МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ ВИЩИЙ НАВЧАЛЬНИЙ ЗАКЛАД
«НАЦІОНАЛЬНИЙ ГІРНИЧИЙ УНІВЕРСИТЕТ»

ГЕОЛОГОРОЗВІДУВАЛЬНИЙ ФАКУЛЬТЕТ

Кафедра геоінформаційних систем

К.Л. Сергєєва

РОЗПІЗНАВАННЯ ОБРАЗІВ ТА ОБРОБКА ЗОБРАЖЕНЬ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ З ДИСЦИПЛІНИ

для студентів спеціальностей
122 Комп’ютерні науки та інформаційні технології
(спеціалізація “Комп’ютерний еколого-економічний моніторинг“),
193 Геодезія та землеустрій
(спеціалізація “Геоінформаційні системи і технології”)

Дніпро
НГУ
2016

Сергєєва К.Л.
Розпізнавання образів та обробка зображень. Методичні рекомендації до лабораторних занять з дисципліни для студентів спеціальностей 122 Комп'ютерні науки та інформаційні технології, 193 Геодезія та землеустрій / К.Л. Сергєєва ; М-во освіти і науки України, Нац. гірн. ун-т. – Дніпро: НГУ, 2016. – 70 с.

Автор:
К.Л. Сергєєва, канд. техн. наук.

Затверджено до видання редакційною радою ДВНЗ «НГУ» (протокол № 9 від 29.10.2016) за поданням кафедри геоінформаційний систем (протокол № 3 від 26.09.2016).

Методичні матеріали призначено для виконання лабораторних робіт з дисципліни "Розпізнавання образів та обробка зображень" студентами спеціальностей 122 Комп'ютерні науки та інформаційні технології та 193 Геодезія та землеустрій.

Розглянуто рекомендації до використання методів розпізнавання образів та обробки зображень у середовищах MATLAB і Borland Delphi. Подано рекомендації до практичної реалізації статистичних, детерміністських та нейромережевих методів розпізнавання образів, обробки растрових даних у просторовій та частотній областях, сегментації зображень та морфологічної обробки. Надано етапи створення елементів програмних додатків обробки растрових зображень у середовищі Borland Delphi.

Рекомендації орієнтовано на активізацію виконавчого етапу навчальної діяльності студентів.

Відповідальний за випуск: завідувач кафедри геоінформаційних систем, д-р техн. наук, проф. Б.С. Бусигін.

ЗМІСТ

ПЕРЕДМОВА

Метою викладання навчальної дисципліни “Розпізнавання образів та обробка зображень” є закладення фундаменту уявлення студентами основних понять розпізнавання образів та складових інформаційних систем обробки зображень, основних принципів оптимізації простору ознак, методів розпізнавання образів, обробки растрових даних в просторовій та частотній областях за допомогою застосування мови макрокоманд та пакету прикладних програм MATLAB і середовища розробки Borland Delphi.

У лабораторному практикумі розглянуто:

– основні поняття розпізнавання образів та складові частини систем розпізнавання образів;

– методи обробки даних в системах розпізнавання;

– основні принципи логічних, статистичних, нейромережевих методів розпізнавання образів;

– методи та алгоритми вирішення загальних питань цифрової обробки растрових даних.

По виконанню лабораторних робіт студенти повинні вміти:

– реалізовувати в середовищі MATLAB статистичні, детерміністські та нейромережеві методи розпізнавання образів;

– здійснювати обробку растрових даних в просторовій та частотній областях;

– здійснювати реалізацію алгоритмів просторової та частотної обробки растрових зображень, сегментації зображень та розпізнавання об’єктів;

– вирішувати прикладні задачі просторового аналізу у системі MATLAB і середовищі розробки Borland Delphi.

Для успішного виконання лабораторних робіт студент повинен володіти теоретичними складовими дисципліни, а також знаннями, отриманими під час вивчення дисциплін «Інформатика та програмування», «Теорія ймовірностей, ймовірнісні процеси та математична статистика», «Математичні методи і моделі».

У результаті виконання лабораторних робіт студент повинен:

- отримати навички роботи з комп'ютером, з його технічним та програмним забезпеченням (носіями інформації, системами програмування, графічними редакторами, зображеннями тощо);

- оволодіти базовими знаннями про гетерогенну просторову інформацію, способи її обробки, аналізу та відображенню;

- отримати знання методів та технологій обробки цифрових зображень та розпізнавання образів.

Лабораторна робота 1
Основні поняття розпізнавання образів.
Міри подібності між об'єктами в багатовимірному просторі ознак

Об'єкт – образи еталонних об’єктів та об’єктів, що класифікуються.

Предмет – методи розпізнавання образів на підставі розрахунку мір близькості.

Мета – закріпити знання відносно методів розрахунку мір подібності між об'єктами у багатовимірному просторі ознак та розпізнавання образів. Використовуючи пакет прикладних програм MATLAB, розробити програму розпізнавання об'єктів на підставі функції міри близькості між об’єктами та за методом найближчого сусіда.

Теоретичні положення

Роботу виконувати в пакеті прикладних програм MATLAB за наступною схемою:

1. Розрахунок міри подібності (відстані) *d* між об'єктами, що розпізнаються, та еталонами залежно від варіанту (реалізувати як окрему функцію). Результати можуть бути представлені у вигляді таблиці 1.1.

Таблиця 1.1

Міри подібності між об'єктами, що розпізнаються, та еталонами

		Навчальна вибірка							
		1 клас (c_1)				***2 клас (c_2)***			
Об'єкти, що розпізнаються		$X_1^{(1)}$	$X_2^{(1)}$	…	$X_n^{(1)}$	$X_1^{(2)}$	$X_2^{(2)}$	…	$X_n^{(2)}$
	X_1	$d(X_1, X_1^{(1)})$	$d(X_1, X_2^{(1)})$	…	…	$d(X_1, X_1^{(2)})$	$d(X_1, X_2^{(2)})$	…	…
	X_2	$d(X_2, X_1^{(1)})$	…	…	…	$d(X_2, X_1^{(2)})$	…	…	…
	…	…	…	…	…	…	…	…	…
	X_l	…	…	…	$d(X_1, X_n^{(1)})$	…	…	…	$d(X_1, X_n^{(2)})$

2. Безпосереднє розпізнавання об'єктів на підставі функції міри близькості. Функція міри близькості між об'єктом X і класом c_j знаходиться наступним чином:

$$d(X, c_j) = \left[\prod_{X_i^{(j)} \in c_j} d\left(X, X_i^{(j)}\right) \right]^{\frac{1}{n}}, \; j = 1,2; \;\; i = 1,\ldots,n,$$

де n – кількість еталонних об'єктів кожного класу (c_1 і c_2),
$d\left(X, X_i^{(j)}\right)$ – міра подібності між об'єктами X і $X_i^{(j)}$.

Вирішальне правило: об'єкт X зараховується в клас c_i, якщо

$$d(X, c_i) < d(X, c_j),\ i \neq j,\ i, j = 1,2.$$

3. Розпізнавання за методом найближчого сусіда. Вирішальне правило: об'єкт X зараховується в той клас, до якого належить еталонний об'єкт з мінімальним значенням міри подібності по відношенню до об'єкту X.

4. Відображення результатів розпізнавання (функція *plot3* в MATLAB). Точки, що відповідають різним класам, відобразити різними кольорами і маркерами. Виділити об'єкти навчальної вибірки. Результати представити у вигляді 3D-графіка, а також в кожній проекції (*Контекстне меню → Go to XY view, Go to XZ view, Go to YZ view*).

5. Формування класу "Країни, що розвиваються". З числа об'єктів розпізнавання виділити еталонні об'єкти для третього класу – "Країни, що розвиваються". Виконати процедуру розпізнавання для трьох класів ("Розвинені країни", "Країни, що розвиваються", "Відсталі країни").

Перераховані нижче країни зазвичай відносяться до числа ринків, що розвиваються: Аргентина, Бразилія, Чилі, Китай, Колумбія, Чехія, Єгипет, Угорщина, Індія, Індонезія, Малайзія, Мексика, Марокко, Пакистан, Перу, Філіппіни, Польща, Росія, ПАР, Таїланд, Туреччина, Південна Корея, Тайвань, Ізраїль.

6. Виконання процедури розпізнавання на спеціально змодельованих даних. Навчальна вибірка (у вигляді векторів ознак) для двох класів задана в файлах "*Samples_1.txt*", "*Samples_2.txt*", що зберігаються в папці з номером варіанту. Об'єкти, що підлягають розпізнаванню, задані там же у файлі "*Objects.txt*". Читання даних з текстового файлу можна виконати за допомогою функції *dlmread()*.

Постановка завдання

На підставі показників, представлених для 174 країн світу, виконати розпізнавання об'єктів (країн) на два класи: "*Розвинені країни*" і "*Країни, що розвиваються / відсталі країни*". Розпізнавання виконувати на підставі навчальної вибірки, заданої для кожного з двох класів (файл "*Z:\Subjects\POO3\Lab_1\data.xls*").

Варіанти

Номер п/п	Варіант вибірки	Міра відстані *d*	
1	1	Евклідова відстань	Відстань city–block
2	2	Відстань city–block	Косинусна відстань
3	3	Метрика Мінковського (p = 3)	Кореляційна відстань
4	4	Косинусна відстань	Метрика Мінковського (p = 3)
5	5	Кореляційна відстань	Евклідова відстань
6	1	Евклідова відстань	Метрика Мінковського (p = 3)
7	2	Відстань city–block	Кореляційна відстань
8	3	Метрика Мінковського (p = 3)	Відстань city–block
9	4	Косинусна відстань	Евклідова відстань
10	5	Кореляційна відстань	Косинусна відстань
11	1	Евклідова відстань	Косинусна відстань
12	2	Відстань city–block	Метрика Мінковського (p = 3)
13	3	Метрика Мінковського (p = 3)	Евклідова відстань
14	4	Косинусна відстань	Відстань city–block
15	5	Кореляційна відстань	Метрика Мінковського (p = 3)

Питання для підготовки до захисту лабораторної роботи

1. Що являє собою навчальна вибірка?

2. Як виконується розпізнавання об'єктів на підставі функції міри близькості?

3. Як виконується розпізнавання об'єктів за методом найближчого сусіда?

4. Якою є структура навчальної вибірки?

5. Наведіть вираз для розрахунку Евклідової відстані між об'єктом, що розпізнається, та еталоном.

Лабораторна робота 2
Непараметричні методи розпізнавання.
Метод парзенівського вікна. Розпізнавання для випадку декількох класів.

Об'єкт – образи еталонних об’єктів та об’єктів, що класифікуються.

Предмет – метод парзенівського вікна.

Мета – закріпити знання відносно непараметричних методів розпізнавання. Використовуючи пакет прикладних програм MATLAB, розробити програму розпізнавання образів за методом парзенівського вікна.

Теоретичні положення

Локальна емпірична оцінка щільності у методі парзенівського вікна задається формулою Парзена-Розенблата:

$$f(X|c_i) = \frac{1}{n_i} \sum_{X_j \in c_i} K\left[\frac{d(X, X_j)}{h}\right],$$

де $K(z)$ – функція ядра, h – додатний параметр, який називається шириною вікна, n_i – число навчальних об'єктів класу c_i,

Ширина вікна h вирішальним чином впливає на якість відновлення щільності. При дуже вузькому вікні ($h \to 0$) щільність концентрується поблизу навчальних об'єктів, і функція $f(X|c_i)$ зазнає різких скачків. При занадто широкому вікні щільність надмірно згладжується і в межах $h \to \infty$ вироджується у константу. Таким чином, повинно існувати оптимальне значення ширини вікна h^*, при якому відновлена щільність є найбільш адекватною.

Для розпізнавання використовується байєсівське класифікаційне правило:

$$B_i(X) = P(\pi_i) f(X|\pi_i)$$

де $P(c_i)$ – апріорна ймовірність появи класу c_i; $f(X|c_i)$ – щільність розподілу класу c_i у р-вимірному просторі ознак за формулою Парзена-Розенблатта.

Оцінка апріорних ймовірностей $P(c_i)$ може бути обчислена за навчальною вибіркою $P(c_i) = \frac{n_i}{N}$, де n_i – число навчальних об'єктів класу c_i, N – число навчальних об'єктів усіх класів.

Вирішальне правило: об'єкт X, що підлягає класифікації, зараховується у клас c_i на підставі максимуму $B_i(X)$, де $i = 1,2,...,K$; K – число класів.

Постановка завдання

1. На підставі навчальної вибірки для двох класів (файли "Samples_1.txt", "Samples_2.txt", див. лаб. роб. №1) та розпізнаваних об'єктів (файл "Objects.txt") виконати у середовищі MATLAB розпізнавання методом парзенівського вікна з використанням однієї з наведених нижче функцій ядра. Визначити оптимальне

значення ширини вікна h. Для знаходження значень функції ядра використати матрицю відстаней з лабораторної роботи №1.

2. Виконати розпізнавання для випадку шести класів (файли "Samples_1-6.txt", "Objects_2.txt").

3. Відобразити графічно результати розпізнавання. Точки, які відповідають різним класам, відображати різними кольорами і маркерами. Виділити об'єкти навчальної вибірки. Результати представити у вигляді 3D-графіка, а також у кожній проекції (*Контекстне меню → Go to X-Y view, Go to X-Z view, Go to Y-Z view*).

