$ при порогах обмеження значення пікселя `[low_in high_in],[low_out high_out]` по замовчуванню `[0 1][0 1];`

б) `[0.3 0.7],[0 1]` при $\gamma = 1$;

в) `[0 1],[0.3 0.7]` при $\gamma = 1$;

г) `[0.3 0.7],[0 1]` при $\gamma = 3$;

д) `[0 1],[0.3 0.7]` при $\gamma = 3$

Проаналізувати результат та зробити висновок про вплив коефіцієнту гамма-корекції та порогів обмеження на характер зображення.

## Контрольні запитання

1. Доповісти сутність методу просторового покращення зображень.
2. Дати визначення гістограми?

2. Доповісти сутність процесу вирівнювання гістограми.

3. Що називається гамма-корекцією?

4. Охарактеризувати степеневе перетворення значення початкового пікселя у результуючий.

## Лабораторна робота № 5. Просторові методи покращення зображення з використанням методу фільтрації

**Мета** – ознайомитись з просторовими методами покращення зображень які базуються на методах фільтрації, застосовуючи програмний комплекс MATLAB

### Теоретичні відомості

### Моделі шуму

**Гаусів шум.** Гаусів шум – це шум у якого щільність ймовірності дорівнює щільності ймовірності нормального розподілу. Широке поширення моделі Гаусова шуму на практиці обумовлено його простотою як у просторовій так і частотній областях. Щільність розподілу імовірності миттєвих значень *x(t)* процесу Гауса визначається виразом:

$$w(x) = (1/\sqrt{2\pi\sigma^2})\exp(-(x-m)^2/(2\sigma^2))$$

**Імпульсний шум.** Функція щільності розподілу ймовірностей імпульсного шуму описується виразом

$$P(z) = \begin{cases} P_a \text{ при } z=a \\ P_b \text{ при } z=b \\ 0 \text{ в інших випадках} \end{cases}$$

У випадку $b > a$, піксель із яскравістю $b$ виглядає як світла крапка на зображенні. Пиксель із яскравістю $a$ виглядає, навпаки, як темна крапка. Якщо одне зі значень імовірності (*Pa* або *Pь*) дорівнює нулю, то імпульсний шум називається уніполярним. Якщо $Pa \neq 0$ *та* $Pь \neq 0$, і у випадку коли вони приблизно рівні, імпульсний шум на зображенні візуально виглядає як крапки чорного і білого кольору. Чому чорного і білого?

Значення імпульсів шуму будуть позитивними і негативними. Під час оцифровки звичайного зображення відбувається масштабування і обмеження значень яскравості. Величина спотворень, викликаних імпульсним шумом, велика у порівнянні з величиною корисного сигналу, тому імпульсний шум після оцифровки буде мати екстремальні значення. Внаслідок цього на зображенні з’являться абсолютно чорні і білі крапки. Тому значення $a$ й $b$ є рівні мінімальному й максимальному значенням, які в можуть бути присутніми

в оцифрованому зображенні. Таким чином негативні імпульси на зображенні будуть виглядати як чорні крапки ("перець" (умовно)). З тих же міркувань позитивні імпульси будуть виглядати як білі крапки ("сіль"). Для 8-бітових зображень це означає, що a = 0 (чорне) і b = 255 (біле). Умовно імпульсний шум і називають "сіль та перець".

***Додавання шуму функцією* imnoise.**

У MATLABI є функція imnoise, яка моделює спотворення зображення певним шумом. Синтаксис цієї функції наступний:

s = imnoise(h, type, parameters)

де:

h — це первинне зображення. Команда imnoise спочатку перетворює зображення в клас double в діапазоні [0,1]. Це необхідно враховувати перед завданням параметрів шуму. Наприклад, для того щоб додати шум з середнім 50 і дисперсією 300 до зображення класу uint8, необхідно стиснути середнє до рівня 50/256, а дисперсію прирівняти до 300/(256) 2. Тільки після цього ці параметри можна підставляти у функцію imnoise. Нижче наведені приклади використання функції imnoise:

s = imnoise(h, 'gaussian', m, var) додає до зображення h шум гауса з середнім m і дисперсією var. За умовчанням, m = 0 і var = 0.01;

s = imnoise(h, 'localvar', V) додає до зображення h локальний шум гауса з нульовим середнім, дисперсія якого в кожній точці зображення h задається матрицею V, розміру як в h;

s = imnoise (h, 'salt & paper', d) спотворює зображення h імпульсним шумом («сіль і перець»), де d – щільність шуму (відсоток вашого зображення, схильного до цього шуму). При цьому приблизно d*numel(h) пікселів буде зіпсовано. За умовчанням, d = 0.05;

