МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»

Л. О. Краснов, К. Ю. Дергачов,

# УПРАВЛІННЯ В УМОВАХ НЕВИЗНАЧЕНОСТІ (ОБРОБЛЕННЯ ЗОБРАЖЕНЬ І ВІДЕОІНФОРМАЦІЇ)

Навчальний посібник

Харків «ХАЇ» 2017

УДК 004.942(075.8)
ББК 32.973.26−018.2Я73
К78

Рецензенти: канд. техн. наук В. О. Кочура,
канд. техн. наук С. М. Флерко

**Краснов, Л. О.**
К78 Управління в умовах невизначеності (оброблення зображень і відеоінформації) [Текст]: навч. посіб. / Л. О. Краснов, К. Ю. Дергачов. – Харків: Нац. аерокосм. ун-т ім. М. Є. Жуковського «Харків. авіац. ін-т», 2017. – 124 с.

**ISBN 978-966-662-530-7**

Подано матеріали з теоретичного і практичного вивчення методів аналізу і оброблення зображень у системах комп`ютерного зору за допомогою програми Matlab. Розглянуто приклади оброблення. Описано можливість роботи з реальними зображеннями і даними відеокамер.

Для студентів напрямів підготовки «Авіоніка», «Аеронавігація» і «Системна інженерія» спеціальностей «Системи управління літальними апаратами і комплексами», «Комп`ютеризовані системи управління і автоматика», «Аеронавігаційні системи і системи аеронавігаційного обслуговування».

Іл. 72. Табл. 4. Бібліогр.: 17 назв

**УДК 004.942(075.8)**
**ББК 32.973.26−018.2Я73**



**ISBN 978-966-662-530-7**

# ВСТУП

Пропонований посібник присвячено вирішенню актуального завдання аналізу і оброблення зображень і відеоданих при проектуванні систем збирання і зберігання інформації, робототехніки і систем комп'ютерного зору. Практичні навички в цій області дозволять уникнути помилок при проведенні наукових досліджень, синтезі алгоритмів оброблення інформації, істотно скоротити витрати і час на проектування.

При підготовці рукопису автори передбачали, що читач має необхідні знання в області цифрового оброблення сигналів і достатні навички програмування в середовищі MATLAB, що дає великі можливості для ефективної роботи із сигналами і зображеннями.

У роботі коротко викладено теоретичні й практичні аспекти оброблення зображень і відеоінформації. Велику увагу приділено питанням фільтрації й стискування кольорових і півтонових зображень, їх спектральному аналізу. Досить детально описано методи аналізу відеоданих (збережених у вигляді файла або таких, що отримані безпосередньо з відеокамери) з використанням додатків Image Processing Toolbox, пакета Computer Vision System Toolbox в додатку Simulink і пакета Image Acquisition Toolbox.

За допомогою цього пакета можна спрямувати потік зображень з камери безпосередньо в середовище MATLAB, використовуючи всього декілька нескладних команд. Це забезпечує дослідників в області оброблення зображень взаємно доповнюючими технічними засобами, необхідними для ефективної роботи.

Описано методи пакета Wavelet Toolbox, які істотно розширюють можливості користувача в тих областях, де зазвичай застосовують техніку Фур'є-розкладання при аналізі сигналів і зображень.

Вивчення матеріалів цієї роботи дозволяє провести цілий ряд самостійних досліджень. Це принесе безперечну користь для набуття теоретичних і практичних навичок в області аналізу зображень різної природи і дозволить надалі перейти до освоєння найсучасніших методів оброблення інформації в системах комп'ютерного зору.

Необхідно зазначити, що через обмежений обсяг публікації у посібнику відбито далеко не всі аспекти проблем, що вивчаються. Сподіваємося, що читачу удасться самостійно подолати труднощі вивчення і використовувати отримані навички в повсякденній практиці дослідження і проектування систем управління, орієнтованих на використання візуальної інформації і даних відеоспостережень.

# 1. ПРАКТИЧНЕ ВИКОРИСТАННЯ СУЧАСНИХ МЕТОДІВ АНАЛІЗУ ЗОБРАЖЕНЬ У СИСТЕМАХ УПРАВЛІННЯ

Бурхливий розвиток теорії цифрового оброблення сигналів і зображень та істотне збільшення можливостей сучасних комп'ютерних систем породили величезну кількість практичних додатків цих технологій в галузі медицини, космічних досліджень і екологічного моніторингу, всіляких систем управління рухом наземних об'єктів і літальних апаратів, робототехніки з використанням візуальної інформації.

Це породжує потребу залучення фахівців в області оброблення зображень і відеоданих. Сподіваємося, що зроблена нами спроба викласти суть основних завдань і методів їх практичного вирішення в цій області виявиться корисною і дасть імпульс до подальшого вивчення.

## 1.1. Основні засоби і методи цифрового оброблення зображень

Всілякі завдання цифрового оброблення зображення можна умовно розділити на два великі класи:

- аналіз і оброблення статичних зображень;
- робота з динамічними зображеннями (відеоданими).

До першого класу відноситься оброблення величезної кількості окремих статичних зображень. Це завдання оброблення медичних діагностичних знімків, аналізу окремих зображень поверхні Землі з космосу, розпізнавання текстової інформації та ін. Основою роботи із зображеннями статичного характеру є методи просторового оброблення даних, що використовують різні варіанти просторової двовимірної лінійної і нелінійної фільтрації, геометричних і морфологічних перетворень, сегментації, пошуку об'єктів на зображеннях і обчислення їх ознак. Розроблено і широко використовується велика кількість алгоритмів аналізу і поліпшення якості статичних зображень, їх спектрального аналізу і стискування.

У цей час досить глибоко пропрацьовано питання автоматизації цих методів. Розроблені і широко використовуються комп'ютерні бібліотеки алгоритмів оброблення зображень. Як приклад наведемо широко використовувану бібліотеку OPENCV (Open Source Computer Vision). Вона доступна на різних мовах: C, C++, Python, CUDA, Java. Дотримуються основні операційні системи: MS Windows, Linux, Mac, Android, ios. Фактично OPENCV – це набір типів даних, функцій і класів для оброблення зображень алгоритмами комп'ютерного зору.

Бібліотека складається з набору модулів, кожен з яких виконує певні завдання. Один з основних модулів CV – модуль оброблення зображень і комп'ютерного зору, який забезпечує:

- базові операції над зображеннями (фільтрація, геометричні перетворення, перетворення колірних просторів і т. д.);
- аналіз зображень (вибір відмітних ознак, морфологія, пошук контурів, гістограми);
- аналіз руху, стеження за об'єктами;

- виявлення об'єктів, зокрема осіб;
- калібрування камер, елементи відновлення просторової структури.

Існує ще цілий ряд програмних продуктів, алгоритмів оброблення статичних зображень, що успішно реалізовують роботу. Другий клас завдань полягає в обробленні динамічних зображень (відеоданих). На відміну від статичних зображень у відеопослідовності, яка складається з окремих кадрів, міститься інформація про зміни, що відбуваються в спостережуваній сцені з часом. Ці відмінності тим більше, чим вище швидкість взаємного переміщення спостережуваних об'єктів і точки відеоспостереження. При цьому з часом може істотно змінитися як положення спостережуваного об'єкта, так і умови освітленості сцени. Це приводить до значних змін розподілу яркостей як об'єкта спостереження, так і фону, що оточує його.

Інформаційна мінливість послідовності відеоданих сформувала необхідність уже не просто враховувати просторовий розподіл яскравості, а застосовувати методи просторово-часового оброблення зображень. Такий підхід дозволяє вирішити актуальні завдання виявлення і високоточного супроводу об'єктів відеоспостереження в системах автоматичного управління. При цьому з'являється можливість оцінювання параметрів руху об'єктів як на площині, так і в тривимірному просторі, визначення параметрів траєкторії. У даний час розроблення ефективних алгоритмів оброблення зображень, орієнтованих на використання в системах управління рухом, є однією з актуальних науково-дослідних проблем.

## 1.2. Оброблення зображень у програмі MATLAB

Одним із найбільш доступних і конструктивних підходів для вирішення комплексу завдань з аналізу і оброблення зображень і відеоінформації є використання програми MATLAB з додатками Image Processing Toolbox, пакета Computer Vision System Toolbox в додатку Simulink, пакетів Image Acquisition Toolbox і Wavelet Toolbox.

***Пакет Image Processing Toolbox*** надає великий набір засобів для цифрового оброблення і аналізу зображень. Будучи тісно пов'язаним із середовищем розроблення додатків MATLAB, Image Processing Toolbox звільняє від виконання тривалих операцій кодування і налагодження алгоритмів, дозволяючи зосередити зусилля на вирішенні основної наукової або практичної задачі. Основні властивості пакета:

- відновлення і виділення деталей зображень;
- робота з виділеною ділянкою зображення;
- аналіз зображення;
- лінійна фільтрація;
- перетворення зображень;
- геометричні перетворення;
- збільшення контрастності важливих деталей;
- бінарні перетворення;

- оброблення зображень і статистика;
- колірні перетворення;
- зміна палітри;
- перетворення типів зображень.

Пакет Image Processing Toolbox дає великі можливості для створення і аналізу графічних зображень у середовищі MATLAB. Він забезпечує надзвичайно гнучкий інтерфейс, що дозволяє маніпулювати зображеннями, інтерактивно розробляти графічні картини, візуалізувати набори даних і анотувати результати для технічних описів. Гнучкість, з'єднання алгоритмів пакета з такою особливістю MATLAB, як матрично-векторний опис, роблять пакет дуже вдало пристосованим для вирішення практично будь-яких завдань щодо розроблення і подання графіки. У MATLAB входять спеціально розроблені процедури, що дозволяють підвищити ефективність графічної оболонки.

Все це дозволяє користувачеві витрачати значно менше часу і сил на створення стандартних графічних зображень і, таким чином, сконцентрувати зусилля на аналізі важливих деталей і особливостей зображень.

MATLAB і пакет Image Processing Toolbox максимально пристосовані для розвитку, впровадження нових ідей і методів користувача. Для цього є набір пакетів, що сполучаються, спрямованих на вирішення всіляких специфічних завдань.

***Пакет Computer Vision System Toolbox***. Одним із найбільш продуктивних методів оброблення відеоданих і зображень у MATLAB є використання пакета Computer Vision System Toolbox у додатку Simulink. Він містить велику бібліотеку блоків для оброблення відеофайлів і зображень. Основні з них:

- блоки аналізу і поліпшення зображень (Analysis & Enhancement);
- фільтрації (Filtering),
- геометричних перетворень (Geometric Transformations);
- морфологічних перетворень (Morphological Operations).

***Пакет Image Acquisition Toolbox*** дозволяє безпосередньо підключати, набудовувати і управляти засобами формування зображень і потокового відео. Це істотно розширює середовище технічних обчислень MATLAB. Разом з іншими застосуваннями, такими, як Image Processing Toolbox і Computer Vision System Toolbox, забезпечує високий рівень проведення аналізу і оброблення даних.

***Пакет Wavelet Toolbox*** надає користувачеві великий набір програм для дослідження багатовимірних нестаціонарних явищ за допомогою вейвлетів (коротких хвилевих пакетів). Порівняно недавно створені методи пакета Wavelet розширюють можливості користувача в тих областях, де зазвичай застосовують техніку Фур'є-розкладання. Він може бути корисним для таких застосувань, як оброблення мови і аудіосигналів, телекомунікації, геофізика, фінанси і медицина. Основні властивості пакета:

- вдосконалений графічний призначений для користувача інтерфейс і набір команд для аналізу, синтезу, фільтрації сигналів і зображень;
- перетворення багатовимірних безперервних сигналів;
- дискретне перетворення сигналів;
- декомпозиція і аналіз сигналів і зображень;
- широкий вибір базисних функцій, включаючи корекцію граничних ефектів;
- пакетне оброблення сигналів і зображень;
- аналіз пакетів сигналів, оснований на ентропії;
- фільтрація з можливістю встановлення жорстких і нежорстких порогів;
- оптимальне стискування сигналів і зображень.

Користуючись цим пакетом, можна аналізувати такі особливості, які упускають інші методи аналізу сигналів, тобто тренди, викиди, розриви в похідних високих порядків. Пакет дозволяє стискувати і фільтрувати сигнали без явних втрат навіть у тих випадках, коли потрібно зберегти і високо-, і низькочастотні компоненти сигналу. Є алгоритми стискування і фільтрації і для пакетного оброблення сигналів. Програми стискування виділяють мінімальне число коефіцієнтів, що подають вихідну інформацію найточніше, що дуже важливо для подальших стадій роботи системи стискування. У пакет включені такі базисні набори вейвлетів: біортогональний, Хаару, «Мексиканський капелюх», Майєру та ін. Детальний опис пояснює принципи роботи з методами пакета, а наведені приклади дозволяють отримати навички практичної роботи.

Таким чином, ураховуючи ефективність і поширеність середовища MATLAB для вирішення завдань оброблення зображень, можна рекомендувати її використання для проведення досліджень у такій області. Далі застосовуватимемо програмні коди MATLAB у матеріалах нашої роботи.

Умовимося, що для опису форматів команд, функцій і операторів MATLAB тут і далі використовуватимемо шрифт `Courier New`.

## 2. ВИКОРИСТАННЯ ДОДАТКА IMAGE PROCESSING TOOLBOX ДЛЯ ОБРОБЛЕННЯ ЗОБРАЖЕНЬ

У цьому розділі розглянемо основні питання оброблення зображень за допомогою програми MATLAB.

Додаток Image Processing Toolbox є набором функцій, які розширюють можливості числових обчислень у середовищі MATLAB. Додаток підтримує різні операції оброблення зображень, включаючи:

- просторові перетворення зображень;
- морфологічні операції;
- ковзаюче і блокове оброблення;
- різні види лінійної і нелінійної фільтрації;
- аналіз і поліпшення зображень;
- відновлення зображень;

- видалення розмитостей;
- оброблення області інтересу.

Багато функцій додатка подаються в системі MATLAB у вигляді M-файлів. У них реалізовані найбільш відомі алгоритми оброблення зображень. Крім того, існує можливість перегляду програмних кодів функцій, які реалізують ці алгоритми:

```
type function_name.
```

Усі M-файли додатка Image Processing Toolbox мають однакові властивості і можуть використовуватися в комбінації з M- файлами інших застосувань, таких, як, наприклад, Signal Processing Toolbox і Wavelet Toolbox.

У цьому розділі розглянемо питання прочитування, візуалізацію і записи зображень на диск.

Нагадаємо, що для початку роботи необхідно очистити робочий простір MATLAB від усіх змінних і закрити відкриті вікна відображень

```
clear, close all
```

### 2.1. Прочитування даних зображення

Для прочитування зображень використовують функцію `imread`. Покажемо на прикладі прочитування одного зображення `pout.tif`, яке включено в додаток Image Processing Toolbox, і його запам'ятовування у вигляді масиву `I`.

```
I = imread('pout.tif');
```

Функція `imread` прочитує дані з графічного формату, поданого як TIFF (Tagged Image File Format). Список усіх підтримуваних форматів знаходиться в описі функції `imread`. Найбільш відомі формати, підтримувані в MATLAB, наведені нижче:

- `BMP` Windows Bitmap
- `GIF` Graphics Interchange Format
- `HDF` Hierarchical Data Format
- `JPEG, JPG` Joint Photographic Experts Group
- `PBM` Portable Bitmap
- `PCX` Paintbrush
- `PGM` Portable Graymap
- `PNG` Portable Network Graphics
- `TIFF, TIF` Tagged Image File Format

Для прочитування зображення, розташованого в деякій конкретній теці, треба в явному вигляді вказати повний шлях до цієї директорії, наприклад:

```
F=imread('C:\Users\krasnov.MEDICA\Desktop\MED_IMAGE\phono.
jpg');
```

У першому прикладі функція `imread` прочитує дані з графічного формату `TIFF`, в другому – `JPEG`.

Функція `imread` дозволяє прочитувати зображення з графічних файлів різних форматів і з різною глибиною кольору (кількість бітів на піксель у кожній колірній компоненті). Більшість форматів використовують для запам'ятовування значень пікселів 8 бітів на піксель.

Якщо ці дані прочитуються в пам'ять, то система MATLAB запам'ятовує їх у форматі `uint8`.

Для файлових форматів `PNG` і `TIFF`, які підтримують 16-бітові дані, система MATLAB запам'ятовує зображення у форматі `uint16`.

Для індексних зображень функція `imread` завжди прочитує палітру `colormap` у матрицю, яка подана у форматі `double`. Наприклад, наведемо код, який прочитує зображення в робочий простір MATLAB у форматі `RGB`:

```
RGB=imread('football.jpg');
```

За допомогою цього коду прочитується індексне зображення з відповідною палітрою `colormap` у робочий простір MATLAB у вигляді двох окремих змінних.

```
[X,map]=imread('trees.tif');
```

У даному прикладі функція `imread` при прочитуванні використовує формат, який вказаний в змісті файла. Крім того, формат можна вказати у функції `imread` як аргумент. Для отримання іншої інформації, наприклад, відносно глибини колірності, див. опис функцій `imread` і `imformats`.

***Інформація про зображення в робочому просторі***. Функція `imread` передає ці зображення в робочий простір. У робочому просторі може відображатися інформація про усі змінні, які були створені в MATLAB упродовж одного сеансу роботи. Функція `imread` повертає ці зображення змінної `I`, яка подається масивом у форматі `uint8` з розмірністю 291x240 елементів. Система MATLAB може відображати також масиви й інших форматів – `uint8, uint16` або `double`.

Існує можливість одержання інформації про всі змінні, які знаходяться в робочому просторі. Для цього використовують команду `whos`. Наприклад:

```
>> RGB = imread('football.jpg');
>> whos
  Name         Size              Bytes  Class      Attributes

  BW         256x256             65536  logical
  RGB        256x320x3          245760  uint8
```

## 2.2. Типи зображень

Пакет Image Processing Toolbox працює з такими типами зображень:

- півтонові зображення (зображення в градаціях сірого);
- подвійні (чорно-білі) зображення;
- кольорові зображення `RGB`.

***Півтонові зображення*** – це матриця, елементи якої наведені у вигляді числових значень класів `uint8 [0,255]` або `uint16 [0,65535]`.

***Бінарні (двійкові) зображення*** в MATLAB є матрицею логічних елементів `0` або `1`. Для перетворення числових масивів у логічні служить функція `logical - B=logical(A)`.

***Повнокольорові зображення*** – такий тип зображень, де кожен піксель описується трьома значеннями червоною, синьою і зеленою складових. Система MATLAB запам'ятовує повнокольорові зображення у вигляді масиву даних із розмірністю `mxnx3`. У цьому масиві зберігається кожна компонента кольору для кожного окремого пікселя.

Графічний файловий формат запам'ятовує повнокольорові зображення як 24-бітові дані, де червоні, зелені та сині компоненти подані вісьмома бітами кожна. У сумі це дає близько 16 мільйонів кольорів. Повнокольорові зображення можуть бути наведені у форматі `uint8, uint16, single` або `double`. Зведемо основні дані різних типів зображень у табл. 2.1.

Таблиця 2.1

| Тип зображення | Опис |
|---|---|
| Бінарне (Binary) | Логічний масив, що містить лише одиниці і нулі, які інтерпретуються як чорний і білий колір відповідно. Крім того, існують так звані дворівневі зображення, пікселі якого містять лише два рівні інтенсивностей, не обов'язково `1` і `0` |
| Індексне (Indexed) | Масив у форматі `logical, uint8, uint16, single` або `double`, значення пікселів якого є індексами з палітри. Палітра є масивом з розмірністю mx3, який поданий у форматі `double`. Також відомі псевдокольорові зображення.<br>*Примітка*. Для масивів, які подані у форматі `single` або `double`, значення знаходяться в діапазоні `[1,p]`. Для масивів, які подані у форматі `logical, uint8` або `uint16`, діапазон значень становить `[0,p-1]` |

Закінчення табл. 2.1.

| Тип зображення | Опис |
|---|---|
| Півтонове (Grayscale) | Масив зображення у форматі `uint8`, `uint16`, `int16`, `single` або `double`. Значення пікселів описують значення інтенсивностей зображення. Вони відомі також як зображення яскравості.<br>***Примітка***. Для зображень у форматі `single` або `double` значення пікселів знаходяться в діапазоні `[0, 1]`. Якщо зображення подані у форматі uint8, то значення пікселів знаходяться в діапазоні `[0,255]`. Якщо ж зображення наведено у форматі `uint16`, то значення пікселів знаходяться в діапазоні `[0, 65535]`. Для формату `int16` значення можуть знаходитися в діапазоні `[-32768, 32767]` |
| Повнокольорове (Truecolor) | Зображення подаються масивом з розміром `mxnx3` у форматі `uint8`, `uint16`, `single` або double. Значення пікселя дорівнює значенню інтенсивності. Ці зображення відомі ще як `RGB` зображення.<br>***Примітка***. Для зображень у форматі `single` або `double` значення пікселів знаходяться в діапазоні `[0,1]`. Якщо зображення подано у форматі `uint8`, то значення пікселів знаходяться в діапазоні `[0,255]`. Якщо ж зображення наведено у форматі `uint16`, то значення пікселів знаходяться в діапазоні `[0, 65535]` |

## 2.3. Просторові координати

У піксельній координатній системі піксель є дискретною областю, яка однозначно визначається парою координат, наприклад `(5,2)`. З цього виходить, що позначення координат у вигляді `(5.3,2.2)` не має сенсу.

Проте в деяких випадках для деяких піксельних систем це може бути виправдано. У системі просторових координат локалізація елементів зображення описується значеннями `x` і в (не `r` і `c` як в піксельній системі координат). На зображенні (рис. 2.1) продемонстровано просторову систему координат, яка використовується для подання зображень. Відзначимо, що змінна `y` збільшується вниз.

Просторова система координат у деяких випадках відповідає піксельній координатній системі. Наприклад, у просторовій і піксельній системах координат центри збігаються для всіх пікселів зображення.

Відзначимо деякі істотні відмінності. У піксельних координатах верхній лівий кут зображення має координати (1,1), а в просторовій системі координат ці індекси дорівнює відповідно `(0.5,0.5)`. Ще одна істотна різниця між ними полягає в тому, що піксельна система координат дискретна, а

просторова система координат безперервна. Проте в більшості випадків, якщо говорять, що піксель у лівому верхньому кутку набуває координати `(1,1)`, то це є просторовою системою координат.

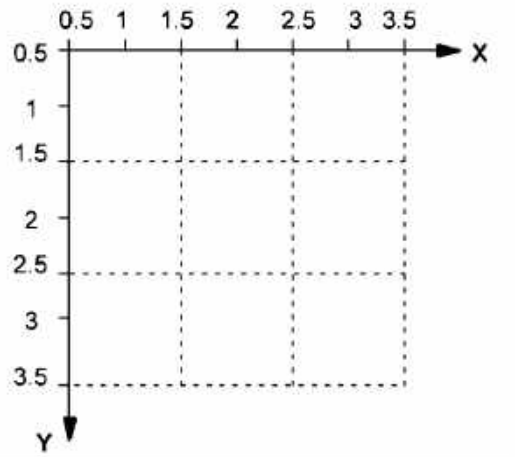



Рис. 2.1. Просторова система координат

Далі, якщо мова йтиме про піксельну систему координат, то використовуватимуться індекси `(r,c)`, а при роботі з просторовою системою координат – індекси `(x,y)`.

***Використання нестандартних просторових координатних систем***. За умовчанням просторова система координат сприймається як піксельна. Наприклад, координати центрального пікселя в `5`-му рядку і `3`-му стовпці мають такі просторові координати: `x=3`, `y=5`. (Запам'ятаємо, що порядок координат зворотний). Деякі функції працюють переважно з просторовими координатами, а інші – з піксельними.

У деяких випадках є необхідність працювати з нестандартними системами просторових координат. Для визначення просторової координатної системи потрібно описати дані зображення `Xdata` і `Ydata`. Ці властивості будуть описані у вигляді двоелементного вектора і задаватимуть діапазон координат зображення, що відображується. За умовчанням для зображення `A` `Xdata` подається діапазоном `[1 size(A,2)]`, а `Ydata` – `[1 size(A,1)]`.

Наприклад, якщо зображення `A` складається з 100 рядків і 200 стовпців, то за умовчанням `Xdata` дорівнює `[1 200]`, а `Ydata` – `[1 100]`. Значеннями цих векторів є координати центрів пікселів. Насправді ж діапазон координат, що відображуються, трохи більше. Якщо `Xdata` дорівнює `[1 200]`, то діапазон уздовж `x` – осі зображення, що дорівнює `[0.5 200.5]`.

Продемонструємо процедуру візуалізації зображення з використанням `XData` і `YData` (рис. 2.2).

```
A = magic(5);
x = [19.5 23.5];
y = [8.0 12.0];
image(A,'XData',x,'YData',y),axis image, colormap(jet(25)
```

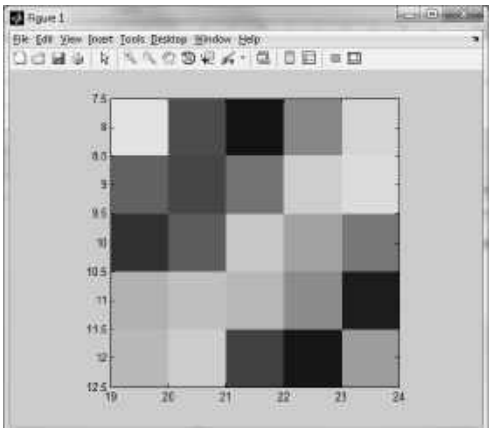


Рис. 2.2. Візуалізація зображення з використанням `Xdata` і `Ydata`

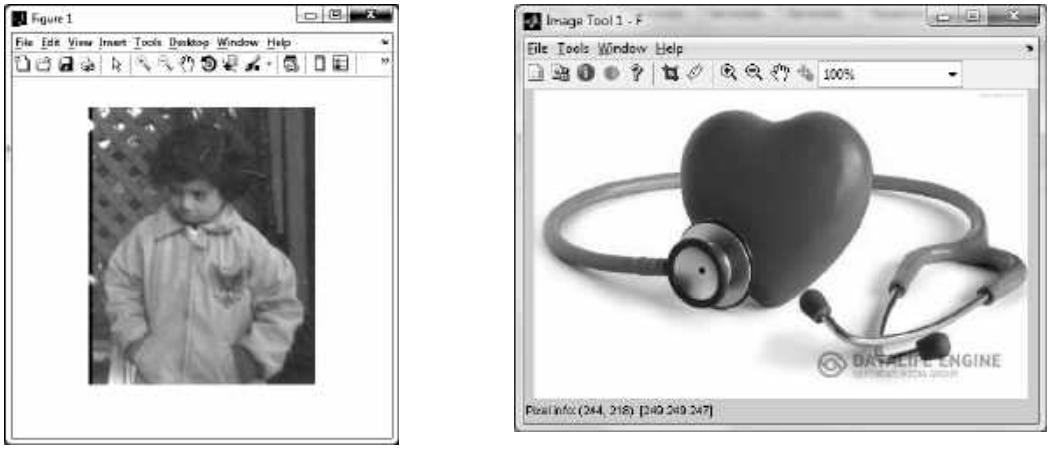
Для детальнішої інформації відносно синтаксису системи просторових координат див. опис функції `imshow`.

## 2.4. Візуалізація зображень

Тепер розглянемо питання візуалізації ліченого зображення. Додаток містить дві функції відображення зображень: `imshow` і `imtool`. Функція `imshow` є фундаментальною функцією відображення зображень. Функція `imtool` запускає інструментарій `Image Tool`, який є інтегрованим середовищем для візуалізації зображень і виконує деякі операції оброблення зображень. Засіб відображення зображень `Image Tool` містить ряд додаткових функцій, які дають можливість надавати інформацію про конкретний піксель або зображення в цілому, підвищувати контраст зображення і т. д. Розглянемо приклад використання функції `imshow` для відображення зображення (див. рис. 2.3,а).

```
imshow(I)
```

У другому прикладі для введення зображення використовують функцію `imtool(F)` (див. рис. 2.3,б).



а б

Рис. 2.3. Функція візуалізації зображень:
а – півтонове зображення `pout.tif` ; б – кольорове зображення `phono.jpg`

***Візуалізація індексних зображень***. Для візуалізації індексного зображення застосовують функції `imshow` або `imtool`, які використовують опис матриці зображення і палітри. Для подання індексного зображення в робочому просторі використовують змінну з назвою `X` для подання матриці індексного зображення і змінну `map` для подання палітри `imshow(X,map)` або `imtool(X,map)`.

Для кожного пікселя функція використовує при відображенні той колір, який відповідає конкретному рядку `map`. Якщо матриця даних зображення наведена у форматі `double`, значення першого відліку відображується першим рядком у `colormap`, значення другого відліку – другим рядком і т. д. Проте якщо дані матриці зображень подані у форматі `uint8` або `uint16`, то значення нульового відліку відображується першим рядком палітри, значення першого відліку – другим рядком і т. д. Це автоматично визначається функціями `imtool` і `imshow`.

Якщо кількість кольорів у палітрі більше, ніж на зображенні, то останні кольори палітри функція ігнорує. Якщо палітра містить менше кольорів, чим присутній на зображенні, функція встановлює всім пікселям значення, які вийшли за діапазон палітри. Наприклад, при візуалізації зображення, яке подано у форматі `uint8` і містить 256 кольорів, використовується палітра, яка містить 16 кольорів. У цьому випадку всі пікселі, значення яких більше 15, відображуватимуться останнім кольором палітри.

***Візуалізація півтонових зображень***. Для відображення півтонових зображень використовують функцію `imshow` або `imtool` з описом назви матриці зображення як аргумент. Далі застосовуватимуться різні імена (`I`) для подання півтонових зображень у робочому просторі

`imshow(I)` або `imtool(I)`

Обидві функції відображують зображення з масштабованими значеннями інтенсивностей, які подані відповідними індексами в палітрі.

Якщо параметр I наведений у форматі `double`, пікселі із значенням `0.0` відображуються як чорні, пікселі із значенням `1.0` – як білі, а пікселі з проміжними значеннями – різними відтінками сірого. Якщо матриця зображення `I` подана у форматі `uint8`, то пікселі із значеннями `255` відображуються білими. Коли матриця зображення `I` подана у форматі `uint16`, то пікселі із значеннями 65535 також відображуються білими.

Півтонові зображення, як і індексні, що використовують палітру `RGB`, у більшості випадків не потребують опису палітри. Система MATLAB відображує півтонові зображення за допомогою півтонової палітрової системи (де `R=G=B`). За умовчанням число відліків сірого в палітрі дорівнює `256` для системи з поданням 24 біти на піксель.

***Візуалізація бінарних зображень***. У системі MATLAB бінарні зображення подаються у форматі `logical`. Бінарні зображення містять пікселі

із значенням лише нуль і одиниця. Пікселі із значенням `0` відображуються як чорні, а пікселі із значенням `1` – як білі.

Для того, щоб додаток інтерпретував зображення як бінарне, необхідно, щоб це зображення було подано у форматі `logical`.

Для візуалізації бінарних зображень використовують одну з функцій – `imshow` або `imtool` з описом матриці зображення як аргумент. Розглянемо приклад прочитування бінарного зображення в робочий простір MATLAB і подальшу візуалізацію зображення. Далі використовуватимемо назву `BW` для подання бінарних зображень у робочому просторі:

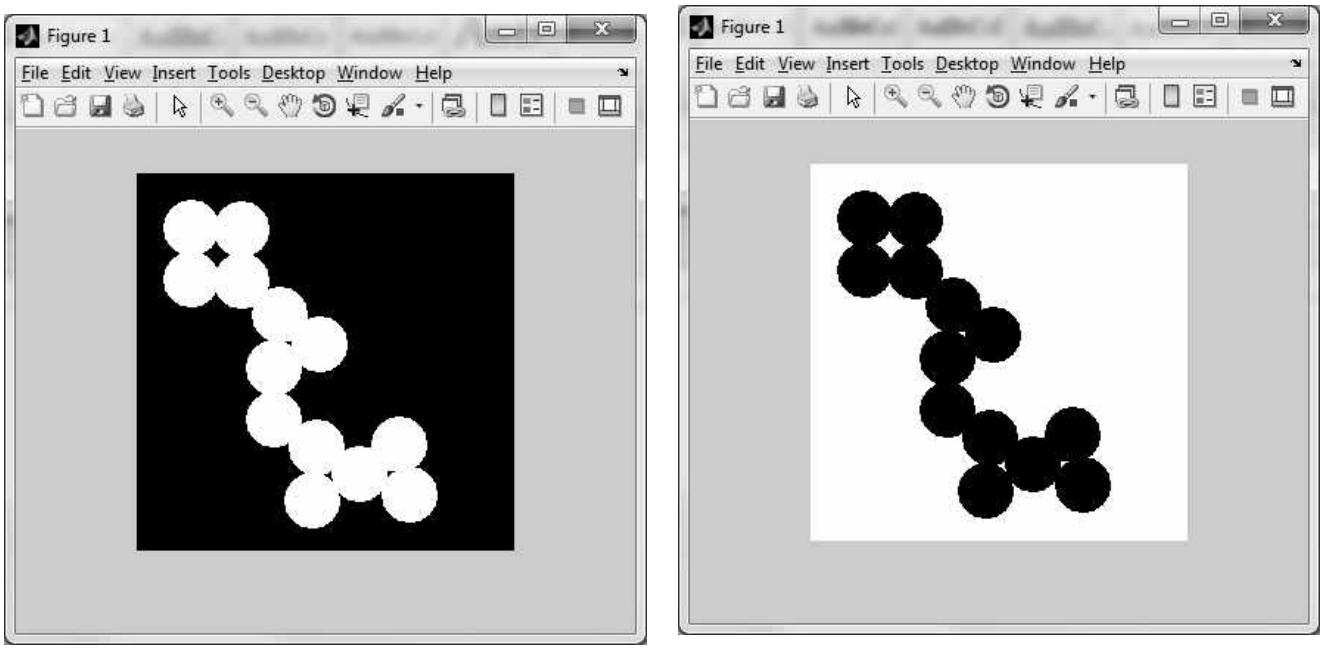
```
BW = imread('circles.png');
```

`imshow(BW)` або `imtool(BW)`

***Зміна кольорів, що відображуються, на бінарному зображенні***. У деяких випадках для зручності бінарні зображення необхідно інвертувати, тобто пікселі із значенням `0` відображувати як білі, а пікселі із значенням `1` – як чорні. Для цього необхідно використовувати оператор NOT `(~)`. Наприклад:

`imshow(~BW)` или `imtool(~BW)`

Приклади візуалізації прямого і інвертованого зображень показані на рис. 2.4.




а б

Рис. 2.4. Бінарні зображення: а – пряме, б – інвертоване

Існує також можливість відображення бінарного зображення з використанням синтаксису палітри індексних зображень. Наприклад, розглянемо команду, в якій задається палітра, в результаті використання якої пікселі з

нульовими значеннями відображуватимуться червоним кольором, а із значенням `1` – синім кольором (рис. 2.5).

`imshow(BW,[1 0 0; 0 0 1])` або `imtool(BW,[1 0 0; 0 0 1])`

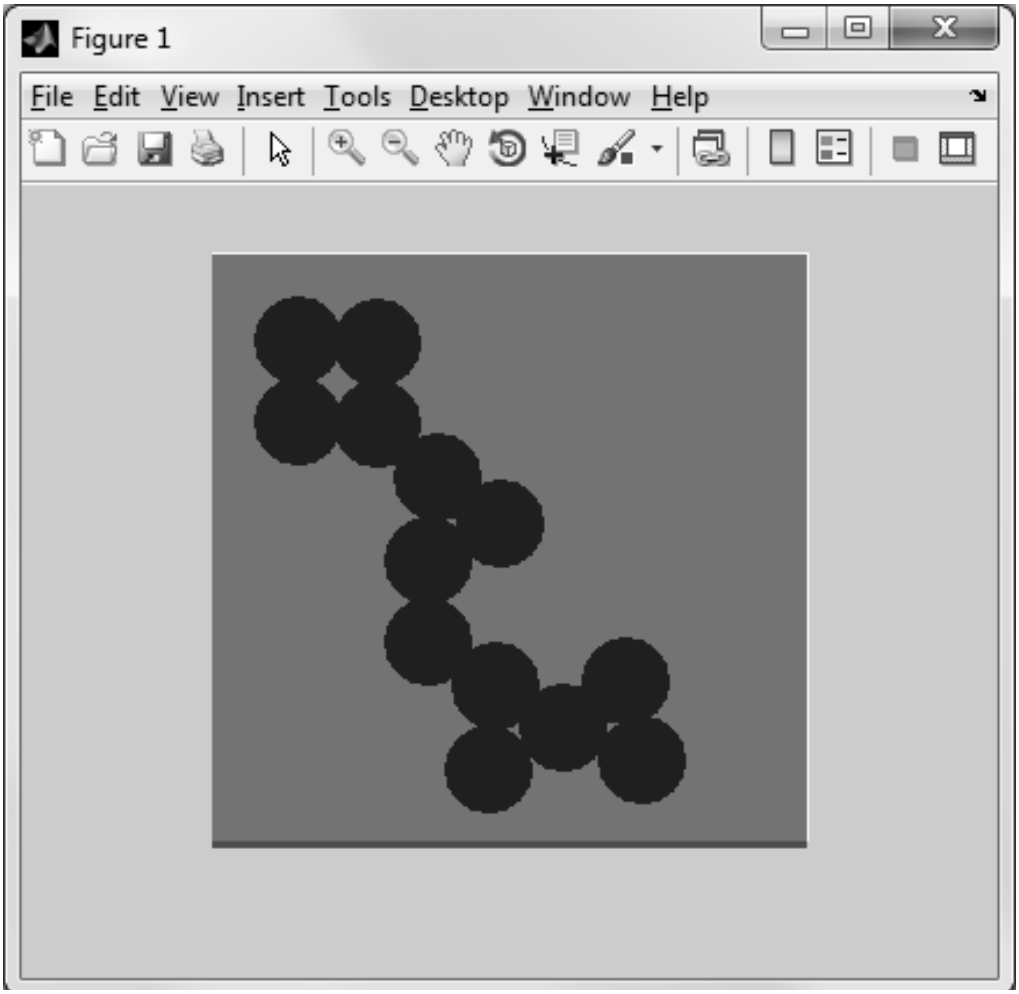



Рис. 2.5. Використання палітри при візуалізації бінарних зображень

***Візуалізація повнокольорових зображень***. Повнокольорові зображення, які також називають RGB зображеннями, подають значення кольорів безпосередньо, без використання палітри. Повнокольорові зображення наводяться тривимірним масивом `mxnx3`. Для кожного пікселя (`r,c`) на зображенні колір подається триплетом (`r,c,1:3`).

При візуалізації повнокольорових зображень використовують функції `imshow` або `imtool` з описом матриці зображення як аргумент. Розглянемо приклад прочитування повнокольорового зображення в робочий простір MATLAB і його подальшу візуалізацію (див. рис. 2.6). Далі назву `RGB` використовуватимемо для наведення повнокольорових зображень у робочому просторі.

```
RGB = imread('C:\Users\krasnov.MEDICA\Desktop\6.jpg');
imshow(RGB)
```

або

```
imtool(RGB)
```

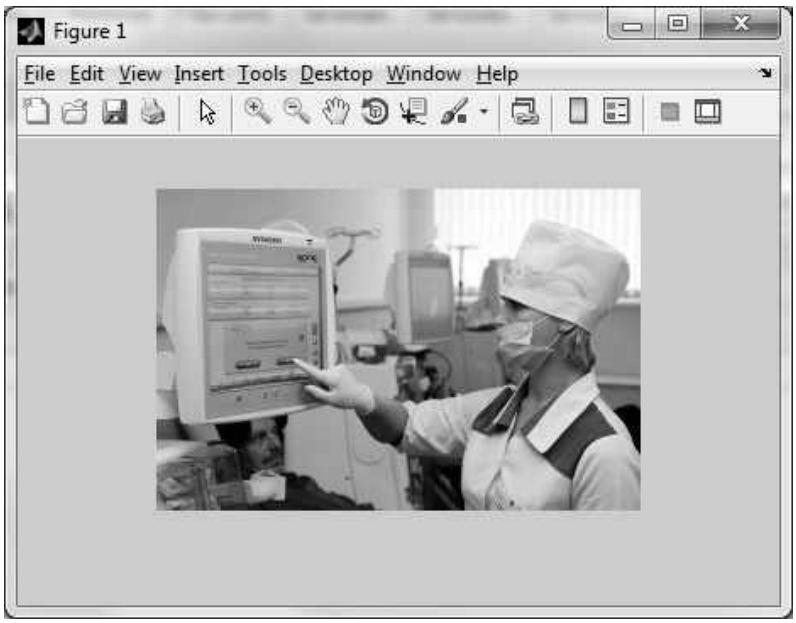



Рис. 2.6. Візуалізація повнокольорового зображення

Система, яка використовує формат 24 біти на піксель, може відображувати повнокольорові зображення прямо, тобто по 8 бітів у форматі червоною, зеленою і синьою складовою відповідно. У системах із меншою кількістю кольорів функція `imshow` відображує зображення за допомогою методу моделювання кольорів (`dithering`).

Якщо при візуалізації кольорового зображення воно відображується як чорно-біле, то необхідно перевірити, чи є це зображення індексним. Для індексних зображень необхідно описати палітру і пов'язати її із зображенням.