4. Зробити змістовні висновки щодо якості розпізнавання. Оформити і захистити звіт.

Функції ядра

Ядро	Вираз
Квадратичне	$Q(r)=\frac{15}{16}\left(1-r^2\right)^2, r \le 1$
Трикутне	$T(r)=\left(1-\lvert r\rvert\right), r \le 1$
Гаусівське	$G(r)=(2\pi)^{-\frac{1}{2}} exp(-r^2/2)$

Питання для підготовки до захисту лабораторної роботи

1. Наведіть функції ядра, що використовуються у формулі Парзена-Розенблата.

2. Що являє собою параметр ширини вікна у функції ядра?

3. Розкрийте зміст байєсівського класифікаційного правила.

4. Як може бути обчислена оцінка апріорних ймовірностей?

5. Назвіть вирішальне правило розпізнавання за методом парзенівського вікна.

Лабораторна робота 3
Вивчення інтерфейсу MATLAB для роботи з нейронними мережами. Створення простих нейронних мереж.

Об'єкт – вектори характеристик об’єктів, що класифікуються.

Предмет – штучні нейронні мережі.

Мета – закріпити знання відносно нейромережевих методів розпізнавання. Ознайомитися з пакетом прикладних програм Neural Network Toolbox. розробити програму розпізнавання об'єктів за методом парзенівського вікна. Отримати навики створення штучних нейронних мереж.

Теоретичні положення

1. GUI-інтерфейс для пакету прикладних програм Neural Network Toolbox

До складу пакету прикладних програм Neural Network Toolbox (ППП NNT) MATLAB входить інструментальний засіб *NNTool*. Цей графічний інтерфейс дозволяє, не звертаючись до командного вікна MATLAB, виконувати створення, навчання, моделювання, а також імпорт та експорт нейронних мереж і даних, використовуючи тільки інструментальні можливості GUI-інтерфейсу. Зазвичай, такі інструменти найбільш ефективні лише на початковій стадії роботи з пакетом, оскільки мають певні обмеження. Зокрема, інтерфейс *NNTool* допускає роботу тільки з найпростішими одношаровими і двошаровими нейронними мережами, але при цьому користувач виграє в часі й ефективності освоєння нових об'єктів.

Виклик GUI-інтерфейсу *NNTool* можливий або командою *nntool* з командного рядка, або з вікна запуску програм *Launch Pad* за допомогою опції *NNTool* з розділу *Neural Network Toolbox*. Після виклику на екрані з'являється вікно *Network / Data Manager* (Управління мережею / даними) (рис. 3.1):

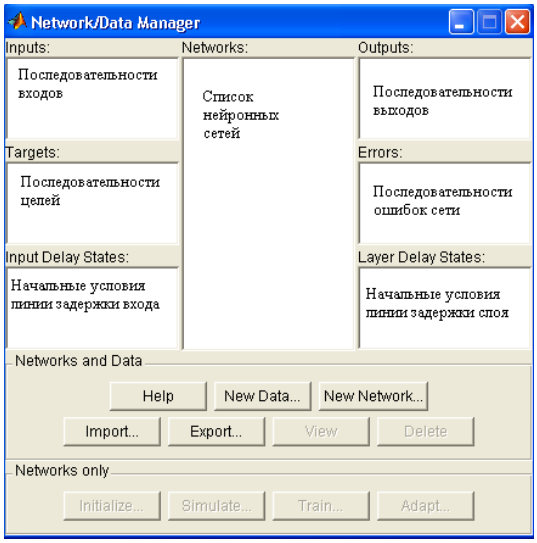


Рис. 3.1 Вікно *Network / Data Manager*

– *Help* – кнопка виклику вікна підказки *Network / Data Manager Help* (рис. 3.2);

– *New Data ...* – кнопка виклику вікна формування даних *Create New Data* (рис. 3.3);

– *New Network ...* – кнопка виклику вікна створення нової нейронної мережі *Create New Network* (рис. 3.4);

– *Import ...* – кнопка виклику вікна для імпорту або завантаження даних *Import or Load to Network / Data Manager* (рис. 3.5);

– *Export ...* – кнопка виклику вікна для експорту або запису даних у файл *Export or Save from Network / Data Manager* (рис. 3.6).

Кнопки *View*, *Delete* стають активними тільки після створення та активізації даних, що відносяться до послідовностей входу, цілі, виходу або помилок мережі. Кнопка *View* дозволяє переглянути, а кнопка *Delete* – видалити активізовані дані.

Кнопки *View*, *Delete*, *Initialize...*, *Simulate...*, *Train...*, *Adapt...* стають активними після створення та активізації самої нейронної мережі. Вони дозволяють переглянути, видалити, ініціалізувати, промоделювати, навчити або адаптувати нейронну мережу і будуть описані нижче.

Насамперед, розглянемо призначення та способи роботи з перерахованими вище вікнами.

Вікно *Network / Data Manager Help*. Це вікно підказки показано на рис. 3.2 і описує правила роботи з диспетчером *Network / Data Manager* під час створення нейронної мережі.

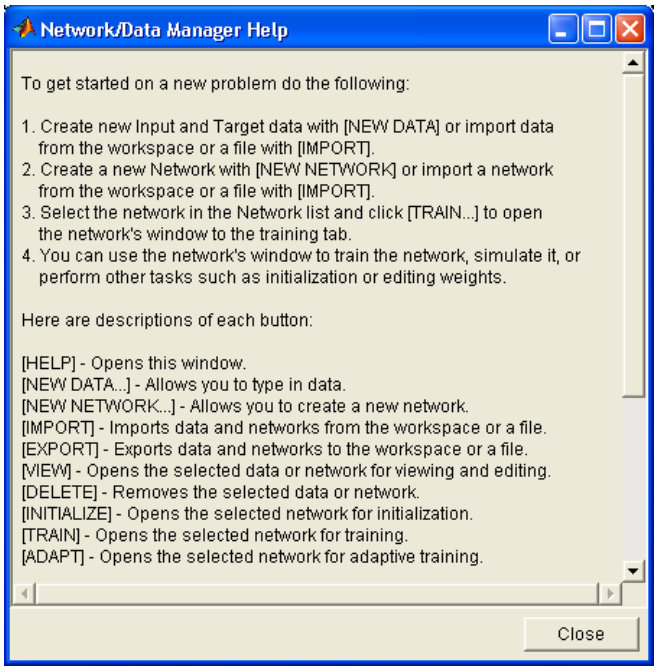


Рис. 3.2 Вікно *Network / Data Manager Help*

Щоб створити нейронну мережу, необхідно виконати наступні операції:

– сформувати послідовності входів і цілей (кнопка *New Data*), або завантажити їх з робочої області системи MATLAB або з файлу (кнопка *Import*);

– створити нову нейронну мережу (кнопка *New Network*), або завантажити її з робочої області системи MATLAB або з файлу (кнопка *Import*);

– вибрати тип нейронної мережі і натиснути кнопку *Train...*, щоб відкрити вікно для завдання параметрів процедури навчання;

– відкрити вікно *Network* для перегляду, ініціалізації, моделювання, навчання та адаптації мережі.

Вікно *Create New Data*. Це вікно показано на рис. 3.3 і включає 2 області редагування тексту для запису імені даних (область *Name*) і введення самих даних (область *Value*), а також 6 кнопок для зазначення типу вхідних даних.

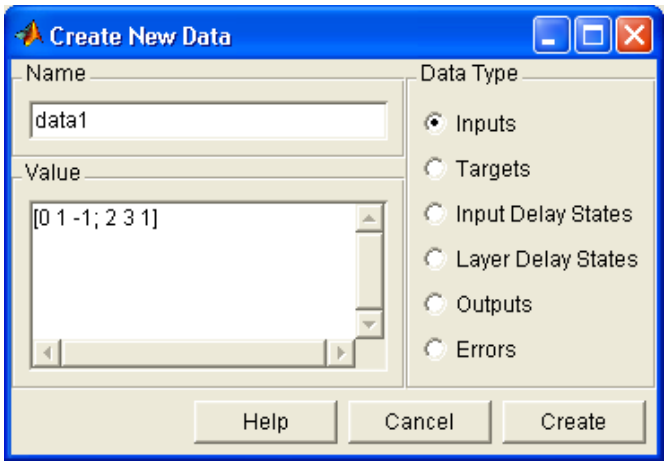


Рис. 3.3 Вікно *Create New Data*

Розрізняють такі типи даних:

Inputs (*Входи*) – послідовність значень входів;

Targets (*Цілі*) – послідовність значень мети;

Input Delay States (*Стани ЛЗ входу*) – початкові умови лінії затримки на вході;

Layer Delay States (*Стани ЛЗ шару*) – початкові умови лінії затримки у шарі;

Outputs (*Виходи*) – послідовність значень виходу мережі;

Errors (*Помилки*) – різниця значень цілей і виходів.

Як правило, користувач задає тільки послідовності входу і цілі, тобто типи даних *Inputs* і *Targets*. При цьому слід пам'ятати, що при адаптації нейронної мережі дані повинні бути представлені у вигляді масиву осередків.

Вікно *Create New Network*. Це вікно показано на рис. 3.4 і включає поля для завдання параметрів створюваної мережі. Залежно від типу мережі кількість полів та їх назви змінюються.

Звернемося до опису полів.

Network Name (*Ім'я мережі*) – стандартне ім'я мережі, що привласнюється GUI-інтерфейсом *NNTool*; у процесі створення нових мереж порядковий номер буде змінюватися автоматично.

Network Type (*Тип мережі*) – список мереж, доступних для роботи з інтерфейсом *NNTool*. Для зручності цей список повторений у таб. 3.1. Інтерфейс *NNTool* дозволяє створювати нейронні мережі тільки з одним або двома шарами.

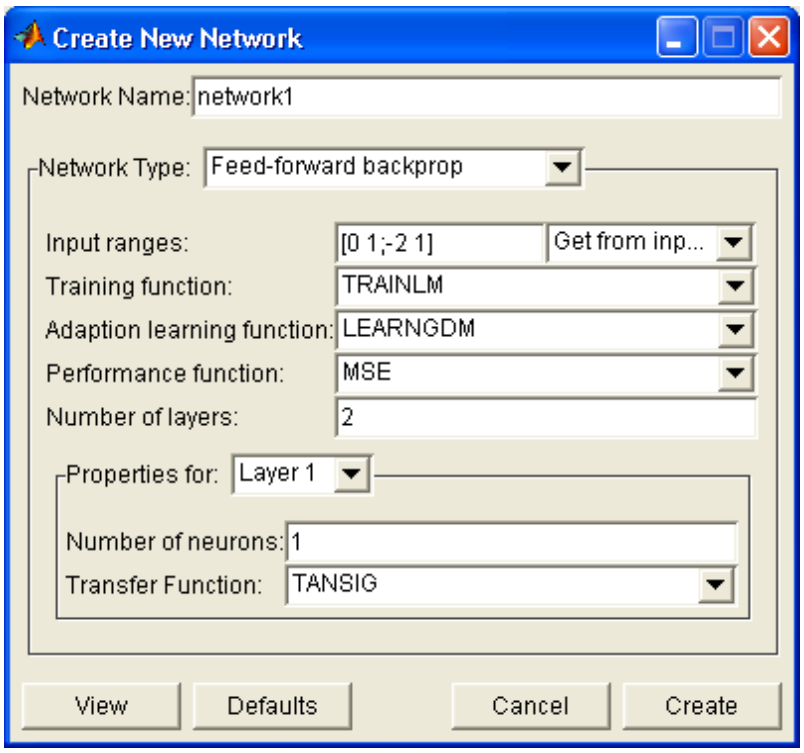


Рис. 3.4 Вікно *Create New Network*

Примітки:

– для мереж 2, 3, 7 інтерфейсу *NNTool* версії MATLAB 6.1 не забезпечується перегляд структурних схем;

– мережі 5, 9 допускають введення ліній затримок на вході;

– мережі 3 допускають введення ліній затримок у шарі;

– мережі з двома шарами мають послідовну структуру, коли вихід першого шару слугує входом другого шару. Виняток становлять мережі 3, які допускають наявність зворотного зв'язку у першому шарі і передачу вхідного сигналу на входи обох шарів.

Продовжимо опис полів.

Input ranges (*Діапазони входу*) – допустимі межі входів, які або призначаються користувачем, або визначаються автоматично за іменем вхідної послідовності, обраної зі списку *Get from Input*.

Training function (*Функція навчання*) – список функцій навчання.

Adaption learning function (*Функції настройки для режиму адаптації*) – список функцій налаштувань.

Performance function (*Функція якості навчання*) – список функцій оцінки якості навчання.

Number of layers (*Кількість шарів*) – кількість шарів нейронної мережі.

Properties for (*Властивості*) – список шарів: *Layer 1* (шар 1), *Layer 2* (шар 2).

Number of neurons (*Кількість нейронів*) – кількість нейронів у шарі.

Transfer function (*Функція активації*) – функція активації шару.

Вікно *Import or Load to Network / Data Manager*. Це вікно показано на рис. 3.5 і включає 3 поля.