s = imnoise(h, "speckle", var) додає до h мультиплікативний шум по формулі s = h "+" n*h, де n - це рівномірно розподілений шум з нульовим середнім і дисперсією var. За умовчанням, var = 0.04;

s = imnoise(h, 'poisson') створює пуасоновський (дробовий) шум, який залежить від вихідних даних, замість додавання штучного шуму. Як відомо пуасоновський шум це хаотичні флуктуації числа частинок відносно їх середнього значення пов’язані з їх дискретністю. Виходячи з цього для узгодження із статистиками Пуасона, пікселі зображень uint8 і uintl6 повинні відповідати числу фотонів, або іншим квантам інформації. Використовуються зображення з подвійною точністю, коли число фотонів на один піксел більше, ніж 65535 (але менше 1012). Величини яскравості пікселів міняються в межах від 0 до 1 і відповідають числу фотонів, що ділиться на $10^{12}$.

## Принцип просторової фільтрації зображень.

Просторова фільтрація зображень – являє собою різновид обробки зображень в просторовій області. Вона відрізняється від класичного поняття «фільтрації», яка базується на спектрах зображень та спектральних характеристик фільтрів. Сутність просторової фільтрації заключається у тому, що для розрахунку яскравості пікселя результуючого зображення використовуються яскравості пікселів в деякій околиці початкового зображення.

Процес просторової фільтрації схематично зображено на рис. 4.1.

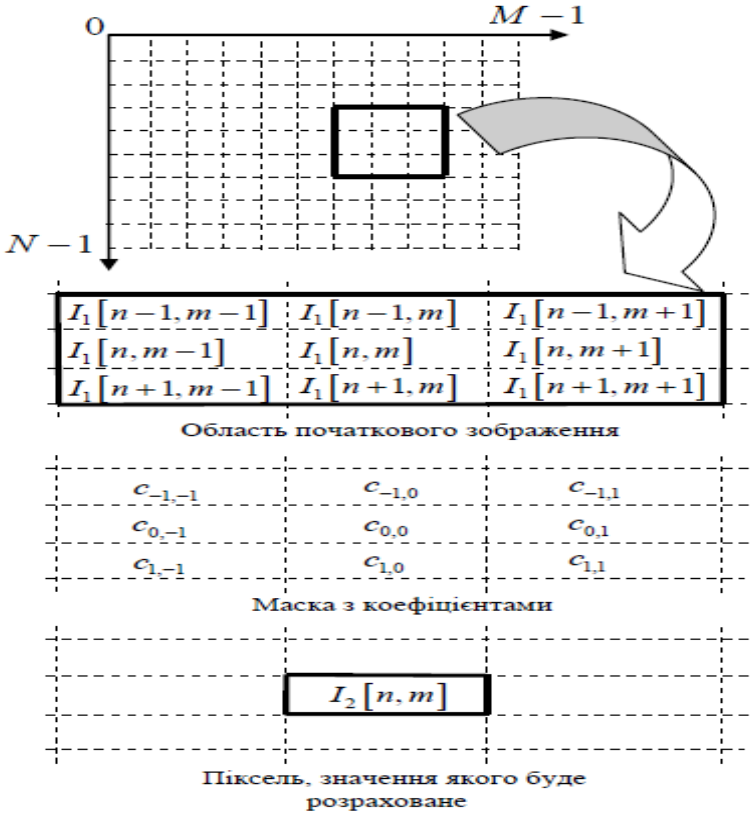



Рис. 5.1. Процес просторової фільтрації

Обробка проводиться послідовно для кожного пікселя зображення. Поширеним різновидом околиці є квадрат 3x3 з робочим елементом у центрі В початковому зображені обирається околиця розміром 3×3 пікселя з центром в деякому пікселі $I1\ [n,m]$. Виходячи зі значення яскравості пікселя $I1\ [n,m]$ та його околу буде розраховуватись яскравість одного пікселя результуючого зображення $I2\ [n,m]$. Для цього обирається «маска» коефіцієнтів. Застосовують різні різновиди масок. Маска містить числа, які мають смисл коефіцієнтів, з якими яскравість кожного пікселя з околиці на початковому зображенні буде використана для отримання яскравості пікселя в результуючому зображенні. Розрахунок

$$I2\ [n,m] = c_{-1,-1}I1[n-1, m-1] + c_{-1,0}\,I1\ [n-1, m] + c_{-1,1}I1\ [n-1, m+1] +$$

$$+ c_{0,-1}I1\ [n, m-1] + c_{0\,0}I1\ [n, m] + c_{0,1}I1\ [n, m+1] + c_{1,-1}I1\ [n+1, m-1] + c_{1,0}I1\ [n + 1, m] + c_{1,1}I1\ [n + 1, m + 1]$$

Видно, що маска центрується в пікселі з номером $[n,m]$, і яскравість цього пікселя множиться на коефіцієнт $C_{0\,0}$, що знаходиться в центрі маски.