## 2.5. Попереднє оброблення зображень

***Перетворення типів зображень***. У практиці оброблення медичних зображень часто виникають завдання перетворення повнокольорових зображень у півтонові. Це може бути, наприклад, для проведення спектрального аналізу або інших завдань. Процедура перетворення виконується за допомогою функції `I=rgb2gray(RGB)`. Покажемо це на прикладі (див. рис. 2.7).

```
>> RGB=imread('C:\Users\krasnov.MEDICA\Desktop\8.jpg');
>> imshow(RGB)
>> I=rgb2gray(RGB);
>> figure,imshow(I)
>> whos
  Name         Size                Bytes  Class    Attributes

  I          190x262             49780  uint8
  RGB        190x262x3          149340  uint8
```

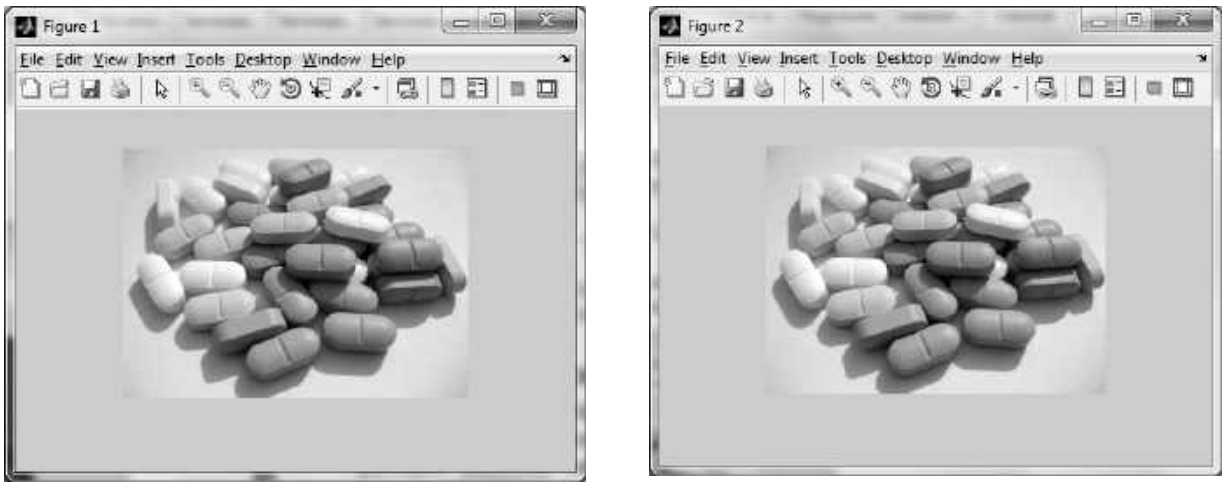
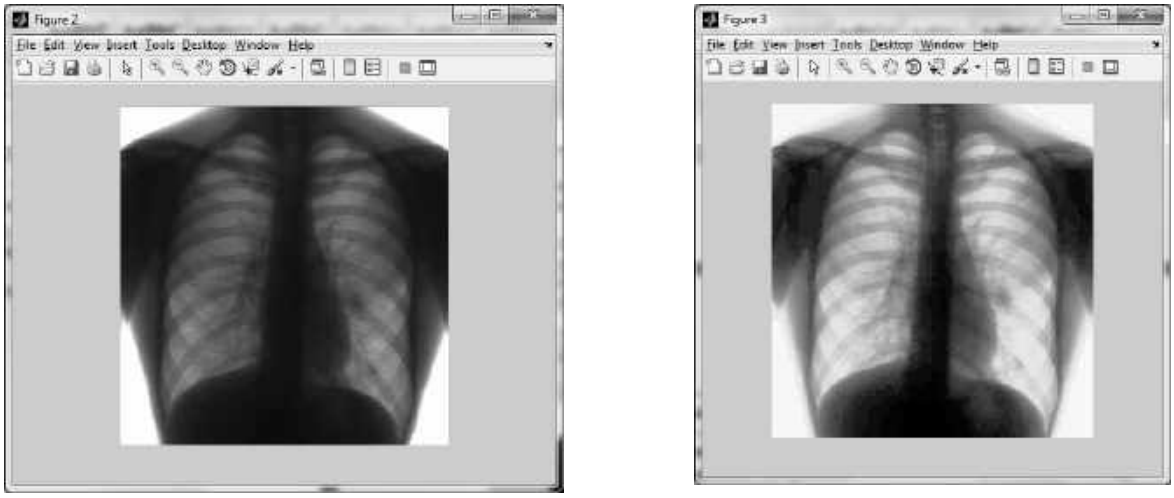



Рис. 2.7. Перетворення повнокольорового зображення в півтонове

За допомогою команди `whos` нескладно відмітити зміни у форматі даних після виконаного перетворення.

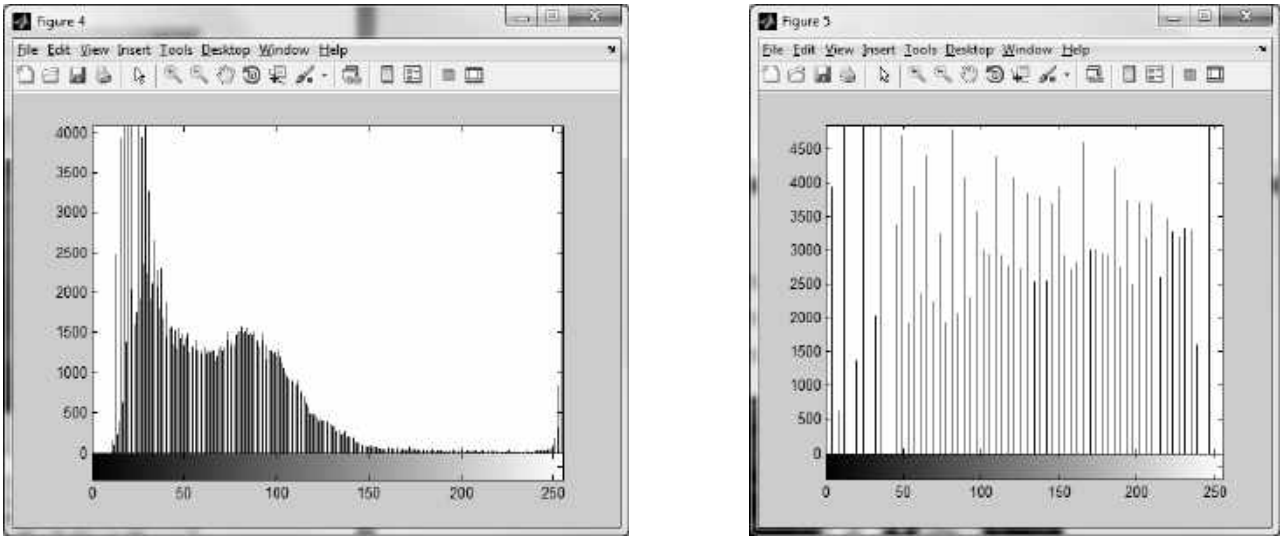
***Поліпшення контрасту оброблюваного зображення***. Функція `histeq` збільшує контраст зображення шляхом розтягування значень інтенсивностей динамічного діапазону. Для детальнішої інформації див. опис функції `histeq`. Відображуватимемо поліпшуване зображення `I3` (рис. 2.8).

```
I=imread('C:\Users\krasnov.MEDICA\Desktop\129.jpg');
imshow(I)
I2=rgb2gray(I);
figure, imshow(I2)
I3=histeq(I2);
figure,imshow(I3)
figure,imhist(I2)
figure,imhist(I3)
```




а б

Рис. 2.8. Поліпшення контрасту зображення: а – початкове, б – контрастоване




а б

Рис. 2.9. Гістограми розподілу яскравості зображень: а – початкового, б – контрастованого

Функція `imhist` створює гістограму перетвореного зображення. Це дозволяє порівнювати гістограми до і після вирівнювання. Відзначимо, що діапазон інтенсивностей вихідного зображення є досить вузьким (рис. 2.9,а). Він не покриває весь діапазон `[0,255]`, що є однією з причин низької контрастності зображення.

### 2.6. Запис зображень у файл на диск

Для запису зображення I у файл на диск використовують функцію `imwrite`. Якщо ця функція містить назву і розширення `'.png'`, то функція `imwrite` записує зображення у форматі `PNG` (Portable Network Graphics ). Крім того, існує можливість записати зображення і в інших форматах, наприклад `tif` або `jpg`.

```
imwrite (I,'pout2.png');
```

Список всіх доступних форматів можна проглянути в описі функції `imwrite`.

## 3. ФІЛЬТРАЦІЯ І ЗАГЛУШЕННЯ ШУМІВ

Одним із найчастіше вирішуваних завдань оброблення зображень є фільтрація або заглушення шумів на зображенні. Цифрові зображення схильні до дії різних типів шумів. Існує декілька основних причин появи шуму, які залежать також від способу формування зображень. Наприклад:

- якщо зображення отримано шляхом сканування фотографічної плівки, то зерна плівки є джерелом шуму. Поява шуму може пояснюватися також пошкодженням самої плівки або вноситься скануючим пристроєм;
- якщо зображення отримано в цифровому форматі, то механізм формування даних (квантування) є джерелом шуму;

- електронна передача даних зображення по каналах зв'язку також може бути джерелом шумів.

Додаток забезпечує декілька доріг повного або часткового усунення шумів на зображеннях. Для усунення різних видів шуму розроблено різні методи. Найбільш ефективними є такі:

- використання лінійної фільтрації;
- застосування медіанної фільтрації;
- використання адаптивної фільтрації.

Для демонстрації ефекту від роботи перерахованих вище методів у додатку існує функція `imnoise`, яка накладає на зображення шуми різних типів. Розглянемо приклади використання цієї функції.

### 3.1. Накладення шумів на зображення

Оскільки зображення є двовимірними функціями, відповідно і шуми на зображенні також мають двовимірний характер. Для зашумлення зображень використовують функцію `imnoise`, що має такий синтаксис:

```
Q = imnoise(I,type)
Q = imnoise(I,type,parameters).
```

Строкова змінна `type` може набувати одного з таких значень:

- "gaussian" – для білого гауссова шуму;
- `'localvar'` – для білого гауссова шуму з нульовим математичним сподіванням і локальною потужністю, залежною від яскравості поточної ділянки зображення;
- `'poisson'` – пуассонівський шум;
- `'salt & pepper'` – для шуму у вигляді чорних і білих пікселів (шум називається – «сіль і перець»).

Крім цього, `Q=imnoise(I,type,parameters)` – додатково задаються параметри відповідного типу шуму. Наведемо деякі приклади:

- `J=imnoise(I,'gaussian',m,v)` – додавання до зображення I білого гауссова шуму з середнім m і дисперсією v. За умовчанням m=0, v=0.01;
- `Q = imnoise(I,'poisson')` – генерує пуассонівський шум з даних замість того, щоб додавати до даних штучний шум;
- `Q = imnoise(I,'salt & pepper',d)`, де параметр `d` – щільність шуму. У результаті генерується приблизно `d*prod(size(I))` пікселів шуму. За умовчанням `d=0.05`.

Зображення `I` може належати класу `uint8` або `uint16`. Результуюче зображення `Q` належить тому ж класу, що і `I`, та має той же розмір.

Для зручності аналізу доцільно візуалізувати початкове і зашумлене зображення. Це спростить оцінювання якості подальшої фільтрації. Покажемо це на прикладі накладення на зображення гауссова шуму (рис. 3.1).

```
I=imread('C:\Users\krasnov.MEDICA\Desktop\MED_IMAGE\
  9.jpg');
Q = imnoise(I,'gaussian',0,0.1);
imshow(I)
title('Вихідне зображення');
figure,imshow(Q)
title('Зашумлене зображення');
```

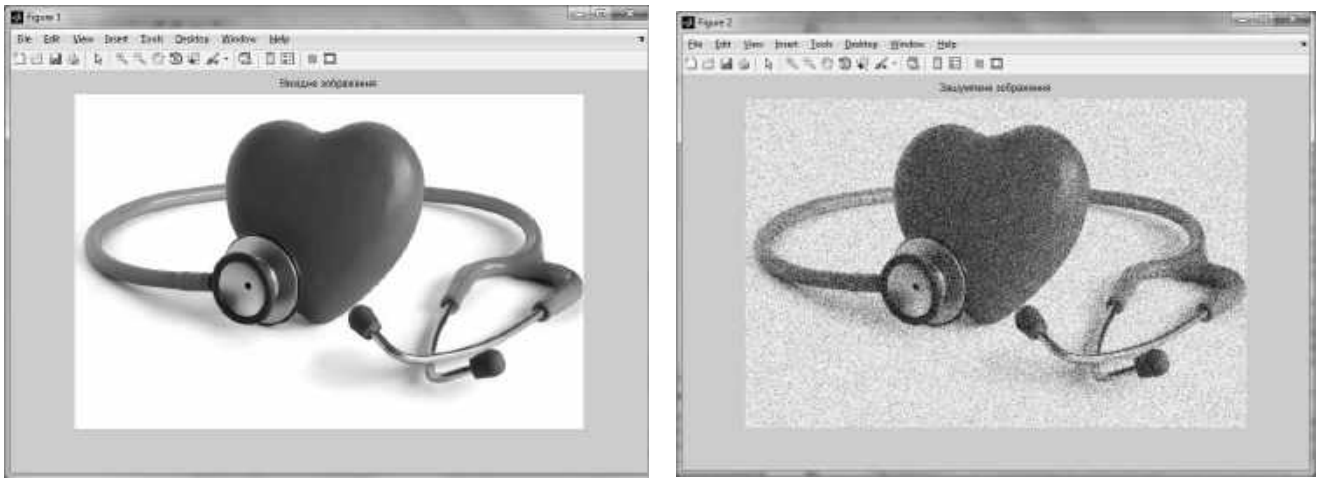


Рис. 3.1. Зображення на фоні нормального (гауссова) білого шуму

### 3.2. Використання лінійної фільтрації для заглушення шумів

Зображення, отримані на виході оптико-електронних перетворювачів, спотворені перешкодами. Це утрудняє як візуальний аналіз зображень людиною-оператором, так і їх автоматичне комп'ютерне оброблення. При обробленні зображень перешкодами є і деякі області самого зображення. Наприклад, при аналізі об'єктів на складному фоні фон теж є перешкодою.

При цифровому обробленні зображень необхідно усувати геометричні спотворення зображень, пригнічувати шуми різної природи, виробляти апертурну корекцію. Ослаблення дії перешкод досягається фільтрацією.

Фільтрація зображень здійснюється в просторовій і частотній областях. При просторовій фільтрації зображень перетворення виконується безпосередньо над значеннями відліків зображення. Результатом фільтрації є оцінювання корисного сигналу зображення. Зображення являє собою двовимірну функцію просторових координат, що змінюється повільніше, ніж двовимірна функція, яка описує перешкоду. Тому при оцінюванні корисного сигналу в кожній точці кадру розглядають межу цієї точки (деяка безліч сусідніх з нею точок), використовуючи загальні характеристики сигналу в цій межі. В інших випадках ознакою корисного сигналу є різкі перепади яскравості. Проте, як правило, частота цих перепадів є відносно невеликою, так що на значних проміжках між ними сигнал або постійний, або змінюється повільно. І в цьому випадку властивості сигналу виявляються

при спостереженні його не лише в окремій точці, але і при аналізі її межі. Відзначимо, що поняття межі є досить умовним.

Таким чином, фільтрація грунтується на використанні як даних поточної точки, так і її межі.

Лінійну фільтрацію можна використовувати для видалення шумів певного типу. Для цього придатними є такі фільтри, що усереднюють, і фільтр Гаусса. Наприклад, фільтр, що усереднює, використовується для видалення зернистості на зображеннях. Оскільки значення інтенсивності кожного пікселя дорівнює середній інтенсивності пікселів межі, то це приводить до заглушення зернистості.

У практиці цифрового оброблення зображень широко використовують маскову фільтрацію. Її лінійний різновид є одним із варіантів двовимірної фільтрації з кінцевою імпульсною характеристикою (КІХ) фільтра. Як маску використовують безліч вагових коефіцієнтів, заданих в усіх точках межі тих, що симетрично оточують поточну точку кадру.

Поширеним виглядом межі, часто вживаним на практиці, є квадрат 3х3 з поточним елементом у центрі. Застосовують різні маски, одним з евристичних варіантів яких є рівномірна маска, всі дев'ять вагових коефіцієнтів якої дорівнюють 1/9. Такий вибір коефіцієнтів відповідає умові збереження середньої яскравості, унаслідок чого вихідний сигнал виявляється вписаним у діапазон вхідного сигналу.

Використання процедур фільтрації приводить до істотного зниження рівня шуму в зображенні.

Просторова фільтрація виконується як операція двовимірної згортки імпульсної характеристики фільтра $\boldsymbol{h}$ із зображенням $\boldsymbol{f(x,y)}$. Вхідним сигналом фільтра й ІПХ є двовимірні функції $\boldsymbol{x_{ij}}$, $\boldsymbol{y_{ij}}$ і $\boldsymbol{h_{kl}}$ відповідно, і дискретна згортка також має двовимірний характер:

$$\boldsymbol{y_{ij} = \sum_{k}\sum_{l} x_{i-k,j-l} h_{kl}} .$$

Точка зображення на виході двовимірного лінійного фільтра є сумою (середнє значення) всіх сусідніх точок фільтрованого зображення, узятих з вагами, визначуваними виглядом ІПХ фільтра. Така процедура ще називається віконною фільтрацією, де вікно фільтра – це його ІПХ. Всі елементи фільтрованого зображення, що потрапили у вікно, усереднюються і тим самим формують вихідний сигнал фільтра.

Процедура фільтрації для всього зображення – це послідовне виконання фільтрації в кожній точці. Вікно фільтра переміщається по всіх елементах фільтрованого зображення і в кожному своєму положенні формує усереднене значення яскравості по всіх точках, що потрапили у вікно (рис. 3.2).

Наведемо приклади масок, які використовують для заглушення шумів різного характеру.

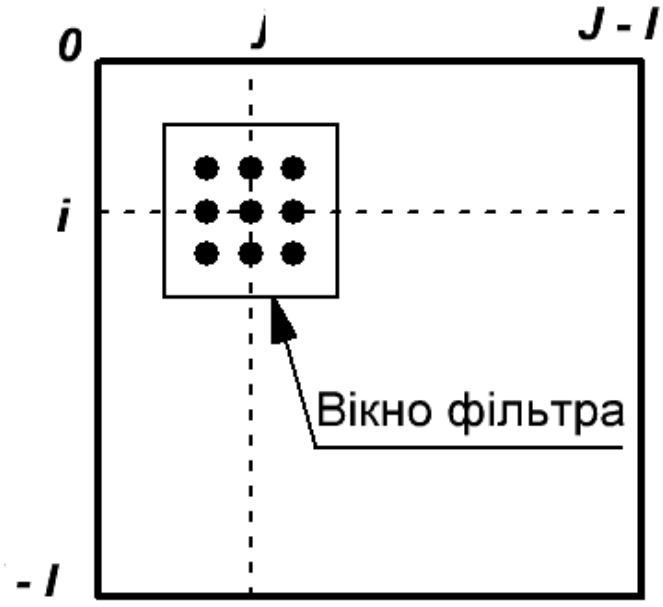



Рис. 3.2. Вікно фільтра розміром 3x3

***Фільтри НЧ***. Для зменшення шумів широко застосовують НЧ фільтри, оскільки шум є ВЧ сигналом. Зокрема, для НЧ фільтрації використовують усереднювання сигналу в масці, наприклад, при `n=m=3`:

$$H = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad H = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad H = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}.$$

***Фільтри ВЧ***. Якщо зображення має нечіткі межі (низьку різкість), то його можна профільтрувати ВЧ фільтром, який підкреслить перепади яскравості на зображенні і зробить його чіткішим. Для ВЧ фільтрації зазвичай використовують такі типові маски:

$$H = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}, \quad H = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}, \quad H = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix}.$$

Нормування є необхідним для того, щоб привести значення відгуку фільтра до діапазону вхідних даних. Нормувальний коефіцієнт визначають з умови рівності одиниці суми всіх коефіцієнтів ІПХ фільтра.

Процедура лінійної фільтрації зображення в MATLAB виконується з використанням команди

```
imfilter(I,h),
```

де параметр h визначає вигляд вікна фільтра. Наприклад, якщо фільтр має прямокутне вікно розміром `3x3` точки з однаковими значеннями $h_{kl}$ у межах вікна, то

```
h = ones(3,3)/9;
```

Визначити довільну маску фільтра у вигляді матриці можна таким чином:

```
>> h=[1, 2, 3; 3, 4, 5; 5, 7, 5]

                    h    = 1     2     3

                           3     4     5

                           5     7     5
```

Наведемо приклади лінійної фільтрації зображень, схильних до впливу шумів з різними характеристиками.

***Приклад 1***

```
close all
I=imread('C:\Users\krasnov.MEDICA\Desktop\129.jpg');
imshow(I)
I1=rgb2gray(I);
figure, imshow(I1)
I2=histeq(I1);
figure,imshow(I2)
title('Вихідне зображення');
Q=imnoise(I2,'gaussian',0,0.1);
figure,imshow(Q)
title('Зашумлене зображення');
h=ones(3,3)/9;
J=imfilter(Q,h);
figure, imshow(J)
title('Зображення на виході фільтра НЧ-1');
h1=[1, 2, 1; 2, 4, 2; 1, 2, 1]/16;
J1=imfilter(Q,h1);
figure, imshow(J1)
title('Зображення на виході фільтра НЧ-2');
h2=[0, -1, 0; -1, 5, -1; 0, -1, 0];
J2=imfilter(Q,h2);
figure,imshow(J2)
title('Зображення на виході фільтра ВЧ-1');
h3=[-1, -1,-1; -1, 9, -1; -1, -1, -1];
J3=imfilter(Q,h3);
figure, imshow(J3)
title('Зображення на виході фільтра ВЧ-2');
```

У цьому прикладі використано фронтальний рентгенівський знімок грудної клітки (див. рис. 3.3,а), замаскований гауссовими шумами (рис. 3.3,б). На рис. 3.3,в і г показано результати лінійної НЧ фільтрації з рівно-

мірною і ваговою масками, що заглушують шум, а на рис. 3.3,д і е результати ВЧ фільтрації з різними масками, що заглушують шум.

Якість фільтрації оцінюватимемо візуально. Зазвичай це найбільш продуктивний спосіб оцінювання при аналізі медичних зображень. Наведений приклад наочно показує, що лінійна НЧ фільтрація не дозволяє повною мірою відфільтрувати гауссові шуми, а ВЧ фільтрація лише погіршує якість зображення, оскільки частково видаляє регулярні низькочастотні компоненти і практично не впливає на високочастотні компоненти.

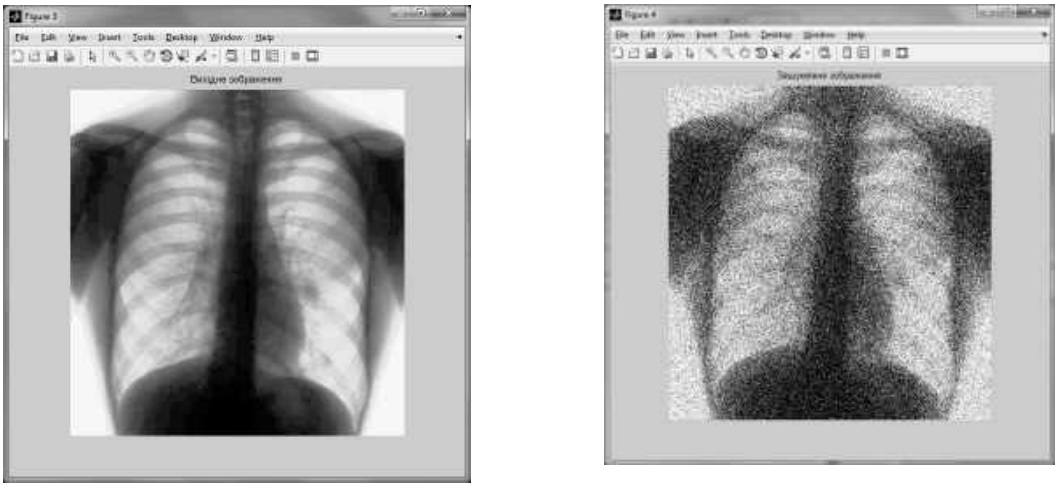
***Приклад 2.*** Змінимо характер шумів у раніше наведеному прикладі. Хай вони мають характер імпульсних перешкод. При їх дії на зображенні спостерігаються білі та чорні точки. Для цього використовуємо команду

```
Q = imnoise(I2,'salt & pepper',0.02);
```

Результати моделювання показані на рис. 3.4. Добре видно, що ефективність лінійної фільтрації помітно стала кращою, особливо при використанні алгоритмів НЧ фільтрації (див. рис. 3.4,в і г).

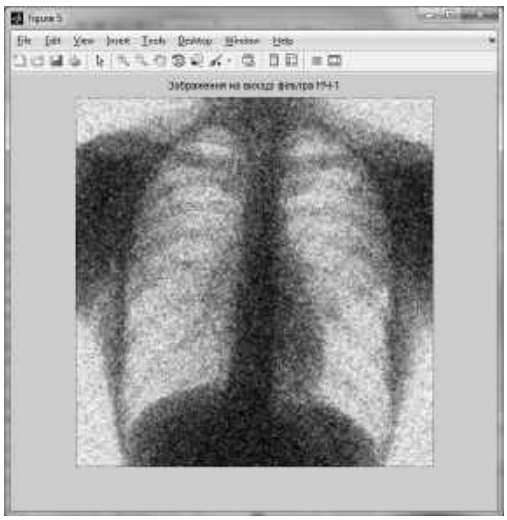
Можна відмітити, що методи лінійної фільтрації не вирішують проблему видалення шумів повною мірою. Це відбувається з ряду причин.

Всі лінійні алгоритми фільтрації приводять до згладжування різких перепадів яскравості зображень, що пройшли оброблення. Цей недолік, особливо істотний, якщо споживачем інформації є людина, принципово не може бути виключений в рамках лінійного оброблення. Справа в тому, що лінійні процедури є оптимальними при розподілі гауссівських сигналів, перешкод і спостережуваних даних. Реальні зображення зазвичай не підпорядковуються цьому розподілу вірогідності. Причому одна з основних причин цього є в наявності на зображеннях всіляких меж, перепадів яскравості, переходів від однієї текстури до іншої і т. п. Піддававшись локальному опису Гаусса в обмежених ділянках, багато реальних зображень у зв'язку з цим погано подаються як глобально гауссові об'єкти. Саме це і служить причиною поганої передачі меж при лінійній фільтрації.

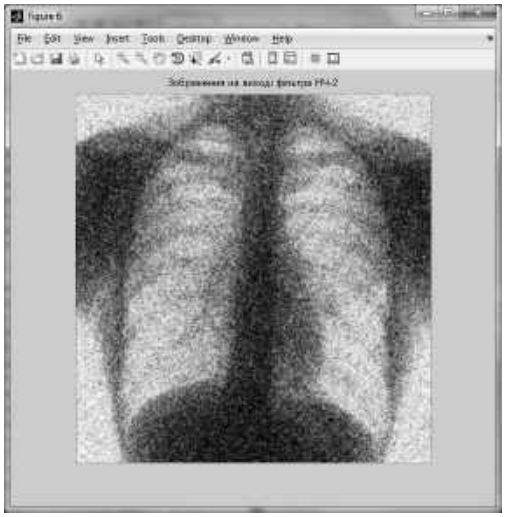


а б

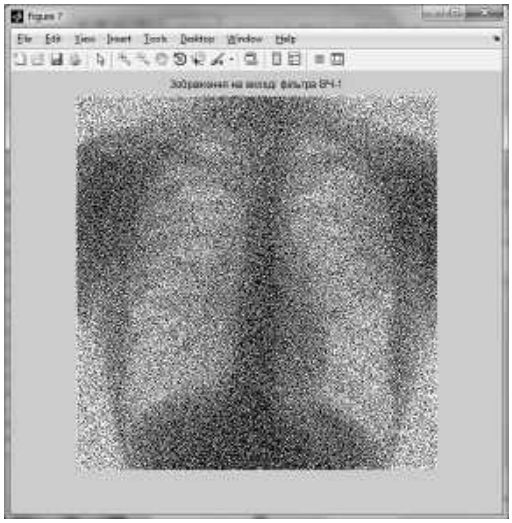
Рис. 3.3. Лінійна фільтрація гауссових шумів при використанні різних заглушуючих масок



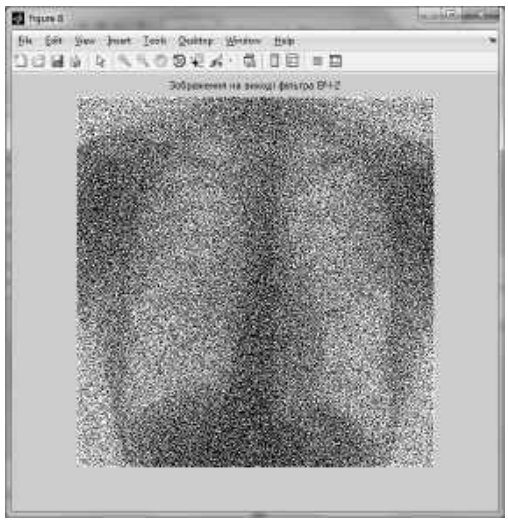
в



г



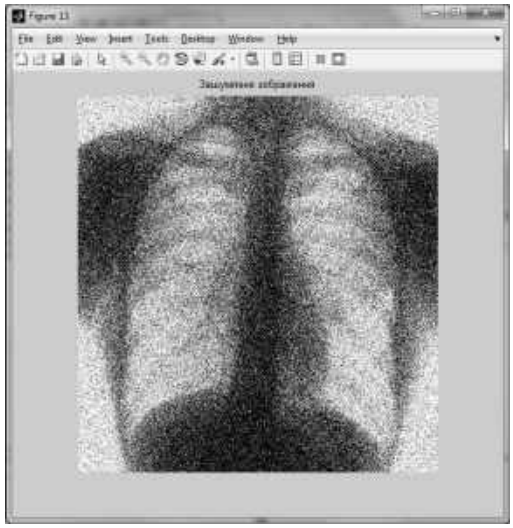
д



е

Рис. 3.3. Закінчення

а

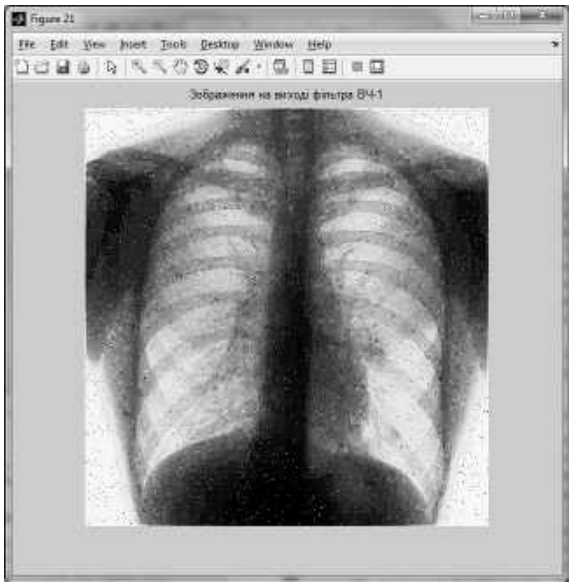


б

Рис. 3.4. Лінійна фільтрація імпульсних шумів


в


г




д


е

Рис. 3.4. Закінчення

## 3.3. Медіанна фільтрація зображень

Ефективність усунення імпульсних шумів істотно підвищується при використанні медіанної фільтрації, запропонованої Дж. Тьюкі в 1971 р. Відзначимо, що медіанна фільтрація є евристичним методом оброблення, її алгоритм не є математичним рішенням строго сформульованої задачі. Тому оцінимо ефективність оброблення зображень на її основі і зіставимо її з іншими методами.

При використанні медіанного фільтра (МФ) відбувається послідовне оброблення кожної точки кадру, внаслідок чого утворюється послідовність оцінок. Оброблення в різних точках незалежне (цим МФ схожий на масковий фільтр), але з метою його прискорення доцільно алгоритмічно на кожному кроці використовувати раніше виконані обчислення.

При медіанній фільтрації використовують двовимірне вікно (апертура фільтра), що зазвичай має центральну симетрію, при цьому його центр розташовується в поточній точці фільтрації. На рис. 3.5 показано два приклади найбільш часто вживаних варіантів вікон у вигляді хреста або квадрата. Розміри апертури належать до параметрів, що оптимізуються в процесі аналізу ефективності алгоритму. Відмінності зображення, що виявилися в межах вікна, утворюють робочу вибірку поточного кроку.

Двовимірний характер вікна дозволяє виконувати, по суті, двовимірну фільтрацію, оскільки для утворення оцінки залучаються дані як з поточного рядка і стовпця, так і з сусідніх. Позначимо робочу вибірку у вигляді одновимірного масиву $\boldsymbol{Y} = \{\boldsymbol{y_1}, \boldsymbol{y_2}, \dots, \boldsymbol{y_n}\}$; число його елементів дорівнює розміру вікна, а їх розташування є довільним. Зазвичай застосовують вікна з непарним числом точок n (це автоматично забезпечується при центральній симетрії апертури і при входженні самої центральної точки в її склад). Якщо упорядкувати послідовність $\{\boldsymbol{y_i}, \boldsymbol{i} = \overline{\boldsymbol{1,n}}\}$ за збільшенням, то її медіаною буде той елемент вибірки, який займає центральне положення в цій впорядкованій послідовності. Отримане таким чином число і є продуктом фільтрації для поточної точки кадру. Зрозуміло, що результат такого оброблення насправді не залежить від того, в якій послідовності подані елементи зображення в робочій вибірці $\boldsymbol{Y}$.

Розглянемо далі процедуру медіанної фільтрації в Image Processing Toolbox. Її виконують з використанням команд `Medfilt2` або `Xd=medfilt2(Xs, 'indexed'...)`.

Опишемо синтаксис цих функцій:

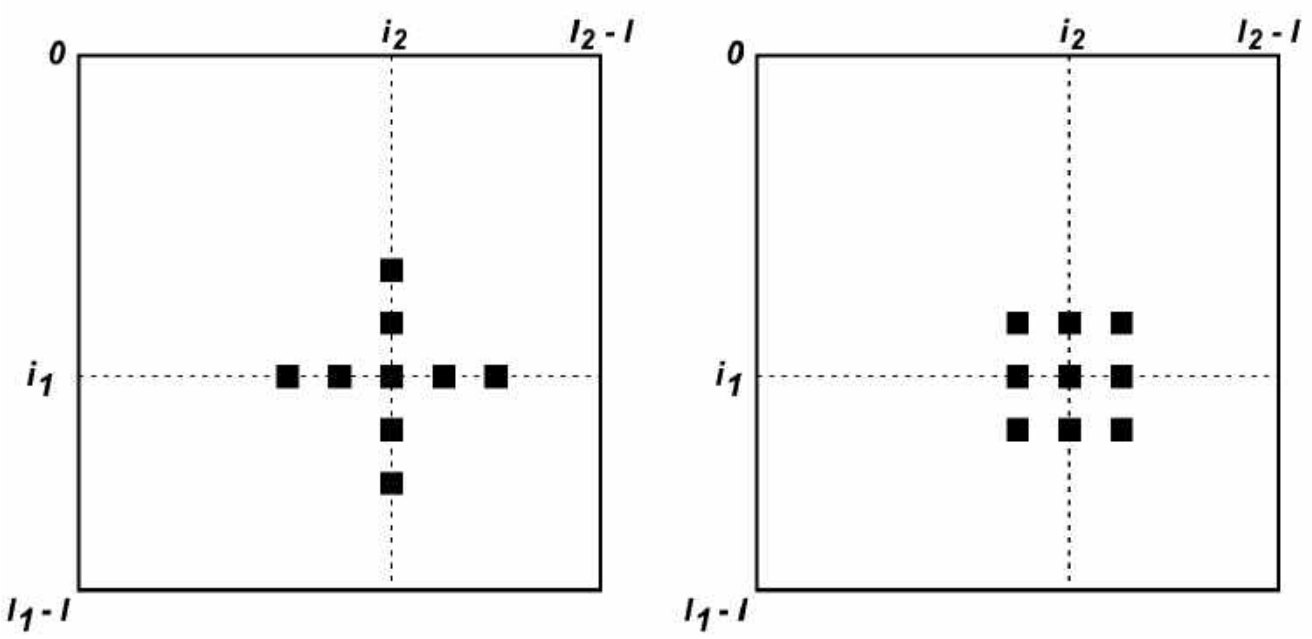
```
D=medfilt2(S,[m n])
Xd=medfilt2(Xs, 'indexed',...)
```

Функція `D=medfilt2(S[m n])` створює півтонове зображення `D`, кожен піксель якого формується таким чином. Пікселі вихідного півтонового зображення `S`, відповідні всім елементам маски фільтра розміром `mxn`, складають упорядковану послідовність `A`. Пікселю `D(r,c)`, де `r` і `c` – координати поточного положення центрального елемента маски, привласнюється значення медіани послідовності `A`. Операцію застосовують нерекурсивною для всіх положень маски.

Медіаною впорядкованої послідовності `A(i)`, де `i=1...N`, називається величина `A((N+1)/2)`, якщо `N` – непарне, і `(A(N/2)+A((N+2)/2))/2`, якщо N – парне.

Для того, щоб розміри зображень `S` і `D` були однаковими, при проведенні обчислень зображення `S` тимчасово доповнюється необхідною кількістю рядків і стовпців нульових пікселів. Формат подання даних результуючого зображення `D` збігається з форматом вихідного зображення `S`.

Якщо вектор `[m n]` при виклику функції `D = medfilt2(S)` не заданий, то як маску фільтра використовують маску розміром `3x3`.



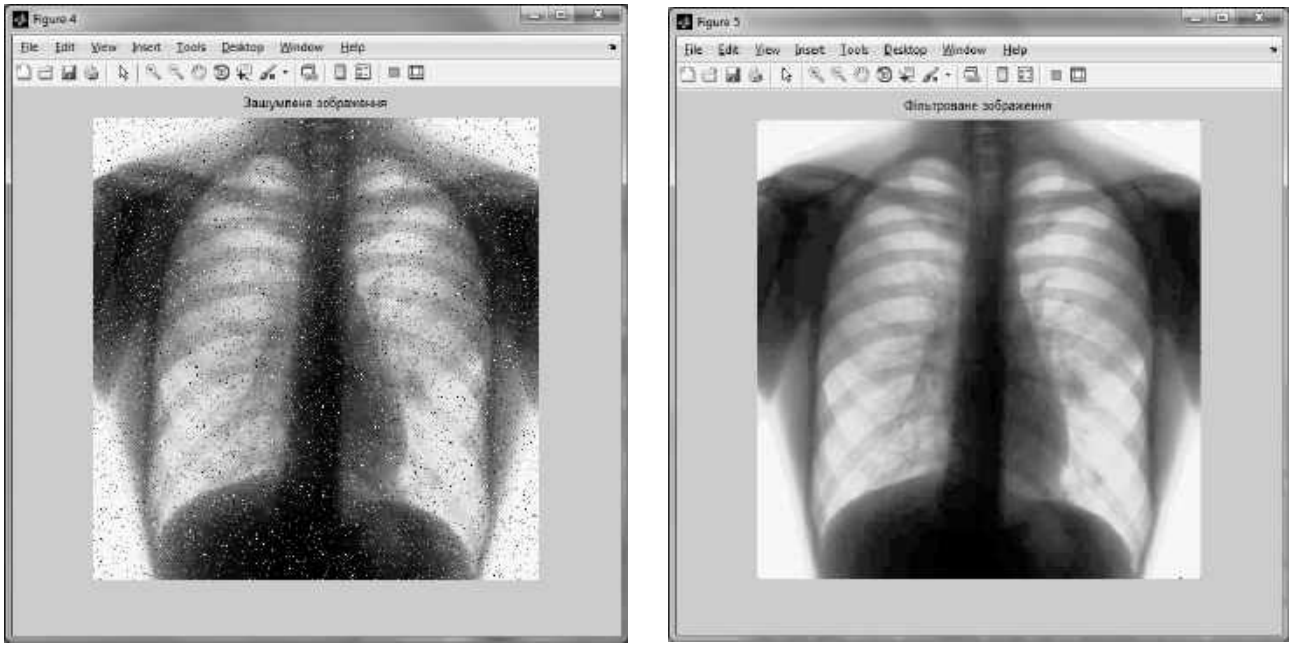

а б

Рис. 3.5. Приклади вікон при медіанній фільтрації

Функція `Xd=medfilt2(Xs,'indexed'...)` аналогічна розглянутій вище. Вона призначена для оброблення палітрових зображень. При проведенні обчислень вихідне зображення тимчасово доповнюється або одиницями при форматі подання даних `Xs – double`, або нулями при форматі подання даних `Xs – uint8`.

Наведемо приклад використання медіанної фільтрації для усунення імпульсного шуму. На рис. 3.6 показано результат медіанної фільтрації з маскою `3x3` зображення, використаного раніше в прикладах з лінійною фільтрацією.

```
close all
I=imread('C:\Users\krasnov.MEDICA\Desktop\129.jpg');
imshow(I)
I1=rgb2gray(I);
figure, imshow(I1)
I2=histeq(I1);
figure,imshow(I2)
title('Вихідне зображення');
Q=imnoise(I2,'salt & pepper',0.05);
figure,imshow(Q)
title('Зашумлене зображення');
K_med=medfilt2(Q);
figure,imshow(K_med)
title('Фільтроване зображення');
```




а б

Рис. 3.6. Приклади зашумленого (а) і відфільтрованого медіанним фільтром (б) зображень

У цьому прикладі завдання заглушення імпульсних шумів удалося вирішити повністю. Це вказує на кращу ефективність медіанної фільтрації порівняно з лінійною.

### 3.4. Адаптивна вінерівська фільтрація

Реальні зображення зазвичай мають неоднорідну структуру. Вони містять велике число фрагментів різних розмірів і з різною яскравістю. Шуми на зображенні також мають нестаціонарний характер: одні частини зображення зашумлені більше, а інші – менше.

Тому при фільтрації зображень дуже часто використовують адаптивні методи. Для ефективної фільтрації такого зображення фільтр має адаптивно підстроювати свої характеристики під характеристики поточного фрагмента зображення і шуму на цьому фрагменті. Прикладом такої фільтрації є адаптивна вінерівська фільтрація.

Адаптивна вінерівська фільтрація в MATLAB виробляється за допомогою функції Wiener2. Її синтаксис має такий вигляд:

```
Id=wiener2(ls,[m n],noise)
[Id,noise]=wiener2(ls,[m n])
```

Функція `Id=wiener2(Is,[m n],noise)` формує півтонове зображення `Id`, яке є результатом адаптивної фільтрації Вінера вихідного півтонового зображення `Is`. Параметри `m` і `n` задають розміри ковзаючого вікна, в межах якого оцінюються середнє і середньоквадратичне відхилення значень яскравості. Якщо при виклику функції параметри `m` і `n` опущені, то розмір вікна встановлюється таким, що дорівнює `3x3`.

Параметр `noise` встановлює потужність гауссова білого шуму, яким пошкоджено зображення. Цей параметр має бути визначений з яких-небудь апріорних відомостей про зображення. Якщо це зробити не вдається, то при виклику функції `wiener2` параметр `noise` можна опустити. В цьому випадку потужність шуму оцінюватиметься автоматично.

Функція `[Id,noise]=wiener2(Is[m n])` працює аналогічно описаній вище і додатково повертає оцінку потужності гауссова білого шуму.