Таблиця 3.1

Список мереж, доступних для роботи з інтерфейсом NNTool

№ п/п	Тип мережі	Назва мережі	Число шарів	Параметри навчання
1	Competitive	Конкуруюча мережа	1	IW{l,l},b{l}
2	Cascade-forward backprop	Каскадна мережу з прямим поширенням сигналу і зворотним поширенням помилки	2	IW{l,l},b{l}, LW{2,1}, IW{2, l},b{2}
3	Elman backprop	Мережа Елмана зі зворотним поширенням помилки	2	IW{l,l},b{l}, LW{2,l},b{2}, LW{1,1}
4	Feed-forward backprop	Мережа з прямим поширенням сигналу і зворотним поширенням помилки	2	IW{l,l},b{l}, LW{2, l},b{2}
5	Time delay backprop	Мережа із запізненням і зворотним поширенням помилки	2	IW{l,l},b{l}, LW{2, l},b{2}
6	Generalized regression	Узагальнена регресійна мережа	2	IW{l,l},b{l}, LW{2,1}.
7	Hopfield	Мережа Хопфілда	1	LW{l,l},b{l}.
8	Linear Layer	Лінійний шар (створення)	1	IW{l,l},b{l}
9	Linear Layer (train)	Лінійний шар (навчання)	1	IW{l,l},b{l}
10	LVQ	Мережа для класифікації вхідних векторів	2	IW{1,1},LW{2,1}
11	Perseptron	Персептрон	1	IW{l,l),b{l}
12	Probabilistic	Імовірнісна мережа	2	IW{l,l},b{l), LW{2,1}
13	Radial basis (exact fit)	Радіальна базисна мережа з нульовою помилкою	2	IW{1, 1},b{l}, LW{2, 1}
14	Radial basis (fewer neurons)	Радіальна базисна мережа з мінімальним числом нейронів	2	IW{l,l},b{l}, LW{2, l},b{2}
15	Self-organizing map	Самоорганізуюча карта Кохонена	1	IW{1,1}

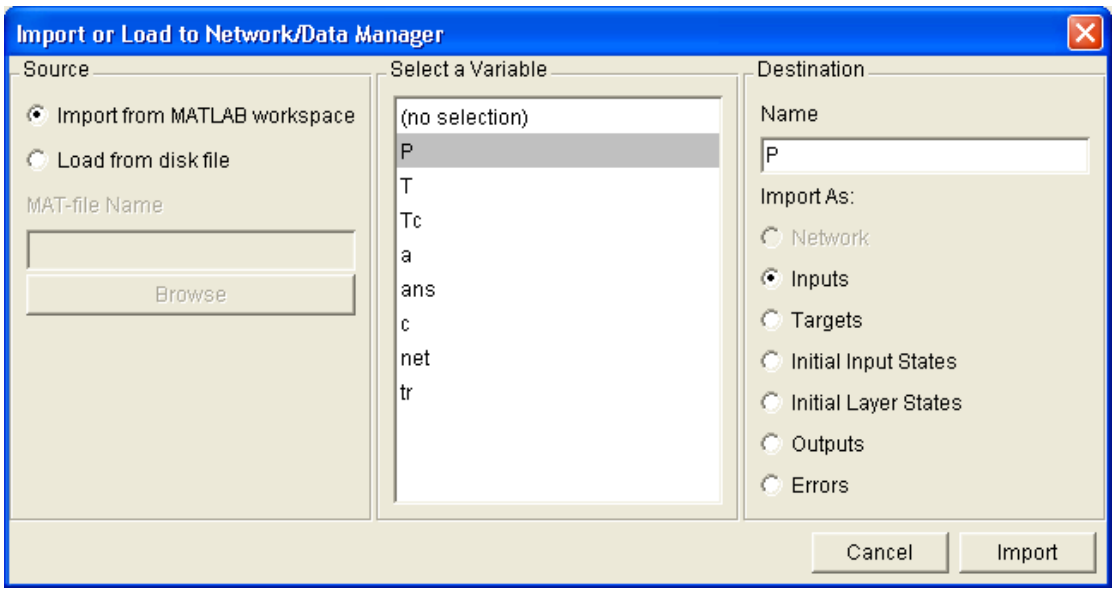


Рис. 3.5 Вікно *Import or Load to Network / Data Manager*

Source (*Джерело*) – поле для вибору джерела даних. Це або робоча область системи MATLAB (кнопка вибору *Import from MATLAB Workspace*), або файл (кнопка вибору *Load from disk file*).

Якщо вибрана перша кнопка, то у полі *Select a Variable* ви можете бачити всі змінні робочої області і, вибравши одну з них, наприклад *P*, можете визначити її у полі *Destination* (*Призначення*) як послідовність входу *Inputs* (*Входи*).

Якщо вибирається кнопка *Load from disk file*, то активізується поле *MAT-file Name* і кнопка *Browse*, що дозволяє почати пошук і завантаження файлу з файлової системи.

Вікно *Export or Save from Network / Data Manager*. Це вікно показано на рис. 3.6 і дозволяє передати дані з робочої області GUI-інтерфейсу *NNTool* у робочу область системи MATLAB або записати їх у вигляді файлу на диску.

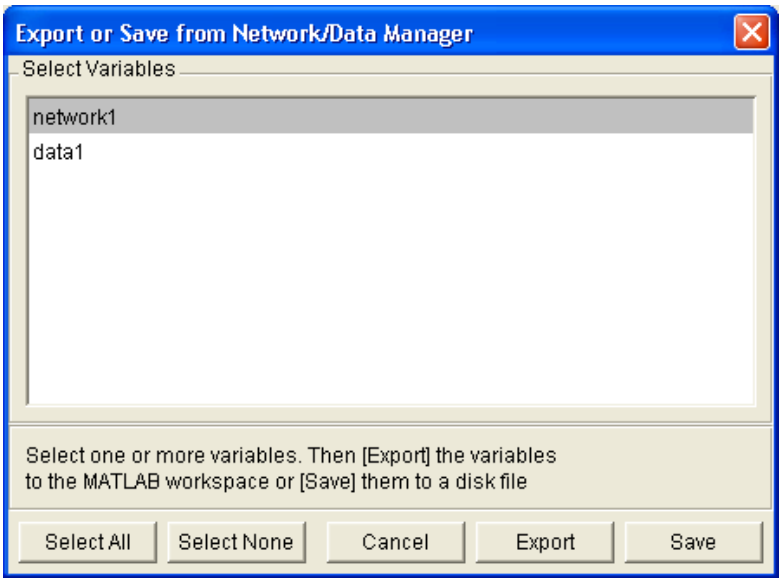


Рис. 3.6 Вікно *Export or Save from Network / Data Manager*

В даному випадку обрана змінна *network 1*, яка належить до класу *network object* і описує нейронну мережу. Після того як ця змінна експортована у робочу область, можна, наприклад, побудувати модель нейронної мережі у системі *Simulink* за допомогою оператора *gensim*.

Діалогова панель *Network* показана на рис. 3.7.

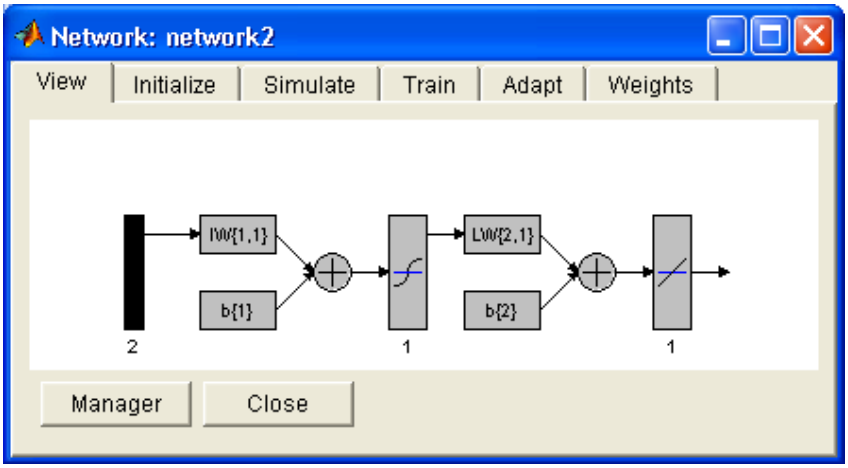


Рис. 3.7 Діалогова панель *Network*

Вона відкривається тільки у тому випадку, коли у вікні *Network / Data Manager* виділена створена мережа і стають активними кнопки *View*, *Initialize*, *Simulate*, *Train*, *Adapt*.

Панель має 6 закладок:

– *View* (*Переглянути*) – структура мережі;
– *Initialize* (*Ініціалізація*) – завдання початкових ваг і зміщень;
– *Simulate* (*Моделювання*) – моделювання мережі;
– *Train* (*Навчання*) – навчання мережі;
– *Adapt* (*Адаптація*) – адаптація та налаштування параметрів мережі;
– *Weights* (*Ваги*) – перегляд встановлених ваг і зміщень.

Особливості роботи з відповідними вікнами будуть розглянуті на наведених нижче прикладах створення конкретних нейронних мереж.

Приклад 1. Нейронна мережа з прямою передачею сигналу

Завдання: створити і навчити нейронну мережу виконанню операції $y = x_1 + x_2$, якщо задані послідовності входу $P = [1\ 0.5\ 0\ 1;\ -2\ 0\ 0.5\ 1]$ і цілі $T = [-1\ 0.5\ 0.5\ 2]$.

Сформуємо послідовності входів і цілей в робочій області GUI-інтерфейсу *NNTool*, використовуючи вікно *Create New Data*. Виберемо нейронну мережу типу *feed-forward backprop* з прямою передачею сигналу і зі зворотним поширенням помилки. Схема цієї мережі показана на рис. 3.7.

Виконаємо ініціалізацію мережі, для чого виберемо закладку *Initialize*, відкриється діалогова панель, показана на рис. 3.8. Діапазони значень вихідних даних виберемо зі спадаючого меню *Get from input*. Для введення встановлених

діапазонів і ініціалізації ваг треба скористатися кнопками *Set Ranges* (*Установити діапазони*) і *Initialize Weights* (*Ініціалізувати ваги*). Якщо потрібно повернутися до попередніх діапазонів, то слід вибрати кнопки *Revert Ranges* (*Повернути діапазони*) і *Revert Weights* (*Повернути ваги*).

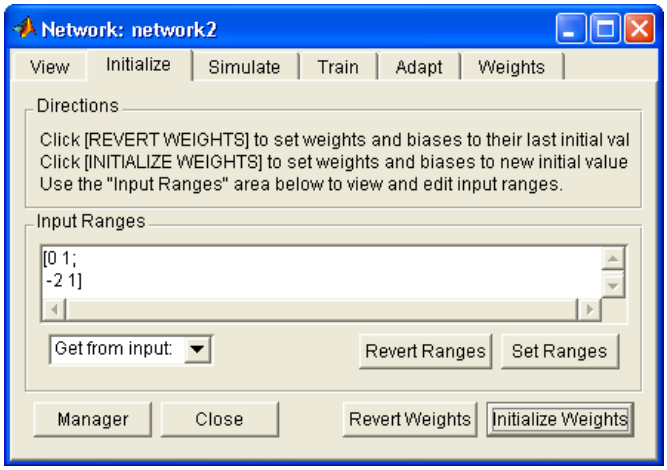


Рис. 3.8 Закладка *Initialize* вікна *Create New Data*

Потім виконується навчання мережі, для чого вибирається закладка *Train* і відкривається діалогова панель, показана на рис. 3.9.

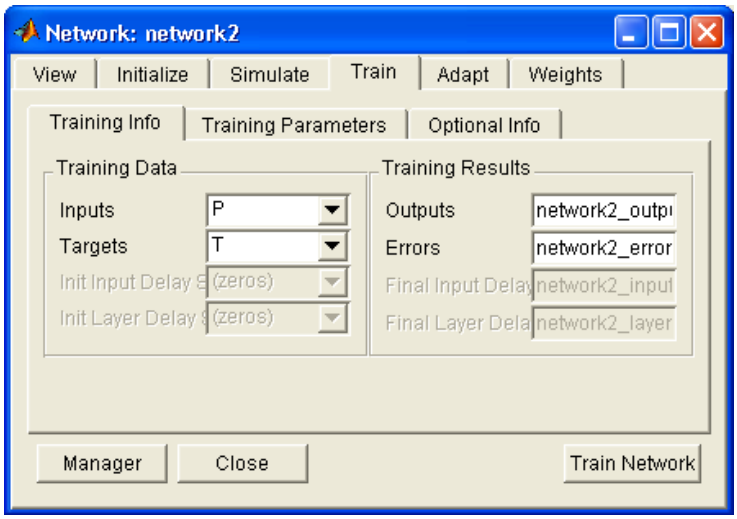


Рис. 3.9 Панель *Train* вікна *Create New Data*

Панель має три закладки:

Training Info (*Інформація про навчальні послідовності*) (рис. 3.10);

Training Parameters (*Параметри навчання*) (рис. 3.11);

Optional Info (*Додаткова інформація*) (рис. 3.12).

Остання закладка використовується в тому випадку, коли у процесі навчання задіяні контрольні й тестові послідовності.

Використовуючи ці закладки, можна встановити імена послідовностей входу і цілі, а також параметрів процедури навчання. Тепер можна приступити до навчання мережі (кнопка *Train Network*).