Константа $C$ є нормувальним множником. Вона числено дорівнює сумі всіх коефіцієнтів маски, тому при діленні на неї сума коефіцієнтів при всіх яскравостях пікселів дорівнює одиниці. Звідси слідує, що яскравість пікселя відфільтрованого зображення буде не більшою, ніж максимальна припустима для даного зображення яскравість. Нормувальний множник застосовується в тому випадку, якщо сума коефіцієнтів маски не дорівнює нулю.

## Фільтри для зображень.

### Лінійні (усереденюючі) фільтри.

Сутність лінійної (усереднючої) фільтрації полягає у тому, що відповідь лінійного фільтра усереднює значення пікселів, що містяться в апертурі, і таким чином згладжує зображення.

Припустимо $Gxy$ позначає прямокутну околицю (множину координат точок зображення) розмірами $m$x$n$ із центром у точці $(x,y)$. Усереденююча фільтрація означає обчислення середнього арифметичного значення спотвореного зображення $h(x,y)$ по околиці $Gxy$. Значення відновленого зображення $f$ в довільній точці $(x,y)$ являє собою середнє арифметичне значень у точках, що належить околиці $Gxy$.

Характерно, що лінійна фільтрація показуэ хороші результати тільки у разі наявності гауссова адитивного шуму.

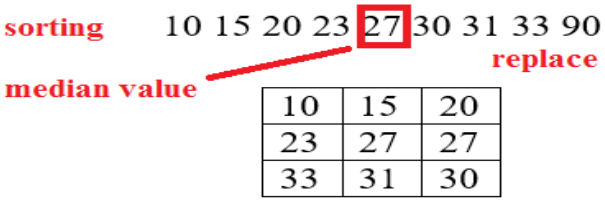
**Медіанні фільтри.**

Суть медіанного фільтру полягає у виборі медіани із набору пікселів околиці. Математичний запис процесу::

$$Im_{i,j} = med[Im_{i+s,j+t}; (s,t) \in W]; i,j \in Z^2$$

Спочатку усі значення пікселів околиці сортуються у певному порядку (зростання) та вибирається медіанне значення, яким замінюється центральний пік-сель:

| | | |
|---|---|---|
| 10 | 15 | 20 |
| 23 | 90 | 27 |
| 33 | 31 | 30 |




| | | |
|---|---|---|
| 10 | 15 | 20 |
| 23 | 27 | 27 |
| 33 | 31 | 30 |

Медіанні фільтри показують велику ефективність при наявності біполярного і уніполярного імпульсного шуму.

**Адаптивні медіанні фільтри.**

Медіанна фільтрація має і недоліки. Зокрема, у даного методу відносно слабка ефективність при фільтрації так званого флуктуаційного шуму. Також при збільшенні розміру маски відбувається розмиття контурів зображення і, внаслідок цього, зниження чіткості зображення.

Таких недоліків можна позбутись якщо застосувати адаптивну медіанну фільтрацію.

Відмінність адаптивної медіанної фільтрації від просто медіанної фільтрації полягає у тому, що у ній змінюються (збільшуються) розміри околиці *Gxy* під час роботи. На скільки потрібно збільшити розмір маски? Позначимо пороговий коефіцієнт відхилу яркости як:

$$S_{поріг} \text{ е } [0, 1].$$

Величини відхилу яркісті сусідніх пікселів *A(r,n),* що попали у вікно розміром n×n відносно яркісті центрального відліку *A(r)* запишуться у вигляді:

$$max\ [S_{mn}(\ г\ )] < S_{поріг}.$$

Маску потрібно збільшувати до тих пір, поки даний вираз буде істиним.

**Порядок виконання роботи.**

1. Зчитати зображення зі свого файлу, використовуючи команду *imread,* та вивести його на екран за допомогою команди *imshow*:

```
F=imread     ;
   imshow(F)
```

2. Додати до початкового зображення шум за допомогою команди *imnoise*:

```
F_noise=imnoise(I,'вид шума');
```

Види шуму, які можна задати в Image Processing Toolbox:
- гаусівський шум ('gaussian');
- імпульсний шум ('salt & pepper');
- пуасонівський шум ('poisson');
- мультиплікативний шум('speckle').

Побудувати гістограми зашумлених зображень для шумів ('gaussian' та "salt & pepper") зробити висновки про характер розподілу яскравості пікселів залежно від виду шуму.

3. Виконати фільтрацію зображень, зашумлених *гаусівським* шумом, за допомогою наступних видів просторових фільтрів:
- медіанного фільтру;
- фільтру усереднення (згладжувального фільтру).
- адаптивного медіанного фільтру.