Формати подання даних вихідного і результуючого зображень збігаються.

Алгоритм: функція `wiener2` використовує алгоритм адаптивної вінерівської фільтрації для заглушення адитивного гауссова білого шуму. Цей алгоритм оснований на статистичних оцінках фрагментів зображення в межах ковзаючого вікна розміром `nxm` пікселів. Для всіх положень ковзаючого вікна з центральним пікселем і координатами `(r,c)` обчислюються:

$$\mu = \frac{1}{mn}\sum_{r}\sum_{c\in\eta} I_s(r,c)$$

– середнє значення яскравості;

$$\sigma^2 = \frac{1}{mn}\sum_{r}\sum_{c\in\eta} I^2{}_s(r,c)$$

– дисперсія.

Таку формулу застосовують нерекурсивною для всіх положень ковзаючого вікна. Якщо потужність гауссова білого шуму не задана, то вона оцінюється як середнє зі всіх точок.

```
K_gaus = wiener2(J_g,[m n],noise)
[K_gaus,noise] = wiener2(J_g,[m n])
```

Команда `[K_gaus,noise]=wiener2(J_g[m n])` дозволяє обчислити потужність шуму в межі матриці `mxn` пікселів. Команда `K_gaus = = wiener2(J_g,[m n],noise)` реалізує власне вінерівську фільтрацію. Наведемо приклад вінерівської фільтрації на фоні гауссових шумів (рис. 3.7).

```
close all
I=imread('C:\Users\krasnov.MEDICA\Desktop\129.jpg');
imshow(I)
I1=rgb2gray(I);
figure, imshow(I1)
I2=histeq(I1);
figure,imshow(I2)
title('Вихідне зображення');
```

```
J_g=imnoise(I2,'gaussian',0,0.005);
K_gaus=wiener2(J_g,[5 5]);
figure,imshow(J_g)
title('Зашумлене зображення (гауссова перешкода))');
figure,imshow(K_gaus)
title('Відфільтроване зображення (гауссова перешкода)');
```

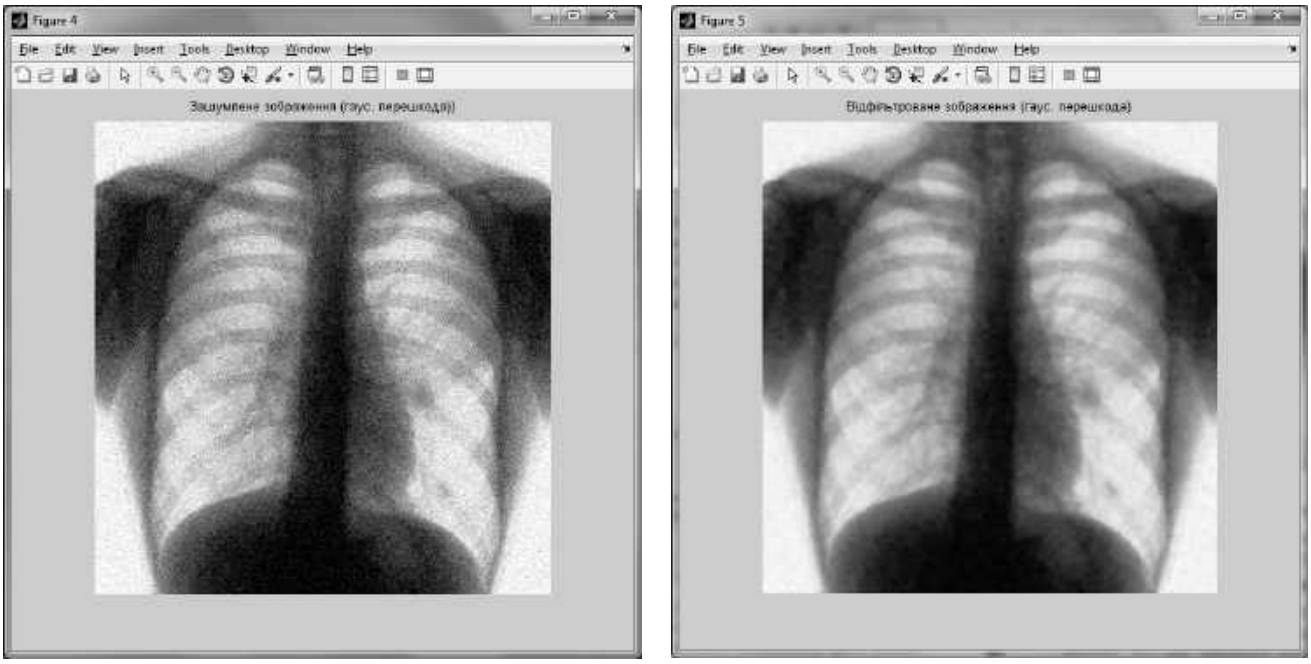



Рис. 3.7. Вінерівська фільтрація зображення на фоні гауссових шумів

Цей приклад показав абсолютно очевидну перевагу алгоритмів вінерівської фільтрації порівняно з лінійними.

### 3.5. Використання спеціальних фільтрів

У бібліотеці MATLAB є функція `fspecial, що дає` можливість формувати фільтри з деякими заздалегідь визначеними властивостями. Розглянемо методи завдання маски зумовленого типу за допомогою функції `fspecial` та її синтаксис.

```
h=fspecial(type,P1,P2)
```

Функція `h=fspecial(type,P1,P2)` повертає маску `h` зумовленого двовимірного лінійного фільтра, що задається рядком `type`. Маска `h` призначена для передачі у функції `filter2` або `conv2`, що виконують двовимірну лінійну фільтрацію. Залежно від типу фільтра для нього можуть бути визначені один або два додаткові параметри `P1`, `P2`. Нижче будуть розглянуті можливі варіанти функції `fspecial`.

Функція `h=fspecial('gausian',n,sigma)` повертає маску `h` фільтра нижніх частот Гаусса. Розмір маски визначає параметр `n`. Якщо `n` – двокомпонентний вектор, то розмір маски `n(1)xn(2)`. Якщо `n` скаляр, то

розмір маски nxn. Параметр sigma задає середньоквадратичне відхилення розподілу Гаусса, який використовують при формуванні маски h. Якщо при виклику функції параметри n і sigma опущені, то розмір маски встановлюється таким, що дорівнює 3x3, а середньоквадратичне відхилення 0.5.

Функція h=fspecial('sobel') повертає маску фільтра Собеля для виділення горизонтальних меж:

$$\boldsymbol{h} = \begin{bmatrix} \mathbf{1} & \mathbf{2} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-2} & \mathbf{-1} \end{bmatrix}.$$

Для виділення вертикальних меж необхідно транспонувати дану маску h.

Функція h=fspecial('prewitt') повертає маску фільтра Превіта для виділення горизонтальних меж:

$$\boldsymbol{h} = \begin{bmatrix} \mathbf{1} & \mathbf{1} & \mathbf{1} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{-1} & \mathbf{-1} & \mathbf{-1} \end{bmatrix}.$$

Для виділення вертикальних меж потрібно транспонувати маску h.

Функція h=fspecial('laplasian',a) повертає маску h ВЧ фільтру Лапласа. Розмір маски – 3x3. Параметр a управляє співвідношенням між центральним і граничними елементами маски. Цей параметр має встановлюватися в діапазоні [0,1]. За умовчанням a=0.2.

Функція h=fspecial('log',n,sigma) повертає маску h фільтра, аналогічного послідовному використанню фільтрів Гаусса і Лапласа, так званого лапласіана-гауссіана. Розмір маски визначає параметр n. Якщо n – двокомпонентний вектор, то розмір маски n(1)xn(2). Якщо n – скаляр, то розмір маски nxn. Параметр sigma задає середньоквадратичне відхилення Гаусса, яке використовується при формуванні маски h. Якщо при виклику функції параметри n і sigma опущені, то розмір маски встановлюється таким, що дорівнює 5x5, а середньоквадратичне відхилення 0.5 використовується при формуванні маски h.

Функція h=fspecial('average',n) повертає маску h, що усереднює НЧ фільтра. Розмір маски визначає параметр n. Якщо n – двокомпонентний вектор, то розмір маски n(1)xn(2). Якщо n – скаляр, то розмір маски nxn. Якщо при виклику функції параметр n опущений, то розмір маски встановлюється таким, що дорівнює 3x3.

Функція h=fspecial('unsharp',a) повертає маску h фільтра, що підвищує різкість зображення. Розмір маски 3x3. Параметр a управляє співвідношенням між центральним і граничними елементами маски. Цей параметр має встановлюватися в діапазоні [0,1]. За умовчанням a=0.2.

***Алгоритми роботи спеціальних фільтрів***. Усереднюючий фільтр відноситься до фільтрів нижніх частот. Він призначений для фільтрації високочастотного шуму, і його робота супроводжується розмиттям зображення. Кожен елемент маски дорівнює `1/mn`, де `M` і `N` − розміри маски (кількість рядків і стовпців).

Фільтр Гаусса також відноситься до НЧ фільтрів. На відміну від усереднюючого фільтра він менше розмиває оброблюване зображення. Маска фільтра є такою, що центральний елемент маски має найбільше значення, він відповідає піку розподілу Гаусса. Значення останніх елементів зменшуються у міру видалення від центрального елемента. Зменшення відбувається відповідно до розподілу Гаусса. Маска формується з використанням таких співвідношень:

$$\boldsymbol{h_g(r,c) = e^{-(r^2+c^2)/(2\sigma^2)}};$$

$$\boldsymbol{h(r,c) = \frac{h_g(r,c)}{\sum_{r=1}^{M}\sum_{c=1}^{N} h_g(r,c)}},$$

де $\boldsymbol{M}$ і $\boldsymbol{N}$ – розміри маски; $\boldsymbol{\sigma}$ – середньоквадратичне відхилення розподілу Гаусса.

Фільтр Лапласа відноситься до ВЧ фільтрів і призначений для виділення меж (перепадів) на всіх напрямках. Маска фільтра конструюється таким чином:

$$\boldsymbol{h = \frac{4}{(a+1)}\begin{bmatrix} \frac{a}{4} & \frac{1-a}{4} & \frac{a}{4} \\ \frac{1-a}{4} & -1 & \frac{1-a}{4} \\ \frac{a}{4} & \frac{1-a}{4} & \frac{a}{4} \end{bmatrix}},$$

де `a` – параметр у діапазоні `[0,1]`, який передається у функцію `fspecial`.

Лапласіан-гауссіан також відноситься до ВЧ фільтрів, але на відміну від фільтра Лапласа виділяє різкіші перепади. Маска фільтра створюється за формулою

$$\boldsymbol{h(r,c) = \frac{(r^2+c^2-2\sigma^2)h_g(r,c)}{2\pi\sigma^6\sum_{r=1}^{M}\sum_{c=1}^{N} h_g(r,c)}},$$

де $\boldsymbol{M}$ і $\boldsymbol{N}$ – розміри маски; $\boldsymbol{\sigma}$ – середньоквадратичне відхилення розподілу Гаусса. Формула для обчислення $\boldsymbol{h_g}$ виведена вище.

Маска фільтра, що підвищує різкість зображення, створюється таким чином:

$$\begin{bmatrix} -a & a-1 & -a \\ a-1 & a+5 & a-1 \\ -a & a-1 & -a \end{bmatrix},$$

де `a` – параметр у діапазоні `[0,1]`, який передається у функцію `fspecial`.

Розглянемо на прикладі, як впливає середньоквадратичне відхилення розподілу Гаусса на елементи маски фільтра Гаусса, який повертається функцією `fspecial`. Одержимо значення маски фільтра Гаусса розміром `5x5` для середньоквадратичного відхилення `0.5` і `0.7`. На рис. 3.8 показано частотну характеристику фільтра Гаусса з маскою `h05` і частотну характеристику фільтра Гаусса з маскою `h07`.

```
>> h05=fspecial('gaussian',5,0.5)

h05 =

    0.0000    0.0000    0.0002    0.0000    0.0000
    0.0000    0.0113    0.0837    0.0113    0.0000
    0.0002    0.0837    0.6187    0.0837    0.0002
    0.0000    0.0113    0.0837    0.0113    0.0000
    0.0000    0.0000    0.0002    0.0000    0.0000
>> h07=fspecial('gaussian',5,0.7)

h07 =

    0.0001    0.0020    0.0055    0.0020    0.0001
    0.0020    0.0422    0.1171    0.0422    0.0020
    0.0055    0.1171    0.3248    0.1171    0.0055
    0.0020    0.0422    0.1171    0.0422    0.0020
    0.0001    0.0020    0.0055    0.0020    0.0001

>> freqz2(h05,[32 32]);
>> figure, freqz2(h07,[32 32]);
```

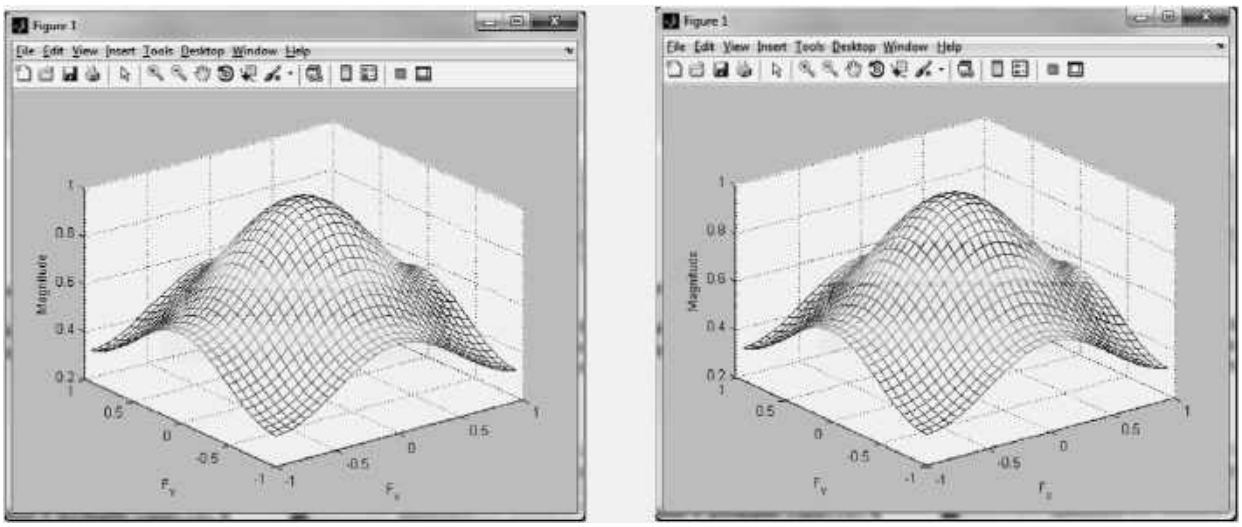



Рис. 3.8. Частотні характеристики фільтра Гаусса з маскою h05 і h07

Після визначення фільтра `fspecial` можна використовувати його безпосередньо в команді `imfilter`.

### 3.6. Особливості фільтрації з використанням функції `imfilter`

Стандартне використання функції `imfilter` було показано раніше на прикладі проведення лінійної фільтрації. Проте її можливості істотно є ширшими. Наприклад, з її допомогою можна реалізувати фільтрацію зображень з використанням кореляції і конволюції. Розглянемо деякі особливості використання функції `imfilter`.

***Типи даних*** у функції `imfilter` встановлюються також, як і при використанні інших схожих арифметичних функцій. Результуюче зображення буде подано в тому ж форматі даних, що і вихідне зображення. Функція `imfilter` обчислює значення кожного результуючого пікселя у форматі подвоєної точності, проводячи арифметичні операції з плаваючою комою. Якщо результуючі значення перевищують діапазон, який визначений для такого типу даних, то функція `imfilter` може обрізувати їх до потрібного діапазону. Це залежить від формату подання даних. Якщо дані наведені у форматі `integer`, то функція `imfilter` округлює дробову частину.

Таким чином, при використанні функції `imfilter` необхідно враховувати формат подання даних зображення. Розглянемо приклад, де в результаті використання функції `imfilter` дані набули негативного значення. Вихідні дані були подані у форматі `double`.

```
A = magic(5)

A =
    17    24     1     8    15
    23     5     7    14    16
     4     6    13    20    22
    10    12    19    21     3
    11    18    25     2     9

h = [-1 0 1]

h =
    -1     0     1

imfilter(A,h)

ans =
    24   -16   -16    14    -8
     5   -16     9     9   -14
     6     9    14     9   -20
    12     9     9   -16   -21
```

```
    18    14   -16   -16    -2
```

Відзначимо, що результат наведений також і негативними значеннями. Якщо матрицю `A` подати не у форматі `double`, а у форматі uint8, то отримаємо такий результат:

```
A=uint8(magic(5));
imfilter(A,h)

ans =

    24     0     0    14     0
     5     0     9     9     0
     6     9    14     9     0
    12     9     9     0     0
    18    14     0     0     0
```

Після того, як вихідні дані були наведені у форматі `uint8`, то результуючі дані також подаються у форматі `uint8`, а негативні значення будуть обрізані до `0`. Таким чином, вихідні дані перед використанням функції `imfilter` можна перетворити і в інші формати, наприклад, `signed integer, single` або `double`.

***Опції кореляції і згортки***. Функція `imfilter` може виконувати фільтрацію з використанням кореляції або згортки. За умовчанням використовують кореляцію, оскільки функції формування фільтрів формують кореляційні ядра.

Проте, якщо необхідно виконати фільтрацію з використанням згортки, потрібно вказати опцію "`conv`" як необов'язковий аргумент у функції `imfilter`. Розглянемо приклад.

```
A=magic(5);
h=[-1 0 1]
imfilter(A,h)
% фільтрація з використанням кореляції

ans =
    24   -16   -16    14    -8
     5   -16     9     9   -14
     6     9    14     9   -20
    12     9     9   -16   -21
    18    14   -16   -16    -2
imfilter(A,h,'conv') % фільтрація з використанням згортки
```

```
ans =
   -24    16    16   -14     8
    -5    16    -9    -9    14
    -6    -9   -14    -9    20
   -12    -9    -9    16    21
   -18   -14    16    16     2
```

***Опція доповнення краю зображення***. При обчисленні результуючих пікселів на краю зображення частина маски фільтра вийде за межі зображення так, як продемонстровано на зображенні (див. рис. 3.9,а).

Зазвичай при виході маски фільтра за межі зображення функція `imfilter` заповнює бракуючі вічка нульовими значеннями. Приклад такого заповнення показаний на рис. 3.9,б.

Якщо відфільтрувати зображення, заповнюючи бракуючі пікселі нулями, то на результуючому зображенні на межі зображення з'явиться темна лінія. Це показано на рис. 3.10.

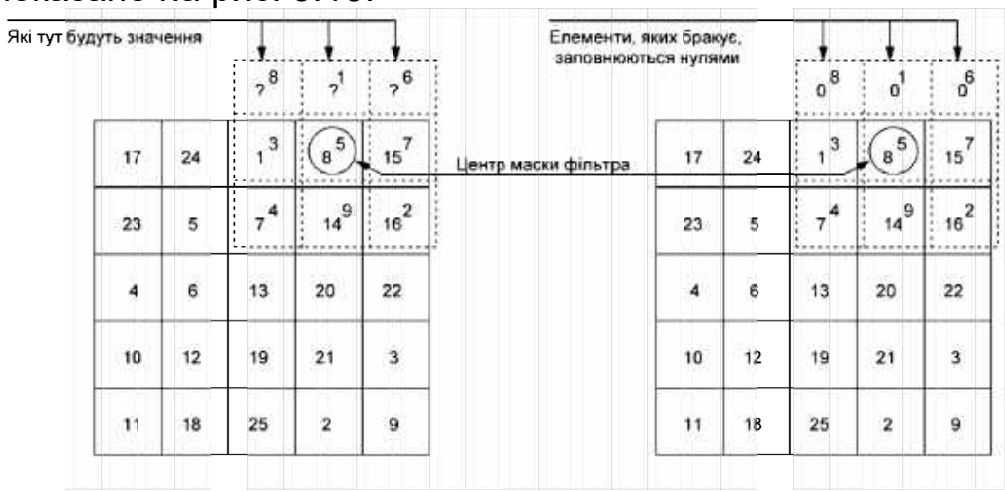



Рис. 3.9. Доповнення межі зображення: а – вихід значень маски фільтра за межі зображення; б – заповнення пікселів, яких бракує, нульовими значеннями

```
I=imread('eight.tif');
h=ones(5,5)/25;
I2=imfilter(I,h);
imshow(I),title('Вихідне зображення');
figure,imshow(I2),title('Зображення після фільтрації');
```

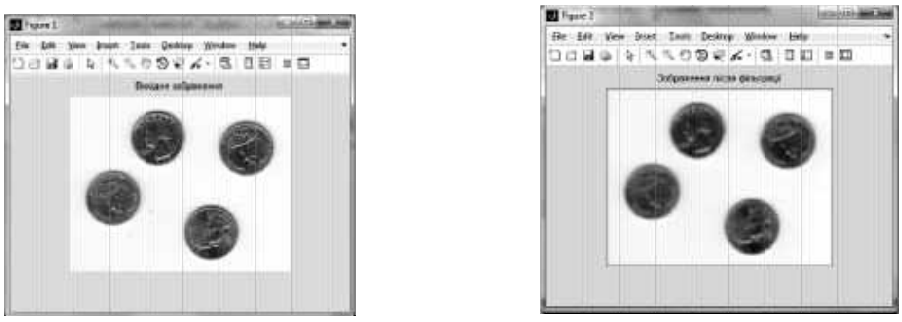


Рис. 3.10. Крайові ефекти при фільтрації меж зображення

Для усунення артефактів, які обумовлені тим, що бракуючі елементи були заповнені нулями, `imfilter` застосовує альтернативні методи заповнення бракуючих елементів. Один із цих методів полягає в тому, що бракуючі пікселі заповнюються значеннями крайніх пікселів зображення. Цей метод продемонстровано на рис. 3.11.

При реалізації фільтрації з використанням заповнення елементів, яких бракує, крайніми елементами зображення необхідно додатково у функції `imfilter` указувати опцію '`replicate`'.

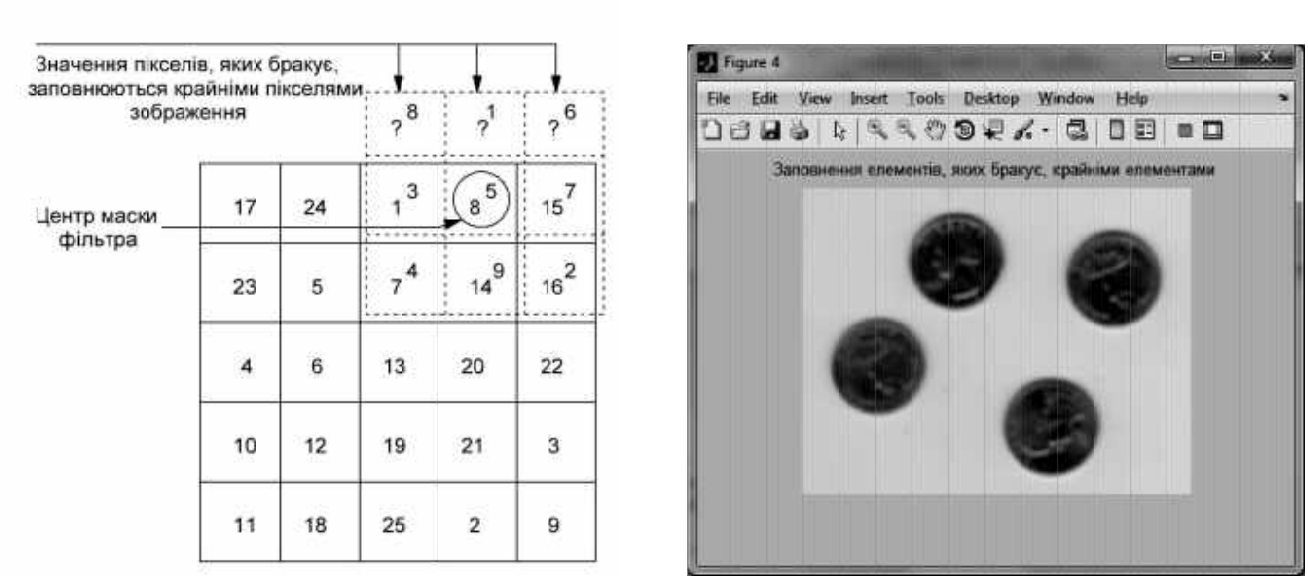



Рис. 3.11. Заповнення маски крайніми пікселями зображення

```
I3=imfilter(I,h,'replicate');
figure,imshow(I3);
```

Функція `imfilter` підтримує також інші способи заповнення пікселів, яких бракує, при обробленні крайніх елементів зображення. Для цього є такі опції, як '`circular`' і '`symmetric`'. Для детальнішої інформації див. опис функції `imfilter`.

Система MATLAB має в своєму розпорядженні декілька двовимірних і багатовимірних функцій фільтрації. Функція `filter2` виконує двовимірну лінійну фільтрацію, `conv2` виконує згортку двовимірних сигналів, а `convn` – згортку багатовимірних сигналів. При роботі з кожною з цих функцій необхідно, щоб вихідні дані були подані у форматі `double`. Результат також буде наведений у форматі `double`. Ці функції завжди використовують заповнення нулями пікселів, яких бракує, при роботі на краю зображення і не підтримують інших способів заповнення.

На відміну від названих функцій `imfilter` не вимагає перетворення вихідних даних у формат `double`. Крім того, функція `imfilter` має в своєму розпорядженні декілька варіантів заповнення бракуючих елементів на краю зображення.

***Використання зумовлених типів фільтрів***. Функція `fspecial` характеризується можливістю формування декількох видів зумовлених фільтрів, що відбивається на формі маски. Після створення фільтра за допомогою функції `fspecial` його можна застосувати для оброблення даних зображення за допомогою функції `imfilter`. Розглянемо приклад реалізації фільтра типу нечіткого маскування для оброблення повнокольорового зображення (рис. 3.12). Метод нечіткого маскування ефективно застосовують для посилення меж і підвищення деталізації зображень.

```
clear;close all;
I=imread('moon.tif');
h=fspecial('unsharp')
h =

   -0.1667   -0.6667   -0.1667
   -0.6667    4.3333   -0.6667
   -0.1667   -0.6667   -0.1667
I2=imfilter(I,h);
imshow(I),title('Вихідне зображення')
figure,imshow(I2),title('Filtered Image')
```


Рис. 3.12. Приклад нечіткого маскування для підвищення деталізації зображення

***Частотні методи перетворень***. Їх використовують як в одновимірних, так і двовимірних фільтрах з кінцевою імпульсною характеристикою, а також з матрицею перетворень, елементи якої визначають частотні перетворення.

Функція `ftrans2` призначена для реалізації частотних методів оброблення зображень. Ця функція формує маску лінійного двовимірного фільтра, використовуючи метод перетворення частот для трансформації одновимірного фільтра з кінцевою імпульсною характеристикою. Відзначимо, що частотні методи перетворень забезпечують гарні результати оброблення. Розглянемо приклад.

```
b=remez(10,[0 0.4 0.6 1],[1 1 0 0]);
h=ftrans2(b);
[H,w]=freqz(b,1,64,'whole');
colormap(jet(64))
plot(w/pi-1,fftshift(abs(H)))
figure,freqz2(h,[32 32])
```

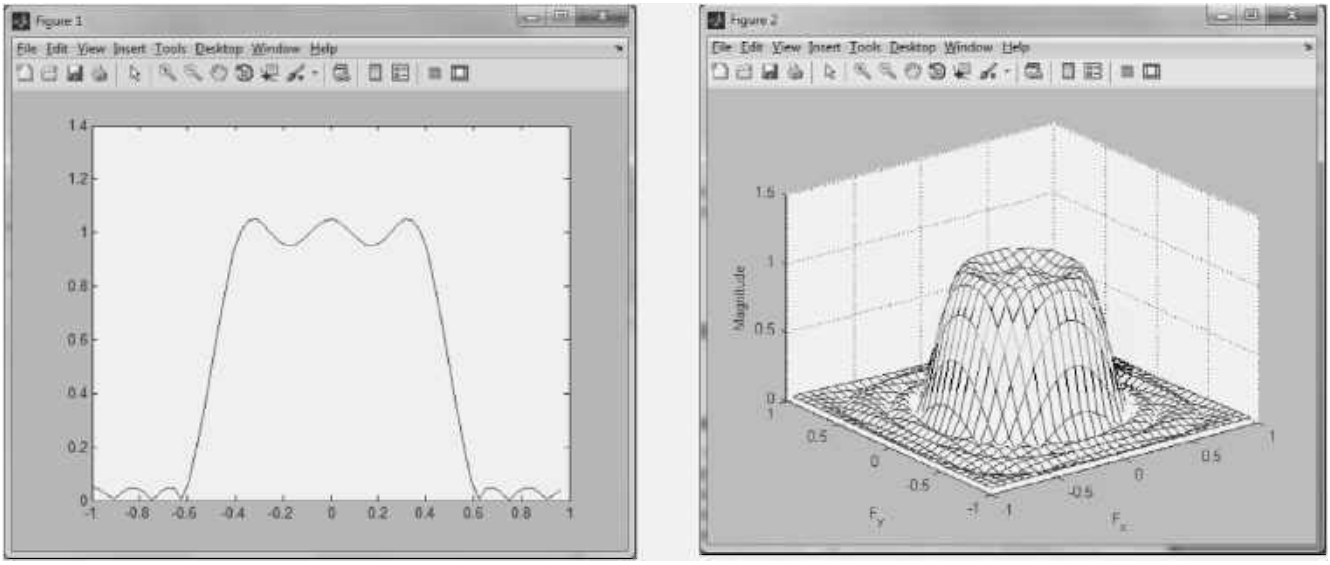


Рис. 3.13. Одновимірний частотний відгук (зліва) і відповідний йому двовимірний частотний відгук (справа)

***Приклад частотного методу***. При частотних методах маска лінійного фільтра формується на основі бажаної амплітудно-частотної характеристики. Функція `fsamp2` формує маску лінійного двовимірного фільтра на основі бажаної амплітудно-частотної характеристики двовимірного фільтра. Сформована маска призначена для передачі у відповідні функції (наприклад, `filter2` або `conv2`), які виконують двовимірну лінійну фільтрацію. Розглянемо сказане вище на конкретному прикладі (рис. 3.14).

```
Hd=zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2]=freqspace(11,'meshgrid');
mesh(f1,f2,Hd),axis([-1 1 -1 1 0 1.2]),colormap(jet(64))
```

```
h=fsamp2(Hd);
figure,freqz2(h,[32 32]),axis([-1 1 -1 1 0 1.2])
```

***Локальні (віконні) методи***. У додатку Image Processing Toolbox існують функції, які формують маски лінійного фільтра за бажаною амплітудно-частотною характеристикою з використанням одновимірного або двовимірного вікна. При формуванні матриці бажаної амплітудно-частотної характеристики для одержання точнішого результату рекомендується використовувати відліки частоти, які повертаються функцією `freqspace`. Розглянемо ці можливості на прикладі.

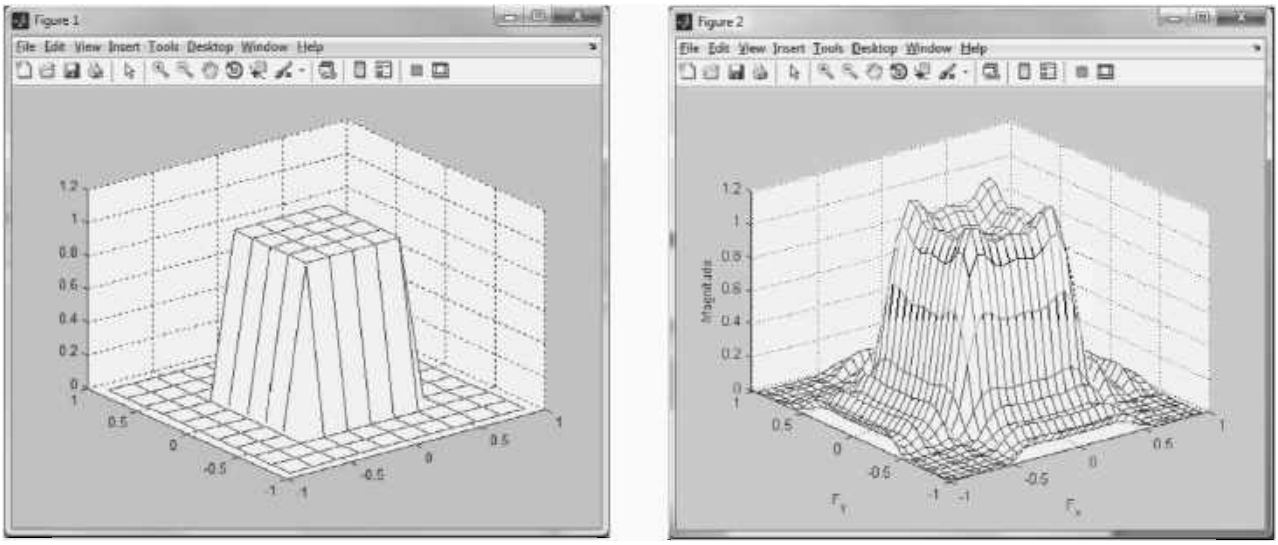


Рис. 3.14. Бажаний двовимірний частотний відгук (зліва) і реальний двовимірний частотний відгук (справа)

```
Hd=zeros(11,11); Hd(4:8,4:8) = 1;
[f1,f2]=freqspace(11,'meshgrid');
mesh(f1,f2,Hd),axis([-1 1 -1 1 0 1.2]),colormap(jet(64))
h=fwind1(Hd,hamming(11));
figure,freqz2(h,[32 32]),axis([-1 1 -1 1 0 1.2])
```

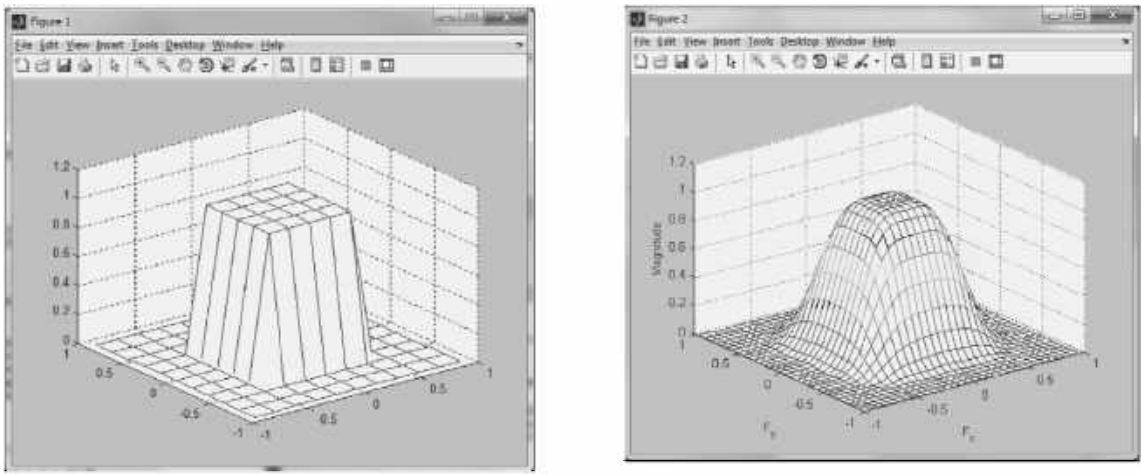


Рис. 3.15. Бажаний двовимірний частотний відгук (зліва) і реальний двовимірний частотний відгук (справа)

***Створення бажаної амплітудно-частотної характеристики***. Функції проектування фільтрів `fsamp2`, `fwind2` і `fwind2` створюють фільтри на основі матриці значень бажаної амплітудно-частотної характеристики. Відповідну бажану амплітудно-частотну характеристику можна створювати за допомогою функції `freqspace`. Розглянемо приклад створення колоподібного низькочастотного фільтра із зрізом в `0.5`. Для цього використовують такий код:

```
[f1,f2] = freqspace(25,'meshgrid');
Hd = zeros(25,25); d = sqrt(f1.^2 + f2.^2) < 0.5;
Hd(d) = 1;
mesh(f1,f2,Hd)
```

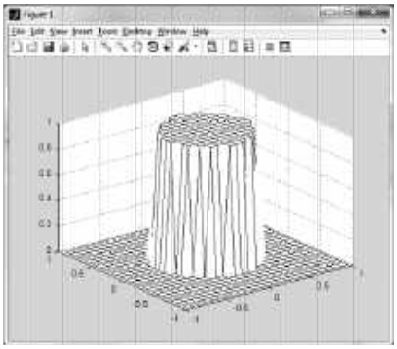


Рис. 3.16. Ідеальний колоподібний низькочастотний відгук

***Обчислення частотного відгуку фільтра***. Функція `freqz2` обчислює частотний відгук двовимірного фільтра. Якщо для фільтра з кінцевою імпульсною характеристикою прийняти, що

```
h =[0.1667    0.6667    0.1667
    0.6667   -3.3333    0.6667
    0.1667    0.6667    0.1667],
```

то розглянемо обчислення і візуалізацію `64x64` точок частотного відгуку `h`.

```
freqz2(h)
```

Для знаходження частотного відгуку `H` і вектора частот `f1` і `f2` використовуємо такий код:

```
[H,f1,f2] = freqz2(h);
```

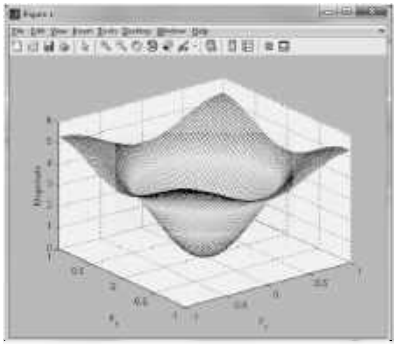


Рис. 3.17. Частотний відгук двовимірного фільтра

Як приклад реалізуємо процедуру створення двовимірного фільтра низьких частот (ФНЧ) з кінцевою імпульсною характеристикою (КІХ), який сформований з одновимірного ФНЧ методом перетворення частот.

```
% Створимо одновимірний КІХ ФНЧ 10-го порядку з частотою
   зрізу 0.15
b=fir1(10,0.15);
% Візуалізація АЧХ і ФЧХ створеного одновимірного фільтра
freqz(b,1,256);
% Формування і виведення на екран АЧХ двовимірного фільтра
h=ftrans2(b);
```

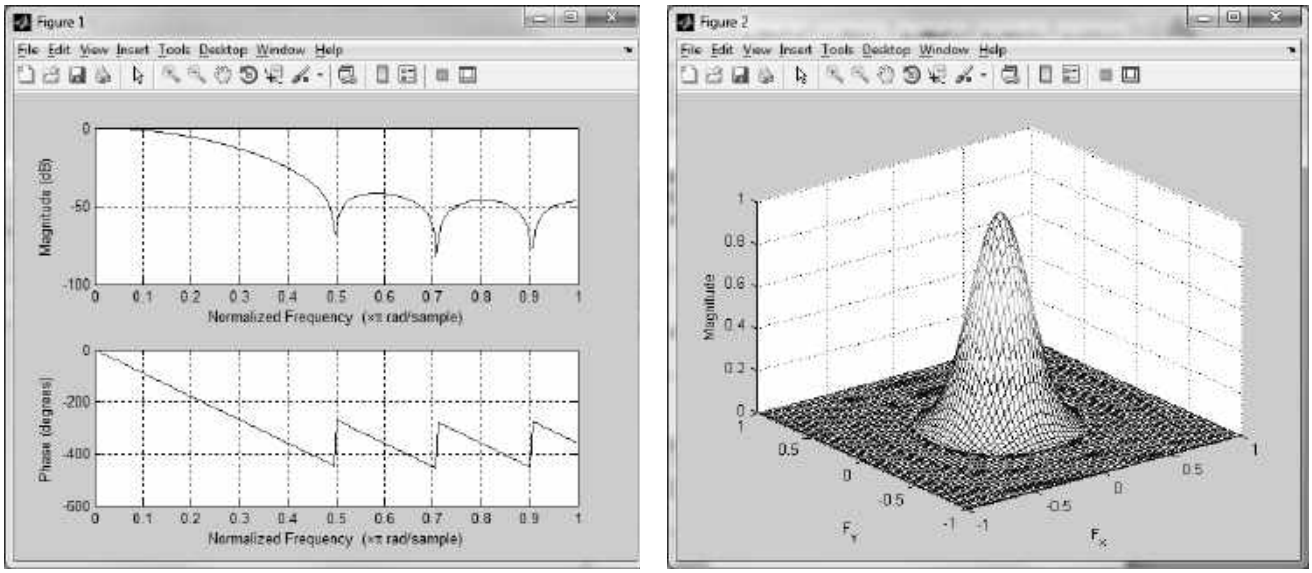



Рис. 3.18. АЧХ одновимірного і двовимірного фільтрів

### 3.7. Робота з областями інтересу

Областю інтересу є частина зображення, над якою можна виробляти деякі операції, зокрема фільтрацію. Область інтересу визначається шляхом створення бінарної маски, розмір якої збігається з розміром оброблюваного зображення. Маска містить одиницю для всіх пікселів, які є частиною області інтересу, і нуль – для всіх інших.