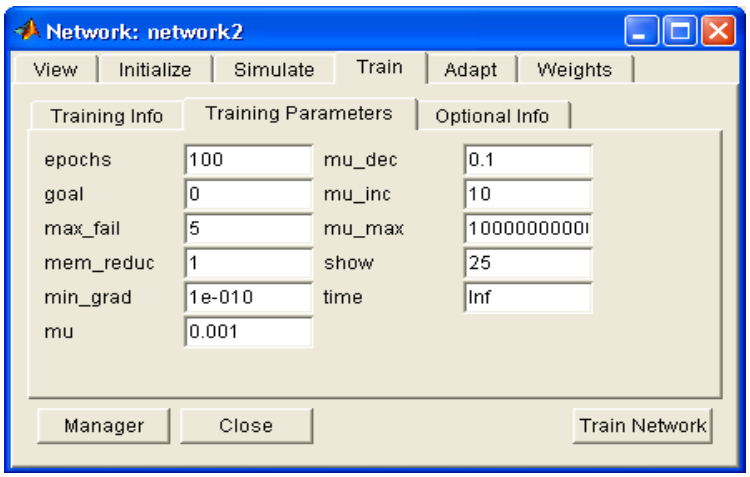


Рис. 3.10 Закладки панелі *Train* вікна *Create New Data*

Якість навчання мережі з прямою передачею сигналу на обраній навчальної послідовності пояснюється на рис. 3.12. Практично нульова точність досягається за 10 циклів навчання.

Відповідні ваги і зміщення можна побачити, якщо вибрати закладку *Weights* (рис. 3.13).

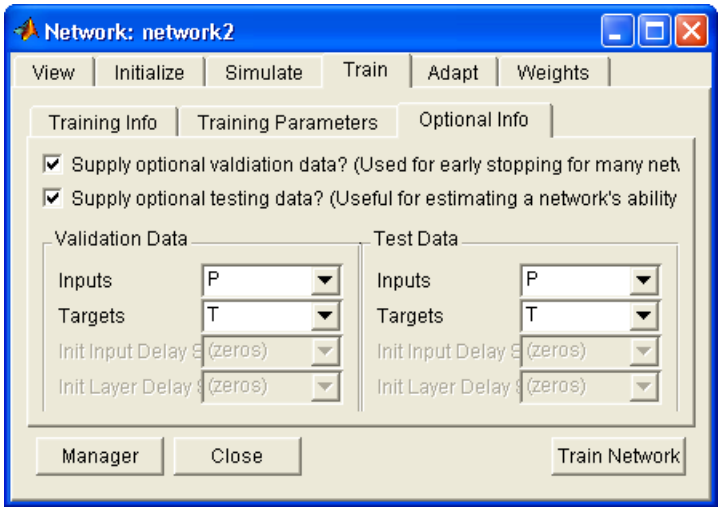


Рис. 3.11 Закладка *Optional Info* панелі *Train* вікна *Create New Data*

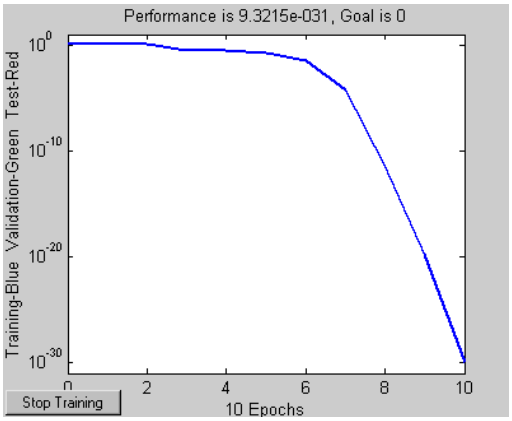


Рис. 3.12 Навчання нейронної мережі

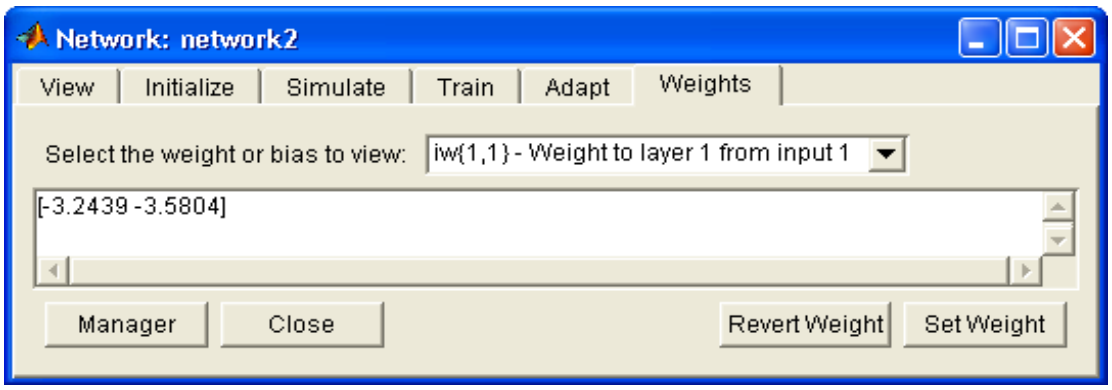


Рис. 3.13 Закладка *Weights* вікна *Create New Data*

Для зручності роботи можна експортувати створену нейронну мережу у робочу область системи MATLAB і отримати інформацію про ваги і пороги (зсуви) безпосередньо у робочому вікні системи:

```
>> network2.IW{1,1},network2.b{1}
ans =   -3.2439   -3.5804
ans =    1.7902
>> network2.LW{2,1},network2.b{2}
ans =   -1.5001
ans =    0.5000
```

Результати навчання можна переглянути у вікні *Network / Data Manager*, вибравши кнопку *Manager* (рис. 3.13). З'являється вікно (рис. 3.14); тепер, активізуючи імена послідовностей виходу або помилок *network2_outputs* і *network2_errors*, можна переглянути результати, використовуючи кнопку *View*. Отримуємо вихідні дані, що практично дорівнюють цілям, і помилки порядку 10^{-16}.

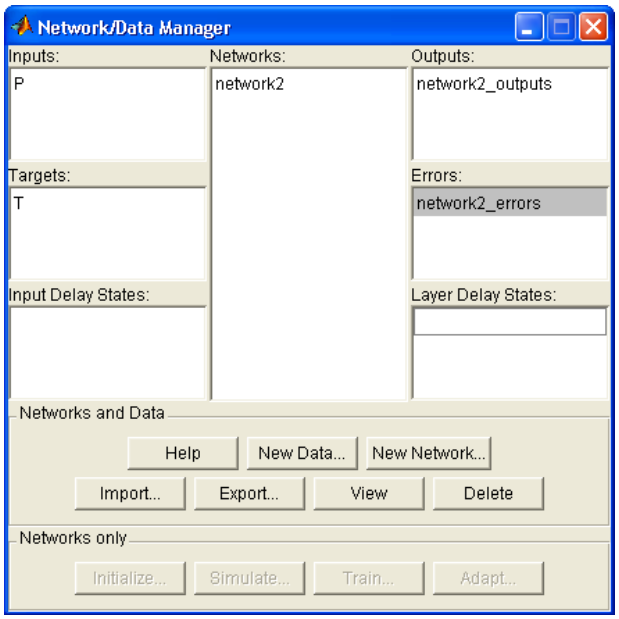


Рис. 3.14 Вікно *Network / Data Manager*

Приклад 2. Класифікація вхідних векторів

Завдання: створити нейронну мережу у вигляді персептрона, який розділяє вектори входу на два класи.

Позначимо ці класи як 0 і 1. Навчальну послідовність сформуємо у вигляді двох масивів: масиву входів *P = {[2; 2] [1; 2] [-2; 2] [-1; 1] [1; -2]}* і масиву цілей *T = {0 0 1 1 1}*, який завдає приналежність кожного вектора входу до певного класу. Виконаємо цю операцію в робочій області системи MATLAB, а потім імпортуємо їх у робочу область інтерфейсу *NNTool*.

Задамо персептрон з одним нейроном, функцією активації *HARDLIM* і правилом налаштування *LEARNP*. Для цього зі списку нейронних мереж виберемо тип мережі *Perceptron* (рис. 3.15) і вкажемо необхідні параметри.

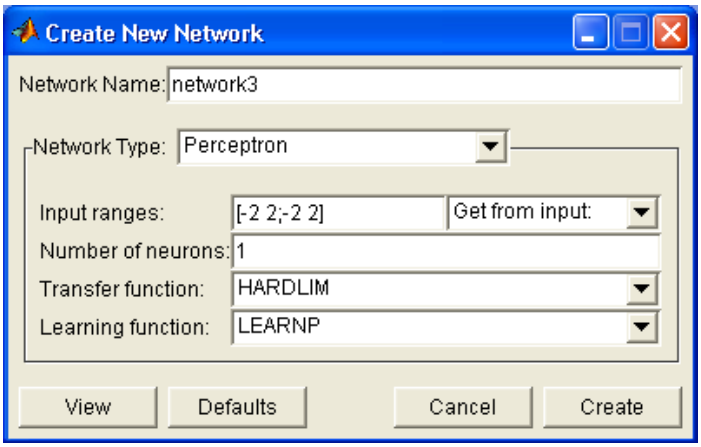


Рис. 3.15 Створення нейронної мережі у вигляді персептрона

Щоб побачити структурну схему мережі, скористаємося кнопкою *View* (рис. 3.16).

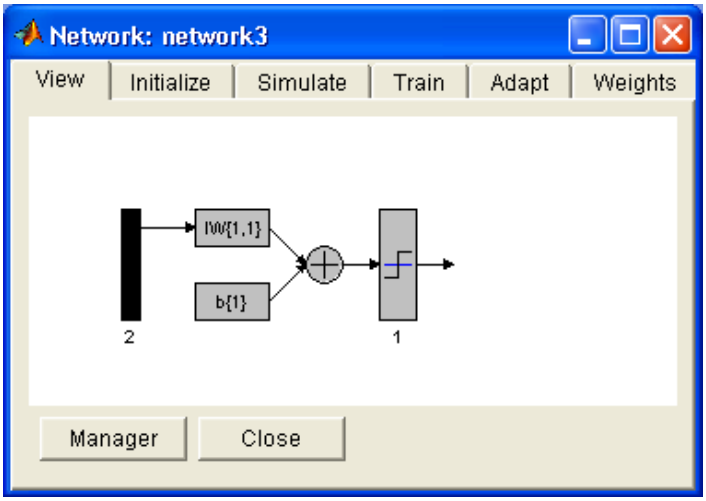


Рис. 3.16 Відображення структури нейронної мережі

Тепер виконаємо ініціалізацію мережі, використовуючи закладку *Initialize*, а потім адаптацію та налаштування параметрів мережі,

використовуючи закладку *Adapt* вікна *Network*. Для цього слід вказати імена входу і цілі, а також задати кількість циклів адаптації (у даному випадку достатньо трьох циклів) і натиснути кнопку *Adapt Network*.

В результаті налаштування будуть встановлені наступні значення ваг і зміщень, які можна побачити, вибравши закладку *Weights*. Для даної мережі вектор ваг дорівнює *IW{1, 1} = [-3 -2]*, а зсув *b{1} = 1*. Таким чином, лінія, що розділяє площину на 2 області, описується таким чином:

$$L: -3p_1 - 2p_2 + 1 = 0.$$

Перейшовши у вікно *Network / Data Manager*, можна переглянути значення сигналів на виході і помилку мережі (рис. 3.17). Неважко переконатися, що на навчальної послідовності мережа навчена точно класифікувати вхідні вектори.

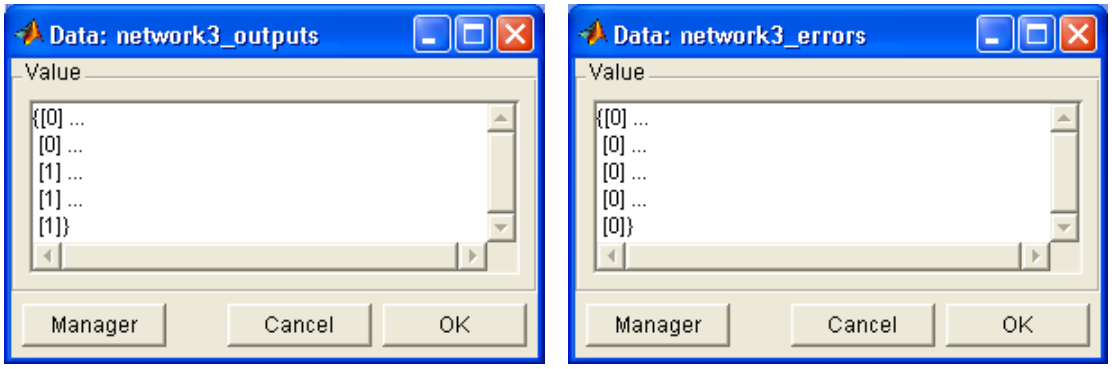


Рис. 3.17 Вікно *Network / Data Manager*

Постановка завдання

У середовищі MATLAB за допомогою GUI-інтерфейсу інструментального засобу NNTool:

1. Створити нейронну мережу, яка дозволяє виконувати класифікацію заданих векторів на 2 класи, і записати рівняння лінії, що розділяє площину на 2 відповідні області.

2. Оформити звіт з лабораторної роботи, включивши до нього вікна налаштування параметрів нейронної мережі та вихідні дані.