Привести відфільтровані зображення. Зробити висновки щодо ефективності роботи наведених фільтрів залежно від виду шуму.

а) Медіанна фільтрація здійснюється за допомогою команди *medfilt2:*

```
I_filt=medfilt2(I_noise, [n m]);
```

де:
*n*×*m* – розмір фільтруючої маски (околиці). За замовчуванням *n=m=3.*

б) Адаптивна медіанна фільтрація виконується командою *adpmedian:*

```
I_filt=adpmedian(I_noise, Smax);
```

де:

$S_{max}$ – максимальний розмір маски.

Для порівняння дії фільтрів *med-filt2* та *adpmedian* встановити *Smax=n=m=7.* Зробити висновки про ефективність застосування цих фільтрів.

в) Фільтр усереднення створюється командою формування стандартних фільтрів *fspecial* та використання його до зображення виконується за командою *imfilter:*

```
filt=fspecial('average', [n m]);
I_filt=imfilter(I_noise, filt);
```

Перевірити вплив розміру маски на якість фільтрації та вид зображення, зробити висновки.

**Контрольні запитання.**

1. Перерахувати методи просторової фільтрації.
2. Пояснити роботу лінійного фільтру усереднення.
3. Пояснити принцип дії медіанного фільтру.
4. Пояснити рангову фільтрацію.
5. Чим відрізняється робота адаптивного медіанного фільтру від медіанного фільтру.
6. Назвати команди MATLAB які реалізують методи просторової фільтрації.

## Лабораторна робота 6. Частотні методи покращення зображень

**Мета** – ознайомитись з частотним способом представлення зображень; ознайомитись з частотними методами покращення зображень застосовуючи програмний комплекс MATLAB.

### Теоретичні відомості

Фундаментальний математичний апарат перетворення Фур'є дозволяє поставити у відповідність часовому представленню сигналу його частотне, а також здійснити перехід від частотного до часового уявлення сигналу. Частотне представлення сигналу більш інформативніше ніж часове і дозволяє наглядно показати сутність методу частотної фільтрації.

**Дискретне перетворення Фур'є**. Аналогічно тому як для одновимірного сигналу у часовій області можна знайти його спектральне представлення Фур'є, також і можна знайти спектр зображення. Якщо зображення *f(x, y)* дискретне і має розмірність *N x M*, то його перетворення Фур'є запишеться у вигляді:

$$F(u,v) = \sum_{N=0}^{N-1}\sum_{M=0}^{M-1} f(x,y)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

Пряме перетворення Фур'є будь якого зображення дозволяє представити зображення у частотній області у вигляді ортогональних синусоїд з відповідними частотами (коефіцієнти розкладання Фур'є). Частотною областю будемо називати координатну систему, яка задає аргументи *F(u,v)* частотними змінними *u* та *v*.

**Зворотнє дискретне перетворення Фур'є**. Зворотнє перетворення Фур'є визначиться виразом:

$$F(x,y) = \sum_{u=0}^{N-1}\sum_{v=0}^{M-1} f(u,v)e^{-j2\pi(\frac{ux}{M}+\frac{vy}{N})}$$

Тобто, знаючи коефіцієнти розкладання Фур'є можемо отримати функцію *F(x, y)* - наше зображення.

**Принцип частотного методу покращення зображень.** В основі методу лежить принцип застосування фільтрації. Оскільки коефіцієнти розкладання

Фур’є це ортогональні синусоїди (гармоніки), можемо легко потрібні нам гармоніки видалити (відфільтрувати) зі спектру отриманого у результаті прямого перетворення Фур’є. Потім застосовуючи зворотнє перетворення Фур’є відновлюємо наше покращене зображення. Математичний апарат, який описує процес фільтрації, базується на використанні математичної операції згортка:

$$f(x,y)^* h(x, y) <=> H(u,v) F(u,v)$$

і зворотньо:

$$f(x,y) h(x, y) <=> H(u,v) * F(u,v).$$

Знак «*» означає операцію згортки двох функцій, а вирази по обидві сторони від подвійних стрілок визначають відповідні дії при перетворенні Фур'є. Перший вираз показує, що згортку двох функцій можна отримати якщо обчислити добуток прямих перетворень Фур'є цих двох функцій. І навпаки, пряме перетворення Фур'є згортки двох просторових функцій дає добуток їх прямих перетворень Фур'є.

Відповідно до теореми про згортку, той же результат можна отримати в частотній області, помноживши *F(u,v)* на *H(u,v)* — перетворення Фур'є просторового фільтру. У даному випадку *H(u,v) означає* передаточну функцію фільтру (рис. 6.1.)