Далі розглянемо методи створення бінарних масок:

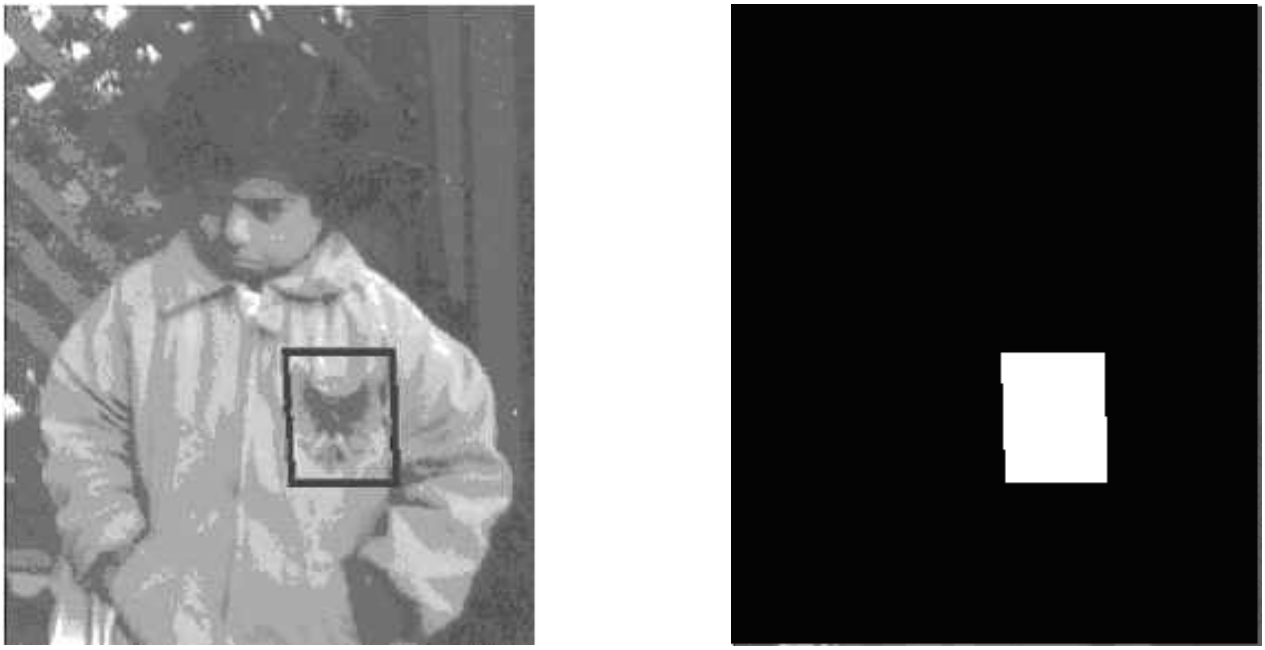
1. Вибір області інтересу.
2. Інші методи вибору області інтересу (використання довільної бінарної маски або функції `roicolor`).

***Вибір області інтересу***. Для вибору багатокутної області інтересу можна використовувати функцію `roipoly`. При виклику функції `roipoly` без аргументів курсор змінює свою форму на хрестик, якщо він знаходиться над зображенням, яке показано в поточних координатах. Область інтересу можна вказати, якщо на зображенні кликнути мишкою і таким чином задати вершини багатокутника. Після цього необхідно натискувати `Enter`, і функція `roipoly` поверне бінарне зображення, розмір якого збігається з

початковим. Це бінарне зображення містить `1` на місці відміченої області і `0` – за її межами.

Розглянемо приклад, який демонструє інтерактивний процес використання функції `roipoly` для створення бінарної маски. На зображенні межі вибраної за допомогою мишки області забарвлені червоним кольором.

```
I=imread('pout.tif');
imshow(I)
BW = roipoly;
imshow(BW)
```



а б

Рис. 3.19. Вибір області інтересу: а – за допомогою функції `roipoly`; б – створена бінарна маска

Функцію `roipoly` можна використовувати і не інтерактивно. Для детальнішої інформації див. опис функції `roipoly`.

***Заповнення області інтересу за допомогою полігона***. Функція `Bw=roipoly(S)` дозволяє інтерактивно задати область інтересу на вихідному зображенні `S` будь-якого типу і помістити результат у бінарне зображення `BW`. Отримане бінарне зображення `BW` можна використовувати як область інтересу для функції `roifilt2` або для завдання положення об'єктів і їх подальшого пошуку за допомогою функції `bwlabel`.

Опишемо її синтаксис:

```
BW=roipoly(S)
BW=roipoly(S,c,r)
BW=roipoly(XData,YData,S,xi,yi)
[BW,xi,yi]=roipoly(...)
[XData,YData, BW,xi,yi]=roipoly(...)
```

Дана функція виводить зображення S у вікно і чекає від користувача визначення області інтересу. Якщо при виклику функції параметр S опущений, то зображення береться з поточного вікна. Область зображення, що нас цікавить, має бути обведена полігоном, вершини якого задаються однократним натисненням лівої клавіші мишки. Попередню задану вершину можна видалити, якщо натискувати на клавіші Backspace або Delete. Натиснення на праву клавішу мишки або подвійне клацання лівою клавішею задає останню вершину полігона. Крім того, завершити процес завдання вершин без вказівки останньої можна натисненням на клавішу Enter.

Зображення BW і S мають однаковий розмір. Пікселю бінарного зображення BW(r,c) привласнюється значення 1, якщо піксель S(r,c) знаходиться усередині полігона. Інакше BW(r,c) дорівнює 0. Для визначення точок, що лежать усередині полігона, використовують правило Non-zero Winding.

Функція BW=roipoly(S,c,r) дозволяє явно задати координати вершин полігона, що обмежує область інтересу, у векторах r (номери рядків) і c (номери стовпців) однакової довжини.

Функція BW=roipoly(Xdata,ydata,s,xi,yi) аналогічна попередній, але в ній координати полігона, що передаються у векторах xi і yi, задаються в просторовій системі координат. Двоелементні вектори Xdata і Ydata визначають діапазон зміни значень по осях просторової системи координат.

Якщо для розглянутих функцій додатково визначити два вихідних параметри xi і yi: [Bw,xi,yi]=roipoly(...), то в них будуть повернені координати вершин полігона в просторовій системі координат. Якщо ж додатково визначити ще два вихідних параметри Xdata і Ydata: [Xdata,ydata,bw,xi,yi]=roipoly(...), то в них будуть повернені діапазони значень по осях просторової системи координат, яку використовували для зображення S.

Якщо функцію roipoly викликати без вихідних параметрів, то результуюче бінарне зображення виводиться на екран у новому вікні. У будь-якій з розглянутих функцій можна замінити параметр S (вихідне зображення) на два параметри m і n, які визначають розмір вихідного зображення. Наприклад, якщо викликати функцію Bw=roipoly(100,200), то на екран буде виведено чорне зображення в 100 рядків і 200 стовпців і функція чекатиме від користувача інтерактивного завдання вершин полігона. Результуюче зображення BW має формат подання даних uint8.

***Заповнення областей інтересу*** здійснюється за допомогою функції roifill(Is). Опишемо її синтаксис:

```
Id=roifill(Is)
```

```
Id=roifill(Is,c,r)
Id=roifill(Is,BW)
Id=roifill(XData,YData,Is,xi,yi)
[Id,BW]=roifill(...)
 [XData,YData,Id,BW,xi,yi]=roifill(...)
```

Функція `roifill` призначена для усунення невеликих за площею дефектів півтонових зображень. Вона заповнює області, що задаються, плавними переходами яскравості, використовуючи інтерполяцію за значеннями яскравості граничних пікселів області.

Функція `Id=roifill(Is)` дозволяє інтерактивно задати область на вихідному півтоновому зображенні `Is` і створює нове півтонове зображення `Id`, яке відрізняється від вихідного зображення `Is` тим, що задана область заповнена. Ця функція виводить зображення Is у вікно і чекає від користувача завдання області інтересу, призначеної для заповнення. Область інтересу має бути поміщена у полігон (багатокутник), вершини якого задаються однократним натисненням лівої клавіші мишки. Попередню задану вершину можна видалити, якщо натискувати на клавіші `Backspace` або `Delete`. Натиснення на праву клавішу мишки або подвійне клацання лівою клавішею задає останню вершину полігона. Крім того, завершити процес завдання вершин без вказівки останньої можна натисненням на клавішу `Enter`. Для визначення точок, що лежать усередині полігона, використовують правило `Non-zero Winding`. Якщо при виклику функції параметр `Is` опущений, то зображення береться з поточного вікна.

Функція `Id=roifill(Is,c,r)` дозволяє явно задати координати вершин полігона, що обмежує область, яка заповнюється у векторах `r` (номери рядків) і `c` (номери стовпців) однакової довжини.

Функція `Id=roifill(Xdata,ydata,is,xi,yi)` аналогічна попередній, але в ній координати полігона, які передають у векторах `xi` і `yi`, задаються в просторовій системі координат. Двоелементні вектори `Xdata` і `Ydata` визначають діапазон зміни значень по осях просторової системи координат.

Області, які необхідно заповнити, можуть бути задані за допомогою бінарного зображення `BW`. У цьому випадку використовують функцію `Id=roifill(Is,bw)`. Зображення `Is` і `BW` мають однаковий розмір. Пікселі зображення `BW`, що відносяться до областей, призначених для заповнення, мають дорівнювати `1`.

Для розглянутих функцій можна додатково визначити чотири вихідних параметри: `[Xdata,ydata,id,bw,xi,yi]=roifill(...)`. Тоді в параметрах `xi` і `yi` будуть повернені координати вершин полігона, що обмежує заповнену область, а в `Xdata` і `Ydata` будуть повернені діапазони значень

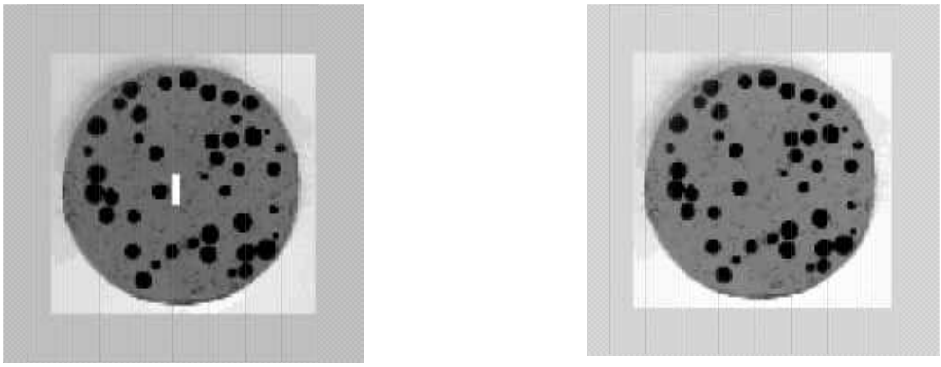
по осях просторової системи координат, яку використовували для зображення `Is`.

Якщо функцію `roifill` задавати без вихідних параметрів, то результуюче зображення виводиться на екран у новому вікні. Формати подання даних вихідного `Is` і результуючого `Id` зображень збігаються.

Розглянемо використання функції `roifill` для усунення невеликих дефектів півтонового зображення. На рис. 3.20, а показано вихідне зображення зі штучним білим прямокутником, який імітує деякий дефект на зображенні. Усунемо цей невеликий дефект, розташований в центрі зображення. Для цього за допомогою функції `roipoly` інтерактивно створимо бінарне зображення, що задає область інтересу, а потім скористаємося функцією `roifill`.

***Приклад***. Усунення на зображенні малих за площею дефектів

```
% Читання вихідного зображення і виведення його на екран
I=imread('shot1.tif');
imshow(I);
% Завдання області інтересу (інтерактивно)
BW=roipoly;
% Заповнення області інтересу
I1=roifill(I,BW);
% Виведення результату на екран
figure,imshow(I1);
```



а б

Рис. 3.20. Заповнення області інтересу для усунення малих за площею дефектів

***Інші методи вибору області інтересу***. Функція `roipoly` надає зручний шлях для створення бінарної маски. В принципі будь-яке бінарне зображення можна використовувати як маску. Проте це бінарне зображення має бути того ж розміру, що і вихідне зображення.

Розглянемо приклад. Хай необхідно відфільтрувати зображення яскравості `I`, проте обробляти необхідно лише ті пікселі, значення яких більше, ніж `0.5`. Створимо відповідну маску

```
BW=(I > 0.5);
```

Для формування бінарної маски можна також використовувати функцію poly2mask. На відміну від roipoly функція poly2mask при створенні маски не потребує вихідного зображення. Для детальнішої інформації див. опис функції poly2mask. Для формування області інтересу на основі колірного або яскравості діапазонів можна використовувати також функцію roicolor. Для детальнішої інформації див. опис функції roicolor.

***Фільтрація області інтересу***. Для оброблення області інтересу можна використовувати функцію roifilt2. При виклику функції roifilt2 необхідно вказати вихідне зображення (яскравість), бінарну маску і фільтр. Функція roifilt2 піддає фільтрації вихідне зображення і повертає (накладає) результат на вихідне зображення в тій області, яка була відмічена маскою. Таким чином, область вихідного зображення, яка на масці відмічена 1, піддається фільтрації, остання частина відповідає вихідному зображенню. Цей тип операції називається маскованою фільтрацією.

Особливістю використання функції roifilt2 є те, що ця операція повертає дані в тому ж діапазоні, що і вихідне зображення. Проте деякі операції фільтрації можуть привести до того, що значення пікселів обробленого зображення можуть вийти за вказаний діапазон (наприклад, [0,1] для зображень, які подані у форматі подвоєної точності, [0,255] – для зображень у форматі uint8 і [0,65535] – для зображень у форматі uint16). Розглянемо детальніше опис функції roifilt2 та її синтаксис.

```
Id=roifilt2(h,Is,BW)
Id=roifilt2(Is,BW,fun)
Id=roifilt2(Is,BW,fun,P1,P2,...)
```

Функція roifilt2 призначена для фільтрації окремих фрагментів півтонових зображень. Ці фрагменти – області інтересу (regions of interest) – задаються за допомогою бінарного зображення BW. Розміри BW і вихідного півтонового зображення Is мають збігатися. Одиничні значення у BW указують на те, що пікселі вихідного зображення з такими ж координатами належать області інтересу. Таким чином, якщо BW(r,c) дорівнює 1, то Id(r,c) дорівнює значенню пікселя з координатами (r,c), отриманого в результаті фільтрації вихідного зображення Is. Якщо BW(r,c) дорівнює 0, то Id(r,c) дорівнює Is(r,c), тобто зміна зображення відбувається лише в областях інтересу.

Функція Id=roifilt2(h,is,bw) створює зображення Id як результат фільтрації областей інтересу BW вихідного зображення Is лінійним фільтром з маскою h. Для фільтрації функція roifilt2 викликає функцію filter2. Результуюче зображення Id має формат подання даних double.

Функція Id=roifilt2(Is,bw,fun) створює зображення Id як результат фільтрації областей інтересу BW вихідного зображення Is за до-

помогою функції fun. Існує три варіанти завдання параметра fun. Вони наведені в таблиці в описі функції blkproc. Ім'ям X має бути позначено оброблюване півтонове зображення Is.

Функція Id = roifilt2(Is,bw,fun) створює зображення Id як результат фільтрації областей інтересу BW вихідного зображення Is за допомогою функції fun. Вони наведені в таблиці в описі функції blkproc. Ім'ям X має бути позначено оброблюване півтонове зображення Is. Результуюче зображення, що повертається функцією fun, повинно мати розміри, які дорівнюють розмірам зображення X.

Функція Id=roifilt2(Is,bw,fun,pi,p2) формує зображення Id, передаючи у функцію fun додаткові параметри PI, P2 ...

Формат подання даних зображень Is і Id визначається реалізацією функції fun. Особливості операції фільтрації – функція roifilt2 також дає можливість обробляти область інтересу за допомогою інших функцій.

Наведемо приклад використання функції imadjust для освітлення частини зображення:

```
I = imread('cameraman.tif');
BW = imread('text.png');
mask = BW(1:256,1:256);
f = inline('imadjust(x,[],[],0.3)');
I2 = roifilt2(I,mask,f);
imshow(I2)
```


Рис. 3.21. Бінарна маска, що містить текст при обробленні

### 3.8. Критерії якості оброблення зображень

Якість зображення може визначатися статистичними і спектральними характеристиками яскравості зображення. У більшості практичних випадків якість розглядається як міра близькості двох зображень: реального і ідеального або перетвореного і початкового. При такому підході можна як оці-

нювати суб'єктивну міру схожості зображень, так і отримувати об'єктивні оцінки параметрів сигналів зображення: моменти першого і другого порядку різницевого сигналу порівнюваних зображень, такі параметри перетворення, як відношення сигнал/шум, коефіцієнти стискування інформації та ін.

***Суб'єктивні критерії*** – це критерії візуального сприйняття, що оцінюються в процесі експертизи деякою групою спостерігачів (експертів). Найбільшого поширення набув метод оцінок, при якому спостерігач оцінює якість зображення в балах за певною шкалою, вважаючи, що ідеальне зображення має максимальний бал. Цей метод дозволяє оцінити такі характеристики зображення, як правильність перенесення кольорів, координатні спотворення, чистота переходів і т. ін. Основні шкали оцінок при використанні методу порівняння наведені в табл. 3.1.

Для інтерпретації отриманих експертних оцінок розроблено методи їх подання, наприклад побудова кумулятивних кривих розподілу оцінок як функції від спотворень. Середню оцінку визначають за формулою

$$\boldsymbol{g}_{\text{сер}} = (\boldsymbol{1}/\boldsymbol{N}) \sum_{\boldsymbol{i}=\boldsymbol{1}}^{\boldsymbol{r}} \boldsymbol{i}\boldsymbol{n}_{\boldsymbol{i}} \ ,$$

де $\boldsymbol{N}$ – загальне число оцінок; $\boldsymbol{n}_{\boldsymbol{i}}$ – число оцінок, які дорівнюють $\boldsymbol{i}$ балам; $\boldsymbol{r}$ – кількість видів різних оцінок. Нормалізовані оцінки p виражають відносну якість у діапазоні $[\boldsymbol{0}, \boldsymbol{1}]$. При п'ятибальній системі, коли $\boldsymbol{g} \in [\boldsymbol{1}, \boldsymbol{5}]$:

$$\boldsymbol{p} = \frac{\boldsymbol{g} - \boldsymbol{1}}{\boldsymbol{4}},$$

а середню оцінку обчислюють відповідно до формули

$$\boldsymbol{p}_{\text{сер}} = (\boldsymbol{n_5} + \boldsymbol{0,75n_4} + \boldsymbol{0,5n_3} + \boldsymbol{0,25n_2})/\boldsymbol{N} \ .$$

Одиницею погіршення якості телевізійних (ТВ) зображень є імп (від impairment – погіршення, пошкодження). Ця одиниця введена Проссером, Аллнатом і Льюісом у 1964 р. і використовується МККР (Міжнародним консультативним комітетом з радіозв'язку (CCIR)). Погіршення обернено пропорційно до нормалізованої оцінки якості і змінюється від ∞ до 0 при зміні $\boldsymbol{p}$ від 0 до 1 відповідно до формули $\boldsymbol{I} = \boldsymbol{1}/(\boldsymbol{p} - \boldsymbol{1})$. Ці характеристики зведені в табл. 3.1.

Таблиця 3.1

Основні шкали суб'єктивних оцінок якості зображення

| Нормалізована | П`ятибальна шкала | | Семибальна шкала погіршення |
|---|---|---|---|
| | Шкала якості | Шкала погіршення | |
| 1 | 5 (відмінно) | 5 (непомітно) | 1 (непомітно) |

Закінчення табл. 3.1

| Нормалізована | П`ятибальна шкала | | Семибальна шкала погіршення |
|---|---|---|---|
| | Шкала якості | Шкала погіршення | |
| 0,875 | – | – | 2 (ледве помітно) |
| 0,75 | 4 (добре) | 4 (помітно, але не заважає) | 3 (помітно, але лише злегка погіршує) |
| 0,625 | – | – | 4 (погіршує, але не заважає) |
| 0,5 | 3 (задовільно) | 3 (помітно, трохи заважає) | 5 (декілька заважає) |
| 0,25 | 2 (погано) | 2 (заважає) | 6 (явно заважає) |
| 0 | 1 (дуже погано) | 1 (сильно заважає) | 7 (украй заважає) |

Позитивність методики оцінювання погіршення полягає в тому, що результуючу оцінку погіршення одержують арифметичним підсумовуванням оцінок погіршення, викликаних різними видами спотворень сигналів зображення. Грунтуючись на психофізичних властивостях спостерігача, суб'єктивні оцінки дозволяють характеризувати сприйняття зображення. Інтегральний критерій якості формується за узагальненою формулою

$$I_{\Sigma} = \sum_{i=1}^{M} I_i^{\gamma},$$

де $M$ – число параметрів, за якими оцінюють якість зображення; $\gamma$ – показник міри.

Значення показника міри мають дорівнювати 1, але можуть бути використані, наприклад, такі значення, як 0,78 або 2. У даний час застосовують й інші оцінки якості зображень. При розробленні апаратних засобів спеціального призначення велике значення має оцінювання об'єктивних характеристик якості перетвореного зображення.

Об'єктивні критерії дозволяють отримати просто обчислювану характеристику зображення різницевого сигналу. До таких критеріїв відноситься, перш за все, середньоквадратичний критерій. За ним мірою відмінності двох зображень $f(x,y)$ і $f_{\text{пр}}(x,y)$ є середньоквадратичне значення різницевого сигналу двох зображень. Для безперервних зображень, заданих при $x \in [0, N]$ і $y \in [0, M]$, середньоквадратичне відхилення (СКВ) обчислюють за формулою

$$\sigma^2 = \int_0^M \int_0^N \left[f(x,y) - f_{\text{пр}}(x,y)\right]^2 dxdy.$$

У деяких випадках використовують критерій максимальної помилки, який дозволяє встановити значення максимальної помилки перетворення

$$\varepsilon_{max} = \max_{x,y} \left| f(x,y) - f_{\text{пр}}(x,y) \right|.$$

Застосовують й інші об'єктивні критерії якості зображень. Розглянемо їх детальніше.

Зазвичай в MATLAB зображення задається таблицею чисел, що складається з M рядків і N стовпців. Кожне число в цій таблиці описує один піксель, який подається K бітами. У всіх таких критеріях порівняння міра близькості зображень визначається числом, яке деяким чином обчислюється за даними зображеннями. Для розрахунку оцінок відмінності зображень можна використовувати основні об'єктивні критерії якості, наведені в табл.. 3.2.

Середньоквадратична помилка (СКВ) дуже ненадійна, оскільки не відповідає системі візуального сприйняття людини (human visual system, HVS). Слід зазначити, що значення СКВ може трохи змінюватися при істотному погіршенні суб'єктивно сприйманої якості стислого зображення. Тому СКВ так само, як і пікове відношення сигнал/шум (PSNR), не може бути узято за основу при побудові оптимальних з візуальної точки зору систем перетворення зображень.

Використання логарифмів згладжує MSE і робить її менш чутливою до малих змін відновлюваного зображення. Тому на практиці використовують модифікацію міри MSE. Її називають PSNR (peak of signal-to-noise ratio). PSNR частіше за інші параметри застосовують для оцінювання схожості між вихідним і відновленим зображеннями. Порівняно з MSE цей захід є позитивним тому, що обчислюється в логарифмічній шкалі по амплітуді (у децибелах). Це важливо, оскільки око сприймає сигнал також у логарифмічній шкалі по амплітуді і тому посилення амплітуди сигналу в два рази не означає для людини поліпшення якості зображення в стільки ж разів.

Одним із недоліків цієї міри є висока чутливість до середньої відмінності сигналів по амплітуді, що може призвести до помилкового результату, у разі, коли сигнали трохи відрізняються в середньому по амплітуді. Фізіологія зору і психологія сприйняття зображення людини є настільки складними, що до цих пір не існує способу математичного розрахунку міри візуальної схожості двох зображень.

Таблиця 3.2

| Основні критерії оцінювання якості зображень | | |
|---|---|---|
| № п/п | Найменування критерію | Математичний опис |
| 1 | Середньоквадратична помилка (mean square error) або середній квадрат помилок | $MSE = \frac{1}{MN} \sum_{m=1}^{M} \sum_{n=1}^{N} \left[ B(m,n) - \tilde{B}(m,n) \right]^2$ |

Закінчення табл. 3.1

| № п/п | Найменування критерію | Математичний опис |
|---|---|---|
| 2 | Середня абсолютна помилка mean absolute error) | $\boldsymbol{MAE = \frac{1}{MN}\sum_{m=1}^{M}\sum_{n=1}^{N}\left\|B(m,n) - \widetilde{B}(m,n)\right\|}$ |
| 3 | Нормована середньоквадратична помилка (normalized MSE) | $\boldsymbol{NMSE = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}\left[B(m,n) - \widetilde{B}(m,n)\right]^2}{\sum_{m=1}^{M}\sum_{n=1}^{N}[B(m,n)]^2}}$ |
| 4 | Нормована абсолютна помилка (normalized absolute error) | $\boldsymbol{NAE = \frac{\sum_{m=1}^{M}\sum_{n=1}^{N}\left\|B(m,n) - \widetilde{B}(m,n)\right\|}{\sum_{m=1}^{M}\sum_{n=1}^{N}\|B(m,n)\|}}$ |
| 5 | Відношення сигнал/шум (signal to noise ratio) | $\boldsymbol{SNR = 10log_{10}\frac{\sum_{m=1}^{M}\sum_{n=1}^{N}[B(m,n)]^2}{\sum_{m=1}^{M}\sum_{n=1}^{N}\left[B(m,n) - \widetilde{B}(m,n)\right]^2}}$ |
| 6 | Пікове відношення сигнал/шум (peak signal to noise ratio) | $\boldsymbol{PSNR = 10log\frac{\left(2^b - 1\right)^2}{MSE}}$, де $\boldsymbol{b}$ – число бітів на значення пікселя зображення |

Необхідно відзначити, що значення PSNR не може повною мірою відображати дію на зображення різних видів перешкод, тобто за наявності в зображенні різних видів шумів його значення може залишатися постійним, а якість зображення істотно змінюватися.

Відзначимо, що існує певна розбіжність в оцінках якості, що даються людським оком (суб'єктивних), і об'єктивних, отриманих у вигляді кількісних показників. Око є досконалим винаходом природи, з ним не можуть змагатися досить примітивні об'єктивні оцінки типу СКВ, пікового відношення сигнал/шум (ПСШ) та ін. Тому деякі результати, що розглядаються з точки зору об'єктивних оцінок як однакові, візуально можуть сприйматися порізному. Проте об'єктивні критерії використовують при комп'ютерному обробленні зображень у системах з автоматичним ухваленням рішень. Функціонування автоматичних комп'ютерних систем повністю підпорядковане математичним критеріям, і якість їх роботи оцінюється лише об'єктивними показниками. Зрозуміло, що і якість зображень, використовуваних у цих системах, також має оцінюватися лише за об'єктивними критеріями.

### 3.9. Порівняння двох зображень

Для порівняння двох зображень у MATLAB використовують декілька функцій. Одна з найчастіше використовуваних – функція визначення абсолютної різниці елементів двох зображень. Опишемо її синтаксис і властивості:

```
Z=imabsdiff(X,Y)
```

Функція `Z=imabsdiff(X,Y)` віднімає кожен елемент зображення `Y` з відповідного елемента зображення `X` і поміщає абсолютну різницю цих елементів у результуючий масив `Z`. Зображення `X` і `Y` мають бути нерозрідженим числовим масивом одного формату і розмірності. Масив `Z` має той же формат і розмірність, що і `X` і `Y`. Коли зображення `X` і `Y` подаються масивом дійсних чисел, результуючі елементи усікаються, оскільки вони перевищують межі.

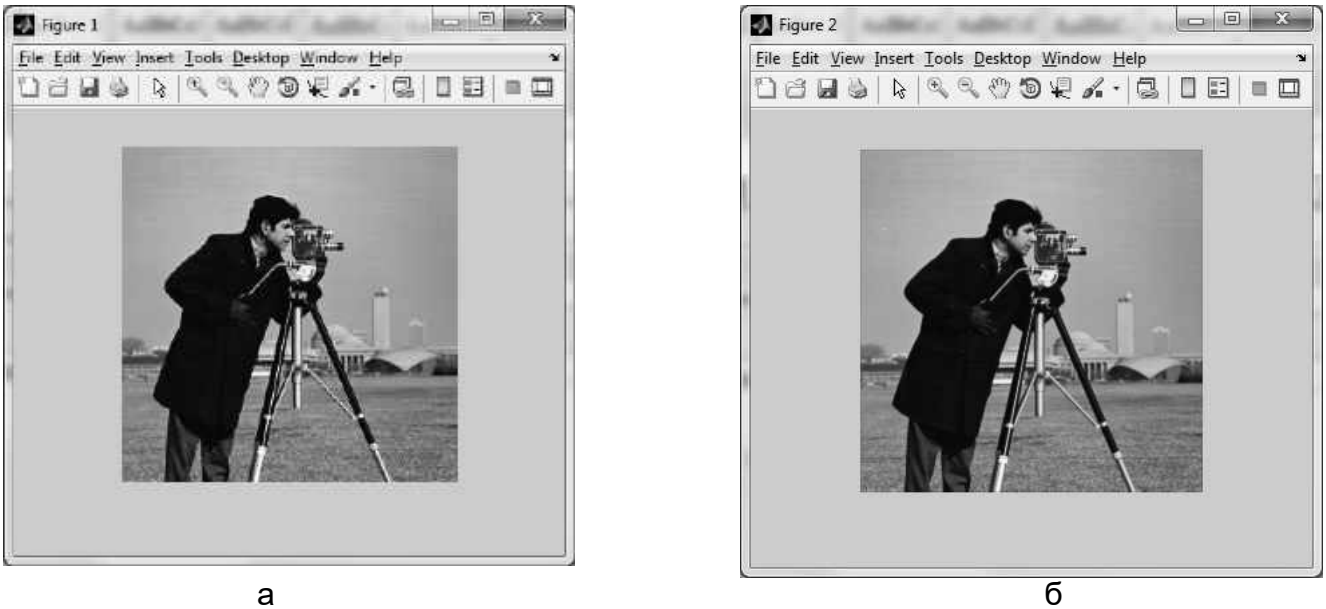
Коли масиви `X` і `Y` мають формат подання даних `double`, то замість цієї функції можна використовувати вираз `abs(X-Y)`. Наведемо приклад обчислення абсолютної різниці між масивами формату `uint8`:

```
X=uint8([ 255 10 75; 44 225 100]);
Y=uint8([ 50 50 50; 50 50 50 ]);
Z=imabsdiff(X,Y)
Z=
205   40     25
 6    175    50
```

На рис. 3.22 показано вихідне зображення і результати його фільтрації фільтром Гаусса.

```
I=imread('cameraman.tif');
J=uint8(filter2(fspecial('gaussian'),I));
K=imabsdiff(I, J);
imshow(K, []) % []=scale data automatically
```

Візуально ці зображення практично невідмітні. Але різниця між ними очевидна, особливо на межах розділу яскравостей. Цю різницю необхідно оцінити кількісно.




а б

Рис. 3.22. Визначення різниці яскравостей двох зображень


в

Рис. 3.22. Закінчення

Наведемо приклад оцінювання якості фільтрації зображення з використанням показника середньоквадратичної помилки MSE.

```
I=imread('cameraman.tif');
imshow(I)
J=uint8(filter2(fspecial('gaussian'),I));
figure,imshow(J)
M=(I-J).^2;
MSE=mean(M(:))

MSE =

   11.5116

h = ones(3,3)/9;
Q = imfilter(I,h);
figure,imshow(Q)
P=(I-Q).^2;
MSE=mean(P(:))

MSE =

   26.7191
```

З цього прикладу очевидно, що якість фільтрації спеціальним фільтром Гаусса (`MSE = 11.5116`) значно краща, ніж якість фільтрації усереднюючим лінійним фільтром (`MSE = 26.7191`).

# 4. ОБРОБЛЕННЯ КОЛЬОРОВИХ ЗОБРАЖЕНЬ

## 4.1. Подання кольорових зображень у MATLAB

Кольорові зображення в додатку Image Processing Toolbox подаються у вигляді повнокольорових (Truecolor) `RGB` (`Red, Green, Blue`) або індексних (`Indexed`) зображень.

Повнокольорове зображення `RGB` – це масив `MxNx3`, що складається з трьох матриць розміром `MxN`. Вони відповідають трьом колірним компонентам: червоному, зеленому і синьому. На рис. 4.1 показано положення монохромних пікселів, відповідних цим матрицям на кольоровому зображенні `RGB`.

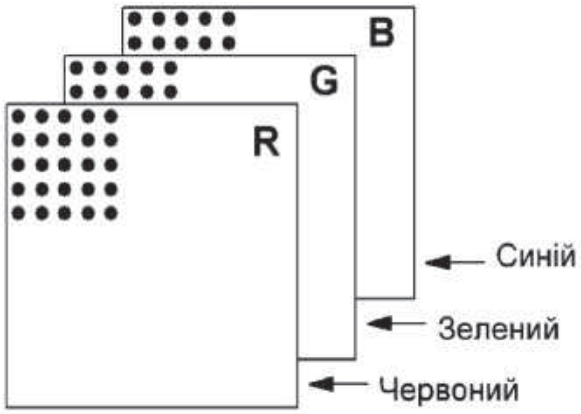



Рис. 4.1. Розподіл монохромних пікселів різного кольору в `RGB` зображенні

Три монохромних зображення, що формують єдине зображення `RGB`, називають червоною, зеленою і синьою компонентами зображення. Їх подають у форматі `uint8, uint16, single` або `double`. Використовують такі числові дані для подання кольорових компонентів:

- `[0,1]` – для форматів `double` і `single`;
- `[0,255]` – для формату `uint8`;
- `[0,65535]` – для формату `uint16`.

Кількість бітів, використовуваних для подання величини кольорового пікселя по всіх складових `RGB`, називається глибиною кольору зображення. Наприклад, якщо кожна компонента є 8-бітовим зображенням, то відповідне зображення `RGB` має глибину 24 біти.

Палітра часто запам'ятовується разом з індексним зображенням і автоматично прочитується разом із зображенням при використанні функції `imread`. Після прочитування зображення і палітри в робочий простір Matlab система сприймає їх як окремі змінні, але між ними існує зв'язок.

Передбачимо, що `IR, IG` і `IB` – три складові `RGB` зображення `I`. Відповідне їм кольорове зображення `RGB` будується за допомогою оператора `cat` (`concatenation` – з'єднання, зчеплення):

```
rgbimage = cat(3,IR,IG,IB)
```

У загальному випадку оператор `cat(dim,a1,A2...)` пов'язує масиви за розмірністю, позначеною в змінній `dim`. Якщо `dim=1`, то масиви розташовуються по вертикалі; при `dim=2` – по горизонталі, а при `dim=3` вони укладаються по третьому виміру так, як показано на рис. 4.1. Очевидно, що для відновлення `RGB` зображення із трьох компонент необхідно вибирати `dim=3`.

Після розкладання вихідного зображення на колірні компоненти можна обробити будь-яку складову як монохромне зображення, а потім за допомогою функції `cat` з'єднати їх в кольорове зображення. Наведемо приклад розкладання `RGB` зображення `I` на колірні компоненти і відновлення за допомогою функції `cat` (рис. 4.2):

```
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\3.jpg');
imshow(RGB)
title('Вихідне зображення');
R=RGB(:,:,1);
G=RGB(:,:,2);
B=RGB(:,:,3);
figure,imshow(R)
title('Red');
figure,imshow(G)
title('Green');
figure,imshow(B)
title('Blue');
RGB1=cat(3,R,G,B);
figure,imshow(RGB1);
title('Відновлене зображення');
```

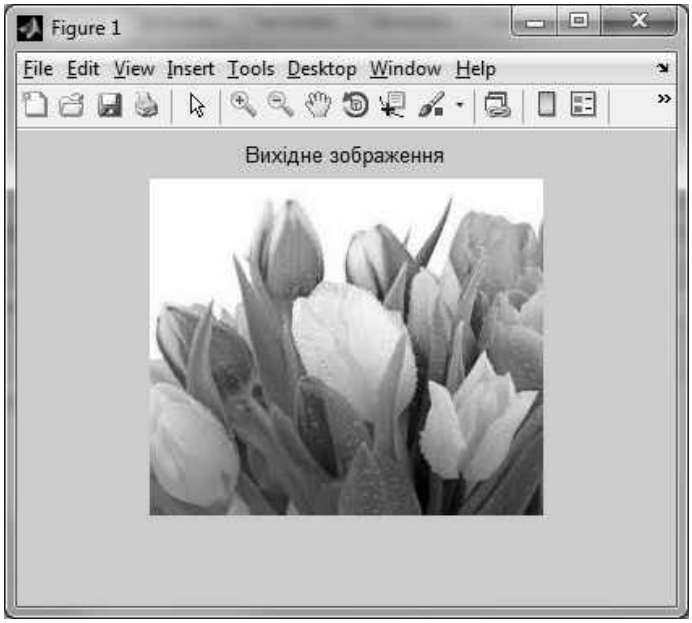



Рис. 4.2. Розкладання зображення на колірні компоненти і відновлення вихідного зображення за допомогою оператора `cat`


Рис. 4.2. Закінчення

***Індексні зображення*** містять два компоненти: числову матрицю даних `X` і матрицю колірної карти `map`, яка є масивом розмірів `mx3` класу `double`. У ньому записані дійсні числа з плаваючою комою з інтервалу `[0,1]`. Довжина m колірної карти дорівнює числу різних кольорів на цьому зображенні. Елементи рядка матриці `map` відображують червону, зелену і синю компоненти кольору. Колір кожного пікселя визначають за допомогою числа, що стоїть у матриці `X`, яке використовується як покажчик на рядок `map`. Якщо `X` наведений у форматі `double`, то всі його компоненти з величинами, меншими або такими, що дорівнюють `1`, вказують на перший рядок `map`, всі компоненти з величинами між `1` і `2` включно вказують на другий рядок і т. д. Якщо `X` наведений у форматі `uint8` або `uint16`, то всі його компоненти, що дорівнюють `0`, вказують на перший рядок `map`, всі компоненти, які дорівнюють `1`, вказують на другий рядок `map` і т. д. Такий метод визначення кольору пікселів показано на рис. 4.3.

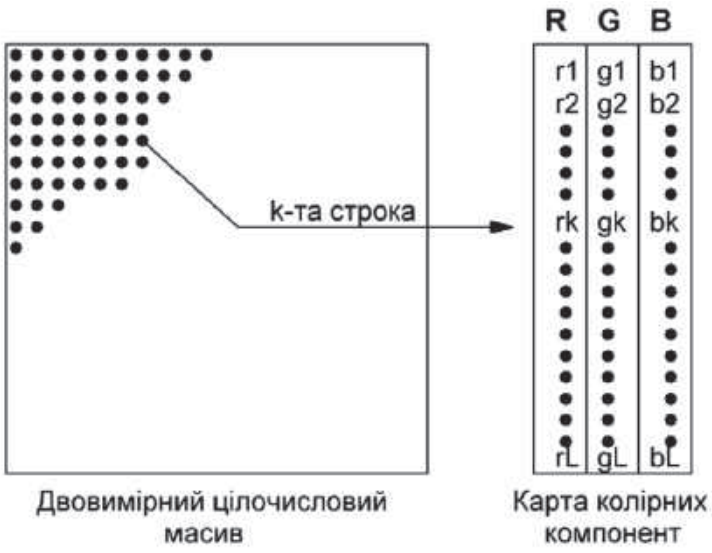



Рис. 4.3. Схема визначення кольору для індексних зображень

Відображення індексного зображення виробляється за командою

```
imshow(X,map);
```

У файлі колірна карта зберігається спільно з індексованим зображенням. Вона автоматично завантажується при прочитуванні зображення за допомогою функції `imread`. Колірну матрицю `map` зазвичай задають у вигляді

```
map(k,:)=[r(k) g(k) b(k)],
```

де `[r(к) g(k) b(k)]` – `RGB` значення, які вказують на рядок карти з номером `k`. Карта заповнюється при зміні параметра `k`.

Для визначення основних кольорів прийнято використовувати три формати їх завдання: довге або коротке ім'я, поміщене в лапки, або числове визначення. Схему кодування основних кольорів індексних зображень наведено в табл. 4.1.

Таблиця 4.1

RGB значення основних кольорів

| Довге им`я | Коротке ім`я | RGB значення |
|---|---|---|
| Black | k | [0 0 0] |
| Blue | b | [0 0 1] |
| Green | g | [0 1 0] |
| Cyan | c | [0 1 1] |
| Red | r | [1 0 0] |
| Magneta | m | [1 0 1] |
| Yellow | y | [1 1 0] |
| White | w | [1 1 1] |

У Matlab міститься декілька стандартних колірних карт, які завантажуються командою `colormap(map_name)`. При цьому матриця колірної карти перетвориться на `map_name`. Наприклад, `colormap(jet)`; де `jet` – одна з колірних карт, наявних у Matlab. Якщо останнє зображення, виведене на екран, було індексним, то після цієї команди колірна карта поміняється на `jet`. Список імен колірних карт у Matlab є таким: `autumn, bone, colorcube, cool, gray, hot, hsv, jet, lines, pink, prism, spring, summer, white, winter`.

Кількість кольорів цих карт можна обмежити, Для цього в круглих дужках встановлюють відповідне число. Наприклад, `gray(16)` задає колірну карту з 16 відтінків сірого кольору.

Інколи буває необхідно апроксимувати індексоване зображення іншим зображенням з меншим набором кольорів. Для цієї мети використовують функцію `imapprox`, виклик якої має вигляд

```
[Y,newmap]=imapprox(X,map,n);
```

Ця функція повертає масив Y з колірною картою `newmap`, який має не більше за `n` кольори. Вхідний масив може належати класу `uint8, uint16` або `double`. Вихід `Y` належить класу `uint8`, якщо `n` не перевершує 256. Якщо `n` більше 256, то `Y` належить класу `double`.

### 4.2. Конвертація колірних систем

Крім `RGB` моделі подання зображень існує ряд моделей, більш відповідних для технічних цілей або більше пристосованих до людського сприйняття.

***Конвертація з RGB в YIQ***. Колірну систему `YIQ` використовують у телевізійній системі `NTSC` подання кольорового телевізійного сигналу. Складова `Y` містить інформацію про яскравість зображення, а ті, що становлять `I` і `Q`, – про його колірність. Конвертація здійснюється за допомогою функції `Yiq=rgb2ntsc(RGB)`, синтаксис якої має вигляд

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

Функція `Yiq=rgb2ntsc(RGB)` створює повнокольорове зображення, значення пікселів якого наведені в колірній системі `YIQ`, з вихідного повнокольорового зображення в колірній системі `RGB`. Результуюче зображення має формат подання даних `double`.

Функція `yiqmap=rgb2ntsc(rgbmap)` створює палітру `yiqmap`, кольори в якій задаються в колірній системі `YIQ`, з вихідної `rgbmap`, що зберігає кольори в колірній системі `RGB`. Перетворення з колірної системи `RGB` в `YIQ` має вигляд

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

***Конвертація з YIQ в RGB***. Зворотне перетворення здійснюється за допомогою такого співвідношення:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.621 \\ 1 & -0.272 & -0.647 \\ 1 & -0.522 & 1.703 \end{bmatrix} * \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}.$$

Конвертація з `YIQ` у `RGB` здійснюється за допомогою функції `RGB=ntsc2rgb(YIQ)`. Вона має такий синтаксис:

```
RGB = ntsc2rgb(YIQ)
gbmap = ntsc2rgb(yiqmap)
```

Функція `RGB=ntsc2rgb(YIQ)` створює повнокольорове зображення, значення пікселів якого наведені в колірній системі `RGB`, з вихідного повнокольорового зображення в колірній системі `YIQ`. Необхідно, щоб вихідне зображення мало формат подання даних `double`. Результуюче зображення також має формат подання даних `double`.