Варіанти

№ п/п	Вектори
1	P_1 = {[1;0] [2;1] [-1;1] [3;2] [-4;0]}; T = {1 1 0 1 0}
2	P_1 = {[0;0] [-5;1] [-1;0] [1;2] [3;1]}; T = {0 1 0 1 1}
3	P_1 = {[0;2] [-5;3] [4;1] [-3;-1] [3;-3]}; T = {0 0 0 1 1}
4	P_1 = {[-1;-1] [-2;0] [0;1] [3;-2] [0;-3]}; T = {1 0 0 1 1}
5	P_1 = {[1;1] [3;1] [2;0] [3;2] [-2;0]}; T = {0 0 1 0 1}

6	P_1 = {[2;3] [6;1] [-1;-5] [4;0] [8;0]}; T = {1 1 0 1 1}
7	P_1 = {[5;2] [4;3] [4;-1] [-3;2] [3;1]}; T = {0 1 0 1 0}
8	P_1 = {[-1;3] [2;1] [-1;1] [1;0] [4;0]}; T = {0 0 0 1 1}
9	P_1 = {[1;0] [-3;-1] [1;5] [2;0] [3;4]}; T = {1 1 1 0 0}
10	P_1 = {[1;3] [1;4] [5;-1] [-3;0] [2;1]}; T = {0 1 0 1 0}
11	P_1 = {[4;1] [2;4] [-1;3] [-2;0] [2;2]}; T = {0 1 1 1 0}
12	P_1 = {[1;2] [0;3] [-4;1] [-3;2] [3;3]}; T = {0 1 0 1 1}

Питання для підготовки до захисту лабораторної роботи

1. Наведіть послідовність етапів створення нейронної мережі.

2. Що є результатом навчання нейронної мережі?

3. Наведіть 3-5 назв нейронних мереж, доступних для роботи з інтерфейсом NNTool.

4. Відтворіть схему нейронної мережі у вигляді персептрона, який розділяє вектори входу на два класи.

5. Наведіть типову структуру нейронної мережі з прямою передачею сигналу.

Лабораторна робота 4
Розпізнавання друкованих символів у середовищі MATLAB

Об'єкт – графічні файли *tif*-формату, що містять символи алфавіту.

Предмет – методи розпізнавання друкованих символів.

Мета – закріпити знання відносно методів розпізнавання. Оволодіти методами викривлення і розпізнавання друкованих символів, а також рядків тексту.

Теоретичні положення

Набір еталонних образів міститься у графічних файлах *tif*-формату (*file_1.tif – file_34.tif*) і являє собою послідовність символів російського алфавіту. У даному прикладі число класів $K = 34$.

На малюнку нижче наведені приклади графічних символів:

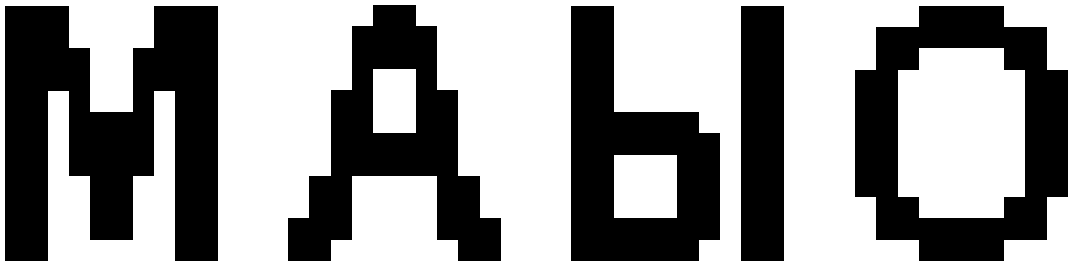


Рис. 4.1 Приклади графічних символів для вирішення задачі розпізнавання

Виконання роботи у середовищі MATLAB здійснюється за допомогою програми *MAIN.m*, код якої наведено нижче:

```
clear;
clc;
Symbols = importdata('alphabet.txt','\t'); % Зчитування символів-міток класів
for i=1:size(Symbols)
    tmp = imread(sprintf('file_%d.tif',i)); % Читання двовимірного масиву даних-
                                             % ознак з графічного файлу
    Samples(i,:) = double(reshape(not(tmp'),1,[])); % Формування одновимірного
                                % масиву даних-ознак для кожного еталону
end;

noise_and_recognize_it('file_27.TIF',0.1,Samples,Symbols) % Функція, що виконує
                                % одночасно зміну образу і розпізнавання

%recognize_it('1.tif',Samples,Symbols)%Функція, що виконує розпізнавання рядку тексту
```

Еталонний образ кожного символу представлений у вигляді вектору-стовпця *[N, 1]*, число елементів *N* якого дорівнює числу ознак (інакше кажучи, *N* – розмірність простору ознак). Такий вектор-стовпець формується з двовимірного масиву-зображення *[N1, N2]*, який, у свою чергу, формується при зчитуванні графічного файлу образу за допомогою команд:

imread (FILENAME) – процедура читання графічного файлу;

X = reshape (A, [N,1]) – процедура перетворення двовимірного масиву *A[N1,N2]* у одновимірний вектор-стовпець *X[N,1]*, де *N=N1*N2*.

noise_and_recognize_it (filename, prob, samples, symbols) – функція, що виконує зміну образу, який міститься у графічному файлі **filename**, з імовірністю зміни **prob**. В якості параметру приймає також **samples** – масив еталонних об'єктів і **symbols** – вектор міток-символів класу.

recognize_it (filename, samples, symbols) – функція, що виконує розпізнавання образу, який міститься у файлі **filename**, і виводить результат розпізнавання на екран.

noise (filename, prob) – функція, що виконує зміну образу, який міститься у файлі **filename** з імовірністю зміни **prob**. Результат роботи функції – викривлений образ – зберігається у файлі **noised_filename.tif.**

distance и **kernel** – функція знаходження відстані у багатовимірному просторі ознак і функція ядра відповідно (див. лабораторні роботи №1-2).

Постановка завдання

1. У середовищі MATLAB виконати процедури викривлення і розпізнавання друкованих символів, а також рядків тексту (див. файл *MAIN.m*).
2. Дослідити залежність якості розпізнавання від ступеня викривлення символів (параметр *prob*).
3. Оформити і захистити звіт з лабораторної роботи.

Питання для підготовки до захисту лабораторної роботи

1. Що являє собою розмірність простору ознак?
2. Розкрийте зміст параметрів функції *noise_and_recognize_it.*
3. Що представляє собою еталонний образ кожного символу?
4. Опишіть принцип роботи функцій *distance* і *kernel*.
5. Що являє собою параметр *prob* функцій *noise_and_recognize_it* та *noise*?

Лабораторна робота 5
Вивчення можливостей пакету MATLAB для вирішення задач обробки зображень

Об'єкт – растрові зображення.

Предмет – методи попередньої обробки зображень пакету MATLAB.

Мета – ознайомитися з можливостями пакету MATLAB щодо читання/запису растрових зображень, отримання інформації про зображення, візуалізації та виконання найпростіших арифметичних операцій над зображеннями.

Теоретичні положення

1. Читання / запис зображень

Для завантаження (читання) зображення у робочий простір MATLAB використовується функція *imread* з наступним синтаксисом:

imread ('filename'),

де *filename* – рядок символів, що утворюють повне ім'я файлу зображення, яке завантажується (включаючи будь-яке розширення). Якщо у ім'я файлу зображення не включена інформація про шлях до даного файлу, то файл *filename* шукається у робочій папці. А якщо його там немає, проводиться пошук даного файлу у всіх папках, шляхи до яких вказані у шляху пошуку MATLAB. Найпростіший спосіб прочитати зображення з деякої конкретної папки – це включити повний або відносний шлях до цієї папки у рядку *filename*. Наприклад, команда

f = imread ('. \ myimage \ chestxray.jpg').

завантажує зображення з підпапки *myimage* поточної робочої папки. Поточна робоча папка MATLAB відображається у рядку інструментів робочого столу Зображення записуються на диск функцією *imwrite*, яка має наступний синтаксис:

imwrite (f, 'filename').

При такому записі рядок *filename* повинен містити розширення, яке підтримується системою MATLAB (див. табл. 5.1). Інакше бажаний формат можна задати явно за допомогою третього аргументу функції. Наприклад, наступна команда записує матрицю *f* в зображення з форматом *TIFF* з ім'ям *my_image*:

>> imwrite(f, ' my_image ', 'tif')

що еквівалентно

>> imwrite(f, ' my_image.tif')

Якщо рядок *filename* не містить інформації про шлях, то функція *imwrite* записує файл у поточну робочу папку.

Більш загальний синтаксис функції *imwrite*, що застосовується лише до файлів у форматі *JPEG*, має наступний вигляд:

imwrite (f, 'filename.jpg', 'quality', q),

де *q* – ціле число в інтервалі від 0 до 100 (чим менше це число, тим вище ступінь викривлення при стисненні файлу у форматі JPEG).

Таблиця 5.1

Деякі графічні формати, які розпізнаються командами *imread* і *imwrite*

Формат зображення	**Розшифровка скорочення**	**Допустимі зображення**
TIFF	Tagged Image File Format	.tif, .tiff
JPEG	Joint Photographic Experts Group	. jpg, .jpeg
GIF	Graphics Interchange Format	.gif
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

2. Отримання інформації про зображення

Щоб дізнатися ступінь досягнутої компресії (стиснення) і отримати інформацію про зображення, можна використовувати функцію *imfinfo*, яка має синтаксис:

imfinfo filename,

або

a = imfinfo(filename).

Функція *size(f)* повертає розмір зображення:

[M, N] = size(f)

При такому записі зміній *M* буде присвоєно число рядків зображення, а зміній *N* – число стовпців.

Функція *whos* повертає додаткову інформацію про зображення. Наприклад, рядок

whos f

дає наступний результат:

Name	*Size*	*Bytes*	*Class*
f	*1024x1024*	*1048576*	*uint8 array*

Grand total is 1048576 elements using 1048576 bytes

3. Візуалізація зображень

Зображення можна вивести на дисплей комп'ютера за допомогою функції *imshow*, яка має наступний синтаксис:

imshow (f, G)

де *f* – матриця зображення, *G* – це число рівнів яскравості, що використовується при відображенні цього зображення. Якщо аргумент *G* опущений, то за замовчуванням приймається 256 рівнів яскравості. Команда

imshow (f, [low high])

означає, що всі пікселі зі значеннями, які не перевищують число *low*, треба відображати чорним кольором, а всі пікселі зі значеннями, не меншими від числа *high* – білим. Всі проміжні значення відображаються проміжною яскравістю з використанням числа рівнів яскравості, прийнятими за замовчуванням. Запис у командному рядку

imshow (f, [])

задає для змінної *low* мінімальне значення масиву *f*, а змінній *high* присвоюється його максимальне значення. Така форма функції *imshow* буває корисною при відображенні зображень, які мають вузький динамічний діапазон значень пікселів, або коли серед них є додатні і від'ємні значення. Функція *pixval* часто використовується для інтерактивного визначення значень яскравості окремих пікселів. Ця функція відображає курсор, розташований поверх зображення. Курсор переміщується по зображенню разом з мишею, а під вікном зображення відображаються поточні координати курсору і значення інтенсивності в поточній точці. При роботі з кольоровим зображенням разом з координатами відображається інтенсивність (яскравість) червоної, зеленої і синьої компоненти кольорового пікселя. При натисканні й утриманні лівої клавіші миші функція *pixval* показує евклідову відстань від початкового до поточного положення курсору. Натискання кнопки *X* у вікні курсору відключає курсор на зображенні.

4. Класи даних і типи зображень

Пакет IPT підтримує наступні типи зображень (табл. 5.2):

- напівтонові зображення (зображення у градаціях сірого кольору);
- бінарні (двійкові) зображення;
- індексовані зображення;
- кольорові зображення RGB.

Таблиця 5.2

Класи даних

Назва	Опис
Числові	
double	Дійсні числа з плаваючою комою подвійної точності в діапазоні приблизно від -10308 до 10308 (8 байт на число).
uint8	Цілі без знака в інтервалі [0,255] (1 байт на число).
uintl6	Цілі без знака в інтервалі [0,65535] (2 байта на число).
uint32	Цілі без знака в інтервалі [0,4294967295] (4 байта на число).
int8	Цілі без знака в інтервалі [-128,127] (1 байт на число).

Продовження таблиці 5.2

intl6	Цілі без знака в інтервалі [-32768, 32767] (2 байта на число).
int32	Цілі без знака в інтервалі [-2147483648, 2147483647] (4 байта).
single	Дійсні числа з плаваючою комою в діапазоні приблизно від -1038 до 1038 (4 байти на число).
Символьні	
char	Символи (літери та знаки) (2 байти на символ).
Логічні	
logical	Значення 0 або 1 (1 байт на елемент).

Напівтонове зображення – це матриця, елементи яскравості якої представлені у вигляді числових значень. Якщо елементи напівтонового зображення належать класу *uint8* або *uintl6*, вони представлені цілими числами, відповідно, в інтервалах [0,255] і [0,65535]. Якщо зображення належить класу *double*, його пікселі є дійсними числами з плаваючою комою. Значення пікселів зображень типу *double* повинні знаходитися в інтервалі [0,1].

Двійкові зображення мають в MATLAB досить специфічний сенс. Вони є логічними масивами, що складаються з 0 і 1. Тому масив елементів 0 і 1 інших класів, наприклад, *uint8*, не є двійковим зображенням.