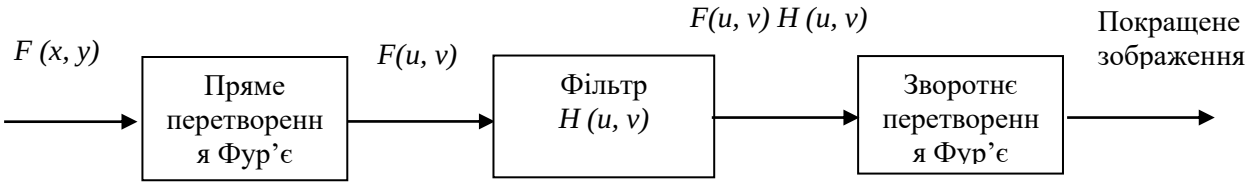



Рис. 6.1. Принцип частотної обробки зображення

**Фільтрація зображень в частотній області.** Як і для одновимірного випадку, можна створити фільтри нижніх (ФНЧ), верхніх (ФВЧ) частот, смугові (СФ) та загороджувальні (ЗФ) фільтри. Оскільки фільтрація зображень має більшу обчислювальну складність, її окрім найбільш простих випадків, виконують у двовимірній частотній області. В частотній області для того, щоб отримати спектр результуючого зображення *F2(u,v)*, необхідно перемножити спектр початкового зображення *F1(u,v)* на комплексну частотну характеристику фільтра *H(u,v)*:

$$F2(u,v) = H(u,v)\, F1(u,v)$$

Для отримання відфільтрованого зображення виконується обернене перетворення Фур'є.

**Алгоритм фільтрації у частотної області**. Покрокова процедура з використанням функцій MATLAB. Дано: $f$ вихідне зображення; $g$ — результат фільтрації.

1. Отримати параметри розширення з допомогою paddedsize:

```
PQ = paddedsize(size(f));
```

2. Побудувати перетворення Фур'є з розширенням:

```
F = fft2(f,PQ(1),PQ(2));
```

3. Сінтезувати функцію фільтру H розміром PQ(1)xPQ(2) одним з описаних далі методів. Якщо він був центрований, до використання його у фільтрації слід виконати команду H = fftshift(H).

4. Помножити перетворення Фур'є на передаточну функцію фільтра:

```
G = H.*P;
```

5. Знайти дійсну частину зворотного перетворення Фур'є від G:

```
g = real(ifft2(G));
```

6. Вирізати верхній лівий прямокутник початкових розмірів:

```
g = g(1:size(f,1), 1:size(f,1));
```

**Низькочастотні фільтри.** Фільтр нижніх частот - електричне коло, смуга пропускання якого лежить в діапазоні від нуля до граничної частоти. Передаточна функція ідеального низькочастотного фільтру має вигляд:

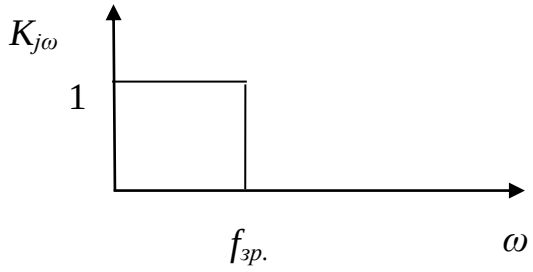



Рис. 5.2. Передаточна характеристика фільтру НЧ

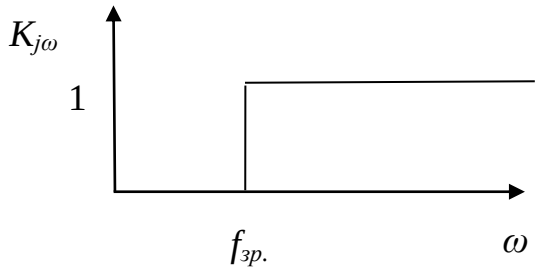
Математично ідеальний ФНЧ описується як:

$$H(u,v) = \begin{cases} 1 \text{ коли } K(u,v) \le K_0 \\ 0 \text{ коли } K(u,v) > K_0 \end{cases}$$

де: $K_0$ – це число >0 (поріг), $K(u,v)$ – відстань від центру фільтра до точки $(u, v)$. Геометричне місце точок $(u,v)$, для яких $K(u, v) = D_0$, представляє собою коло. Передаточна характеристика фільтра множиться на зображення і тому даний фільтр придушує високі частоти в зображенні. По іншому він «зрізає» - множить на нуль всі компоненти $F$, що знаходяться поза цим колом, і залишає незмінними усі компоненти, які знаходяться усередині та на кордоні кола.

Це призводить до розмитості відфільтрованого зображення.