Функція `rgbmap=ntsc2rgb(yiqmap)` створює палітру `rgbmap`, значення кольору в якій задаються в колірній системі `RGB`, з вихідної `yiqmap`, що зберігає кольори в колірній системі `YIQ`. Інші системи телебачення, як аналогові, так і цифрові, використовують схожі колірні системи. Їх відмінність від системи `YIQ` полягає, як правило, в коефіцієнтах, які застосовують для формування складових колірності. Наведемо приклад перетворення зображення в системі `RGB` в систему `YIQ`:

```
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\3.jpg');
imshow(RGB),title('Вихідне зображення у системі RGB');
YIQ=rgb2ntsc(RGB);
figure,imshow(YIQ)
title('Зображення у системі YIQ');
Y= YIQ(:,:,1);
I= YIQ(:,:,2);
Q= YIQ(:,:,3);
figure,imshow(Y)
title('Компонента Y у системі YIQ');
figure,imshow(I)
title(' Компонента I у системі YIQ');
figure,imshow(Q)
title(' Компонента Q у системі YIQ');
YIQ=cat(3,Y,I,Q);
figure,imshow(YIQ);
title('Відновлене зображення YIQ');
```

Результати перетворення зображення з колірної системи RGB у систему YIQ показані на рис. 4.4.

***Конвертація з RGB у Ycbcr***. Колірну систему Ycbcr широко використовують у цифровому відео. Складова Y містить інформацію про яскравість зображення, а складові Cb і Cr (так звані кольоророзницеві складові) – про його колірність. Величина Cb=b-y – це різниця між блакитною компонентою B і світлотою Y, а Cr=r-y – це різниця між червоною компонентою R і Y. Світлота кольорового зображення відповідає характеристиці інтенсивності (півтоновій яскравості) в монохромному випадку. Зазвичай уживається термін «яскравість». Вона (яскравість – рівень сірого кольору) є основною характеристикою монохромних (півтонових) зображень.

Функція Ycbcr=rgb2ycbcr(RGB) призначена для перетворення зображення, наведеного в колірній схемі RGB, у схему Ycbcr і має такий синтаксис:

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

Функція Ycbcr=rgb2ycbcr(RGB) створює повнокольорове зображення, значення пікселів якого подані в колірній системі Ycbcr, з вихідного повнокольорового зображення в колірній системі RGB. Формати подання даних вихідного і результуючого зображень збігаються.

Функція ycbcrmap=rgb2ycbcr(rgbmap) створює палітру ycbcrmap, значення кольору в якій задаються в колірній системі Ycbcr, з вихідної палітри rgbmap, що зберігає кольори в колірній системі RGB.

Перетворення з колірної системи RGB, де складові належать діапазону [0,1], в систему Ycbcr має вигляд:

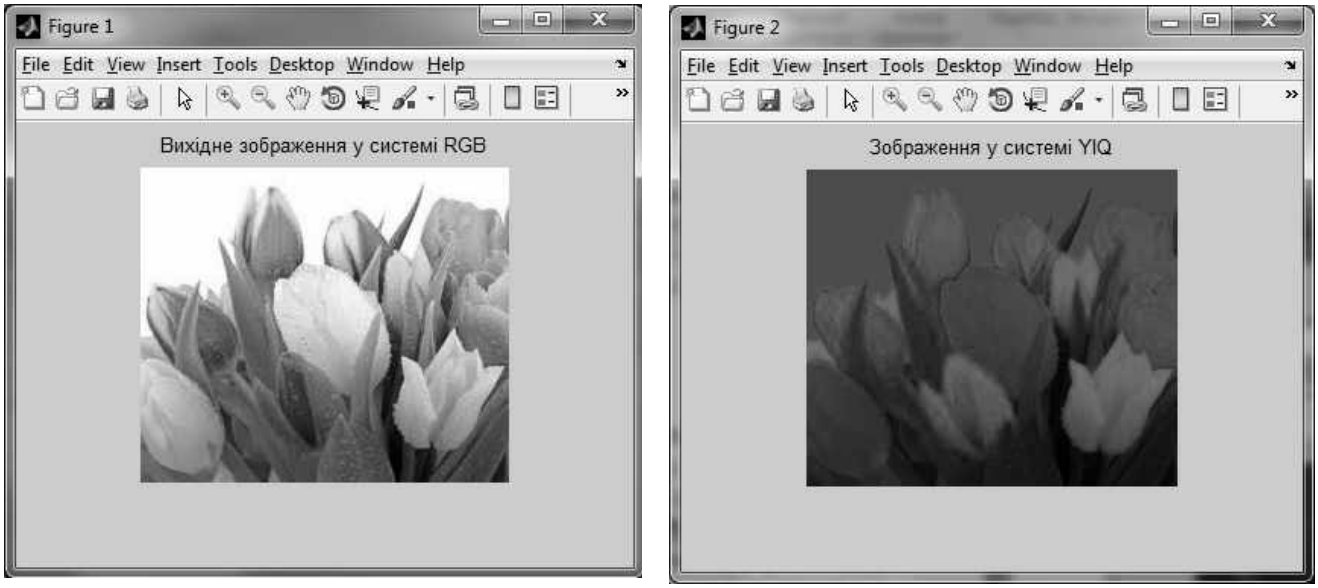



Рис. 4.4. Перетворення зображення з колірної системи RGB у систему YIQ

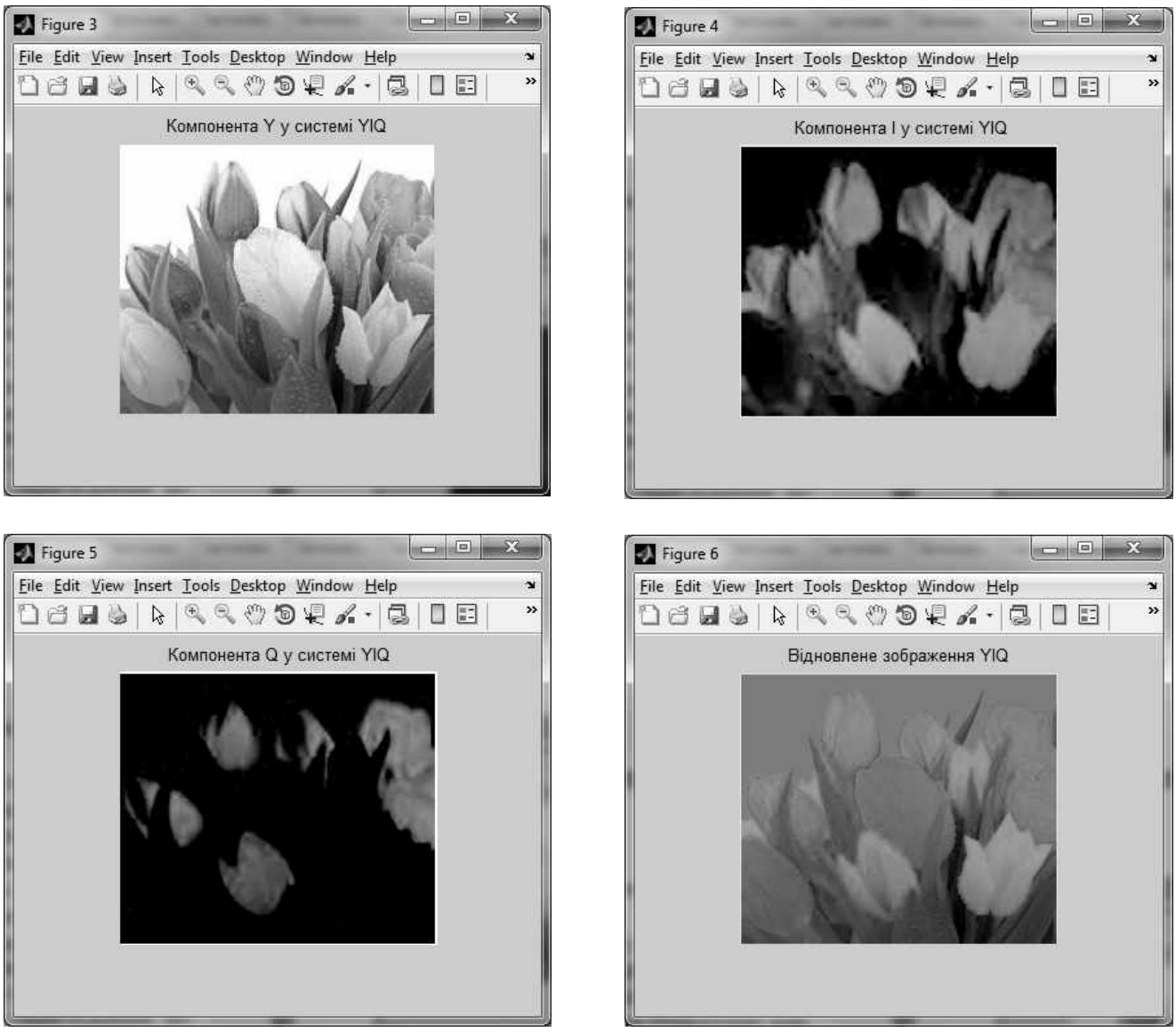



Рис. 4.4. Закінчення

У результаті такого перетворення складова `Y` належить діапазону `[16,235]`, а складові `Cb` і `Cr` належать діапазону `[16,240]`. Допустимі значення, що залишилися, поза вказаними діапазонами (`[0,15]` і `[236,255]` для `Y`, `[0,15]` і `[241,255]` для `Cb, Cr`) використовують для додаткової інформації (наприклад, звуку), яка передається разом із відеопотоком.

***Конвертація з `Ycbcr` у `RGB`*** виробляється за допомогою функції `Rgb=ycbcr2rgb(Ycbcr)`, створює повнокольорове зображення, значення пікселів якого наведені в колірній системі `RGB`, з вихідного повнокольорового зображення в колірній системі `Ycbcr`. Формати подання даних вихідного і результуючого зображень збігаються.

Функція `rgbrmap = ycbcr2rgb(ycbcrmap)` створює палітру `rgbmap`, значення кольору в якій задаються в колірній системі `RGB`, з вихідної палітри `ycbcrmap`, що зберігає кольори в колірній системі `Ycbcr`.

Наведемо приклад (див. рис. 4.5) перетворення зображення із системи RGB у систему Ycbcr:

```
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\3.jpg');
imshow(RGB)
title('Вихідне зображення у системі RGB');
YCbCr=rgb2ycbcr(RGB);
figure,imshow(YCbCr)
title('Зображення у системі YCbCr');
Y= YCbCr(:,:,1);
Cb= YCbCr(:,:,2);
Cr= YCbCr(:,:,3);
figure,imshow(Y)
title('Компонента Y у системі YCbCr');
figure,imshow(Cb)
title('Компонента Cb у системі YCbCr');
figure,imshow(Cr)
title('Компонента Cr у системі YCbCr');
YCbCr =cat(3,Y,Cb,Cr);
figure,imshow(YCbCr);
title('Відновлене зображення HSV');
```

***Конвертація з RGB у HSV***. Перетворення виробляється за допомогою функції `HSV=rgb2hsv(RGB)`, яка має такий синтаксис:

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

Функція `HSV=rgb2hsv(RGB)` створює повнокольорове зображення, значення пікселів якого подані в колірній системі `HSV`, з вихідного повнокольорового зображення в колірній системі `RGB`. Результуюче зображення має формат подання даних `double`.

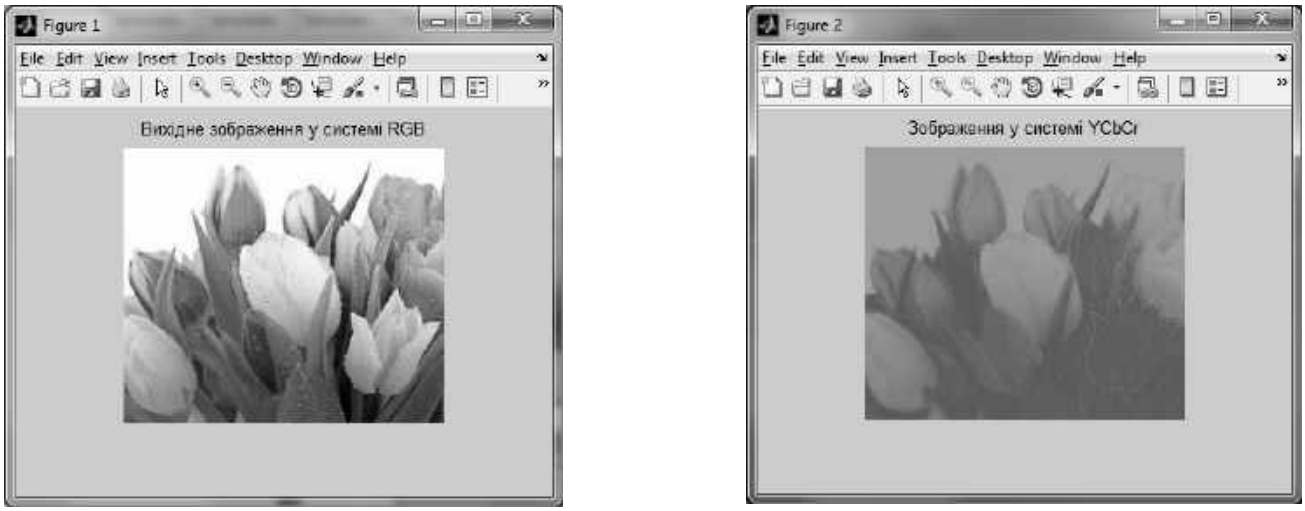



Рис. 4.5. Перетворення зображення з колірної системи `RGB` у систему `Ycbcr`


Рис. 4.5. Закінчення

Функція `hsvmap = rgb2hsv(rgbmap)` створює палітру `hsvmap`, значення кольору в якій задаються в колірній системі `HSV`, з вихідної палітри `rgbmap`, що зберігає кольори в колірній системі `RGB`.

Поширення в комп'ютерній графіці і цифровому обробленні зображень набула колірна система `HSV` (`hue` – колірний фон, `saturation` – насиченість, volume – `світлота`). Часто для назви цієї ж системи використовують абревіатуру `HSB` (`hue, saturation, brightness` – яскравість).

Геометричну модель системи `HSV` вибирають із таких міркувань. Якщо колірний куб `RGB` спроектувати на площину, перпендикулярну діагоналі, на якій в `rgb`-кубі розташовані значення яскравості (відтінки сірого – від чорного до білого), то виходить правильний шестикутник з червоним, жовтим, зеленим, блакитним, синім і пурпурним кольорами у вершинах. При зниженні насиченості `RGB` кольорів зменшується розмір і колірний обхват `rgb`-куба. При цьому відповідна шестикутна проекція також буде менше. Якщо

проекції зібрати навколо осі яскравості V, то вийде перевернутий об'ємний шестигранний конус HSV, показаний на рис. 4.6.

Яскравість V змінюється від 0 у вершині конуса (чорний колір) до 1 всередині підставки конуса (білий колір). На осі V розташовані ахроматичні кольори – відтінки сірого. Насиченість S визначається відстанню до осі V. На ній насиченість дорівнює нулю, а на сторонах конуса – одиниці. Колірний тон H визначається кутом повороту осі S проти годинникової стрілки відносно осі, що проходить через червоний колір.

Колірна система HSV відповідає тому, як складають кольори художники. Чистим кольорам відповідають значення V=1 і S=1, розбілам – кольори зі збільшеним вмістом білого, тобто з меншими значеннями S. Система HSV зручна для вибору кольорів, тому її відносять до колірних систем, наближених до людського сприйняття (perception).

***Конвертація з HSV у RGB***. Перетворення здійснюється за допомогою функції Rgb=hsv2rgb(HSV), яка має такий синтаксис:

```
RGB = hsv2rgb(HSV)
rgbmap = hsv2rgb(hsvmap)
```

Функція Rgb=hsv2rgb(HSV) створює повнокольорове зображення, значення пікселів якого подані в колірній системі RGB з вихідного повнокольорового зображення в колірній системі HSV. Необхідно, щоб вихідне зображення мало формат подання даних double. Результуюче зображення також має формат подання даних double.

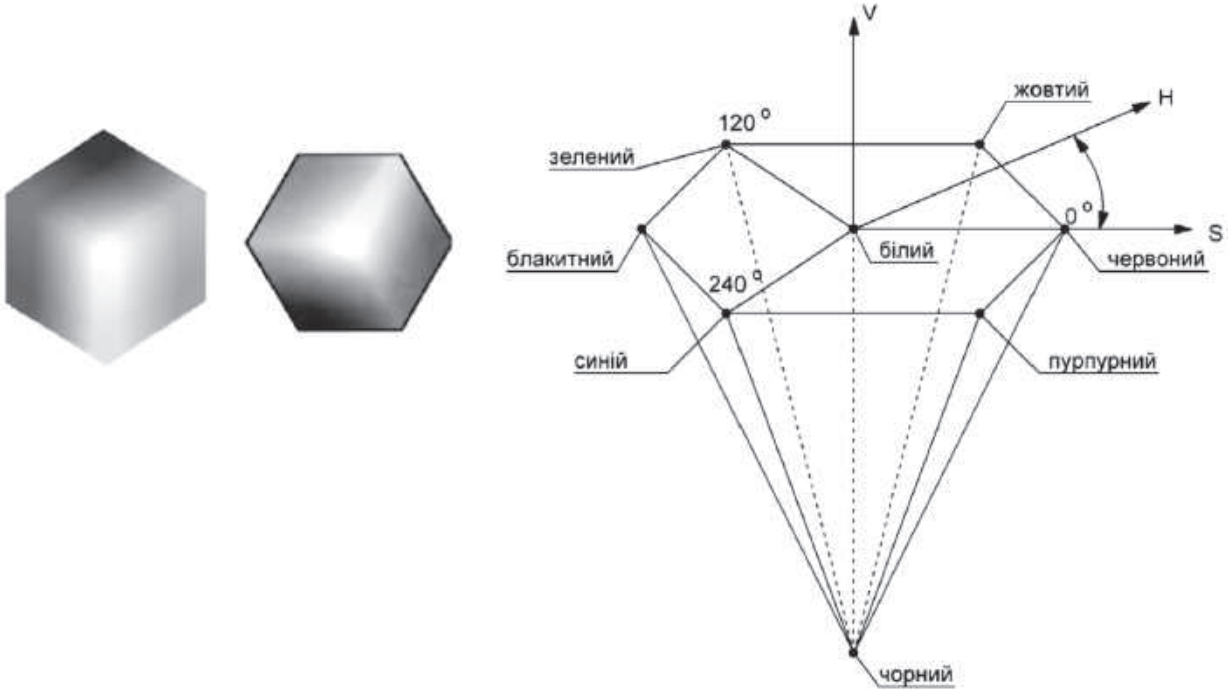



Рис. 4.6. Колірний куб; колірний шестикутник; шестигранний конус HSV

Функція `rgbmap=hsv2rgb(hsvmap)` створює палітру `rgbmap`, значення кольору в якій задаються в колірній системі `RGB`, з вихідної палітри `hsvmap`, що зберігає кольори в колірній системі `HSV`.

Наведемо приклад (див. рис.4.7) перетворення зображення із системи `RGB` у систему `HSV`:

```
>> close all
>> RGB=imread('C:\Users\krasnov.MEDICA\Desktop\3.jpg');
>> imshow(RGB)
>> title('Вихідне зображення у системі RGB');
>> HSV=rgb2hsv(RGB);
>> figure,imshow(HSV)
>> title('Зображення у системі HSV');
>> H = HSV(:,:,1);
>> S = HSV(:,:,2);
>> V = HSV(:,:,3);
>> figure,imshow(H)
>> title('Компонента H у системі HSV');
>> figure,imshow(S)
>> title('Компонента S у системі HSV');
>> figure,imshow(V)
>> title('Компонента V у системі HSV');
>> HSV =cat(3,H,S,V);
>> figure,imshow(HSV);
>> title('Відновлене зображення HSV');
```

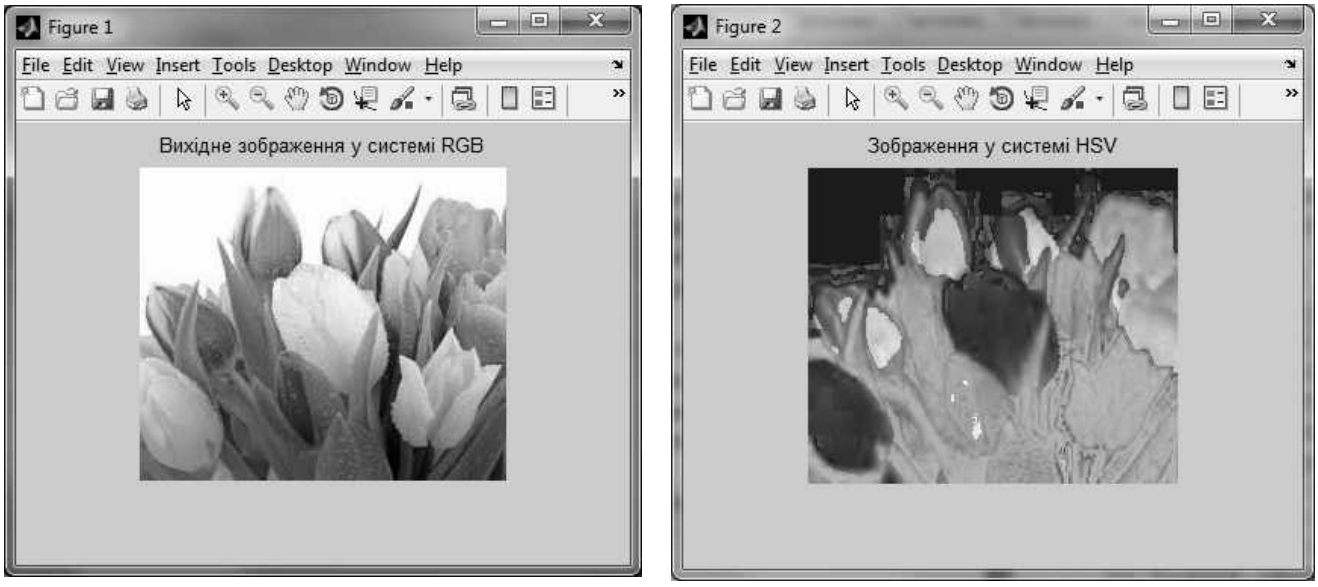



Рис. 4.7. Перетворення зображення з колірної системи `RGB` у систему `HSV`

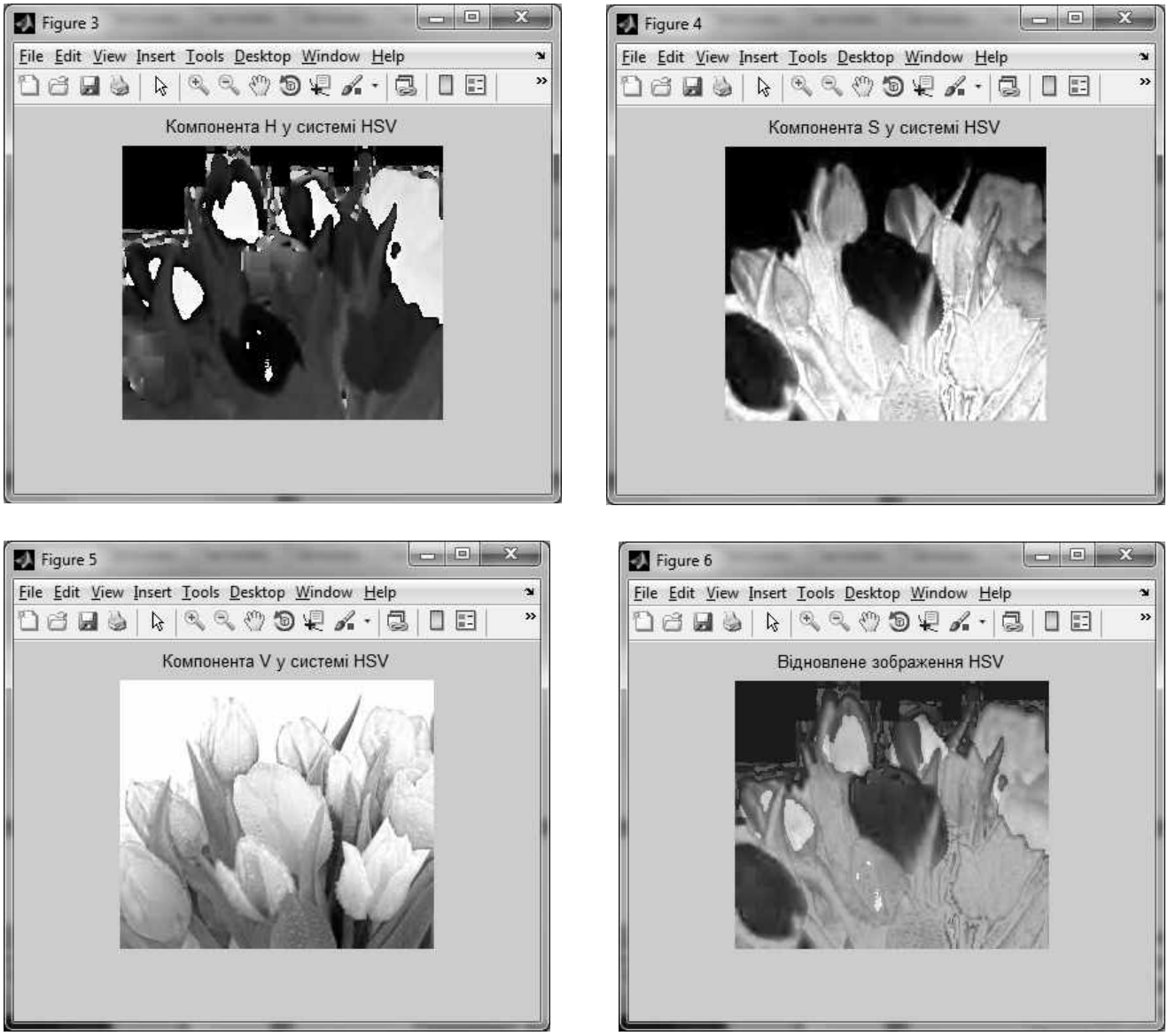



Рис. 4.7. Закінчення

## 4.3. Оброблення кольорових зображень

Існують три основні типи оброблення кольорових зображень:

- перетворення кольорів;
- окреме оброблення колірних компонент;
- розглядання кольору як єдиного вектора в тривимірному просторі.

За допомогою процедур першого типу обробляють пікселі кожної колірної площини, грунтуючись лише на значеннях пікселів, і не враховують їх просторові координати. Ці підходи аналогічні методам, використовуваним при перетворенні яскравості монохромних зображень.

У методах другого типу фактично виконується оброблення `R,G,B` кольорів як окремих півтонових зображень. При цьому можуть використовуватися як лінійні, так і нелінійні алгоритми оброблення окремої колірної площини. У результаті такого незалежного оброблення (наприклад, нелі-

нійного розтягування контрасту) трьох кольорів можуть виникнути неприродні для вихідного зображення спотворення.

У методах третього типу зміни в колірні складові вносяться пропорційно їх «вкладу» в кожен піксель.

Оскільки кольорові зображення мають три компоненти, кольорові пікселі можна розглядати як тривимірні вектори. Наприклад, у системі RGB кожну колірну точку можна розглядати як вектор, проведений з початку координат у відповідну точку колірного простору. Якщо c – це довільний вектор у колірному просторі RGB, то

$$c(x,y) = \begin{bmatrix} C_R(x,y) \\ C_G(x,y) \\ C_B(x,y) \end{bmatrix} = \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix}.$$

Покажемо відмінності при просторовій фільтрації монохромних і кольорових зображень (див. рис. 4.8).

Передбачимо, що проводиться процедура лінійної фільтрації НЧ. При роздільному обробленні усереднювання здійснюється підсумовуванням яскравості всіх півтонових пікселів межі і діленням отриманої суми на загальне число пікселів межі, а при спільному обробленні підсумовують всі вектори межі і розділяють кожну компоненту отриманого сумарного вектора на загальне число векторів межі. Кожна компонента усередненого вектора дорівнює сумі значень, відповідних даній компоненті, що ділиться на число пікселів межі. Це збігається з результатом усереднювання кожної компоненти зображення при роздільному обробленні колірних шарів. Обидві процедури усереднювання дають однаковий результат. Проте така еквівалентність реалізується не завжди, зокрема нелінійні процедури будуть, як правило, давати різні результати.

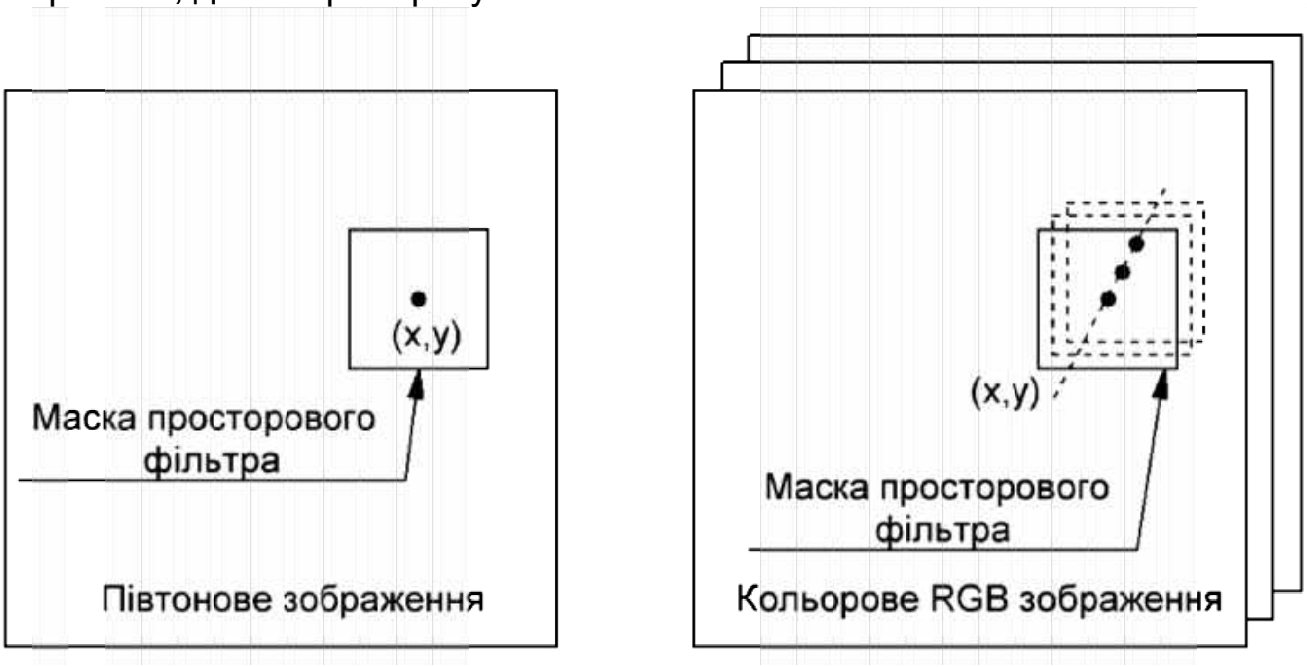



Рис. 4.8. Просторові маски для монохромного і кольорового RGB зображень

Наведемо приклади, що демонструють особливості оброблення кольорових зображень RGB методами роздільного оброблення по окремих колірних шарах і методами узагальненої векторної обробки. Розглянемо приклад лінійної пошарової НЧ-фільтрації зображення, зашумленого гауссовими шумами.

```
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\17.jpg');
imshow(RGB),title(Вихідне зображення');
RGB1 = imnoise(RGB,'gaussian',0,0.01);
figure,imshow(RGB1),title('Зашумлене зображення');
R1=RGB1(:,:,1);
G1=RGB1(:,:,2);
B1=RGB1(:,:,3);
figure,imshow(R1),title('Red');
figure,imshow(G1),title('Green');
figure,imshow(B1),title('Blue');
h=ones(3,3)/9;
R2=imfilter(R1,h);
G2=imfilter(G1,h);
B2=imfilter(B1,h);
figure,imshow(R2),title('Red filter');
figure,imshow(G2),title('Green filter');
figure,imshow(B2),title('Blue filter');
RGB2=cat(3,R2,G2,B2);
figure,imshow(RGB2),title('Відфільтроване зображення по
шарах ');
RGB3=imfilter(RGB1,h);
figure,imshow(RGB3),title('Відфільтроване зображення век-
торно');
```


а б

Рис. 4.9. Початкове (а) і зашумлене (б) RGB зображення


а б

Рис. 4.10. Зашумлена (а) і фільтрована (б) компоненти Red


а б

Рис. 4.11. Зашумлена (а) і фільтрована (б) компоненти `Green`


а б

Рис. 4.12. Зашумлена (а) і фільтрована (б) компоненти `Blue`


а б

Рис. 4.13. `RGB` зображення, фільтроване по шарах (а) і векторно (б)

З наведеного прикладу видно, що пошарова і векторна фільтрації (див. рис. 4.9 – 4.13) привели до передбачено однакового результату. Це зрозуміло, оскільки в цих випадках використовували процедуру лінійної фільтрації. Проте така еквівалентність реалізується не завжди. Нелінійні процедури будуть, як правило, давати різні результати.

Наведемо ще один приклад оброблення цього ж кольорового зображення. Для кожного колірного шару застосуємо процедуру вирівнювання

(еквалізації) гістограми за допомогою функції `histeq` з метою збільшення контрастності.

```
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\17.jpg');
imshow(RGB)
title('Вихідне зображення');
R = RGB(:,:,1);
G = RGB(:,:,2);
B = RGB(:,:,3);
figure,imshow(R),title('Red');
figure, imhist(R),title('Гістограма Red');
figure,imshow(G),title('Green');
figure, imhist(R),title('Гістограма Green');
figure,imshow(B),title('Blue');
figure, imhist(B),title('Гістограма Blue');
RGB1=cat(3,R,G,B);
figure,imshow(RGB1),title('Відновлене RGB зображення');
R1 = histeq(R);
G1 = histeq(G);
B1 = histeq(B);
figure,imshow(R1),title('Red contrast');
figure,imhist(R1),title('Гістограма Red contrast');
figure,imshow(G1),title('Green contrast');
figure,imhist(G1),title('Гістограма Green contrast');
figure,imshow(B1),title('Blue contrast');
figure, imhist(B1),title('Гістограма Blue contrast');
RGB2=cat(3,R1,G1,B1);
figure,imshow(RGB2),title('Відконтрастоване RGB
зображення');
```


Рис. 4.14. Пошарове подання колірних компонент RGB зображення і гістограми розподілу яскравостей окремих шарів

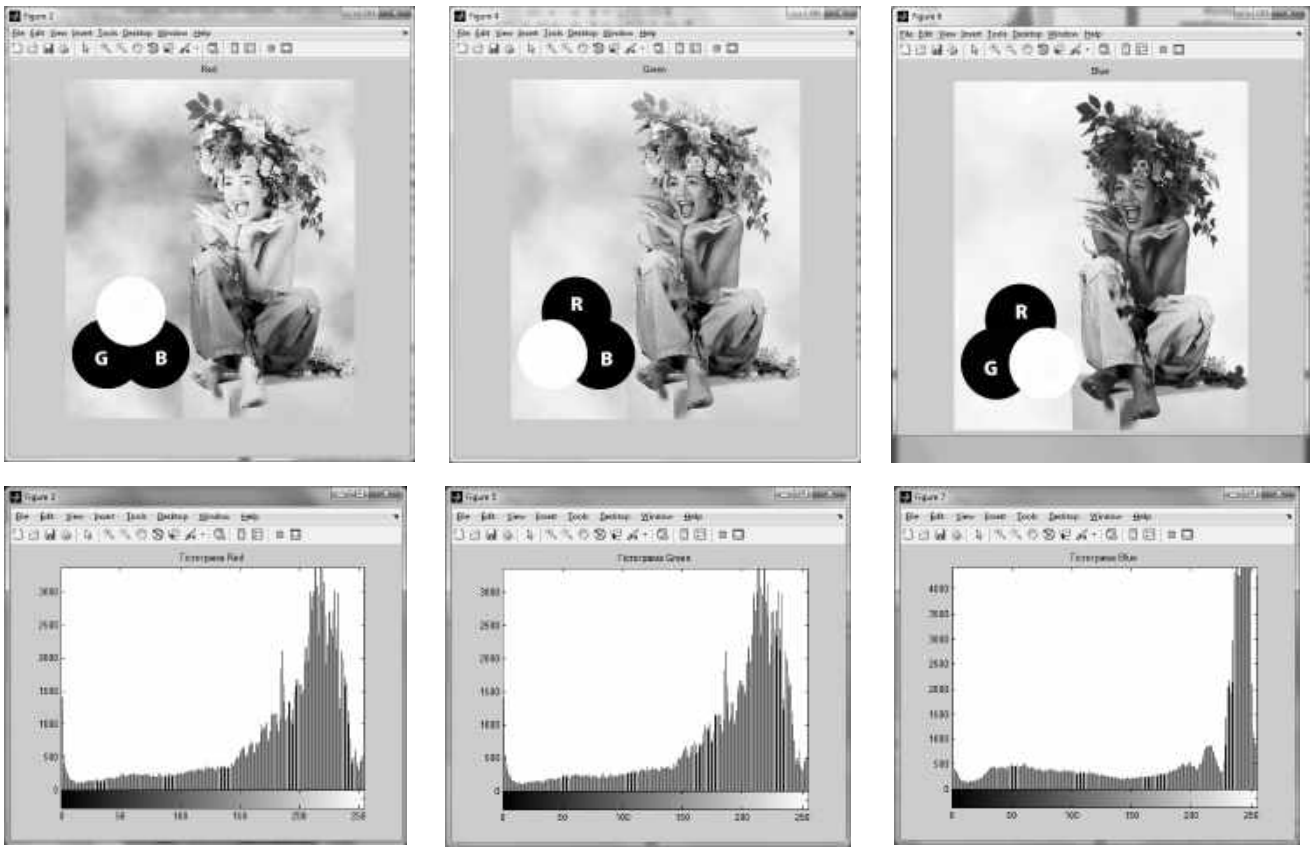



Рис. 4.14. Закінчення

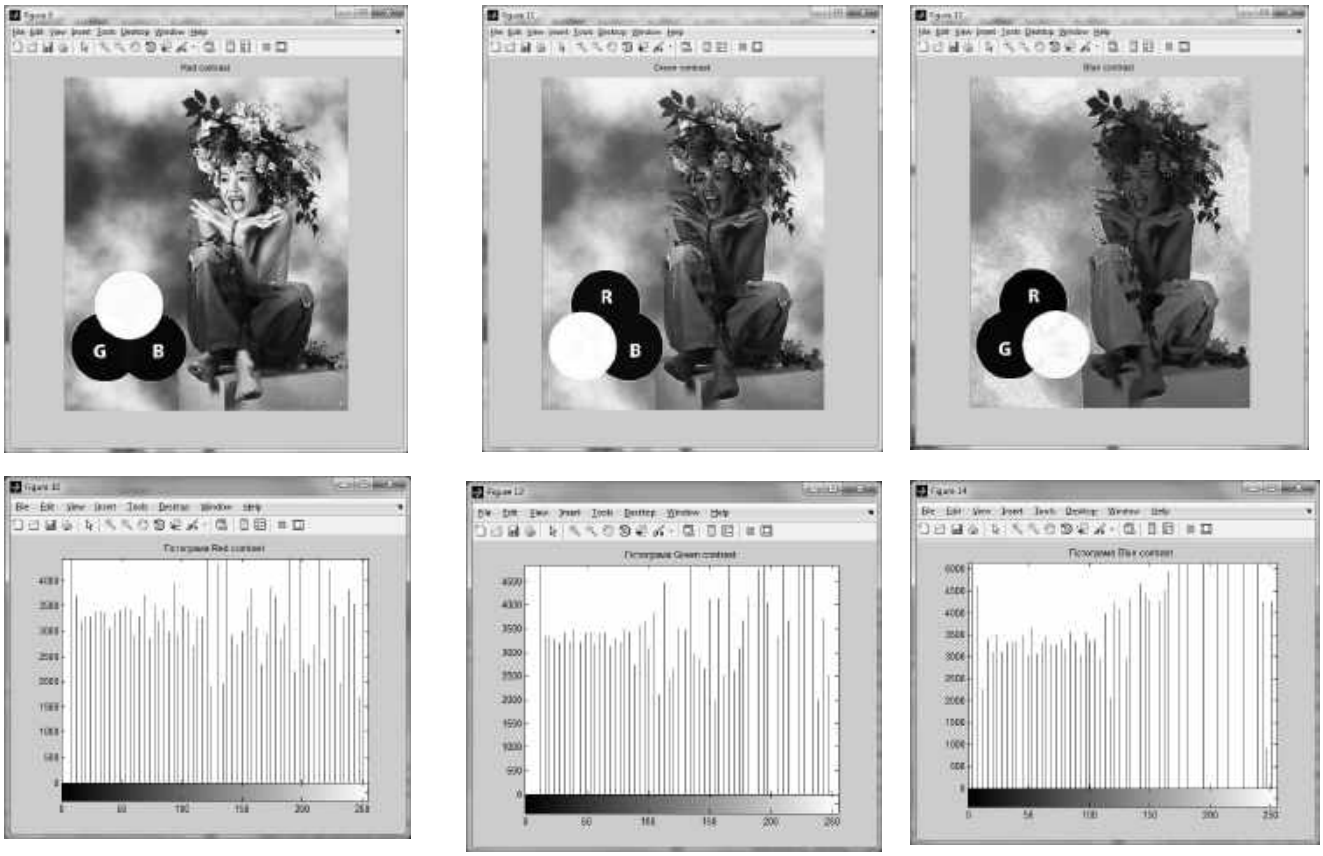



Рис. 4.15. Пошарово контрастовані колірні компоненти, їхні гістограми і відконтрастоване RGB зображення


Рис. 4.15. Закінчення

Цей приклад наочно показує, що процедура еквалізації `RGB` зображення (див. рис. 4.14) приводить до істотного спотворення розподілу яскравостей вихідного зображення після відновлення його за допомогою функції `cat` (рис. 4.15).

Необхідно мати на увазі, що в різних випадках застосування одних і тих же методів оброблення може призвести до різних результатів.