Функція *logical* призначена для перетворення числових масивів в *логічні*. Якщо *A* – числовий масив, що складається з 0 і 1, то для побудови логічного масиву *B* з тими ж елементами слід виконати команду

B = logical (A).

Якщо масив А містить також інші елементи, відмінні від 0 і 1, то функція *logical* перетворює всі його ненульові одиниці в логічний елемент 1, а всі нульові – в логічний елемент 0. Щоб перевірити, чи є даний масив логічним, застосовується функція *islogical*:

islogical (C).

Якщо С – логічний масив, то ця функція повертає 1, в іншому випадку – 0.

5. Конвертація класів даних

Класи конвертуються наступним чином:

B = data_class_name(A),

де *data_class_name* – це одне з імен класів даних з першого стовпця табл. 5.2. Наприклад, нехай *A* – масив класу *uint8*. Масив подвійної точності *B* генерується командою

B = double (A).

Якщо *C* – це масив класу *double*, елементи якого знаходяться в інтервалі [0,255], але серед них можуть зустрічатися дійсні числа, то його можна перетворити в масив *uint8* командою

D = uint8 (C).

Якщо масив класу *double* містить елементи зі значеннями поза інтервалу [0, 255] і він був конвертований в клас *uint8* описаним вище способом, MATLAB перетворює всі від'ємні значення (менші за 0) в 0, значення, більші за 255 – в 255, а для всіх інших елементів відкидаються дробні частини. Це

означає, що перед перетворенням масивів *double* в *uint8* необхідно здійснити відповідне перенормування (перемасштабування) його елементів.

6. Конвертація класів і типів зображень

У пакеті є спеціальні функції, які здійснюють перенормування (перемасштабування) при конвертації одних класів і типів зображень в інші.

Таблиця 5.3

Функції для конвертації класів і типів зображень

Початковий тип	Вихідний тип	Перетворюваний клас
im2uint8	uint8	logical,uint8,uintl6,double
im2uintl6	uintl6	logical,uint8,uintl6,double
mat2gray	double (в область [0,1])	double
im2double	double	logical,uint8,uint16,double
im2bw	logical	uint8,uint16,double

Функція *im2uint8* спочатку розпізнає клас даних на вході і здійснює всі необхідні перетворення. Функція обнуляє всі від'ємні значення вхідного зображення, присвоює значення 255 величинам, більшим за 1, і помножує інші значення на 255, після чого округлює результат до найближчого цілого числа.

Перетворення довільних масивів типу *double* в перенормовані масиви (зображення) *double* зі значеннями в інтервалі [0,1] здійснюється за допомогою функції *mat2gray*, що має наступний синтаксис:

g = mat2gray (A, [Amin, Amax]),

де зображення *g* має значення пікселів в інтервалі від 0 (білий) до 1 (чорний). Зазначені параметри *Amin* і *Amax* мають наступну дію: всі елементи, менші за *Amin*, обнуляються, а всі елементи, більші за *Amax*, замінюються на 1. При виконанні команди

g = mat2gray (A)

значення *Amin* і *Amax* – це фактичні максимум і мінімум масиву *A*. Вхідний масив має бути класу *double*. Вихідний масив також відноситься до класу *double*.

Функція *im2double* перетворює вхідний масив в клас *double*. Якщо вхідний масив був класу *logical*, *uint8* або *uintl6*, то функція *im2double* перетворює його в клас *double* у діапазоні [0,1]. Якщо вхідний масив був класу *double*, то функція *im2double* залишає його без змін. Для перетворення масивів *double* в масиви *double* зі значеннями в інтервалі [0,1] застосовується функція *mat2gray*.

Функція *im2bw* має наступний синтаксис

g = im2bw (f, T)

і породжує двійкове зображення *g* з напівтонового зображення *f*, використовуючи поріг *T*. Значення всіх елементів *f*, менших за *T*, стають логічним 0, а всіх інших – логічними 1. Значення порогу *T* повинно знаходитися в інтервалі [0,1] незалежно від класу вхідного зображення. Вихідний масив автоматично перетворюється на логічний. Якщо написати

g = im2bw (f),

то в IPT буде використовуватися за замовчуванням поріг *T = 0,5*. Якщо вхідне зображення було класу *uint8*, то *im2bw* спочатку ділить його елементи на 255, а потім застосовує заданий поріг або поріг, прийнятий за замовчуванням. Якщо вхідний масив був класу *uintl6*, то значення діляться на 65535. Якщо вхідне зображення належало до класу *double*, то *im2bw* відразу застосовує відповідний поріг. Якщо вхідний масив був логічним, то вихідний масив буде йому ідентичний. Логічний (двійковий) масив можна перетворити на числовий за допомогою будь-якої з перших чотирьох функцій табл. 5.3.

7. Арифметичні операції над зображеннями

У пакеті IPT є кілька арифметичних функцій для роботи із зображеннями (табл. 5.5). Подібні дії можна виконати, безпосередньо застосовуючи базові арифметичні оператори MATLAB (табл. 5.4). Однак перевага використання функцій IPT полягає в тому, що вони враховують клас вхідного зображення, а відповідні математичні оператори MATLAB вимагають використання класу *double*.

Таблиця 5.4

Операції з матрицями і масивами

Оператор	**Назва**	**Функція MATLAB**	**Коментарі і приклади**
+	Додавання матриць і масивів	*plus(A,B)*	a+b, A+B, або a+A
-	Віднімання матриць і масивів	*minus (A,B)*	a-b, A-B, a-A або A-a
.*	Множення масивів	*times(A,B)*	C=A.*B, C(I,J)=A(I,J)*B(I,J).
*	Множення матриць	*mtimes(A,B)*	A*B, стандартний добуток матриць, або a*A, множення елементів матриці A на скаляр
./	Ділення на масив справа	*rdivide(A.B)*	C=A./B,C(I,J)=A(I,J)/B(I,J)
.\	Ділення на масив зліва	*ldivide(A,B)*	C=A.\B,C(I,J)=B(I,J)/A(I,J)
/	Матричне ділення справа	*mrdivide(A,B)*	A/B, майже та ж дія, що і A*inv(B), залежно від точності обчислень
\	Матричне ділення зліва	*lrdivide(A,B)*	A\B, майже та ж дія, що ф inv(A)*B, залежно від точності обчислень
.^	Піднесення до степеня масивів	*power(A,B)*	Якщо C=A.^B, то C(I,J)=A(I,J)^B(I,J)
^	Матричне піднесення до ступеня	*mpower(A.B)*	
.’	Векторне або матричне транспонування	*transpose(A)*	A.’. Стандартне транспонування векторів або матриць
’	Векторне або матричне комплексне сполучення й транспонування	*ctranspose(A)*	A’. Стандартне комплексне сполучення векторів або матриць
+	Унарний плюс	*uplus(A)*	+A еквівалентно 0+A
-	Унарний мінус	*uminus(A)*	-A еквівалентно 0-A або -1*A
:	Двокрапка		A=B(1:10,1:10), завдання діапазону рядків і стовпців

Таблиця 5.5

Арифметичні функції обробки зображень пакету IPT

Функція	Опис
imadd	Складання двох зображень або додавання константи
imsubtract	Віднімання двох зображень чи віднімання константи
immultiply	Множення двох зображень, яке здійснюється поелементно між відповідними парами елементів зображень, або множення всього зображення на константу
imdivide	Ділення двох зображень, яке здійснюється поелементно між відповідними парами елементів зображень, або ділення всього зображення на константу
imabsdiff	Обчислення модуля різниці двох зображень
imlincomb	Знаходження лінійної комбінації двох або більше зображень

Вихідні дані:

Зображення текстур ' Z:\Subjects\POO3\Textures*.bmp'

Номер комп'ютера	Зображення
1	1.1.01.bmp – 1.1.05.bmp
2	1.1.06.bmp – 1.1.10.bmp
3	1.1.11.bmp – 1.2.02.bmp
4	1.2.03.bmp – 1.2.07.bmp
5	1.2.08.bmp – 1.2.12.bmp
6	1.2.13.bmp – 1.3.04.bmp
7	1.3.05.bmp – 1.3.09.bmp
8	1.3.10.bmp – 1.4.01.bmp
9	1.4.02.bmp – 1.4.06.bmp
10	1.4.07.bmp – 1.4.11.bmp
11	1.4.12.bmp – 1.5.04.bmp
12	1.5.05.bmp – brick.030.bmp

Постановка завдання

1. Відкрити зображення, зберегти його без стиснення (1.jpg) і з деяким ступенем стиснення (2.jpg). Використовуючи результати виконання функції *imfinfo*, визначити розмір початкового зображення (1.jpg) і обчислити його ступінь стиснення як відношення вихідного розміру до отриманого (2.jpg).

2. Написати функцію, яка розраховує суму двох вхідних зображення і повертає зображення-суму, максимальний і мінімальний елементи цієї суми і їх індекси, а також відповідне нормоване зображення зі значеннями в інтервалі [0,1]. Приклад пошуку максимального елемента зображення:

[a, s] = max (cat (1, max (b '), max (b))').

3. Написати функцію, яка вирізає підзображення заданого користувачем розміру із зображення 1.jpg.

Питання для підготовки до захисту лабораторної роботи

1. Охарактеризуйте типи зображень, які підтримує пакет MATLAB.

2. Як отримати інформацію про зображення у пакеті MATLAB?

3. Розкрийте зміст поняття «напівтонове зображення».

4. Яку дію виконує функція *mat2gray*?

5. Наведіть функції MATLAB для виконання арифметичних операцій над зображеннями.

Лабораторна робота 6
Просторова обробка растрових зображень

Об'єкт – растрові зображення.

Предмет – методи просторової обробки зображень пакету MATLAB.

Мета – ознайомитися з можливостями пакету MATLAB щодо перетворення яскравості зображень, побудови гістограм та виконання просторової фільтрації.

Теоретичні положення

1. Перетворення яскравості зображень

Функція *imadjust* є базовим інструментом пакета IPT при перетвореннях яскравості півтонових зображень. Вона має наступний синтаксис:

g = imadjust (f, [low_in, high_in], [low_out, high_out], gamma),

де *f* – вихідне зображення, *g* – нове зображення.

Значення яскравості в інтервалі *[low_in, high_in]* перетворюється в значення з інтервалу *[low_out, high_out]*, значення, менші за поріг *low_in*, перетворюються в *low_in*. Всі значення, більші за поріг *high_in*, перетворюються в *high_in*. Вхідне зображення може мати клас *uint8*, *uintl6* або *double*, а клас вихідного зображення збігається з класом вхідного. Всі вхідні параметри функції *imadjust*, за винятком *f*, повинні бути дійсними числами в інтервалі від 0 до 1, незалежно від класу *f*. Якщо *f* належить до класу *uint8*, функція *imadjust* помножує ці параметри на 255; якщо *f* – зображення класу *uintl6*, то всі значення множаться на 65535. Якщо замість векторів *[low_in, high_in]* або *[low_out, high_out]* поставити порожній вектор *([])*, то будуть використовуватися величини за замовчуванням, рівні [0 1]. Якщо *high_out* менше, ніж *low_out*, то вихідні яскравості симетрично перевертаються.

Параметр *gamma* служить для завдання форми кривої, що відображає яскравість *f* в яскравість *g*. Якщо *gamma* менше 1, то яскравість відображення зміщується вгору в бік більш яскравих значень. Якщо *gamma* більша за 1, то яскравість відображення зміщується вниз у бік менш яскравих значень. Якщо параметр *gamma* опущений, то його значення за замовчуванням дорівнює 1 (лінійне відображення).

Основне застосування логарифмічного перетворення полягає в стисненні динамічного діапазону. Наприклад, спектр Фур'є часто має діапазон величин від 0 до 10^e і навіть вище. Якщо лінійно масштабувати цей діапазон в інтервал з 8-ми бітною градацією, то при візуалізації найбільш яскраві пікселі будуть домінувати, що призведе до втрати деталей менш яскравих ділянок спектра.

При виконанні логарифмічних перетворень часто буває необхідно повертати стислий діапазон назад до початкового значення. Для 8 біт таку дію в MATLAB можна здійснити командою

g = im2uint8 (mat2gray (f));

Застосування *mat2gray* переводить величини в діапазон [0 ... 1], а функція *im2uint8* перетворює їх до діапазону [0 ... 255].

Використання логарифмічного перетворення для скорочення динамічного діапазону може здійснюватися із застосуванням наступної послідовності команд (у випадку, якщо вихідне зображення представлене в форматі *uint8*):

g = im2uint8 (mat2gray (log (1 + double (f)))),

або, наприклад

*g = log (f * 255 + 1),*

якщо значення елементів вихідного зображення f – дійсні числа в діапазоні від 0 до 1.

Функція $g = T(f) = \frac{1}{1+(m/f)^E}$ (де f – це яскравість вхідного зображення, g – відповідна яскравість вихідного зображення) називається функцією перетворення розтягування контрастності, оскільки вона стискає вхідні значення, менші за m, у більш вузький піддіапазон темних рівнів на вихідному зображенні, і, відповідно, величини, більші за m – в вужчу смугу яскравих рівнів. Результатом є зображення з більшою контрастністю.