**Покращення різкості при частотній фільтрації.** Фільтр високих частот - електричне коло, смуга пропускання якого лежить в діапазоні від граничної частоти до безкінечності. **В**исокочастотна фільтрація, на противагу низькочастотній фільтрації яка призводить до розмиття зображень, підвищує різкість зображення. Вона ослаблює низькі частоти і залишає високі частоти перетворення Фур'є відносно незмінними. Передаточна характеристика ВЧ фільтру має вигляд:




## Хід виконання роботи.

1. Зчитати зображення з файлу (відповідно до варіанту завдання), використавши команду *imread*.

2. Отримати дискретне перетворення Фур’є від зображення:

```
F=fft2(I);%ДПФ від матриці зображення
F2=log(abs(F));%Для візуалізації ДПФ необхідно взяти його модуль у
                логарифмічній шкалі
imshow(F2[-5 15]); colormap(jet); colorbar
```

Проаналізувати отримане зображення і відповісти на питання, де знаходиться початок координат на отриманій картинці.
Для відцентровування результату ДПФ виконати команду:

```
F1=fftshift(F);
```

Проаналізувати отримане зображення, охарактеризувавши ділянки низьких та високих просторових частот.
3. Відфільтрувати зображення, застовувавши команду *fspecial*:

```
H=fspecial('вид фільтру');
figure
freqz2(H);%Виводить АЧХ фільтру на екран
I2=imfilter(I,H);% Здійснюється фільтрація вихідного зображення
```

Отримати ДПФ відфільтрованого зображення і пояснити результат фільтрації, використовуючи представлення зображення у частотній та просторових областях.
Вид фільтру:

- 'gaussian' – гаусів НЧ фільтр;
- 'average' – фільтр усереднення НЧ;
- 'laplacian' – ВЧ фільтр Лапласа;

4. Проектування фільтрів.
4.1. За заданою АЧХ за допомогою команди *fsamp2*:

```
[u,v]=freqspace(20,'meshgrid');
%Створюємо маску фільтру ВЧ
Hd=zeros(20);
R=sqrt(u.^2+v.^2);Hd(R>0.15)=1;%0.15 - відносна частота зрізу
figure
H=fsamp2(Hd);%Створення фільтру по його масці
freqz2(H);
I2=imfilter(I,H);
```

Отримати ДПФ відфільтрованого зображення і пояснити результат фільтрації, використовуючи представлення зображення у частотній та просторовій

областях. Проаналізувати вплив на кінцеве зображення величини частоти зрізу фільтру.

4.2. Повторити п.4.1, створивши фільтр НЧ. Для цього замінити команду *zeros* на *ones* та присвоїти *Hd(R>0.15)=0.*

4.3. Методом перетворення частот за допомогою команди *ftrans2*:

```
B=fir1(6,0.3);%Створення коеф. одновимірного ФНЧ 6-го порядку з
              відносною частотою зрізу 0,3
H=ftrans2(B);
figure
freqz2(H);
I2=imfilter(I,H);
```

Отримати ДПФ відфільтрованого зображення та пояснити результат фільтрації.

4.4. Повторити п.4.3, створивши фільтр ВЧ. Для цього у команду *fir1* внести наступні зміни:

```
B=fir1(6,0.3,'high');
```

## Контрольні питання

1. Пояснити сутність двовимірного перетворення Фур'є.
2. Пояснити сутність фільтрації зображення у частотній області
3. Що таке передаточна характеристика фільтру?
4. Який вплив фільтрів НЧ та ІЧ на зображення?
5. Якими командами MATLAB виконується фільтрація зображення у частотній області?

## Лабораторна робота № 7. Реставрація зображень

**Мета**: ознайомитись з методами фільтрації, які застосовуються для реставрації зображень, використовуючи програмний продукт моделювання MATLAB.

### Теоретичні відомості

**Інверсна фільтрація.** Реставрація зображень це процес оцінювання вихідного зображення, яке зазнало деякого перетворення. Для того щоб успішно вирішувати таке завдання бажано мати модель системи яка спотворює. В реальній ситуації спотворення виникають через наприклад через турбулентність атмосфери, дефекти фотоматеріалів, датчиків зображень, швидкий рух датчика зображень ін. Для низки таких явищ розроблено відповідні математичні моделі, на основі яких будуються алгоритми реставрації. У випадку якщо просторові спотворення можна моделювати користуючись незалежним від лінійного зсуву імпульсним відгуком і адитивним шумом, реставрацію зображення можна здійснювати методами лінійної фільтрації, до якої наприклад відноситься інверсна фільтрація.

Припустимо, що неспотворене зображення *FI(x,y)* проходить через лінійну систему спотворення з імпульсним відгуком $h_D(x,y)$ і спотворюється адитивним шумом *n(x,y)* некорельованим із зображенням.