## 5. СПЕКТРАЛЬНИЙ АНАЛІЗ ЗОБРАЖЕНЬ

Одним із найбільш ефективних методів аналізу структури зображень є їхній спектральний аналіз. Розглянемо короткі теоретичні відомості з даного питання і наведемо ряд прикладів.

### 5.1. Дискретне перетворення Фур'є в MATLAB

***Одновимірне ДПФ***. Розглянемо застосування стандартних функцій MATLAB для виконання спектрального аналізу.

Опишемо їх синтаксис:

```
Y = fft(X)          X = ifft(Y)
Y= FFT(X,n)         X = iFFT(Y,n).
```

Дискретні пряме і зворотне перетворення Фур'є для одновимірного масиву `x` довжини `N` визначаються таким чином:

$$X(k) = \sum_{j=1}^{N} x(j) e^{2\pi/N(j-1)(k-1)};$$

$$X(k) = \frac{1}{N} \sum_{k=1}^{N} X(k) e^{-2\pi/N(j-1)(k-1)}.$$

Функція `Y=fft(X)` обчислює для масиву даних `X` дискретне перетворення Фур'є, використовуючи `fft`-алгоритм швидкого Фур'є-перетворення. Якщо масив `X` двовимірний, обчислюється дискретне перетворення кожного стовпця.

Функція `Y=fft(X,n)` обчислює `n`-точкове дискретне перетворення Фур'є. Якщо `length(X)<n`, то бракуючі рядки масиву `X` заповнюються нулями; якщо `length(X)>n`, то зайві рядки віддаляються. Функція `X=ifft(Y)` обчислює зворотне перетворення Фур'є для масиву `Y`.

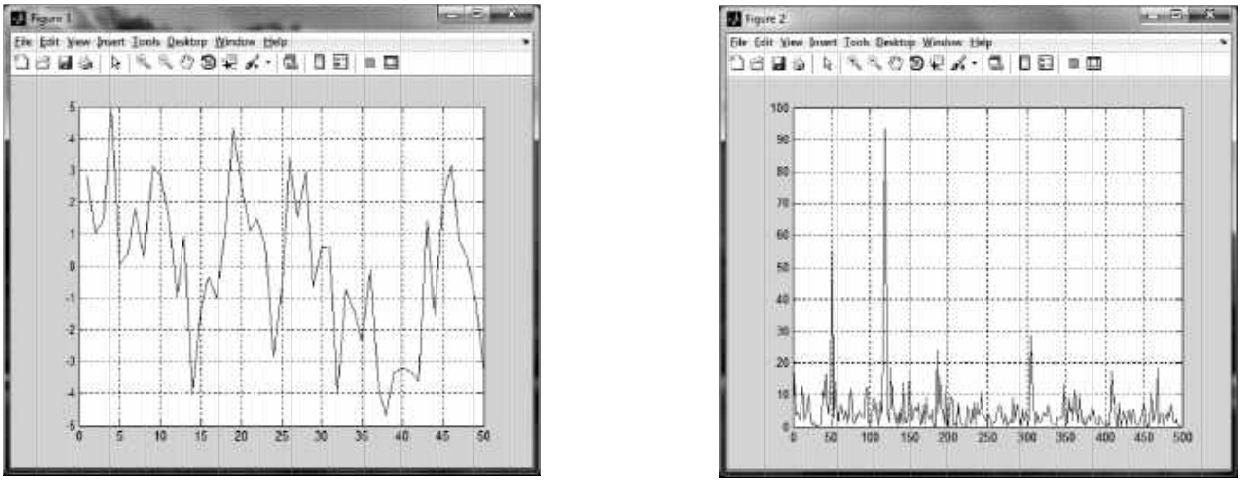
Функція `X=ifft(Y,n)` обчислює `n`-точкове зворотне перетворення Фур'є для масиву `Y`.

***Приклад***. Основне призначення перетворення Фур'є – виділити частоти регулярних складових сигналу, зашумленого перешкодами. Розглянемо дані, що надходять з частотою 1000 Гц. Сформуємо сигнал, що містить регулярні складові з частотами 50 і 120 Гц і випадкову адитивну компоненту з нульовим середнім.

```
t=0:0.001:0.6;
x=sin(2*pi*50*t)+sin(2*pi*120*t);
e=x+2*randn(size(t));
y=x+2*randn(size(t));
plot(y(1:50)),grid
```

На рис. 5.1,а показано цей сигнал. Дивлячись на нього, важко сказати, які частоти мають його регулярні складові. Реалізуючи одновимірне перетворення Фур'є цього сигналу на основі 512 точок і побудувавши графік спектральної щільності, можна виділити дві частоти, на яких амплітуда спектра є максимальною. Це частоти 120 і 50 Гц (рис. 5.1,б).

```
Y=fft(y,512);
Pyy=Y.*conj(Y)/512;
f=1000*(0:255)/512;
figure(2),plot(f,Pyy(1:256)),grid
```



а б

Рис. 5.1. Приклад двовимірного перетворення Фур'є

Двовимірне дискретне перетворення пов'язано з одновимірним дискретним перетворенням Фур'є таким чином:

`fft2(X) = fft(fft(X).').`

### 5.2. Обчислення двовимірного ДПФ досліджуваних зображень

Алгоритм двовимірного масиву ДПФ `f` розміром `MXN` у MATLAB реалізується функцією `F=fft2(f)`. Ця функція повертає як результат також двовимірний масив, але вже коефіцієнтів ДПФ розміром `MXN`, початок масиву знаходиться у верхньому лівому кутку і має координати `(1,1)`.

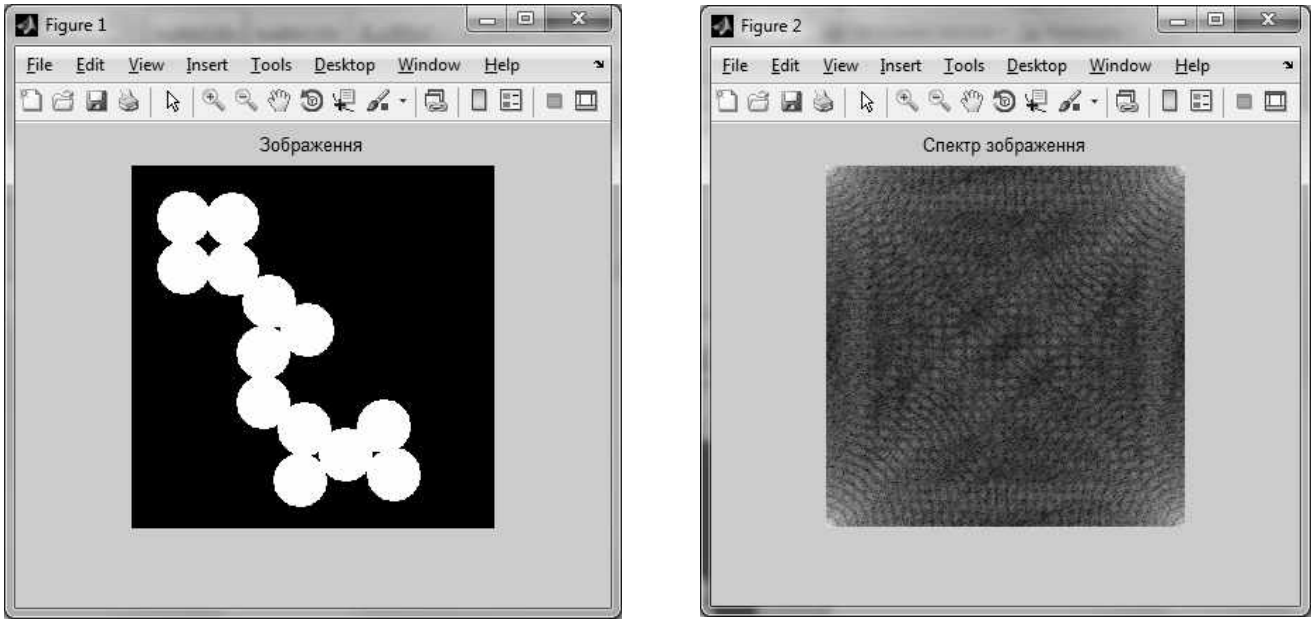
Перетворення Фур'є визначається в комплексному вигляді

`F=Re(F) + jIm(F).`

Тому для його відображення, якщо це потрібно, необхідно знайти модуль спектра – корінь квадратний із суми квадратів дійсної і уявної компонент – `S=abs(F)`.

Відображення спектра (рис. 5.2) так само, як і вихідного зображення, виробляється з використанням функції `imshow(S[ ])`.

```
clear all,close all;
f = imread('circles.png');
imshow(f),title('Зображення');
F = fft2(f);
S=abs(F);
Slog=log(1+S);
figure,imshow(Slog,[ ]),title('Спектр зображення');
```

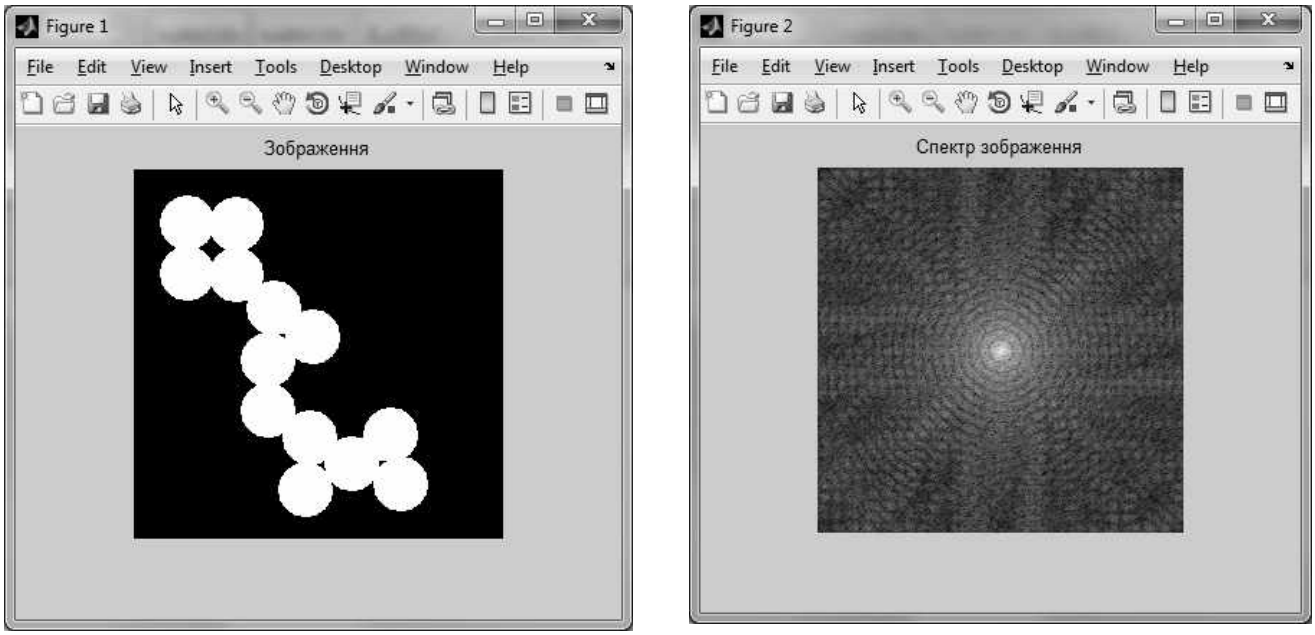



а б

Рис. 5.2. Бінарне зображення (а) і його спектр (б)

У цьому прикладі (рис. 5.2) замість модуля спектра відображувався його логарифм, оскільки масштаб отримуваних у результаті ДПФ чисел значно відрізняється від масштабу самого зображення. Але це не істотно. Інколи зручніше відображувати спектр зображення з початком не в точках `(1,1)`, а в центрі частотної області – в середині прямокутника `MXN` (див. рис. 5.3). Для цього використовують функцію `fftshift`.

```
f=imread('circles.png');
imshow(f),title('Зображення');
F=fft2(f);
S=abs(F);
Fc=fftshift(S);
Slog=log(1+Fc);
figure,imshow(Slog,[ ]),title('Спектр зображення');
```




а б

Рис. 5.3. Бінарне зображення (а) і його спектр у центрі частотної області (б)

За допомогою команди `F=fftshift(Fc)` можна повернути спектр до первинного вигляду (з нумерацією від нульової точки у верхньому лівому куті). Відновлення зображення за його спектром, яке виробляється з використанням функції `f=ifft2(F)`, показано на рис. 5.4.

Приклад:

```
f = imread('rice.png');
imshow(f),title('Зображення');
F = fft2(f);
```

```
Fc = fftshift(F);
Fdc = ifftshift(Fc);
If = ifft2(Fdc);
figure, imshow(If,[ ]),title('Зображення після ДПФ -
ОДПФ');
```

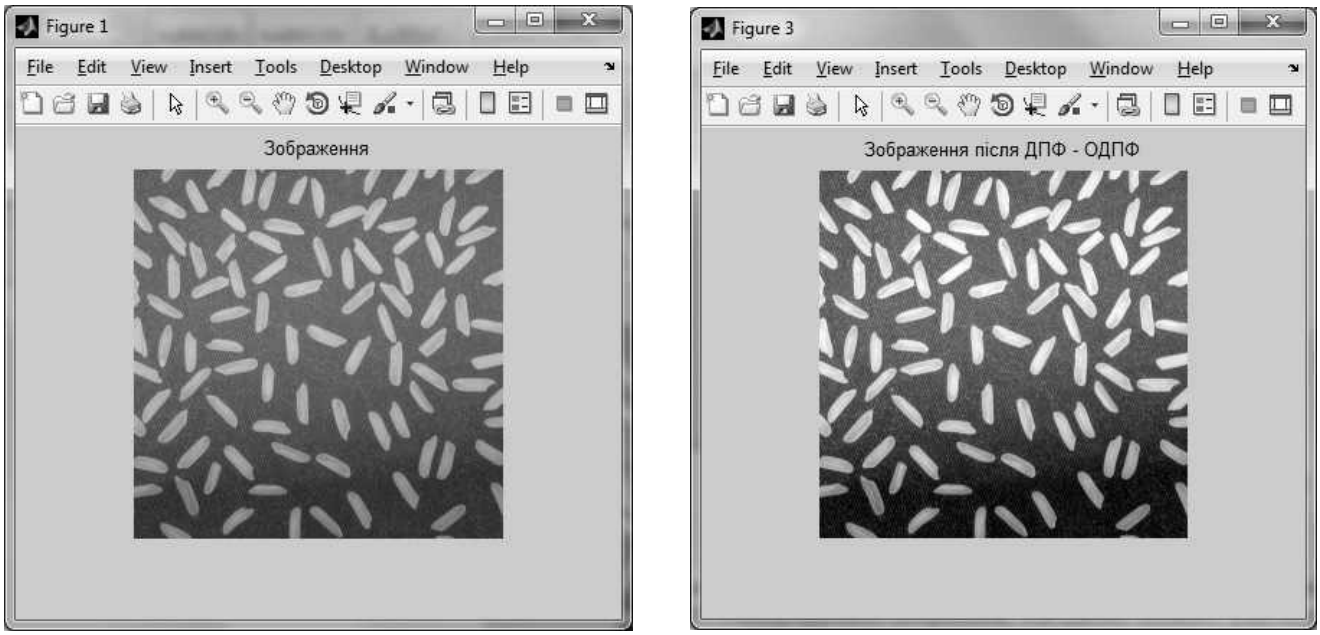



Рис. 5.4. Відновлення зображення за його спектром

## 6. СТИСКУВАННЯ ЗОБРАЖЕНЬ

За останні десятиліття значний розвиток отримали системи зберігання і оброблення цифрових зображень. На цей момент уже існують сховища даних, що налічують сотні мільйонів оцифрованих зображень. Такі бази даних використовують у найрізноманітніших областях, наприклад, у медицині (рентгенівські знімки), криміналістиці (відбитки пальців), у сервісах для зберігання фотографій у мережі Internet. Внаслідок цього витрачаються значні зусилля на розроблення нових технологій, що дозволяють підвищити ефективність стискування цифрових зображень.

У цей час найбільший обсяг даних відповідає зображенням, відеоданим і звуку. Як показує практика, використання методів стискування без втрат не приводить до істотного скорочення їх обсягу, тому актуальним завданням стає дослідження методів стискування з втратами, орієнтованих на конкретні типи даних.

У цьому розділі розглянемо основні теоретичні відомості про методи стискування зображень, основні алгоритми і стандарти стискування графічних даних, а також ресурси додатка Image Processing Toolbox для вирішення цих завдань.

## 6.1. Основні поняття про стискування зображень

Алгоритми стискування, що розглядаються нижче, орієнтовані на реальні зображення, що отримуються, наприклад, за допомогою цифрового фотоапарата або за допомогою сканування, або системами захоплення кадрів із відео і т. п. У будь-якому випадку передбачатимемо, що у нас є повнокольорове зображення, що є двовимірним масивом, елементи якого містять колір відповідної точки. При цьому завданням кодування є подати вихідне зображення якомога меншим числом байтів порівняно з вихідним розміром.

Міру стискування прийнято визначати або як коефіцієнт стискування, який дорівнює

$$k = \frac{S_{\text{код}}}{S_{\text{вих}}},$$

де $S_{\text{код}}$ – розмір закодованого (стислого) зображення; $S_{\text{вих}}$ – розмір вихідного зображення, або як чинник стискування:

$$f = \frac{1}{k} = \frac{S_{\text{вих}}}{S_{\text{код}}}.$$

Таким чином, коефіцієнт завжди менше 1, а чинник стискування більше 1. Зазвичай при порівнянні алгоритмів стискування використовують чинник стискування і свідчать, наприклад, що те або інше зображення є стислим у 2 рази, і це означає, що чинник стискування дорівнює 2.

Всі алгоритми стискування зображень можна умовно розбити на два основні класи – це алгоритми стискування без втрат якості при відновленні закодованого зображення і алгоритми з деякою втратою якості відновлення. У першому випадку гарантується точна відповідність між вихідним і відновленим зображеннями, але при цьому досягається, як правило, невисока міра стискування, зазвичай 2 – 3.

При стискуванні з деякою втратою якості, яка є малопомітною для ока, удається отримати вищу компресію зображень в 10 і більше разів. При цьому вибір на користь того чи іншого алгоритму стискування слід робити залежно від конкретного прикладного завдання.

Коли мова заходить про стискування з втратою якості, то виникає питання: яким чином слід оцінювати якість відновленого зображення? Універсального критерію оцінювання якості зображень не існує, тому для визначення, наскільки добре було відновлено зображення, користуються найбільш відповідними критеріями з тих, що відомі. Розглянемо детальніше найбільш поширені критерії якості оцінювання зображень.

## 6.2. Метрики помилок при стискуванні зображень

Розробникам методів стискування зображень з частковою втратою інформації необхідні стандартні метрики для вимірювання розбіжності від-

новлених зображень і вихідних зображень. Чим ближче відновлений образ до початкового, тим більше має бути ця метрика (її зручно називати «метрикою схожості»). Ця метрика має бути безрозмірною і не дуже чутливою до малих змін відновлюваного зображення. Загальноприйнятою величиною, використовуваною для цих цілей, служить пікове відношення сигнал/шум (PSNR) (peak signal to noise ratio). Воно відоме всім, хто працює в цій області, його легко обчислювати, але воно має досить обмежене, наближене відношення до розбіжностей, які виявляються органами зору людини. Високе значення PSNR означає певну схожість реконструйованого і вихідного зображень, але воно не дає гарантію гарного зорового сприйняття.

Питання використання критеріїв якості оброблення зображень досить детально обговорювалися в розділі 3.5 цієї роботи для аналізу спотворень при проведенні фільтрації. Ці методи оцінювання якості зазвичай застосовують і при стискуванні зображень. Тому не зупинятимемося на цьому детально. Відзначимо лише, що чим більше схожість між образами, тим менше величина MSE, а значить, більше PSNR. Число PNSR є безрозмірним, оскільки одиницями вимірювання і чисельника, і знаменника служать величини пікселів. Проте через використання логарифмів говориться, що число PSNR вимірюється в децибелах (дБ). Використання логарифмів згладжує MSE, робить цю величину менш чутливою. Наприклад, ділення MSE на 10 означає множення PSNR на 2. Відзначимо, що PSNR не має абсолютного значення. Немає сенсу говорити, що якщо PSNR дорівнює, наприклад, 25, то це добре. Величини PSNR використовують лише для порівняння продуктивності різних методів стискування і для вивчення впливу різних параметрів на продуктивність того чи іншого алгоритму. Наприклад, комітет MPEG використовує суб'єктивний поріг Psnr = 0.5 дБ при включенні кодової оптимізації, оскільки вважає, що поліпшення на цю величину буде помітне оку.

Зазвичай величина PSNR варіюється в межах від 20 до 40. Якщо значення пікселів знаходяться в інтервалі [0 … 255], то MSE, яке дорівнює 25.5, дає PSNR, яке дорівнює 20, а при MSE, що дорівнює 2.55, величина PSNR – 40. Значення MSE, яке дорівнює нулю (збіг зображень), дає для PSNR результат − нескінченність (точніше − невизначеність).

### 6.3. Використання алгоритму стискування зображень JPEG

Алгоритм був розроблений групою експертів в області фотографії (Joint Photographic Expert Group) спеціально для стискування 24-бітових і півтонових зображень у 1991 році. Цей алгоритм не дуже добре стискує дворівневі зображення, але він прекрасно обробляє зображення з безперервними тонами, в яких близькі пікселі, як правило, мають схожі кольори. Зазвичай око не в змозі відмітити якої-небудь різниці при стискуванні цим методом в 10 або 20 разів.

Алгоритм базується на двох основоположних процедурах:

1. Кодуванні не самих даних, а деякого лінійного перетворення від цих даних.
2. Квантуванні (округленні) коефіцієнтів лінійного перетворення.

Як лінійне перетворення для JPEG було вибрано Двовимірне Дискретне Перетворення (ДКП або DCT) Косинусне, що забезпечує алгоритму достатню простоту реалізації, високу міру стискування і досить високу якість стислого зображення.

Двовимірне дискретне косинусне перетворення переводить зображення з області просторових змінних (з подання набором відліків або пікселів) у спектральну область (подання набором частотних складових). Розглянемо властивості ДКП детальніше.

***Дискретне косинусне перетворення*** подає зображення у вигляді суми синусоїд з різною амплітудою і частотою. Функція `dct2` в додатку Image Processing Toolbox реалізує двовимірні дискретні косинусні перетворення зображень. Одна з особливостей дискретного перетворення Фур'є полягає в тому, що деякі локальні ділянки зображення можна охарактеризувати невеликою кількістю коефіцієнтів дискретного перетворення Фур'є. Цю властивість дуже часто використовують при розробленні методів стискування зображень.

Двовимірне дискретне косинусне перетворення матриці $\boldsymbol{A}$ з розмірами $\boldsymbol{N \times M}$ реалізується згідно з таким виразом:

$$\boldsymbol{B_{pq} = \alpha_p \alpha_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N}},$$

де $\boldsymbol{0 \le p \le 1}$ и $\boldsymbol{0 \le q \le N-1}$.

$$\boldsymbol{\alpha_p} = \begin{cases} \frac{1}{\sqrt{M}}, \text{ якщо } \boldsymbol{p = 0}; \\ \sqrt{\frac{2}{M}}, \text{ якщо } \boldsymbol{1 \le p \le M-1}. \end{cases} \qquad \boldsymbol{\alpha_q} = \begin{cases} \frac{1}{\sqrt{N}}, \text{ якщо } \boldsymbol{q = 0}; \\ \sqrt{\frac{2}{N}}, \text{ якщо } \boldsymbol{1 \le q \le N-1}. \end{cases}$$

Значення $\boldsymbol{B_{pq}}$ називають коефіцієнтами дискретного косинусного перетворення матриці $\boldsymbol{A}$. Слід зазначити, що індекси матриці в MATLAB завжди починаються з `1`, а не з `0`. Тому елементи матриці, які подані в MATLAB як $\boldsymbol{A(1,1)}$ і $\boldsymbol{B(1,1)}$, відповідатимуть елементам $\boldsymbol{A_{00}}$ і $\boldsymbol{B_{00}}$ з наведеної вище формули.

Зворотне дискретне косинусне перетворення реалізується згідно з виразом

$$\boldsymbol{A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} \alpha_p \alpha_q B_{pq} cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N}},$$

де $\mathbf{0} \leq \boldsymbol{m} \leq \boldsymbol{M} - \mathbf{1}$ і $\mathbf{0} \leq \boldsymbol{n} \leq \boldsymbol{N} - \mathbf{1}$.

Вираз зворотного дискретного косинусного перетворення може інтерпретуватися як подання матриці $\boldsymbol{A}$ з розмірами $\boldsymbol{N} \times \boldsymbol{M}$ у вигляді суми $\boldsymbol{N} \times \boldsymbol{M}$ таких функцій:

$$\boldsymbol{\alpha_p \alpha_q cos \frac{\pi(2m+1)p}{2M} cos \frac{\pi(2n+1)q}{2N}}$$ , де $\mathbf{0} \leq \boldsymbol{p} \leq \boldsymbol{M} - \mathbf{1}$ і $\mathbf{0} \leq \boldsymbol{q} \leq \boldsymbol{N} - \mathbf{1}$.

Ці функції називаються основними (базовими) функціями дискретного косинусного перетворення. Коефіцієнти дискретного косинусного перетворення $\boldsymbol{B_{pq}}$ можна розглядати як вагові при кожній базовій функції. Наприклад, для матриці з розміром 8x8 елементів існує 64 базових функції, що продемонстровано на рис. 6.1.

Горизонтальні частоти збільшуються зліва направо, а вертикальні – зверху вниз. Стосовно двовимірного ДКП (ця функція в лівому верхньому кутку) та її вигляд означає, що спектральна складова $\boldsymbol{DCT}(\mathbf{0}, \mathbf{0})$ виходить як сума всіх пікселів блока зображення 8x8 (тобто як середня яскравість блока).

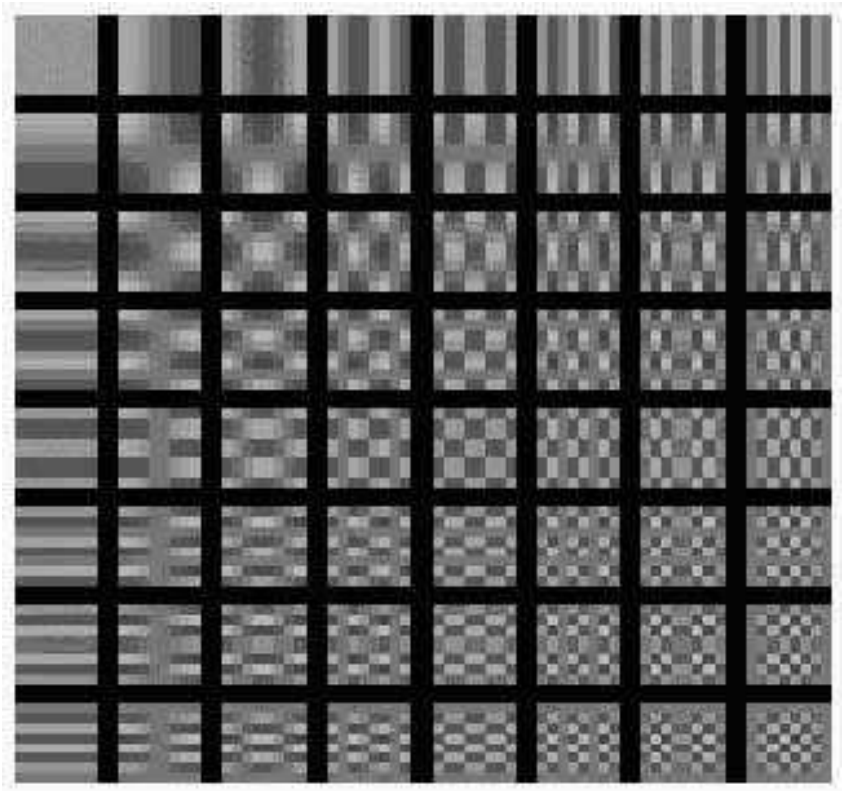


Рис. 6.1. 64 базові функції, отримані для матриці з розмірами 8x8 елементів

Наступна по черзі базисна функція $\boldsymbol{u} = \mathbf{1}$ змальована нижче і є половиною періоду косинусоїди по одній координаті і константою – по іншій. Стосовно двовимірного $\boldsymbol{DCT}$ це означає, що спектральна складова з координатами `(1,0)` є сумою всіх пікселів блока зліва від середини блока, мі-

нус суму всіх пікселів справа (різниця яскравостей лівої і правої частин блока 8x8) і т. д.

При цьому чим нижче і правіше в матриці $DCT$ його компонента, тим більше високочастотним деталям зображення вона відповідає.

***Матриця дискретних косинусних перетворень***. У додатку Image Processing Toolbox пропонується два різних шляхи реалізації дискретних косинусних перетворень. Перший метод реалізований у функції `dct2`. Вона використовує швидке перетворення Фур'є для прискорення обчислень. Другий метод використовує матрицю дискретних косинусних перетворень, яка повертається функцією `dctmtx`. Матриця перетворень `T` формується згідно з таким виразом:

$$T_{pq} = \begin{cases} \frac{1}{\sqrt{M}} \text{ при } p = 0, 0 \le q \le M-1; \\ \sqrt{\frac{2}{M}} cos \frac{\pi(2q+1)p}{2M} \text{ при } 1 \le p \le M-1, 0 \le q \le M-1. \end{cases}$$

Для матриці $A$ з розмірами $M \times M$ $T \cdot A$ є матрицею з розмірами $M \times M$, де кожен стовпець містить одновимірне дискретне косинусне перетворення $A$. Двовимірне дискретне косинусне перетворення $A$ обчислюється як $B = T^* A^* T'$. Зворотне двовимірне дискретне косинусне перетворення B обчислюється як $T'^* B^* T$.

***Дискретні косинусні перетворення і стискування зображень***. В алгоритмі стискування зображень JPEG вихідне зображення розділяється на блоки з розмірами 8x8 або 16x16 елементів. Далі для кожного блока обчислюється двовимірне дискретне косинусне перетворення. Коефіцієнти дискретних косинусних перетворень квантуються, кодуються і передаються. Одержувач jpeg-даних декодує коефіцієнти дискретного косинусного перетворення, обчислює зворотне двовимірне дискретне косинусне перетворення в кожному блоці і далі поєднує їх разом в одне зображення.

Розглянемо приклад обчислення двовимірних дискретних косинусних перетворень у блоках з розмірами 8x8 елементів вихідного зображення. Далі при реконструкції зображення враховуватимемо лише 10 коефіцієнтів з кожного блока, останні прирівняємо до нуля. При проведенні описаних обчислень застосовують також матрицю перетворень.

```
I = imread('cameraman.tif');
I = im2double(I);
T = dctmtx(8);
B = blkproc(I,[8 8],'P1*x*P2',T,T');
```

```
   mask = [1   1   1   1   0   0   0   0
           1   1   1   0   0   0   0   0
           1   1   0   0   0   0   0   0
```

```
     1    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0
     0    0    0    0    0    0    0    0];
B2 = blkproc(B,[8 8],'P1.*x',mask);
I2 = blkproc(B2,[8 8],'P1*x*P2',T',T);
imshow(I);
figure,imshow(I2)
```

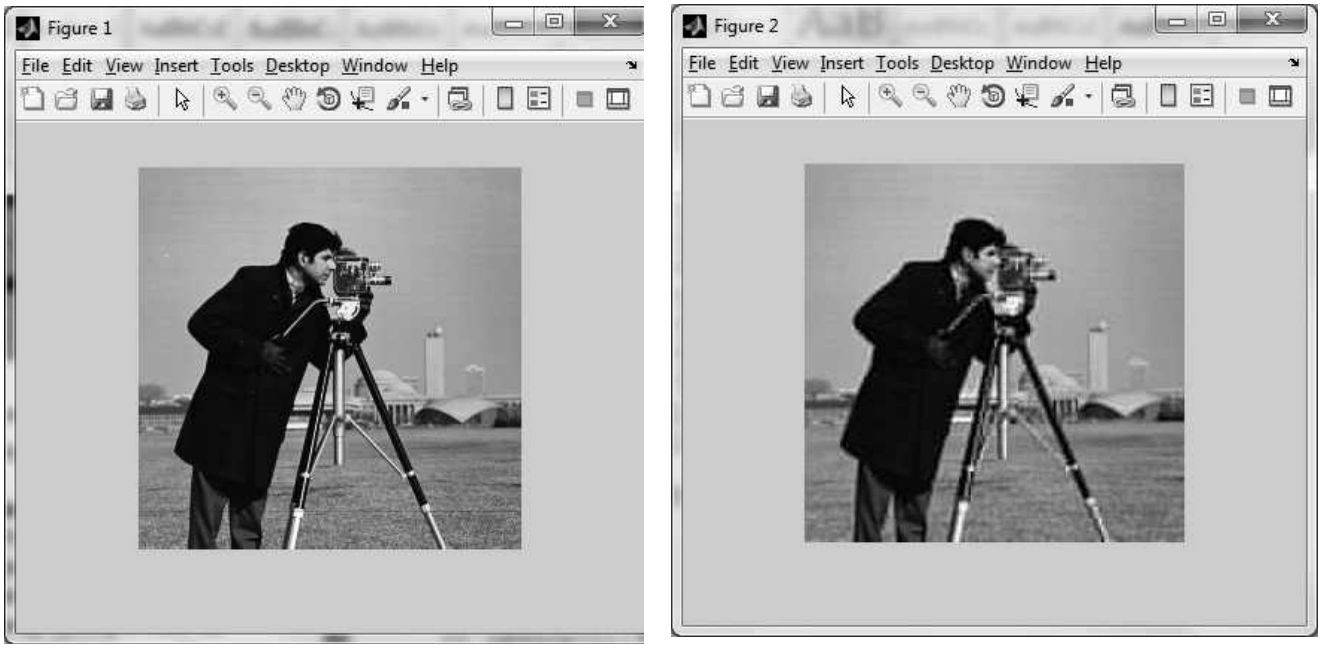



Рис. 6.2. Вихідне і стисле зображення

На рис. 6.2 показано два зображення – початкове (зліва) і реконструйоване (справа). При реконструкції зображення використовували лише 15% коефіцієнтів дискретних косинусних перетворень. Проте слід зазначити, що якість реконструйованого зображення є досить прийнятною. Для перегляду інших властивостей дискретного косинусного перетворення див. функцію `dctdemo`. Процедура двовимірного $\boldsymbol{DCT}$ у MATLAB також реалізується за допомогою функції

```
J = dct2(I).
```

Функція `J = dct2(I)` повертає двовимірне $\boldsymbol{DCT}$ масиву зображення `I`. Матриця DCT `J` має той же розмір, що і зображення `I`. Наведемо приклад процедури прямого і зворотного $\boldsymbol{DCT}$.

```
close all;
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\8.jpg');
imshow(RGB)
```

```
I=rgb2gray(RGB);
figure,imshow(I)
J = dct2(I);
figure; imshow(log(abs(J)),[])
K = idct2(J),figure, imshow(K,[0 255]);
```

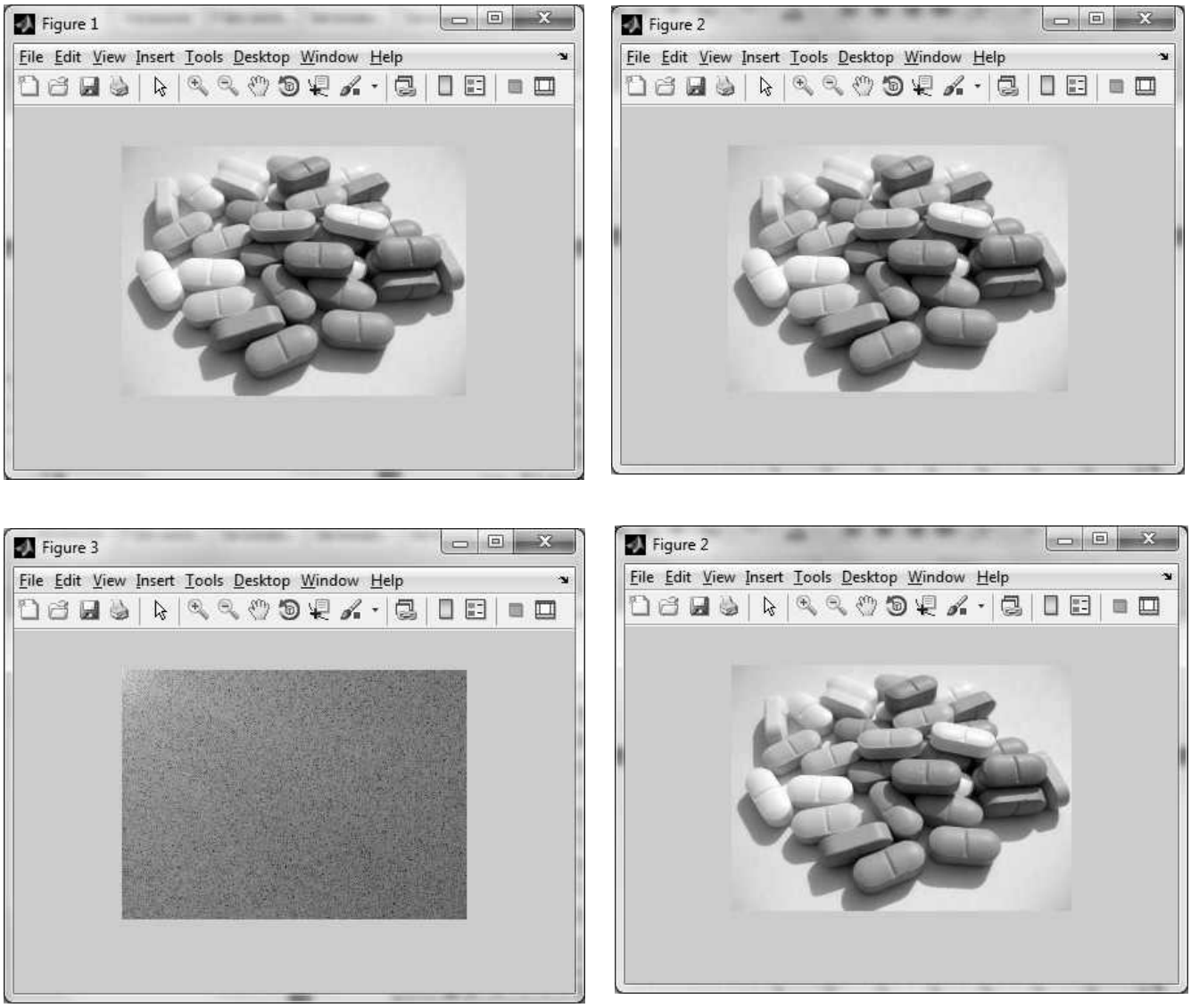



Рис. 6.3. Пряме і зворотне ***DCT***

У цьому прикладі функція `I=rgb2gray(RGB)` виконує перетворення кольорового зображення `RGB` у півтонове `I`. Відображення результату ДКП виробляється в логарифмічному масштабі, оскільки його динамічний діапазон (від мінімального до максимального коефіцієнта ***DCT***) набагато ширший за динамічний діапазон яскравостей самого зображення `(0 … 255)`.

Оскільки ***DCT*** є оборотним лінійним перетворенням, то за його коефіцієнтами можна абсолютно точно відновити вихідне зображення. Обернене перетворення з області спектральних коефіцієнтів ***DCT*** в область просторових змінних виробляється з використанням функції `K=idct2(J)`.

Відзначимо, що найбільші (найбільш яскраві) його значення зосереджені в лівому верхньому куті (це низькочастотні складові $\boldsymbol{DCT}$), вся ж остання частина масиву (права нижня його частина – високочастотні складові) заповнена відносно невеликими значеннями (точками відносно невисокої яскравості).

Це означає, що переведення зображення їх просторової області в спектральну дозволило зосередити основну частину енергії блока зображення у відносно невеликій кількості низькочастотних спектральних коефіцієнтів, які відповідають за великі деталі зображення, що добре розрізняються оком. Високочастотні ж коефіцієнти, які відповідають за дрібні деталі зображення, мають набагато меншу величину.

Таким чином, процедура $\boldsymbol{DCT}$ усуває просторову надмірність зображення, оскільки, чим більш схожими один на одного (корельованими) будуть значення пікселів у блоці, тим більша частина енергії блока буде сконцентрована в компоненті $\boldsymbol{DCT}(\mathbf{0},\mathbf{0})$ і тим меншими будуть значення високочастотних компонент.

Наступним необхідним кроком для процедури стискування є квантування коефіцієнтів $\boldsymbol{DCT}$. Для цього після обчислення $\boldsymbol{DCT}$ від вихідного зображення просто «обнулимо» всі коефіцієнти $\boldsymbol{DCT}$, величина яких за модулем не перевищує, наприклад, 10 (або 30). Цю процедуру можна виконати з використанням функції

```
J(abs(J)<10) = 0;
```

Потім за ненульовими коефіцієнтами, що залишилися, відновимо зображення за допомогою функції `K = idct2(J)`.

```
close all;
RGB=imread('C:\Users\krasnov.MEDICA\Desktop\8.jpg');
imshow(RGB)
I=rgb2gray(RGB);
figure,imshow(I)
J = dct2(I);
figure;imshow(log(abs(J)),[])
J(abs(J)<30) = 0;
figure;imshow(log(abs(J)),[])
K = idct2(J);
figure,imshow(K,[0 255]);
```

Видно (див. рис 6.4), що в результаті «обнулення» невеликих за величиною коефіцієнтів $\boldsymbol{DCT}$ майже вся площа масиву $\boldsymbol{DCT}$ «потемніла». Тобто обнулилися майже всі коефіцієнти (<30), крім розташованих у лівій верхній частині матриці (низькочастотних коефіцієнтів, що відповідають за великі деталі зображення). Природно, що результат кодування квантованих коефіцієнтів $\boldsymbol{DCT}$ буде набагато компактніше, чим для неквантованих. Резуль-

тат відновлення зображення за квантованими коефіцієнтами $\boldsymbol{DCT}$ наведений поруч і істотно відрізняється від вихідного зображення.

Слід зазначити, що наведений алгоритм дозволяє користувачеві встановити потрібний коефіцієнт стискування. Проте при підвищенні міри стискування зображення розпадається на окремі квадрати (8x8). Це пов'язано з тим, що відбуваються великі втрати в низьких частотах при квантуванні, і відновити вихідні дані неможливо. Крім цього, виявляється ефект Гіббса – ореоли по кордонах різких переходів яскравості.