2. Обробка гістограм і побудова графіків функцій

Функція *imhist* пакету IPT побудови гістограм має наступний синтаксис:

h = imhist (f, b),

де *f* – вхідне зображення, *h* – його гістограма і *b* – число інтервалів діапазону яскравості, використаних при формуванні гістограми (якщо аргумент *b* відсутній, то за замовчуванням приймається b = 256). Для того, щоб отримати нормовану гістограму, необхідно виконати дію:

p = imhist (f, b) / numel (f).

Функція *numel(f)* повертає число елементів масиву *f* – кількість пікселів зображення.

Гістограму можна побудувати за допомогою стовпчастих діаграм. Для цього слугує функція:

bar (horz, v, width),

де *v* – це вектор-рядок, графік якого треба побудувати, *horz* – вектор того ж розміру, що й вектор *v*, який складається з прирощень за горизонтальною шкалою, і *width* – число між 0 і 1.

Якщо параметр *horz* опущений, то горизонтальна вісь ділиться з кроком 1 від 0 до *length(v)*. Коли *width* дорівнює 1, сусідні стовпчики стикаються; коли *width* дорівнює 0, стовпчики є вертикальними лініями. За замовчуванням *width* дорівнює 0.8.

Наприклад:

```
f = randn(1000);
imshow(f)
h = imhist(f) ;
hi = h(1:10:256);
horz = 1:10:256;
```

```
bar(horz, hi);
axis([0 255 0 15000]);
set(gca, 'xtick', 0:50:255)
set(gca, 'ytick', 0:2000:15000)
```

Функція *axis* використана з метою розширення області вертикальної і горизонтальної осі для візуального аналізу. Функція *axis* має синтаксис:

axis ([horzmin horzmax vertmin vertmax]),

де встановлюються мінімальні і максимальні значення горизонтальних і вертикальних координат. Параметр *gca* означає *«get current axis»* (використовувати поточні осі, тобто осі раніше побудованого графіка), а *xtick* і *ytick* встановлюють мітки на горизонтальній та вертикальній осях через задані інтервали.

Біля горизонтальної та вертикальної осі графіка можна помістити пояснювальні написи за допомогою функцій

xlabel ('text string', 'fontsize', size),
ylabel ('text string', 'fontsize', size),

де *size* – це розмір шрифту.

Текст можна також розмістити прямо на графіку за допомогою функції *text* наступним чином:

text (xloc, yloc, 'text string', 'fontsize', size),

де *xloc* і *yloc* – координати початку тексту.

Графік можна забезпечити заголовком за допомогою функції *title*:

title ('titlestring'),

де *titlestring* – це рядок символів заголовка, яка буде розміщена по центру над графіком.

Стеблова діаграма будується аналогічно до стовпчастої діаграми. Її синтаксис має вигляд

stem (horz v, 'color_linestyle_marker', 'fill'),

де *v* – це вектор-стовпець, графік якого необхідно побудувати, а змінна *horz* має той же зміст, що й у функції *bar*. Аргумент *color_linestyle_marker* складається з комбінації символів, можливі значення яких наведені в табл. 6.1.

Наприклад, команда *stem(v, 'r-s')* будує стеблову діаграму, на якій вертикальні стебла будуть проведені червоними пунктирними лініями, а вершини стебел будуть квадратними. Якщо присутня змінна *fill*, а змінна *marker* позначає коло, квадрат або ромб, то ці символи всередині закрашуються в колір, привласнений змінній *color*. За замовчуванням колір вважається чорним, лінії стебел суцільними, а вершинами слугують кола.

Таблиця 6.1

Атрибути функцій *stem* і *plot*. Атрибут *none* застосовується тільки до функції *plot*

Символ	**Колір**	**Символ**	**Стебло**	**Символ**	**Вершина**
k	чорний	-	суцільне	+	плюс
w	білий	--	пунктирне	o	круг
r	червоний	:	точкове	*	зірка
g	зелений	-.	штрихпунктирне	.	точка
b	синій	none	відсутнє	x	хрест
c	блакитний			s	квадрат
Y	жовтий			d	ромб
m	пурпурний			none	відсутня

Стеблевая діаграма може бути отримана за допомогою команд

```
h = imhist(f) ;    horz = 1:10:256;       axis([0 255 0 15000]);
hi = h(1:10:256); stem(horz, hi, 'fill') set(gca, 'xtick', 0:50:255)
                                          set(gca, 'ytick', 0:2000:15000)
```

Функція *plot* будує графік по точках, з'єднуючи їх відрізками прямих ліній. Вона має синтаксис:

plot (horz, v, 'color_linestyle_marker'),

де аргументи мають той же сенс, що і у функції *stem*. Допустимі значення атрибутів *color*, *linestyle* і *marker* наведені в табл.6.1.

3. Лінійна просторова фільтрація

Лінійна просторова фільтрація реалізована в пакеті IPT функцією *imfilter*, що має наступний синтаксис:

g = imfilter (f, w, filtering_mode, boundary_options, size_options),

де *f* – вхідне зображення; *w* – фільтруюча маска; *g* – результат фільтрації. Параметр *filtering_mode* визначає, що здійснюється фільтрація, кореляція *('corr')* або згортка *('conv')*. Опція *boundary_options* відповідає за розширення границь, розміри яких визначається розмірами фільтра. Параметр *size_options* визначає розмір вихідної зображення.

Таблиця 6.2

Опції функції *imfilter*

Опції	**Опис**
Режим фільтрації	
‘corr’	Фільтрація проводиться методом кореляції
‘conv’	Фільтрація проводиться методом згортки
Граничні опції	
P	Границі зображення розширюються відповідно до значення P. За замовчуванням P = 0
‘replicate’	Розмір зображення збільшується повторенням величин на його бічних границях
‘symmetric’	Розмір зображення збільшується шляхом дзеркального відображення через границі
‘circular’	Розмір зображення збільшується періодичним повторенням двовимірної функції
Опції розміру	
‘full’	Вихідне зображення має ті ж розміри, що і розширене вхідне зображення
‘same’	Вихідне зображення має ті ж розміри, що і вхідне зображення. Це досягається за допомогою обмеження переміщення центру фільтрує маски точками, які належать до вихідного зображення. Опція за замовчуванням.

Наприклад:

g = imfilter (f, w, 'replicate');

4. Завдання маски заданого типу

Функція *h = fspecial (type, P1, P2)* повертає маску *h* заданого двовимірного лінійного фільтра, що задається рядком *type*. Маска *h* призначена для передачі у функції, які 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('gaussian', n, sigma)* повертає маску *h* фільтра нижніх частот Гаусса. Розмір маски визначає параметр *n*. Якщо *n* – двокомпонентний вектор, то розмір маски *n(1)×n(2)*. Якщо n скаляр, то розмір маски *n×n*. Параметр *sigma* задає середньоквадратичне відхилення розподілу Гаусса, яке використовується при формуванні маски *h*. Якщо під час виклику функції параметри *n* і *sigma* опущені, то розмір маски встановлюється рівним 3×3, а середньоквадратичне відхилення 0,5.

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

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

Для виділення вертикальних границь достатньо транспонувати дану маску *h*.

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

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

Для виділення вертикальних границь достатньо транспонувати дану маску *h*.

Функція *h = fspecial('laplacian', a)* повертає маску *h* фільтра Лапласа. Розмір маски 3×3. Параметр *a* управляє співвідношенням між центральним і граничними елементами маски. Даний параметр повинен встановлюватися в діапазоні [0, 1]. За замовчуванням a = 0,2.

Функція *h = fspecial('log', n, sigma)* повертає маску *h* фільтра, аналогічного до послідовного застосування фільтрів Гауса і Лапласа, так званого лапласіан-гауссіана. Розмір маски визначає параметр *n*. Якщо *n* – двокомпонентний вектор, то розмір маски *n(1)×n(2)*. Якщо *n* – скаляр, то розмір маски *n×n*. Параметр *sigma* задає середньоквадратичне відхилення Гаусса, яке використовується при формуванні маски *h*. Якщо під час виклику функції параметри *n* і *sigma* опущені, то розмір маски встановлюється рівним 5×5, а середньоквадратичне відхилення 0,5.

Функція *h = fspecial('average', n)* повертає маску *h* усереднюючого фільтру. Розмір маски визначає параметр *n*. Якщо *n* – двокомпонентний вектор, то розмір маски *n(1)×n(2)*. Якщо *n* – скаляр, то розмір маски *n×n*. Якщо під час виклику функції параметр *n* опущений, то розмір маски встановлюється рівним 3×3.

Функція *h = fspecial('unsharp', a)* повертає маску *h* фільтра, що підвищує різкість зображення. Розмір маски 3×3. Параметр a управляє співвідношенням

між центральним і граничними елементами маски. Даний параметр повинен встановлюватися в діапазоні [0 ... 1]. За замовчуванням a = 0,2.

Усереднюючий фільтр відноситься до фільтрів нижніх (низьких) частот. Він призначений для фільтрації високочастотного шуму і його використання супроводжується розмиванням зображення. Кожен елемент маски дорівнює *1/MN*, де *M* і *N* – розміри маски (кількість рядків і стовпців).

Фільтр Лапласа є високочастотним фільтром і призначений для виділення границь (перепадів) у всіх напрямках. Маска фільтра конструюється таким чином:

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

де α – параметр в діапазоні [0,1], що передається у функцію *fspecial*.

Лапласіан-Гауссіан також відноситься до високочастотних фільтрів, але на відміну від фільтра Лапласа, виділяє більш різкі перепади. Маска фільтра формується наступним чином:

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

де *M* і *N* – розміри маски; σ – середньоквадратичне відхилення розподілу Гауса.

Формула для обчислення h_g приведена вище. Маска фільтра, що підвищує різкість зображення, створюється наступним чином:

$$h = \frac{1}{(1+\alpha)}\begin{bmatrix} -\alpha & \alpha-1 & -\alpha \\ \alpha-1 & \alpha+5 & \alpha-1 \\ -\alpha & \alpha-1 & -\alpha \end{bmatrix}$$

де α – параметр в діапазоні [0,1], що передається у функцію *fspecial*.

Постановка завдання

1. Вихідними даними є зображення лабораторної роботи № 5. Написати m-функцію, параметрами якої є вихідне зображення і деяка константа. В залежності від значення константи виконується:

– перетворення зображення в негатив;

– логарифмічне перетворення;

– гамма-перетворення;

– розтягнення контрасту.

Додаткові параметри функцій IPT задаються користувачем в інтерактивному режимі.

2. Виконати послідовність дій:

a. Відкрити зображення "1.tif", перетворити значення елементів до типу *double* (зображення *a*);

b. Задати маску фільтра:

-1	-1	-1
-1	8	-1
-1	-1	-1

c. Виконати фільтрацію зображення *a* заданих фільтром з використанням функції *imfilter*. У параметрах функції вказати необхідність збільшення розміру вихідного зображення повторенням величин на його бічних границях (зображення *b*);

d. Обчислити суму значень елементів зображень *a* і *b* (зображення *c*);

e. Виконати фільтрацію зображення *a* маскою фільтра Собеля для виділення спершу горизонтальних, а потім вертикальних границь. Обчислити суму двох результатів фільтрації (зображення *d*);

f. Обробити зображення *d* маскою усереднюючого фільтру розміру 5x5 (зображення *e*);

g. Перемножити значень елементів зображень *c* і *e* (зображення *f*);

h. Скласти значення елементів зображень *a* і *f* (зображення *g*);

i. Виконати градаційну корекцію зображення *g* за степеневим законом (піднести значення елементів зображення до ступіня 0,5) (результуюче зображення).

Питання для підготовки до захисту лабораторної роботи

1. Охарактеризуйте параметри функції *imadjust*.
2. Опишіть фунцію *imfilter*.
3. Наведіть приклад завдання маски усереднюючого фільтра розміром 3×3.
4. Якими є коефіцієнти маски фільтра Собеля?
5. Що являє собою фільтр Лапласа?

Лабораторна робота 7
Частотна обробка растрових зображень

Об'єкт – растрові зображення.

Предмет – методи частотної обробки зображень пакету MATLAB.

Мета – ознайомитися з можливостями пакету MATLAB щодо виконання прямого і зворотного дискретного перетворення Фур'є растрових зображень.

Теоретичні положення

Пряме і зворотне дискретне перетворення Фур'є (ДПФ, DFT Discrete Fourier Transform) на практиці обчислюються за допомогою алгоритму *швидкого перетворення Фур'є (FFT, Fast Fourier Transform)*.

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.