Система реставрації має вигляд фільтра із незалежним від лінійного зсуву імпульсним відгуком $h_R(x,y)$. На виході цього фільтра виходить виправлене зображення. Зворотнє перетворення Фур'є дозволяє отримати виправлене зображення яке описує функція:

$$\hat{F}_i(x,y) = F_i(x,y) + \frac{1}{4\pi^2}\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}\frac{N(\omega_x,\omega_y)}{H_D(\omega_x,\omega_y)}\exp[j(\omega_x x+\omega_y y)]d\omega_x d\omega_y \quad (6.1)$$

$H_D(\omega_x,\omega_y)$ - зображення яке викликане появою шума.

Друга складова у (6.1) - це помилка. У тих областях, де $H_D$ мало (АЧХ), а шум присутній, друга складова у (6.1) може сильно зростати, що виражається, зрештою, у сильному спотворенні дрібних деталей зображення. Тому таке явище може спричинити значні помилки в оцінці вихідного зображення.

Можливо використовувати різні способи послаблення шумів які виникають при інверсній фільтрації. Наприклад один з них полягає у застосуванні реставруючого фільтра з частотною характеристикою виду:

$$H_R(\omega_x,\omega_y) = H_K(\omega_x,\omega_y) / H_D(\omega_x,\omega_y)$$

де:

$$\left|H_K(\omega_x,\omega_y)\right| \approx \begin{cases} 1, \text{при} \left|\Phi_I(\omega_x,\omega_y)\right| > \left|N(\omega_x,\omega_y)\right| \\ 0, \text{при} \left|\Phi_I(\omega_x,\omega_y)\right| \le \left|N(\omega_x,\omega_y)\right| \end{cases}$$

**Вінерівська фільтрація.** Інверсна фільтрація має низьку завадостійкість, тому що цей метод не враховує зашумленість зображення. Значно менш схильний до впливу завад і сингулярностей, обумовлених нулями передаточної функції системи, що спотворює, фільтр Вінера, т.к. при його синтезі поряд з видом функції розсіяння точки використовується інформація про спектральні щільності потужності зображення та шуму.

У даному випадку використовуються відомості про статистичні властивості шуму. Якщо ідеальне зображення $\Phi_N$ спотворене адитивним гаусівським шумом з нульовим середнім та відомим енергетичним спектром, то мінімальна середньоквадратична помилка реставрації досягається використанням вінерівського фільтра, частотна характеристика якого має вигляд

$$H_R(\omega_x,\omega_y) = \frac{H_D^*(\omega_x,\omega_y)}{\left|H_D(\omega_x,\omega_y)\right|^2 + \Phi_N(\omega_x,\omega_y)}$$

де * означає комплексно спряжену частотну характеристику.

**Метод сліпої деконволюції**. Сліпа деконволюція це метод відновлення зображення без апріорної інформації про функцію розмиття точки оптичної системи, яка вносить в корисний сигнал шум, спотворення і т.п.

Принцип методу сліпої деконволюції заключається у наступному: вибирається перше наближення функції спотворення зображення (PSF), далі однією з методів робиться деконволюція (операція зворотня згортки сигналів), після чого деяким критерієм визначається рівень якості, з урахуванням її уточнюється функція PSF і ітерація повторюється до досягнення потрібного результату.

## Порядок виконання роботи

1. Застосувати до зображення размиття типу «гаусів шум»:

```
I = imread('xxxxxxx.xxx');
figure; imshow(I); title('Original Image');
G = fspecial('gaussian',7,10);
Blurred = imfilter(I,H,'symmetric','conv');
figure; imshow(Blurred); title('Blurred Image');
```

2. Відреставрувати зображення методом сліпої деконволюції, припускаючи, що інформація про спотворюючий оператор *G* не відома:
а) спотворюючий оператор *G* на 6 пікселів менше по горизонталі та по вертикалі:

```
UNDERPSF = ones(size(H)-6);
[J1 G1] = deconvblind(Blurred,UNDERPSF);
figure;
imshow(J1);title('Deblurring with Undersized PSF');
```

б) спотворюючий оператор *G* на 6 пікселів більше по горизонталі и вертикалі:

```
OVERPSF = padarray(UNDERPSF,[6
6],'replicate','both');
[J2 G2] = deconvblind(Blurred,OVERPSF);
figure;imshow(J2);
title('Deblurring with Oversized PSF');
```

Зробити висновок про реставрацію зображення методом сліпої деконволюції, у випадку обмеженості інформації про спотворюючий оператор.
в) спотворюючий оператор *G* має початковий розмір:

```
INITPSF = ones(size(G));
[J1 G1] = deconvblind(Blurred,INITPSF,N);
figure;
imshow(J1);title('Deblurring with Initial PSF');
```