Рис. 6.4. Результати стискування зображення

## 7. ОБРОБЛЕННЯ ВІДЕОЗОБРАЖЕНЬ

Одним із найбільш актуальних застосувань у практиці створення систем управління, робототехніки, використання систем відеоспостереження, роботи з безпілотними літальними апаратами є можливість оброблення відеоданих, отриманих при роботі з відеореєстраторами.

У цій главі наведемо дані про можливості MATLAB для роботи з відео. Основну увагу приділимо введенню відеоданих, виділенню окремих кадрів

з потоку відеоданих і вирішенню зворотної задачі − створенню відеопослідовності з окремих зображень. Необхідність виділення окремих кадрів з відео полягає в швидшій роботі алгоритмів над безліччю кадрів, що зберігаються у вигляді файлів окремо, чим над єдиним відеофайлом. Це забезпечує більшу зручність з точки зору програміста і велику швидкість створення, налагодження і роботи алгоритмів.

Крім цього, розглянемо можливості використання пакета Computer Vision System Toolbox, який містить алгоритми і інструменти для розроблення і моделювання систем комп'ютерного зору і оброблення відео. Пакет містить алгоритми виявлення деталей, детектування руху, виявлення і відстежування об'єктів, стереозору, оброблення аналізу відео. Передбачені також інструменти для введення/виведення відеофайлів, відображення відео, побудови графіки і компонування. Всі ці можливості надані у вигляді системних об'єктів і функцій MATLAB, а також блоків Simulink.

Особливу увагу приділимо опису властивостей пакета Image Acquisition Toolbox, що є набором функцій, які істотно розширюють можливості обчислень у середовищі MATLAB. Це застосування підтримує широкий діапазон операцій по захопленню зображень від відеосенсорів, включаючи:

- захоплення зображень з різного роду пристроїв, від професійних пристроїв захоплення до web-камер з usb-входом;
- перегляд потокового відео;
- управління захопленням (включаючи зовнішні пристрої захоплення);
- вибір потрібної конфігурації функцій;
- передачу даних у робочий простір MATLAB.

Більшість функцій додатка подані у вигляді m-файлів. Для перегляду програмного коду цих функцій потрібно використовувати вираз

```
type function_name
```

Для розширення можливостей Image Acquisition Toolbox існує можливість написання власних m-файлів або використовувати це застосування в комбінації з іншими застосуваннями, наприклад, з Image Processing Toolbox.

Додаток містить також simulink-інтерфейс, який називається Image Acquisition Toolbox. Він служить для моделювання введення відеоданих.

### 7.1. Читання відеофайла і створення послідовності з окремих зображень

Розглянемо процедуру читання відео з використанням функції `mmreader` для відкриття файла і надалі використання функції `read` для читання окремих кадрів.

Наведемо приклад прочитування відеофайла і створення послідовності кадрів (фреймів) із подальшим їх обробленням.

```
close all,clear all
% Приклад прочитування відеофайла
Video =
mmreader('C:\Users\Леонід\Desktop\viptraffic.avi');
% Визначення характеристик лічених відеоданих
width = Video.Width;
height = Video.Height;
frameRate = Video.FrameRate;
numOfFrames = Video.NumberOfFrames;
frameNo = 1;
size(frameNo)
% Прочитування першого фрейма і перетворення його в
півтонове зображення
I = read(Video, frameNo);
figure,imshow(I),title('Вихідний кольоровий фрейм');
I1=rgb2gray(I);
figure,imshow(I1),title('Вихідний півтоновий фрейм');
% Циклічне прочитування фреймів та їх перетворення
for k=5:5:15
    I = read(Video,k);
    I1 = rgb2gray(I);
    figure,imshow(I1),title('Змінний півтоновий фрейм');
    BW=edge(I1, 'sobel', 0.09);
    figure,imshow(BW),title('Контури змінного фрейма');
end
```

Всі функції читання фреймів повертають як результат виконання багатовимірну матрицю. Параметри введеної відеопослідовності можна проконтролювати в інтерактивному режимі в командному вікні MATLAB:

```
>> width = Video.Width

width =

   160

>> height = Video.Height

height =

   120

>> frameRate = Video.FrameRate
```

```
frameRate =

   15.0000

>> numOfFrames = Video.NumberOfFrames

numOfFrames =

   120

>> Video_3

ans =

     1     1

>> frame = read(Video, 1);
>> size(frame)

ans =

   120   160     3
```

Після завантаження фрейма з ним можна працювати, як із звичайною матрицею, а так само застосовувати до нього всілякі функції для оброблення зображень, що входять у пакет `Image Processing Toolbox`.

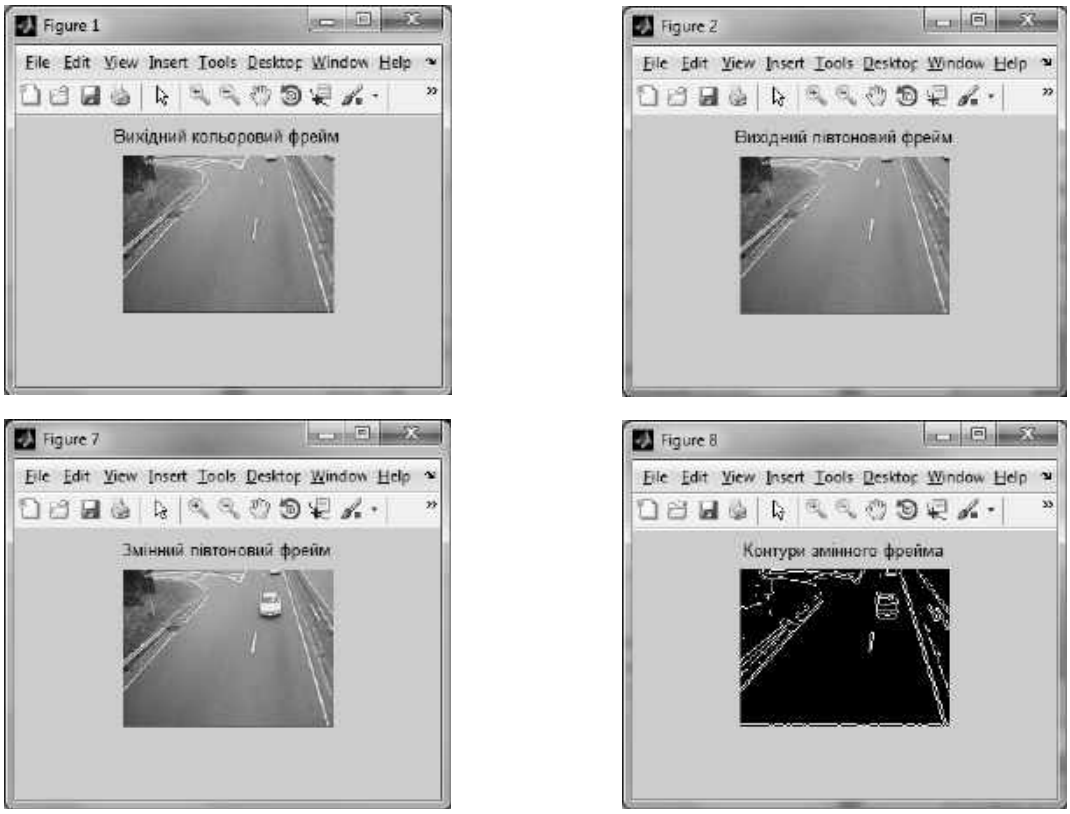



Рис. 7.1. Читання відеофайла і оброблення окремих фреймів

На рис. 7.1 показано результати читання даних відеофайла, розкладання його на окремі фрейми з їх подальшим обробленням і виведенням результатів на екран.

## 7.2. Виведення на екран багатофреймового зображення і створення відеопослідовності

Функція для створення багатокадрового зображення `montage(MS)` має такий синтаксис:

```
montage(MS);
montage(MX,map);
h=montage(...)
```

Функція `montage(MS)` одночасно виводить на екран у поточне вікно всі кадри багатокадрового повнокольорового, півтонового або бінарного зображення `MS`. Багатокадрове півтонове або бінарне зображення складається з декількох відповідно півтонових або бінарних зображень однакового розміру `MXN`. Таке багатокадрове зображення є чотиривимірним масивом `Mxnx1xk`, де `K` – кількість кадрів. Багатокадрове повнокольорове зображення складається з декількох повнокольорових зображень однакового розміру `MXN`. Повнокольорове багатокадрове зображення є чотиривимірним масивом `MxNx3xK`, де `K` – кількість кадрів.

Функція `montage(Mx,map)` одночасно виводить на екран у поточне вікно всі кадри багатокадрового палітрового зображення `MX` з палітрою `map`. Багатокадрове палітрове зображення `MX` складається з декількох палітрових зображень однакового розміру `MXN`, що використовують однакову палітру. `MX` є чотиривимірним масивом `MxNx1xK`, де `K` – кількість кадрів.

Функція `h=montage(...)` повертає описувач виведеного багатокадрового зображення як об'єкта графічного інтерфейсу системи MATLAB.

Відеопослідовності створюються за допомогою функції `immovie`, яка має такий синтаксис:

```
MovX=immovie(MX,map)
```

Функція `Movx=immovie(Mx,map)` формує матрицю `MOVX` з багатокадрового палітрового зображення `MX` з палітрою `map`. Матриця `MovX` є відеопослідовністю, яка може бути показана на екрані за допомогою функції MATLAB `movie`.

Функція `movie` дозволяє програти відеоролик. Особливість використання цієї функції полягає в тому, що потрібно заздалегідь підготувати весь відеоролик, який має бути програним. Додатково ролик має зберігатися в пам'яті у форматі, відмінному від того, який виходить при читанні відео.

Багатокадрове палітрове зображення MX складається з декількох палітрових зображень однакового розміру mxn, що використовують однакову палітру. MX є чотиривимірним масивом MxNx1xK, де k – кількість кадрів.

Наведемо приклад. Хай є шість повнокольорових зображень. Потрібно скласти із цих фреймів багатокадрове зображення (рис. 7.2) і відеопослідовність (маленький фільм) і показати його на екрані (рис. 7.3).

```
close all,clear all
% Прочитування відеофайла
Video =
mmreader('C:\Users\krasnov.MEDICA\Desktop\viptraffic.avi');
% Прочитування послідовності фреймів
frameNo=51;
I1 = read(Video,frameNo);
frameNo=52;
I2=read(Video,frameNo);
frameNo=53;
I3= read(Video,frameNo);
frameNo=54;
I4=read(Video, frameNo);
frameNo=55;
I5=read(Video, frameNo);
frameNo=56;
I6=read(Video, frameNo);
% Перетворення повнокольорових фреймів у палітрові
[I1,map]=rgb2ind(I1,200);
I2=rgb2ind(I2, map);
I3=rgb2ind(I3, map);
I4=rgb2ind(I4, map);
I5=rgb2ind(I5, map);
I6=rgb2ind(I6, map);
% Формування багатокадрового палітрового зображення
MX=cat(4,I1,I2,I3,I4,I5,I6);
Z=montage(MX,map);
figure,imshow(Z);
% Формування матриці відеопослідовності
movMX = immovie(MX,map);
% Виведення відеопослідовності на екран із частотою 8
кадрів на секунду
implay(movMX,8);
```


Рис. 7.2. Багатокадрове зображення, отримане з окремих фреймів вихідної відеопослідовності


Рис. 7.3. Вікно перегляду змонтованої відеопослідовності

## 7.3. Оброблення відеоданих у пакеті Computer Vision System Toolbox

Одним із найбільш продуктивних методів оброблення відеоданих і зображень у MATLAB є використання пакета Computer Vision System Toolbox в додатку Simulink. Цей пакет містить велику бібліотеку блоків для моделювання оброблення відеофайлів і зображень. Основні розділи цієї бібліотеки показані на рис. 7.4.

Основними джерелами відеоінформації для подальшого оброблення є такі блоки, розташовані в розділі Sources:

- `From Multimedia File;`
- `Image From File;`
- `Image From Workspace;`
- `Read Binary File;`
- `Video From Workspace.`

Ресурси виведення і перегляду містяться в розділі бібліотеки Sinks. Це такі блоки:

- `Frame Rate Displey;`
- `To Multimedia File;`
- `To Video Display;`
- `Video To Workspace;`
- `Video Viewer;`
- `Write Binary File.`


Рис. 7.4. Основні розділи бібліотеки пакета Computer Vision System Toolbox

З рис. 7.4 видно, що основними розділами даного пакета є блоки аналізу і поліпшення зображень (`Analysis & Enhancement`), фільтрації (Filtering), геометричних перетворень (`Geometric Transformations`), морфологічних перетворень (`Morphological Operations`) та ін. (рис.7.5).

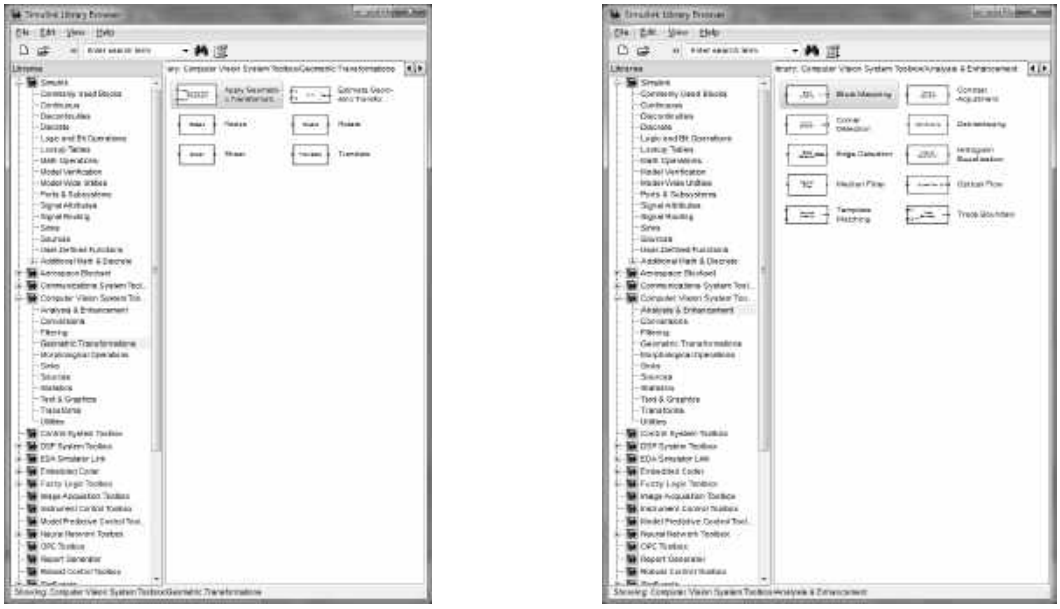


Рис. 7.5. Блоки Geometric Transformations і Analysis & Enhancement бібліотеки Computer Vision System Toolbox

Наведемо простий приклад моделювання введення зображень для подальшого перетворення і перегляду. Схему моделювання показано на рис. 7.6. За допомогою блока `Image From File` здійснюється введення в схему моделі зображення (ТУ-160.jpg), а блок Color Space Conversion здійснює перетворення `RGB`-зображення в півтонове. Відображення вихідного і перетвореного зображень здійснюється за допомогою блоків `Video Viewer`. На рис. 7.7 показані результати перетворення.

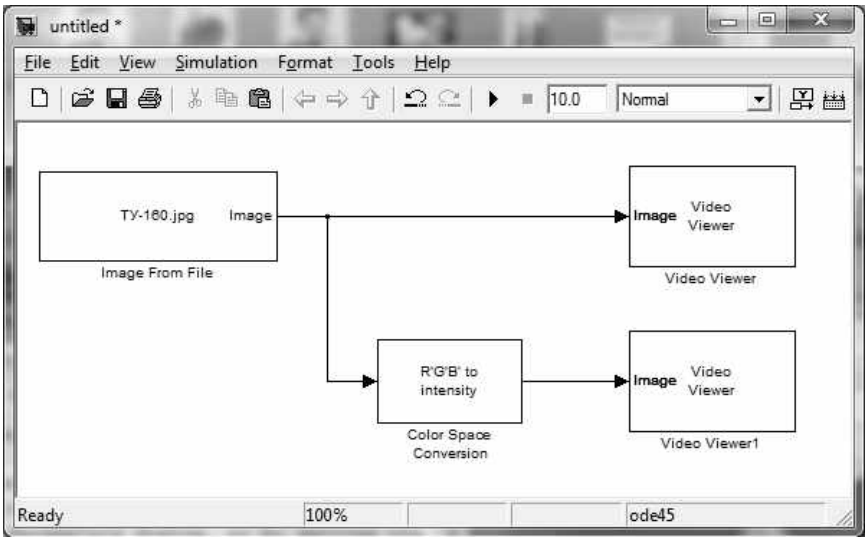



Рис. 7.6. Перетворення `RGB`-зображення в півтонове



Рис. 7.7. Результати перетворення повнокольорового зображення в півтонове

Наведемо ще один приклад введення відеозображення для оброблення оптичного потоку за методом Лукаса – Канаде, що передбачає визначення пересування об'єкта по його контуру від кадру до кадру. Схему і результати моделювання показано на рис. 7.8 і 7.9. За допомогою блока `From Multimedia File` здійснюється введення в моделі відеозображення (`viptraffic.avi`), а блок `Color Space Conversion` здійснює перетворення `RGB`-зображення в півтонове. Потім відеозображення надходить на блок `Optical Flow`, де перетвориться в оптичний потік за методом Лукаса – Канаде. Відображення вихідного і перетвореного зображень здійснюється за допомогою блоків `Video Viewer`.

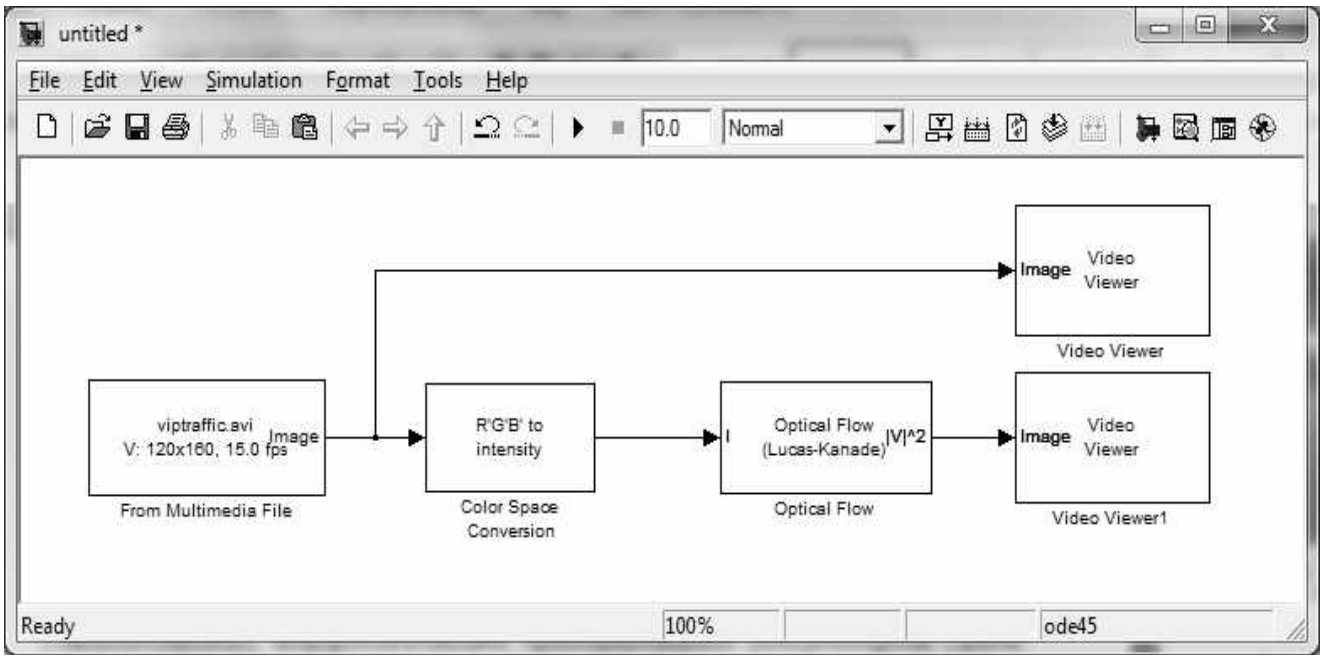



Рис. 7.8. Приклад оброблення оптичного потоку за методом Лукаса – Канаде

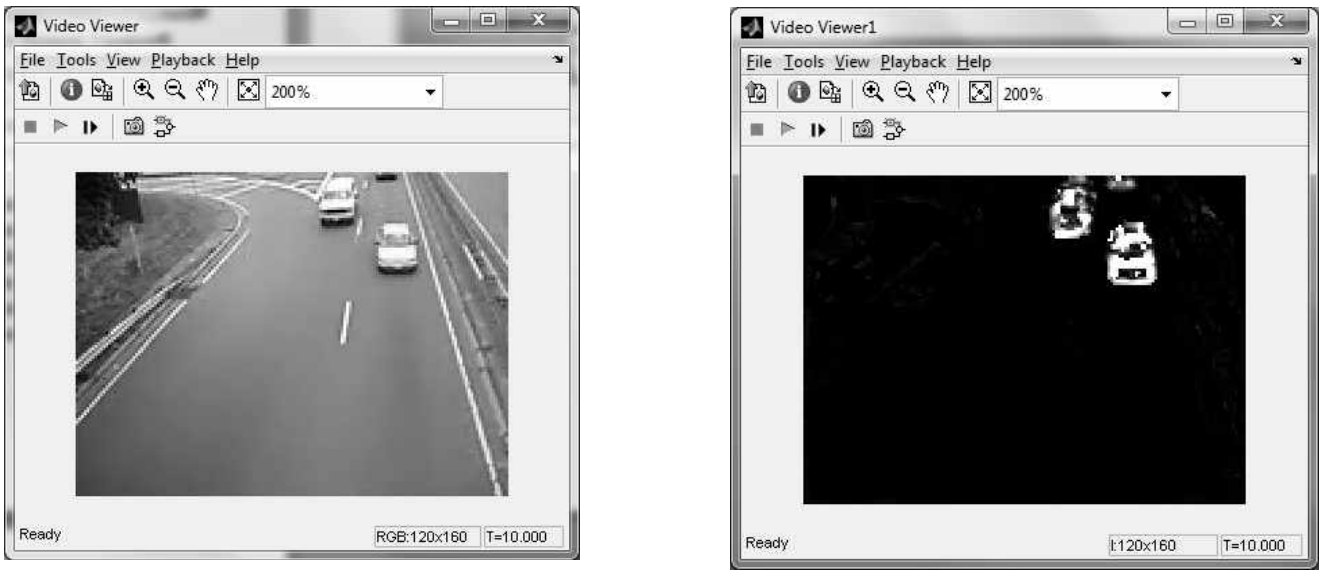



Рис. 7.9. Результати оброблення оптичного потоку за методом Лукаса – Канаде

## 7.4. Використання пакета Image Acquisition Toolbox для підключення відеокамер

Додаток `Image Acquisition Toolbox` дозволяє безпосередньо підключати, набудовувати і управляти засобами формування зображень і потокового відео. Це істотно розширює середовище технічних обчислень MATLAB. Разом з іншими застосуваннями, наприклад з такими, як `Image Processing Toolbox, Image Acquisition Toolbox`, забезпечує високий рівень проведення аналізу і оброблення даних.

Додаток `Image Acquisition Toolbox` працює з таким апаратним забезпеченням:

- плати Matrox і VFW (Video for Windows);
- web-камери;
- відеокамери з інтерфейсами плати Data Translation;
- плати відеозахоплення у форматах WDM (Windows Driver Model USB і Firewire (Ieee-1394);
- цифрові (DV) відеокамери.

Щоб використовувати `Image Acquisition Toolbox` для захоплення зображень, потрібно реалізувати такі кроки:

- інсталяція і установлення конфігурації пристроїв захоплення зображень;
- набуття інформації, яка однозначно ідентифікує пристрої захоплення зображень в `Image Acquisition Toolbox`;
- створення вихідних об'єктів відеопослідовності;
- попередній перегляд відеопослідовності (необов'язково);
- формування властивостей об'єктів захопленого зображення (необов'язково);
- захоплення даних зображення;
- очищення даних.

***Крок 1***. ***Інсталяція пристрою захоплення зображень***. Інсталяцію пристрою захоплення зображень проводять таким чином:

- установлення плати захоплення зображень у комп'ютер;
- установлення потрібних програмних драйверів для такого пристрою;
- приєднання камери до плати захоплення зображень;
- перевірка роботи камери, її характеристик, запуск відповідного ПО;
- перегляд отриманої відеопослідовності.

Пристрої Windows для захоплення зображень, такі, як web-камери або портативні камери, не потребують установлення плати захоплення зображень. Ці пристрої підключаються прямо до комп'ютера через usb-порт.

***Крок 2. Набуття інформації про пристрої захоплення зображень***. Для цього використовують функцію imaqhwinfo без аргументів у командному рядку:

```
>> imaqhwinfo
ans =
          InstalledAdaptors: {'matrox' 'winvideo'}
          MATLABVersion: '7.13 (R2011b)'
          ToolboxName: 'Image Acquisition Toolbox'
             ToolboxVersion: '4.2 (R2011b)'
```

Додаткова інформація про пристрій міститься в полі `Deviceinf`о. Ці дані використовують як аргумент у функції `imaqhwinfo` разом із назвою адаптера.

```
>> dev_info = imaqhwinfo('winvideo',1)
dev_info =
          DefaultFormat: 'YUY2_1280x720'
    DeviceFileSupported: 0
             DeviceName: 'HD WebCam'
               DeviceID: 1
      ObjectConstructor: 'videoinput('winvideo', 1)'
       SupportedFormats: {1x6 cell}
```

Описувати формат для відеопотоку не обов'язково. Щоб визначити, який відеоформат підтримує пристрій захоплення зображення, потрібно проглянути поле `Supportedformats` у структурі `Deviceinfo`, повертаній функцією `imaqhwinfo`.

***Крок 3. Створення вихідних об'єктів відеопослідовності***. Для цього в командному рядку MATLAB необхідно ввести функцію `vidieoinpu`t. У ній використано такі параметри, як назва адаптера, `ID` пристрою і формат відео, які були визначені на кроці 2. Назва адаптера є необхідним аргументом, останні параметри (`ID` пристрою і формат відео) функція `videoinput` може використовувати за умовчанням:

```
>> vid = videoinput('winvideo',1);
```

Для перегляду короткого списку об'єктів відеопослідовності в командному рядку MATLAB необхідно ввести `vid`.

У цій інформації відображуватимуться короткі характеристики об'єктів, такі, як число захоплених фреймів, тип захоплення і черговий стан об'єктів. Властивості вихідних об'єктів відеопослідовності можна використовувати для керування іншими характеристиками. Для детальнішої інформації див. крок 5 (формування властивостей об'єктів захопленого зображення).

```
>> vid
Summary of Video Input Object Using 'HD WebCam'.
Acquisition Source(s): input1 is available.
Acquisition Parameters:'input1' is the current selected
source.
      10 frames per trigger using the selected source.
      'YUY2_1280x720' video data to be logged upon START.
      Grabbing first of every 1 frame(s).
      Log data to 'memory' on trigger.
      Trigger Parameters: 1'immediate'trigger(s) on START.
      Status:  Waiting for START.
      0 frames acquired since starting.
      0 frames available for GETDATA.
```

За умовчанням формати відео перераховані в полі `Defaultformat` у структурі `Devicienfo`. Якщо необхідно використовувати різні формати відео, то при створенні об'єкта необхідно вказати формат.

***Крок 4. Попередній перегляд відеопослідовності.*** Після створення вихідних об'єктів відеопослідовності система MATLAB має доступ до пристроїв захоплення зображень і може готувати захоплені дані. Проте спочатку можна здійснити попередній перегляд вихідних об'єктів відеопослідовності і переконатися в прийнятності результату. Наприклад, можна змінити розташування камери, змінити освітлення, відкоректувати фокус або зробити інші зміни в установках захоплення зображення. Цей крок не є обов'язковим, оскільки попередній перегляд можна проводити у будь-який час після створення вихідних об'єктів відеопослідовності.

Для попереднього перегляду потокового відео в цьому прикладі необхідно ввести функцію попереднього перегляду в командне вікно MATLAB і як аргумент поставити опис вихідних об'єктів відеопослідовності, які були визначені в кроці 3.

```
preview(vid)
```

Функція попереднього перегляду відкриває вікно і відображує існуючу відеопослідовність. Коли вікно попереднього перегляду відкрито, тоді у властивостях встановлюється значення `'on'`. Після зміни характеристик зображення шляхом установлення властивостей захоплення зображень на

зображенні, що відображується, у вікні попереднього перегляду відображуватимуть ці зміни. На рис. 7.10 і 7.11 показані приклади вікна попереднього перегляду. Тут відображується назва пристрою захоплення (`winvideo 1`), час створення фрейма, дозвіл зображення (1280x720) і стан вихідних об'єктів відеопослідовності (`Waiting for Start`).

Рис. 7.10. Вікно попереднього перегляду відео з дозволом 1280x720

Наведемо приклад зміни формату відеопослідовності в режимі попереднього перегляду. Для цього зменшимо дозвіл відеоданих від (1280x720) до (640x480).

```
obj = videoinput('winvideo',1,'YUY2_640x480');
preview(obj);
```

Результати такого перетворення показані на рис. 7.10. Це може виявитися зручнішим у певних практичних ситуаціях.


Рис. 7.11. Вікно попереднього перегляду відео з дозволом 640x480

Для закриття вікна попереднього перегляду використовують кнопку `Close` у верхньому правому куті вікна або функцію `closepreview`. Як аргумент у цій функції необхідно вказати вихідні об'єкти відеопослідовності

```
closepreview(vid)
```

***Крок 5***. Формування властивостей об'єктів захопленого зображення. Вихідні об'єкти відеопослідовності створюються на кроці 3 за допомогою функції `Videoinput`. Після їх створення додаток автоматично формує один або декілька первинних об'єктів відеопослідовності, пов'язаних з вихідними об'єктами відеопослідовності. Кожен первинний об'єкт відеопослідовності являє собою набір фізичних даних, що відображають його суть. Число створюваних первинних об'єктів залежить від пристроїв і описаного формату відео.

У будь-який момент часу лише один із первинних об'єктів відеопослідовності може бути активним. Його використовують при захопленні. Для перегляду списку всіх властивостей, підтримуваних вихідними або первинними об'єктами відеопослідовності, використовують функцію `get` у командному вікні MATLAB.

```
>> get(vid)
```

Функція `get` виводить список всіх властивостей об'єктів і їх поточні значення:

```
>> get(vid)
  General Settings:
    DeviceID = 1
    DiskLogger = []
    DiskLoggerFrameCount = 0
    EventLog = [1x0 struct]
    FrameGrabInterval = 1
    FramesAcquired = 0
    FramesAvailable = 0
    FramesPerTrigger = 10
    Logging = off
    LoggingMode = memory
    Name = YUY2_1280x720-winvideo-1
    NumberOfBands = 3
    Previewing = off
    ROIPosition = [0 0 1280 720]
    Running = off
    Tag =
    Timeout = 10
    Type = videoinput
    UserData = []
    VideoFormat = YUY2_1280x720
```

```
    VideoResolution = [1280 720]
Color Space Settings:
    BayerSensorAlignment = grbg
    ReturnedColorSpace = YCbCr
  Callback Function Settings:
    ErrorFcn = @imaqcallback
    FramesAcquiredFcn = []
    FramesAcquiredFcnCount = 0
    StartFcn = []
    StopFcn = []
    TimerFcn = []
    TimerPeriod = 1
    TriggerFcn = []
  Trigger Settings:
    InitialTriggerTime = []
    TriggerCondition = none
    TriggerFrameDelay = 0
    TriggerRepeat = 0
    TriggersExecuted = 0
    TriggerSource = none
    TriggerType = immediate
  Acquisition Sources:
    SelectedSourceName = input1
    Source = [1x1 videosource]
```

Для перегляду властивостей вибраних первинних об'єктів, пов'язаних з відповідними вихідними об'єктами відеопослідовності, використовують функцію getselectedsource разом із функцією `get`. Вона повертає чергову активну відеопослідовність. Список властивостей вибраних первинних об'єктів відеопослідовності, які пов'язані з вихідними об'єктами відеопослідовності, створюється на кроці 3 і вводиться в командний рядок MATLAB:

```
get(getselectedsource(vid))
```

Функція `get` виводить список всіх об'єктів та їх відповідні значення. Основні установлення:

```
>> get(getselectedsource(vid))
  General Settings:
    Parent = [1x1 videoinput]
    Selected = on
    SourceName = input1
    Tag =
    Type = videosource
```

Особливі властивості пристрою:

```
Device Specific Properties:
    BacklightCompensation = on
```

```
Brightness = 0
Contrast = 0
FrameRate = 7.5000
Gamma = 100
Hue = 0
Saturation = 64
Sharpness = 2
```

***Установлення властивостей об'єктів***. Для установлення значень властивостей вихідних об'єктів відеопослідовності або значень властивостей первинних об'єктів відеопослідовності використовують функцію `set` або посилання на відповідне поле в структурі властивостей об'єкта.

Деякі властивості можуть бути лише прочитані без зміни їх значень. Ці властивості дають інформацію про стан об'єкта. Інші властивості можуть бути прочитані лише у тому випадку, коли об'єкт запущений. Для перегляду списку всіх властивостей необхідно використовувати функцію `set function`, описавши об'єкт як аргумент.

Наприклад, для безперервного захоплення зображень у команді `set` властивості `Triggerrepeat` потрібно поставити значення `Inf`. Це все має бути записано в командному рядку:

```
set(vid,'TriggerRepeat',Inf);
```

***Крок 6***. ***Захоплення даних зображення***. Після створення вихідних об'єктів відеопослідовності і формування їх властивостей можна приступати до захоплення даних. Типові етапи захоплення зображень є такими:

- запуск відеопослідовності;
- запуск захоплення;
- передача даних у робочий простір MATLAB.

***Крок 7***. ***Очищення даних***. Після завершення процедури захоплення зображень дані з пам'яті переміщаються в робочий простір MATLAB.

```
delete(vid)
clear
close(gcf)
```

***Запуск і зупинка об'єктів вихідного відео***. При створенні вихідних відеооб'єктів необхідно встановити зв'язок між системою MATLAB і пристроєм захоплення зображень. Проте, перш ніж виробляти захоплення даних з пристрою, необхідно запустити об'єкт за допомогою функції `start`:

```
>> start(vid);
```

При запуску об'єкта пристрій резервується для Вашого використання зі встановленою конфігурацією параметрів. Таким чином, значення деяких властивостей будуть прочитані лише при запуску. Запуск об'єктів захоп-

лення зображень буде зупинений при виконанні однієї з перерахованих нижче умов:

- запрошуване число фреймів вже захоплене. Це перевіряється умовою `Framesacquired = Framespertrigger*(Triggerrepeat+1)`, де параметри `Framesacquired`, `Framespertrigger` і `Triggerrepeat` є властивостями вихідних відеооб'єктів;
- час запуску помилковий;
- закінчився час захоплення;
- була використана функція `stop`.

Коли об'єкт запущений, то йому встановлюють `Running property` значення `'on'`. У випадку якщо об'єкт не запущений, то додаток встановлює об'єкт `'Running property'` в положення `'off'` і цей стан називається зупиненим.

***Передача окремих фреймів у робочий простір***. Виклик функції getsnapshot приводить до передачі фреймів у робочий простір.

```
>> frame = getsnapshot(vid);
```

Розмірність повертаних даних нерозривно пов'язана з дозволом зображення, а також значенням властивості `Numberofbands`.

```
vid.NumberOfBands
ans =

   1

size(frame)

ans =

   480   0
```

***Передача декількох фреймів у робочий простір***. Виклик функції `getdata` дозволяє передавати декілька фреймів зображень у робочий простір MATLAB.

```
data = getdata(vid,10);
```

Функція `getdata` виконує передачу даних 10 фреймів у робочий простір. Відзначимо, що ці дані подаються у вигляді чотиривимірного масиву: кожен окремий фрейм є тривимірним, а четверта розмірність свідчить про кількість фреймів.

```
size(data)

ans =

   480 640 1 10
```

## 7.5. Використання пакета Image Acquisition Toolbox у додатку Simulink

Крім можливості використання функцій пакета `Image Acquisition Toolbox` у командному режимі або в режимі створення `m`-файлів у додатку `Simulink` передбачено можливість використання цих ресурсів за допомогою блока `From Video Device`. Це створює користувачеві додаткові зручності.

Наведемо ще один приклад введення відеозображення для подальшого перетворення за методом Лукаса-Канаде. В цьому випадку використовують відеозображення, яке отримується безпосередньо від web-камери комп'ютера. Воно вводиться в схему моделювання за допомогою блока `From Video Device`, розташованого в розділі `Image Acquisition Toolbox` бібліотеки `Simulink`. Схему і результати моделювання показано на рис. 7.12 і 7.13. Відзначимо, що оброблення здійснюється в реальному масштабі часу.

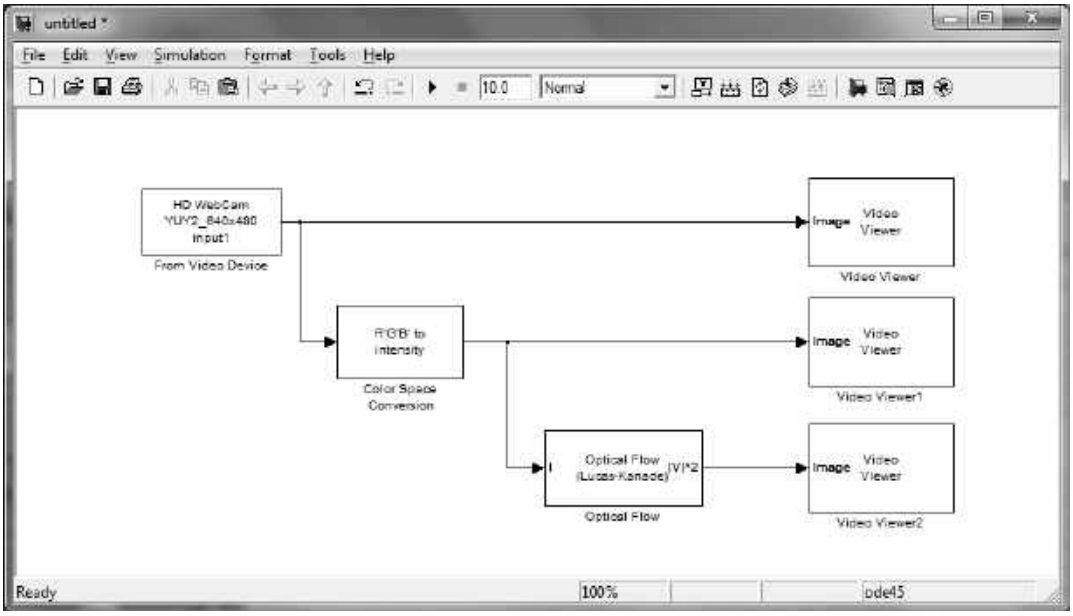



Рис. 7.12. Схема моделювання оброблення оптичного потоку з web-камери

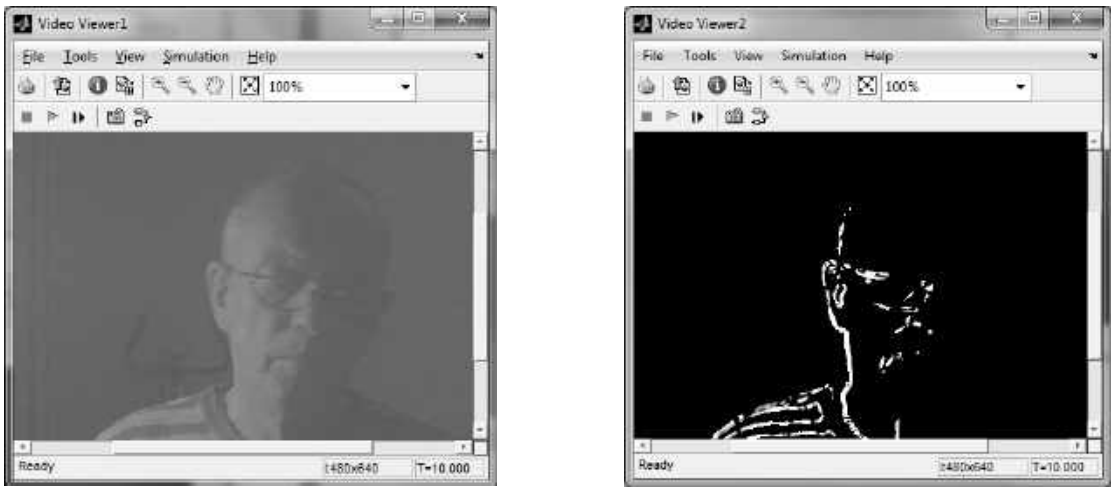



Рис. 7.13. Оброблення відеопотоку в реальному масштабі часу

Блок From Video Device може бути налагоджений на різні варіанти роботи (див. рис. 7.14). У вікні налагодження відображується тип відеореєстратора і його ID код (у цьому випадку – winvideo 1). Можна вибрати відеоформат із відповідним дозволом, наприклад Yuy2_640x480. Задається розмір області інтересу – ROI position [r,c,height,width], вибирається тип даних (наприклад, Data type – single). Можна змінити час зміни фреймів (наприклад, в Block sample time встановити – 1/15).