де N – кількість ітерацій застосування методу. Порівняти результати реставрації для кількості ітерацій N=5, 10, 20.
Оцінити візуально якість отриманого зображення по пунктам а), б) и в) і зробити висновок про обмеження точності реставрації.
3. Застосувати до зображення завдання розмиття типу «рух»:

```
I = imread('xxxxxxx.xxx');
figure;imshow(I);title('Original Image');
G = fspecial('motion',30,10);
Blurred = imfilter(I,G,'circular','conv');
figure; imshow(Blurred);
title('Blurred');
```

5. Відреставрувати зображення застосовуючи метод інверсної фільтрації з використанням команди `deconvwnr`:

```
wnr1 = deconvwnr(Blurred,G);
figure;imshow(wnr1);
title('Restored');
```

Зробити висновок про якість реставрації зображення при впливі на нього розмиття.

6. Додати до розмитого зображення шум та повторити застосування команди:

```
deconvwnr:
noise = 0.1*randn(size(I));
BlurredNoisy = imadd(Blurred,im2uint8(noise));
figure;
imshow(BlurredNoisy);title('Blurred & Noisy');
wnr2 = deconvwnr(BlurredNoisy,G);
figure;
imshow(wnr2);
title('Inverse Filtering of Noisy Data');
```

Зробити висновок про якість реставрації зображення при впливі на нього спотворюючого оператору та шуму.

7. Ввести контроль відношення шум-сигнал, при цьому використання команди `deconvwnr` еквівалентно використанню вінерівської фільтрації:

```
NSR = sum(noise(:).^2) /sum(im2double(I(:)).^2);
wnr3 = deconvwnr(BlurredNoisy,H,2*NSR);
figure;
imshow(wnr3);
title('Restored with NSR');
```

Як змінилась якість відреставрованого зображення?

## Контрольні питання

1. Що називається функцією спотворення (функція розсіяння точки)?
2. Охарактеризувати інверсної фільтрацію. Які обмеження накладаються на результат відновлення?
3. Охарактеризувати вінерівську фільтрацію, її особливості?
4. В чому полягає метод сліпої деконволюції і коли він використовується?

## 3. СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Творошенко І. С. Конспект лекцій з дисципліни «Цифрова обробка зображень» для студентів 4 курсу денної форми навчання напряму 6.080101 – Геодезія, картографія та землеустрій / І. С. Творошенко; Харків. нац. ун-т міськ. госп-ва ім. О. М. Бекетова. – Харків : ХНУМГ ім. О. М. Бекетова, 2017. – 75 с.
2. А.О. Різуненко Р 49 Теорія та практика цифрової обробки зображень: Монографія. – Полтава: РВВ ПУСКУ, 2009. – 195 с.
3. Р. Гонсалес, Р. Вудс Цифровая обработка изображений в среде MatLab 3. Pattern recognition, fourth edition / Sergios Theodoridis, Konstantinos Koutroumbas. – Elsevier Inc., 2009. – 961 p.
4. М. А. Павлейко, В. М. Ромаданов Спектральные преобразования в MatLab. – СПб, 2007. – 160 С.
5. Баскаков С.И. Радиотехнические цепи и сигналы. Учебн для вузов - 3 изд. Перераб и доп. - М. Высш шк. 2000 - 448 с. ил.
6. Прэтт У. Цифровая обработка изображений: Пер. с англ. – М: Мир, 1982. – Кн.2 –480с.
7. Конспект лекцій з дисципліни “Обробка сигналів та зображень” (для студентів денної форми навчання напряму 6.170101 «Безпека інформаційних і комунікаційних систем») / Укладачі: к.т.н., доцент Фриз М.Є., Стадник М. А. – Тернопіль: ТНТУ, 2015 – 97 с.

Навчально-методичне видання

# ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМ

**Методичні вказівки**
**до виконання лабораторних робіт**
для студентів спеціальності
123 «Комп'ютерна інженерія»

Укладачі: **Котенко** Андрій Миколайович,
**Хлапонін** Юрій Іванович

Комп'ютерне верстання *М.М. Власенко*

Підписано до друку 23.09.2022 Формат 60 × 84 1/ 16
Ум. друк. арк. 2,56. Обл.-вид. арк. 1,41.
Електронний документ. Вид № 59/ІІІ-17.

Видавець і виготовлювач
Київський національний університет будівництва і архітектури

Повітрофлотський проспект, 31, Київ, Україна, 03680
Свідоцтво про внесення до Державного реєстру суб'єктів
видавничої справи ДК № 808 від 13.02.2002 р.