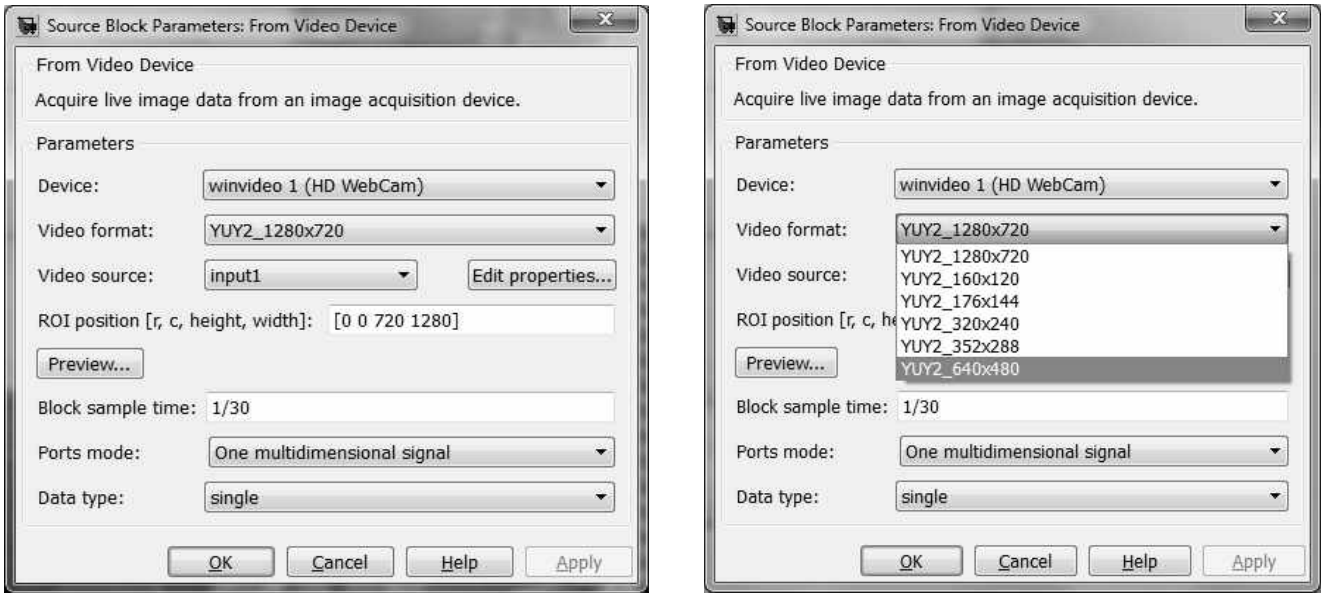



Рис. 7.14. Налагодження параметрів блока Image Acquisition Toolbox

На жаль, обмежений обсяг роботи не дозволяє навести детальніший опис всіх особливостей використання додатка Image Acquisition Toolbox для підключення, налагодження і управління засобами формування зображень і потокового відео. Детальнішу інформацію про це можна отримати за посиланням http://matlab.exponenta.ru/imageacquis/book1.

## 8. ОБРОБЛЕННЯ ЗОБРАЖЕНЬ У ДОДАТКУ WAVELET TOOLBOX

Технологія роботи з пакетом Wavelet Toolbox досить детально описана в розділі 3 цієї роботи на прикладі аналізу одновимірних сигналів. Проте аналіз зображень має свої особливості. Тому розглянемо це питання детальніше.

Розділ Wavelet 2-d головного меню призначений для вейвлет-розкладання і аналізу зображень. Для запуску Wavelet 2-d потрібно викликати головне меню командою wavemenu і натискувати кнопку Wavelet 2-d. Запускається графічне середовище, зручне для роботи комплексу команд, за аналізом зображень. Головне вікно графічного інтерфейсу містить звичайне меню Windows вгорі, спеціальні кнопки меню справа і внизу, а також основне поле для візуалізації результатів аналізу (рис. 8.1).

Використовуючи меню `File`, в це вікно можна завантажити дані для аналізу. При виборі дороги `File → Load → Image` відкривається тека MATLAB, з якої вибирається потрібне для аналізу зображення.

У меню `File` міститься деяка кількість демонстраційних зображень, які можна завантажити таким чином. Укажемо, наприклад, шлях `File → Example Analysis → Indexed Images → At level 3, with bior 6.8 → Noisy Sinsin`. Завантажиться зашумлене зображення набору квадратів (див. рис. 8.2).

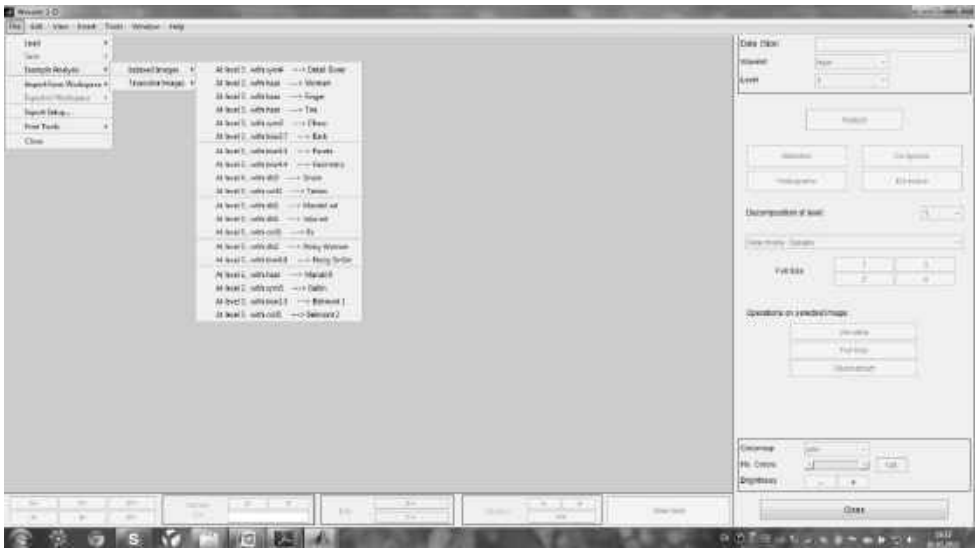


Рис. 8.1. Головне вікно wavelet-аналізу зображень

У вікні аналізу відображуватиметься вихідне зображення (`Original Image`), результати розкладання зображення на вейвлет-компоненти (`Image Selection`) на три рівні (`Decomposition at level 3`) шляхом дискретного вейвлет-перетворення (`dwt`) і відновлене шляхом зворотного перетворення (`idwt`) зображення (`Synthesized Image`).

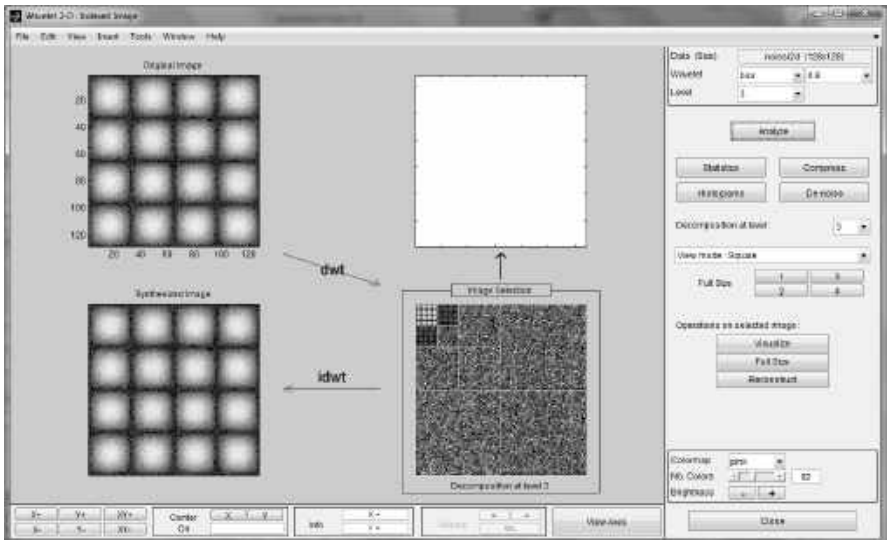



Рис. 8.2. Аналіз демонстраційного зашумленого зображення `Noisy Sinsin`

Як і в разі одновимірного аналізу, нижнє меню надає великі можливості масштабування зображень по осях X і B, а також можливість перегляду зображень у збільшеному масштабі (View Axes).

У правому меню закладені основні ресурси аналізу. По-перше, можна вибрати різні вейвлет-функції для розкладання досліджуваного зображення і переаналізувати його знов. По-друге, можна змінити кількість рівнів розкладання в меню (`Decomposition at level ...`), детально розглянути компоненти розкладання за допомогою меню `View Mode`. На рис. 8.3 показано відображення всіх трьох компонент розкладання в режимі `View Mode: Tree`. Ці компоненти подаються окремо в трьох проекціях (`Horisontal, Diagonal, Vertical Details`). Описані ресурси створюють великі зручності для аналізу зображення на кожному етапі.

Але найбільш істотні можливості аналізу зосереджені в основних чотирьох ресурсах:

- `Statistic;`
- `Histograms;`
- `Compress;`
- `De-noise.`

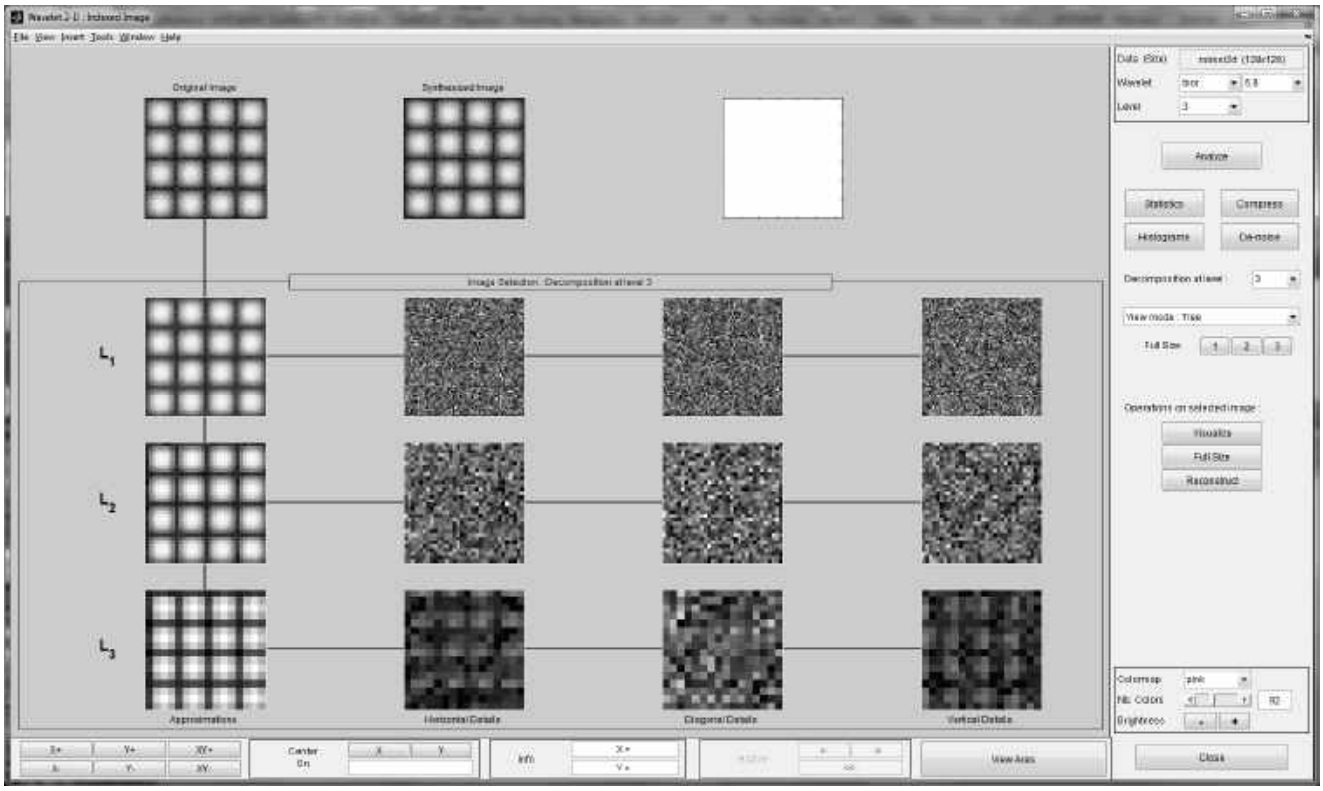


Рис. 8.3. Вікно компонент вейвлет-розкладання

Перші два види аналізу (`Statistic i Histograms`) дозволяють детально досліджувати статистичні характеристики досліджуваного зображення і функції його просторового розподілу. Не розглядатимемо ці режими детально, оскільки читач може зробити це самостійно.

Більшу увагу приділимо режиму очищення зображення від шумів (De-noise), який фактично є режимом фільтрації. При натисненні кнопки De-noise програма переходить у нове вікно (див. рис. 8.4), в якому з'являється вихідне зображення (`Original image`), відфільтроване зображення (`De-noised image`) і вікна розподілу інтенсивностей компонент розкладання в різній площині. У нашому прикладі визначено три рівні вейвлет-розкладання в трьох площинах.

У меню `Select noise structure` можна вибирати відповідні компоненти розкладання (`Horisontal, Vertical або Diagonal`) і за допомогою мишки управляти порогами функцій розподілу компонент розкладання, визначаючи таким чином міру участі тієї або іншої компоненти в процесі фільтрації зображення. Ще важливішим у меню `Select noise structure` є вибір характеру заглушення шумів:

- ненормований білий шум (`Unscaled white noise`);
- нормований білий шум (`Scaled white noise`);
- небілий шум (`Non-white noise`).

Неправильне налагодження фільтра до характеру шумів, що маскують досліджуване зображення, призводить до істотного погіршення якості фільтрації.

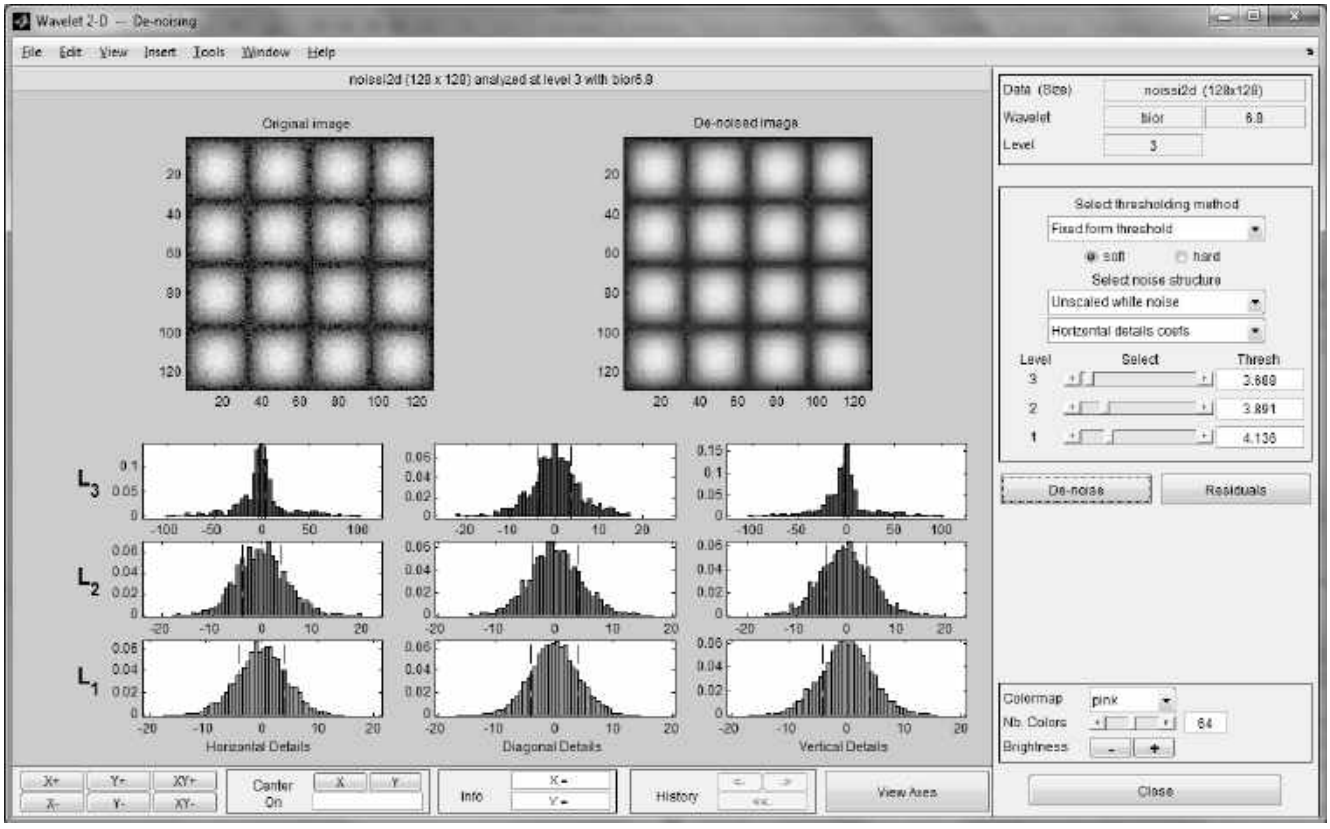
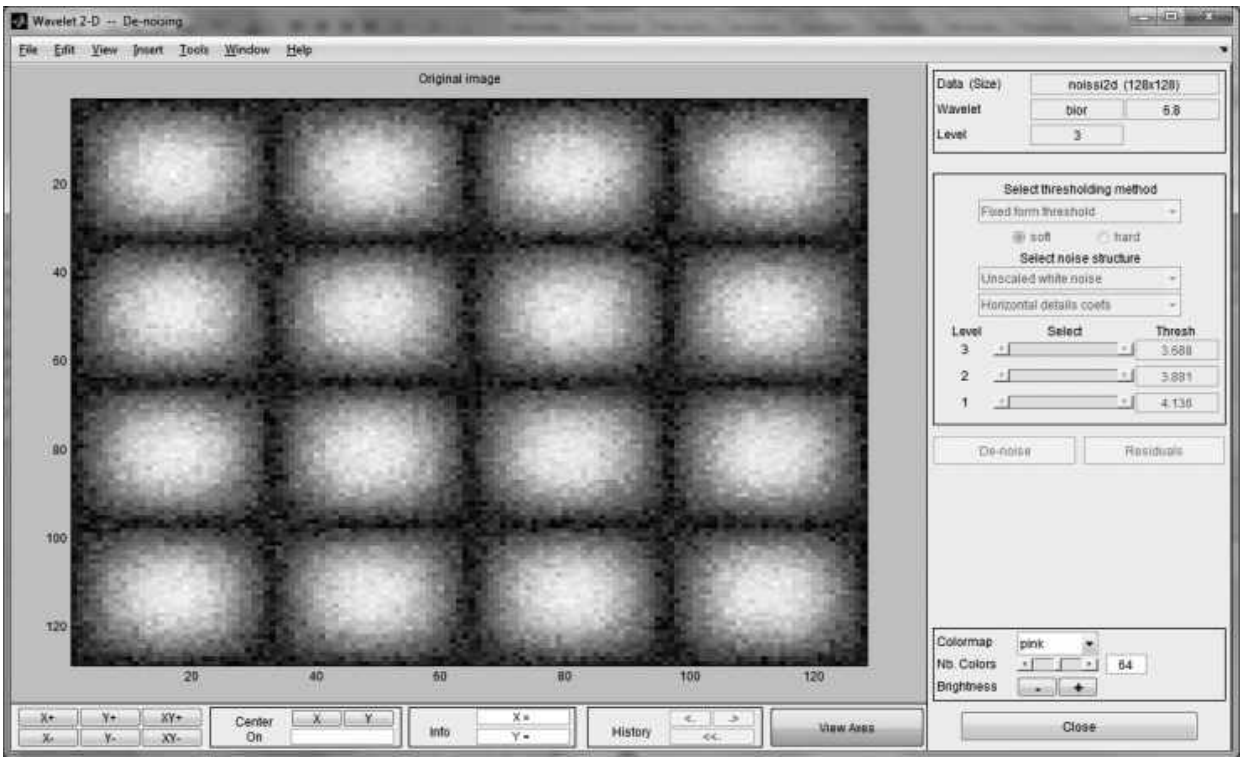
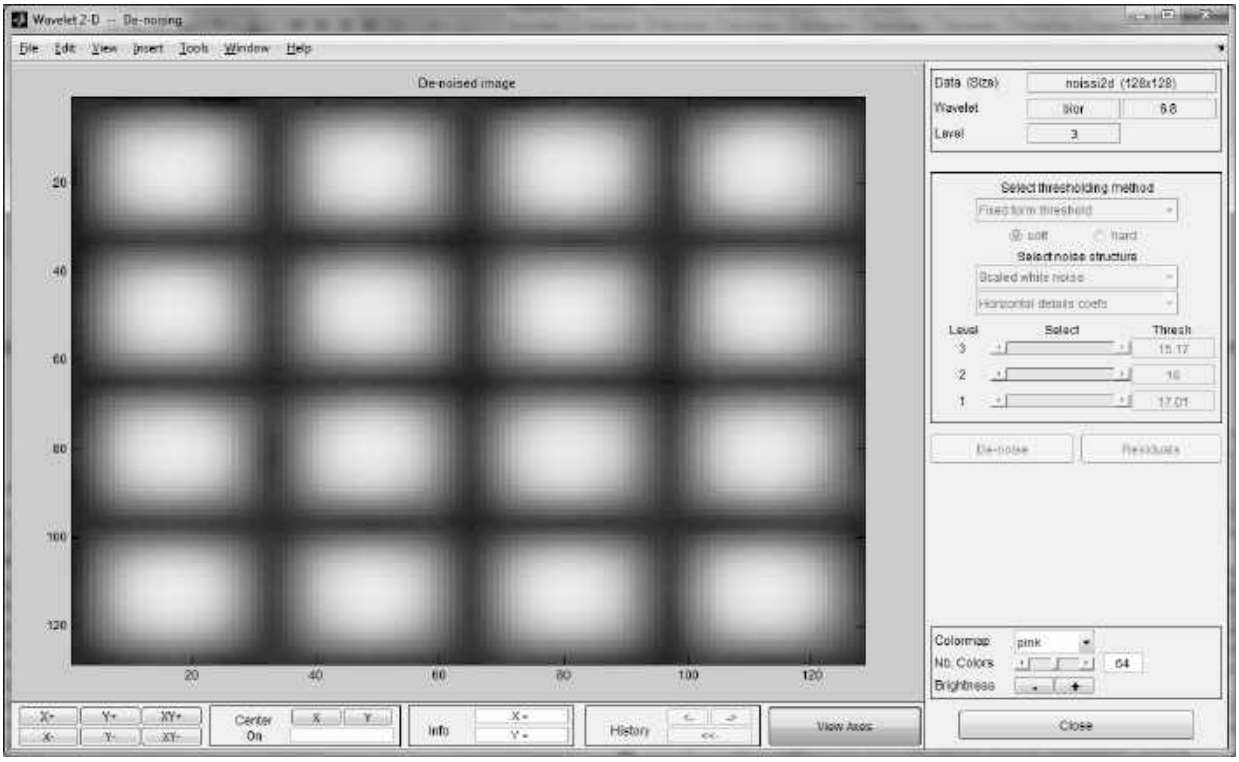



Рис. 8.4. Вікно фільтрації зображення від шумів

На рис. 8.5 показані результати очищення від шумів (фільтрації) досліджуваного зображення.




а




б

Рис. 8.5. Вихідне зашумлене (а) і відфільтроване (б) зображення

При описі технології роботи з пакетом `Wavelet Toolbox` для оброблення зображень не ставилося завдання детального опису всіх режимів, оскільки це значно збільшує обсяг публікації. Читачеві буде нескладно виявити особливості кожного з режимів у процесі практичної роботи.

# ДОДАТОК

## Список функцій Image Processing Toolbox

***Формати подання даних:***

- `xyz2uint16` – перетворення колірних даних з формату `XYZ` в `uint16`;
- `xyz2double` – перетворення даних у xyz-значення у форматі `double`;
- `double` – подання елементів масиву у форматі `double`;
- `uint8` – подання елементів масиву у форматі `uint8`;
- `im2double` – подання зображення масивом у форматі `double`;
- `im2uint8` – подання зображення масивом у форматі `uint8`;
- `im2uint16` – подання зображення масивом у форматі `uint16`;
- `im2mis` – подання зображень в Java Memory Image Source;
- `im2java2d` – перетворення зображення в зображення Java, що буферизує;
- `im2java` – перетворення даних в java-зображення;
- `lab2uint8` – перетворення даних з формату `L*a*b*` в uint8;
- `lab2uint16` – перетворення даних `L*a*b*` у формат `uint16`;
- `lab2double` – перетворення даних `L*a*b*` у рядок.

***Визначення типу зображення***:

- `isbw` – перевірити, чи є зображення бінарним;
- `isgray` – перевірити, чи є зображення півтоновим;
- `isind` – перевірити, чи є зображення палітровим;
- `isrgb` – перевірити, чи є зображення повнокольоровим.

***Робота з графічними форматами файлів***:

- `imfinfo` – читання з файла інформації про зображення;
- `imread` – читання зображення з файла;
- `imwrite` – запис зображення у файл;
- `imageinfo` – створення інформаційних даних про зображення;
- `imagemodel` – доступ до властивостей зображень з точки зору їх відображення.

***Установлення і читання глобальних змінних IPT:***

- `iptsetpref` – установлення глобальних змінних IPT;
- `iptgetpref` – читання глобальних змінних IPT;
- `getline` – вибір ламаної лінії за допомогою мишки;
- `getpts` – вибір точок за допомогою мишки;
- `getrect` – вибір прямокутника за допомогою мишки;
- `ipticondir` – повернення назви директорії, що містить `IPT` і MATLAB ікони;
- `iptgetapi` – доступ до прикладного програмного інтерфейсу.

***Виведення зображень на екран і захоплення їх з екрана:***

- `imshow` – виведення зображення на екран;
- `truesize` – установлення розмірів вікна для відображення зображень;
- `subimage` – виведення декількох зображень в одному вікні;
- `colorbar` – виведення на екран палітри;
- `imcontour` – побудова для зображення ліній рівня;
- `immovie` – створення відеопослідовності;
- `montage` – виведення на екран всіх кадрів багатокадрового зображення;
- `warp` –- накладення зображення на поверхню;
- `zoom` – масштабування зображення у вікні зображення;
- `getimage` – отримання зображення з графічного об'єкта;
- `dicominfo` – читання метаданих з `dicom`-файла;
- `dicomread` – читання DICOM зображень;
- `dicomwrite` – запис зображень у dicom-файл;
- `dicomuid` – генерація ідентифікатора для dicom-файлів;
- `imview` – відображення зображень в `Image Viewer`;
- `imgcf` – отримання останніх зображень, що відображуються;
- `imgca` – отримання недавно оброблюваних даних;
- `imgetfile` – відображення діалогового вікна відкриття зображень;
- `imhandles` – установлення ручного управління зображеннями;
- •`dicomanon` – анонімність `dicom`-файла;
- `dicomdict` – отримання або установлення активного словника `dicom`-даних;
- `impositionrect` – створення прямокутника, що пересувається;
- `imscrollpanel` – панель прокручування для інтерактивного управління зображеннями;
- `analyze75info` – прочитування метаданих зі встановленого заголовка файла даних Analyze 7.5;
- `analyze75read` – прочитування даних зображення з Analyze 7.5.

***Перетворення типів зображень:***

- `im2bw` – бінаризація відсіканням по порогу яскравості;
- `mat2gray` – перетворення матриці чисел у півтонове зображення;
- `rgb2gray` – перетворення повнокольорового зображення в півтонове;
- `ind2gray` – перетворення палітрового зображення в півтонове;
- `gray2ind` – перетворення півтонового зображення в палітрове;
- `grayslice` – перетворення півтонового зображення в палітрове відсіканням по декількох порогах;

- `ind2rgb` – перетворення палітрового зображення в повноцінне;
- `dither` – дифузійне псевдозмішення кольорів;
- `rgb2ind` – перетворення повнокольорового зображення в палітрове;
- `imapprox` – зменшення кількості кольорів палітрового зображення;
- `cmunique` – пошук палітри мінімального розміру;
- `cmpermute` – зміна порядку кольорів у палітрі;
- `label2rgb` – перетворення матриці міток у `RGB`-зображення.

***Конвертація колірних систем:***

- rgb2hsv – конвертація з RGB у HSV;
- hsv2rgb – конвертація з HSV у RGB;
- rgb2ntsc – конвертація з RGB в YIQ;
- ntscc2rgb – конвертація з YIQ у RGB;
- rgb2ycbcr – конвертація з RGB в Ycbcr;
- ycbcr2rgb – конвертація з Ycbcr у RGB;
- rgbplot – зображення компонентів RGB палітри (MATLAB Toolbox);
- graythresh – обчислення глобального порогу зображення з використанням методу Отса;
- iccread – прочитування опису ICC.

***Геометричні перетворення зображень:***

- `imcrop` – кадрування зображень;
- `imresize` – зміна розмірів зображення;
- `imrotate` – поворот зображення;
- `checkerboard` – створення шаховоподібних зображень;
- `findbounds` – визначення меж при просторових перетвореннях;
- `imtransform` – застосування просторових перетворень зображень;
- `makeresampler` – створення структури, що повторюється;
- `maketform` – створення структури просторових перетворень (TFORM);
- `tformarray` – використання просторових перетворень для багатовимірних масивів;
- `tformfwd` – використання прямих просторових перетворень;
- `para2fan` – обчислення віялово-променевих проекцій на підставі паралельно-променевих даних томографії;
- `tforminv` – використання зворотних просторових перетворень;
- `fan2para` – обчислення паралельно-променевих проекцій даних томографії з пучком, що розходиться;
- `fanbeam` – обчислення віялово-променевих перетворень;
- `fliptform` – перестановка вихідних і результуючих даних у структурі TFORM;

- `ifanbeam` – обчислення інверсного віялово-променевого перетворення;
- `applycform` – використання перетворення колірних просторів;
- `makecform` – створення структури перетворення колірних значень;
- `whitepoint` – опис повнокольорової білої точки в колірному просторі;
- `immagbox` – локальне збільшення з використанням панелі прокручування.

***Аналіз зображень:***

- `imhist` – побудова гістограми;
- `improfile` – побудова профілю;
- `impixel` – визначення значення пікселя;
- `pixval` – управління режимом відображення значень пікселів;
- `mean2` – обчислення середнього значення елементів матриці;
- `std2` – обчислення середньоквадратичного відхилення елементів матриці;
- `corr2` – обчислення коефіцієнтів кореляції між двома матрицями;
- `xcorr2` – обчислення двовимірної взаємної кореляційної функції;
- `imabsdiff` – визначення відмітних ознак двох зображень;
- `imadd` – підсумовування двох зображень або підсумовування зображення і константи;
- `imcomplement` – доповнення зображень;
- `imdivide` – розділення двох зображень або розділення зображення на константу;
- `imlincomb` – обчислення лінійної комбінації двох зображень;
- `immultiply` – множення двох зображень або множення зображення на константу;
- `imsubtract` – віднімання двох зображень або віднімання константи із зображення;
- `regionprops` – визначення властивостей області зображення;
- `cpstruct2pairs` – конвертація `cpstruct` у найбільш важливі контрольні точки;
- `cp2tform` – виведення просторових перетворень між парою контрольних точок;
- `cpcorr` – визначення погоджених контрольних точок з використанням крос-кореляції;
- `cpselect` – інструмент вибору контрольних точок;
- `normxcorr2` – нормалізація двовимірної крос-кореляції;
- d`econvblind` – поліпшення зображень з використанням зворотної згортки;

- `deconvlucy` – поліпшення зображень з використанням методу Лакі-Річардсона;
- `deconvreg` – поліпшення зображень з використанням регуляризаційної фільтрації;
- `deconvwnr` – поліпшення зображень з використанням фільтра Вінера;
- `ippl` – перевірка наявності бібліотеки функцій (Intel Performance Primitives Library (IPPL));
- `getrangefromclass` – знаходження динамічного діапазону зображень на основі їх формату;
- `graycomatrix` – півтонова матриця суміжності для зображень;
- `graycoprops` – властивості півтонових матриць суміжності;
- `hough` – перетворення Хо;
- `houghlines` – знаходження ліній сегментації на основі перетворень Хо;
- `houghpeaks` – локалізація піків при перетвореннях Хо;
- `imdisplayrange` – відображення динамічного діапазону яскравості зображення;
- `entropy` – ентропія інтенсивності елементів зображення;
- `entropyfilt` – локальна ентропія інтенсивностей елементів зображення;
- `impixelinfoval` – засоби визначення інформації про піксель без текстових міток;
- `getimagemodel` – відновлення моделей об'єктів зображення на основі об'єктів зображення;
- `impixelregion` – засоби перегляду локального масиву пікселів;
- `impixelregionpanel` – панель інструментарію відображення локального масиву пікселів зображення;
- `impixelinfo` – засоби знаходження інформації про пікселі.

***Поліпшення зображень:***

- `histeq` – вирівнювання гістограми;
- `imadjust` – контрастування з гаммою;
- `brighten` – управління яскравістю палітри;
- `imnoise` – додавання шуму;
- `roifill` – заповнення областей інтересу;
- `stretchlim` – пошук меж підвищення контрасту зображення;
- `edgetaper` – виділення країв з використанням функції протяжності точок;
- `otf2psf` – перетворення оптичної функції у функцію протяжності точок;

- `psf2otf` – перетворення функції протяжності точок в оптичну функцію;
- `adapthisteq` – виконання контрастно обмеженої адаптивної еквалізації гістограми;
- `decorrstretch` – застосування декореляційного розтягування багатоканальних зображень;
- `axes2pix` – інтерактивні методи регулювання контрасту і яскравості;
- `imcontrast` – засоби посилення контрасту зображення.

***Фільтрація зображень:***

- `conv2` – згортка зображень;
- `convn` – згортка N–мірних сигналів;
- `convmtx2` – обчислення матриці згортки;
- `filter2` – двовимірна лінійна фільтрація;
- `freqz2` – двовимірна АЧХ;
- `fspecial` – завдання маски зумовленого фільтра;
- `fsamp2` – формування маски лінійного фільтра за бажаною АЧХ;
- `ftrans2` − формування маски лінійного фільтра методом перетворення частот;
- `fwind1` − формування маски лінійного фільтра за бажаною АЧХ з використанням одновимірного вікна;
- `fwind2` – формування маски лінійного фільтра за бажаною АЧХ з використанням двовимірного вікна;
- `blkproc` − оброблення блоків зображення;
- `bestblk` – визначення розміру блока;
- `nlfilter` – узагальнений нелінійний фільтр;
- `colfilt` – оптимізована операція фільтрації;
- `im2col` – перетворення фрагментів зображення в стовпці;
- `col2im` – перетворення допоміжного зображення;
- o`rdfilt`2 – рангова фільтрація;
- `medfilt2` – медіанна фільтрація;
- `wiener2` – адаптивна вінерівська фільтрація;
- `roifilt2` – фільтрація областей інтересу;
- `imfilter` – фільтрація двовимірних і багатовимірних зображень;
- `freqspace` – визначення відгуку в двовимірній частотній області (MATLAB Toolbox).

***Сегментація зображень:***

- `poly2mask` – перетворення деякої області в маску;
- `qtdecomp` – сегментація методом розділення;
- `qtgetblk` – знаходження блоків із квадродерева результатів сегментації;

- `qtsetblk` – заміна блоків–результатів сегментації;
- `edge` – виділення меж;
- `roipoly` – завдання області інтересу за допомогою полігона;
- `roicolor` – бінаризація за заданими кольорами;
- `watershed` – алгоритм маркерного водорозділу.

***Морфологічні операції над бінарним зображенням:***

- `uintlut` – обчислення нових значень масиву на основі табличних перетворень;
- `applylut` – перетворення бінарного зображення за допомогою таблиці перекодування;
- `bwboundaries` – відстежування локальних меж на бінарному зображенні;
- `bwmorph` – морфологічні операції над бінарним зображенням;
- `bwareaopen` – відкриття бінарних площ (малих об'єктів);
- `bwdist` – визначення періоду перетворення бінарних об'єктів;
- `bwfill` – заповнення областей фону;
- `bwhitmiss` – бінарні `hit-miss` операції;
- `bwlabeln` – установлення мітки зв'язаних елементів у багатовимірних бінарних зображеннях;
- `bwpack` – упакування бінарних зображень;
- `bwperim` – виділення меж бінарних об'єктів;
- `bwselect` – виділення об'єктів;
- `bwtraceboundary` – відстежування контурів бінарних зображень;
- `bwulterode` – гранична ерозія;
- `bwunpack` – розпаковування бінарних зображень;
- `conndef` – відсутність зв'язності;
- `dilate` – нарощування бінарного об'єкта;
- `erode` – ерозія бінарного об'єкта;
- `imbothat` – виконання низькочастотної фільтрації;
- `imclearborder` – заглушення світлової структури, пов'язаної з краями зображення;
- `imclose` – закриття зображення;
- `imdilate` – розширення зображення;
- `imerode` – ерозія зображення;
- `imextendedmax` – максимальна тривалість перетворень;
- `imextendedmin` – мінімальна тривалість перетворень;
- `imfill` – заповнення областей зображення;
- `imhmax` – H-максимальні перетворення;
- `imhmin` – H-мінімальні перетворення;

- `imimposemin` – установлення мінімуму;
- `imopen` – відкриття зображення;
- `imreconstruct` – морфологічне відновлення зображень;
- `imregionalmax` – максимум області;
- `imregionalmin` – мінімум області;
- `imtophat` – виконання високочастотної фільтрації;
- `makelut` – формування таблиці, тієї, що перекодувала.

***Пошук об'єктів і обчислення їхніх ознак:***

- `bwlabel` – пошук об'єктів;
- `bwarea` – обчислення площі об'єктів;
- `bweuler` – обчислення числа Ейлера;
- `imfeature` – обчислення ознак об'єктів;
- `imattributes` – одержання інформації про атрибути зображення.

***Перетворення Фур'є:***

- `fft2` – двовимірне БПФ;
- `fftn` – `n`-мірне БПФ;
- `ifft2` – зворотне двовимірне БПФ;
- `ifftn` – `n`–вимірне зворотне БПФ;
- `fftshift` – перегрупування вихідного масиву перетворення Фур'є.

***Дискретне косинусне перетворення:***

- `dct2` – двовимірне ДКП;
- `idct2` – зворотне двовимірне ДКП;
- `dctmtx` – обчислення матриці коефіцієнтів ДКП.

***Перетворення Радону:***

- `radon` – пряме перетворення Радону;
- `iradon` – зворотне перетворення Радону;
- `phantom` – створення модельного зображення голови.

***Створення і оброблення структурних елементів:***

- `getheight` – створення вертикальних структурних елементів;
- `getneighbors` – визначення місця розташування сусідніх структурних елементів;
- `getnhood` – створення сусідніх структурних елементів;
- `getsequence` – створення послідовності розкладених структурних елементів;
- `isflat` – повернення однакових структурних елементів;
- `reflect` – подання структурних елементів через їх центр;
- `strel` – створення морфологічних структурних елементів;
- `translate` – перетворення структурних елементів.

***Операції з масивами:***

- `padarray` – порожній масив.

## БІБЛІОГРАФІЧНИЙ СПИСОК


Блаттер, К. Вейвлет-анализ. Основы теории [Текст] / К. Блаттер. – М. : Техносфера, 2004. – 352 с.

Воробьев, В. И. Теория и практика вейвлет-преобразования [Текст] / В. И. Воробьев, В. Г. Грибулин. – СПб.: ВУС, 1999. – 285 с.

Гонсалес, Р. Цифровая обработка изображений в среде MATLAB [Текст] / Р. Гонсалес, В. Вудс, С. Эддинс; пер. с англ. В. В. Чепыжова. – М. : Техносфера, 2006. – 512 с.

Добеши, И. Десять лекций по вейвлетам [Текст] / И. Добеши. – М.: НИЦ “Регулярная и хаотическая динамика”, 2004. – 183 с.

Дьяконов, В. П. MATLAB. Обработка сигналов и изображений. Специальный справочник [Текст] / В. П. Дьяконов, И. В. Абраменкова. – СПб. : Питер, 2002. – 434 с.

Дьяконов, В. П. MATLAB 6.0/6.1/6.5/6.5+SP1 + Simulink 4/5. Обработка сигналов и изображений [Текст] / В. П. Дьяконов. – М. : СОЛОН-Пресс, 2005. – 583 с.

Короновский, А. А. Непрерывный вейвлет-анализ и его приложения [Текст] / А. А. Короновский, А. Е. Храмов. – М. : Физматлит, 2003. – 178 с.

Косых, В. П. Цифровая обработка изображений [Текст] : учеб. пособие / В. П. Косых. – Новосибирск : НГУ, 2006. – 95 с.

Лайонс, Ричард. Цифровая обработка сигналов [Текст] : пер. с англ. / Ричард Лайонс; под ред. А. А. Бритова. – 2-е изд. – М. : БИНОМ-Пресс, 2007. – 653 с.

Мала, С. Вейвлеты в обработке сигналов [Текст] / С. Мала. – М. : Мир, 2005. – 276 с.

Новиков, Л. В. Теория всплесков [Текст] / Л. В. Новиков, В. Ю. Протасов, М. А. Скопина. – М. : Физматлит, 2006. – 294 с.

Новиков, Л. В. Основы вейвлет-анализа сигналов [Текст] / Л. В. Новиков. – СПб. : МОДУС, 1999. – 312 с.

Прэтт, У. Цифровая обработка изображений [Текст] / У. Прэтт. – М. : Мир, 1982. – Т. 1, 2. – 791 с.

Сергиенко, А. Б. Цифровая обработка сигналов [Текст] / А. Б. Сергиенко. – 2-е изд. – СПб. : Питер, 2005. – 587 с.

Солонина, А. И. Цифровая обработка сигналов. Моделирование в MATLAB [Текст] / А. И. Солонина, С. М. Арбузов. – СПб. : БХВ-Петербург, 2008. – 437 с.

Яковлев, А. Н. Введение в вейвлет-преобразования [Текст] / А. Н. Яковлев. – Новосибирск : НГТУ, 2003. – 288 с.

**ЗМІСТ**

Навчальне видання

**Краснов Леонід Олександрович**
**Дергачов Костянтин Юрійович**

# УПРАВЛІННЯ В УМОВАХ НЕВИЗНАЧЕНОСТІ (ОБРОБЛЕННЯ ЗОБРАЖЕНЬ І ВІДЕОІНФОРМАЦІЇ)

Редактор Т. Г. Кардаш

Зв. план, 2017
Підписано до друку
Формат 60х84 1/16. Папір офс. № 2. Офс. друк
Ум. друк. арк. 6.9. Обл.-вид. арк.7,75. Наклад 100 пр.
Замовлення 51. Ціна вільна

Видавець і виготовлювач
Національний аерокосмічний університет ім. М. Є. Жуковського
«Харківський авіаційний інститут»
61070, Харків-70, вул. Чкалова, 17
http://www.khai.edu
Видавничий центр «ХАІ»
61070, Харків-70, вул. Чкалова, 17
izdat@khai.edu

Свідоцтво про внесення суб`єкта видавничої справи до Державного реєстру видавців, виготовлювачів і розповсюджувачів видавничої продукції сер. ДК № 391 від 30.02.2001