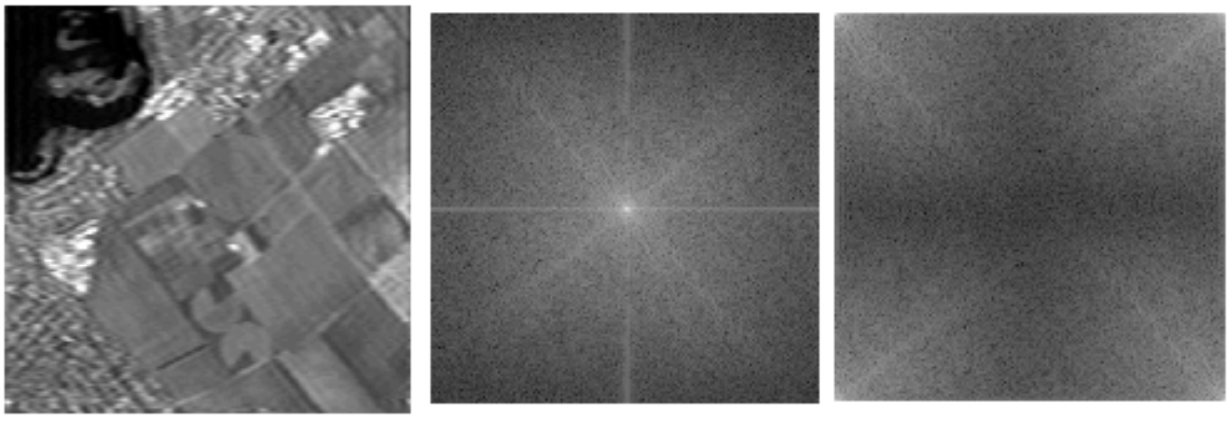
Алгоритм FFT масиву *f* зображення *M*×*N* реалізований в пакеті IPT функцією *fft2*, яка має наступний синтаксис:

F=fft2(f).

Ця функція повертає перетворення Фур'є розміром *M*×*N*, причому початок даних знаходиться у верхньому лівому кутку, і має чотири чверть-періоду (квадранта), які примикають до центральної точки частотної області (рис. 7.1).

Функцію *fftshift* пакету IPT можна використовувати для зміщення початку координат перетворення в центр частотної області. Її синтаксис має вигляд:

Fc=fftshift(F).



а) б) в)

Рис. 7.1 Перетворення Фур'є: а) вихідне зображення; б) результат застосування до зображення *а* функції *fft2*; в) результат застосування до зображення *б* функції *fftshift*

Функція *fftshift* переставляє квадранти F наступним чином. Наприклад, якщо *a = [1 2; 3 4]*, то *fftshift(a) = [4 3; 2 1]*. Якщо застосовувати цю дію після обчислення перетворення Фур'є, то результат буде в точності збігатися з множенням зображення на $(-1)^{x+y}$ перед виконанням перетворення. Ці два

процеси не можна переставити, тобто, якщо позначити оператор перетворення Фур'є через $g[\cdot]$, який застосовується до аргументу, то вираз $g[(-1)^{x+y} f(x, y)]$ еквівалентний обчисленню *fftshift(fft2(f))*, але дія *fft2(fftshift(f))* призведе до зовсім іншого результату.

Функція *ifftshift* обертає центрування спектра. Її синтаксис має вигляд

F = ifftshift(Fc)

Цю функцію можна використовувати для приведення спектра до його початкового вигляду, тобто коли центр спектра розташований у верхньому лівому кутку прямокутника.

Центр частотного прямокутника знаходиться в точці (*M/2, N/2*) у разі, якщо координати *u* і *v* знаходяться в інтервалі від *0* до *M-1* і до *N-1* відповідно. Наприклад, центр частотного квадрата *8×8* знаходиться у точці *(4,4)*, тобто в п'ятій точці по кожній осі, якщо рахувати від точки *(0,0)*. Якщо (в системі MATLAB) ці значення змінюються від *1* до *M* і від *1* до *N* відповідно, то центр квадрата має координати *[(M/2)+1, (N/2)+1]*. У прикладі з квадратом *8×8* центр знаходиться в точці *(5,5)*, якщо рахувати від *(1,1)*.

Якщо *M* і *N* – непарні числа, то центр при обчисленнях в MATLAB визначається округленням *M/2* і *N/2* у бік зменшення до найближчого цілого. Наприклад, центр області *7×7* знаходиться в точці *(3,3)*, якщо рахувати від *(0,0)*, і він має координати *(4,4)*, якщо початок знаходиться у *(1,1)*. В обох випадках центр розташований в четвертій точці від початку відліку. Якщо тільки одна з розмірностей непарна, то координати центру за цим напрямком обчислюються округленням. Використовуючи функцію MATLAB *floor*, центр частотного прямокутника обчислюється за формулою:

[floor(M/2)+1, floor(N/2)+1],

яка дає правильний результат і для парних, і для непарних величин *M* і *N*.

Амплітудно-частотна характеристика (АЧХ) – функція, що показує залежність модуля деякої комплекснозначної функції від частоти. *Частота* – характеристика періодичного процесу, рівна числу повних циклів, здійснених за одиницю часу.

Для отримання спектру Фур'є (амплітудно-частотної характеристики сигналу) слід застосувати команду:

S=abs(F),

яка обчислює абсолютну величину (корінь квадратний із суми квадратів реальною і уявною частин) кожного елемента масиву.

Для масиву дійсних чисел *X* функція *Y=abs(X)* повертає масив *Y* абсолютних значень елементів *X*. Для масиву комплексних чисел *Z* функція *Y=abs(Z)* повертає масив *Y* модулів комплексних елементів *Z*.

Наприклад:

```
abs(-5)
ans=5
abs(3+4i)
ans=5
```

Фаза коливань – аргумент періодичної функції, наприклад, $\cos(\omega t + \varphi_0), \sin(\omega t + \varphi_0)$ (ω – кутова частота, t – час, φ_0 – початкова фаза коливань, тобто фаза коливань в початковий момент часу $t = 0$). Фаза зазвичай виражається в кутових одиницях (радіанах) або частках періоду.

Функція *P=angle(Z)* обчислює фазу комплексного числа (фазо-частотної характеристики сигналу). Для комплексного числа $z = x + iy = re^{ij}$ його модуль r і аргумент j обчислюються наступним чином:

r=abs(z)

j=angle(z)

Зворотне перетворення Фур'є (ЗДПФ) обчислюється за допомогою функції *ifft2*, синтаксис якої має вигляд:

f=ifft2(F).

Якщо вихідне зображення, використане при обчисленні *F*, було речовим, то і зворотне перетворення повинно бути речовим. Однак на практиці вихід *ifft2* часто має дуже малу уявну компоненту, яка обумовлена помилками округлення при виконанні арифметичних дій з плаваючою комою. Тому, як правило, необхідно виділити дійсну частину після виконання зворотного перетворення Фур'є:

f=real(ifft2((F)).

Алгоритми частотної обробки растрових зображень використовуються, зокрема, в задачах фільтрації сигналів і видалення шуму.

Приклад:

1. Завантаження зображення, виділення зрізу (рядка – сигналу), побудова графіка зрізу:

```
f = im2double(imread('selo_gray.bmp'));
l = img(100, :);
figure; plot(l); grid;
```

2. Фільтрація гаусівського шуму

```
lGauss = imnoise(l, 'Gaussian');        Додавання гаусівського шуму
figure; plot(lGauss); grid;             Побудова графіка зашумленного зрізу
lGaussF = fft(lGauss);                  ДПФ
lGaussF(50:206) = 0;                    Видалення високих частот
lGaussFl = real(ifft(lGaussF));         ЗДПФ
figure; plot(lGaussFl); grid;           Побудова графіка відфільтрованого сигналу
lGaussF = fft(lGauss);                  ДПФ
lGaussF(100:156) = 0;                   Видалення високих частот
lGaussFl = real(ifft(lGaussF));         ЗДПФ
figure; plot(lGaussFl); grid;           Побудова графіка відфільтрованого сигналу
```

3. Фільтрація мультиплікативного шуму

```
lSpeckle = imnoise(l, 'speckle');          Додавання мультиплікативного шуму
figure; plot(lSpeckle); grid;              Побудова графіка зашумленного зрізу
lSpeckleF = fft(lSpeckle);                 ДПФ
lSpeckleF(50:206) = 0;                     Видалення високих частот
lSpeckleFl = real(ifft(lSpeckleF));        ЗДПФ
figure; plot(lSpeckleFl); grid;            Побудова графіка відфільтрованого сигналу
lSpeckleF = fft(lSpeckle);                 ДПФ
lSpeckleF(100:156) = 0;                    Видалення високих частот
lSpeckleFl = real(ifft(lSpeckleF));        ЗДПФ
figure; plot(lSpeckleFl); grid;            Побудова графіка відфільтрованого сигналу
```

4. Фільтрація шуму Salt & pepper

```
lSP = imnoise(l, 'salt & pepper');         Додавання мультиплікативного шуму
figure; plot(lSP); grid;                   Побудова графіка зашумленного зрізу
lSPF = fft(lSP);                           ДПФ
lSPF(50:206) = 0;                          Видалення високих частот
lSPFl = real(ifft(lSPF));                  ЗДПФ
figure; plot(lSPFl); grid;                 Побудова графіка відфільтрованого сигналу
lSPF = fft(lSP);                           ДПФ
lSPF(100:156) = 0;                         Видалення високих частот
lSPFl = real(ifft(lSPF));                  ЗДПФ
figure; plot(lSPFl); grid;                 Побудова графіка відфільтрованого сигналу
```

Постановка завдання

1. Відкрити растрове зображення, перетворити його в клас *double*.
2. Виділити для обробки довільний рядок *n* зображення.
3. Виконати побудову графіка і гістограми рядку *n* (кількість інтервалів гістограми – *10*, ширина стовпчика гістограми – *0,7*; додати мітки по осях *X* і *Y*, додати підписи координатних осей і назву гістограми) (див. лаб. 6).
4. Виділити регіональну компоненту сигналу – рядок *n*. Для цього обнулити високочастотні складові перетворення Фур'є рядка *n* (значення складових перетворення Фур'є, наприклад, з 11-го по 189-й).
5. Візуалізувати речову частину сигналу *n*, отриманого після виконання зворотного перетворення Фур'є, візуально порівняти з графіком функції рядка *n*.
6. Виділити локальну компоненту сигналу. Для цього обнулити низькочастотні складові перетворення Фур'є рядка *n*.
7. Візуалізувати речову частину сигналу *n*, отриманого після виконання зворотного перетворення Фур'є.
8. Побудувати графіки амплітудно-частотної і фазо-частотної характеристики сигналу *n*. Графік АЧХ представити у логарифмічній шкалі.
9. Виконати пункти 4-7 послідовно для всіх рядків зображення. Візуалізувати отримані зображення в діапазоні значень від 0 до 1.
10. Побудувати гістограму вихідного зображення, зображення з видаленою високочастотною складовою і зображення з видаленою низькочастотною складовою.

11. Для довільного рядку довільного зображення виконати фільтрацію:

Номер комп'ютера	**Модель шуму**
1, 4, 7, 10	Гаусівський шум
2, 5, 8, 11	Мультиплікативний шум
3, 6, 9, 12	Шум Salt & Pepper

12. Побудувати графік вихідного, зашумленого і відфільтрованого рядку

Вихідні дані: зображення лабораторної роботи № 5.

Питання для підготовки до захисту лабораторної роботи

1. Для вирішення яких задач використовуються алгоритми частотної обробки растрових зображень?

2. Які функції пакету IPT дозволяють виконувати частотну обробку зображень?

3. Що представляє собою амплітудно-частотна характеристика зображення?

4. Дайте визначення фазо-частотної характеристики зображення.

5. Наведіть функції пакету IPT для додавання шуму до зображення.

Лабораторна робота 8
Морфологічна обробка зображень

Об'єкт – растрові зображення.

Предмет – методи морфологічної обробки зображень пакету MATLAB.

Мета – ознайомитися з можливостями пакету MATLAB щодо виконання операцій дилатації, ерозій, розмикання, замикання, перетворення успіх / невдача, виділення компонент зв'язності для двійкових і напівтонових растрових зображень.

Теоретичні положення

1. Дилатація і ерозія

У пакеті IPT операція дилатації реалізується функцією *imdilate*. Її основна форма виклику має вигляд:

A1=imdilate(A, B),

де *A* і *A1* – двійкові зображення, а *B* – матриця з нулів і одиниць, яка позначає структуроутворюючий елемент.

Наступна послідовність команд прочитує зображення з файлу, будує матрицю структуроутворюючого елементу, здійснює дилатацію і показує результат.

У пакеті IPT є функція *strel*, яка будує структуроутворюючі елементи різних форм і розмірів. Її синтаксис має вигляд:

se=strel(shape,parameters),

де *shape* – рядок, що позначає бажану форму структуроутворюючого елемента, а *parameters* задає список параметрів, які уточнюють інформацію про його форму, наприклад, визначають його розмір. Так, команда

strel('diamond', 5)

будує структуроутворюючий елемент у формі ромба, який має розмір ±*5* пікселів по горизонталі і вертикалі. У табл. 8.1 перераховані різні форми, які може будувати функція *strel*.

Таблиця 8.1

Різні синтаксичні форми функції *strel*

Синтаксична форма	Опис
se = strel('diamond', R)	Будує плоский структуроутворюючий елемент у формі ромба, де *R* позначає відстань від центру структуроутворюючого елемента до крайньої точки ромба.
se = strel('disk', R)	Будує плоский структуроутворюючий елемент у формі круга з радіусом *R* (наявні додаткові параметри, які описані в довідці по *strel*).

Продовження таблиці 8.1

se = strel('line', LEN, DEG)	Будує плоский лінійний елемент, де LEN позначає його довжину, а DEG – кут (в градусах) нахилу лінії, виміряний проти годинникової стрілки від горизонтальній осі.
se = strel('octagon', R)	Будує плоский шестикутний елемент, де R позначає відстань від центру елемента до сторони шестикутника, виміряну вздовж горизонтальної або вертикальної осі. Число R має бути додатнім і кратним 3.
se = strel('pair', OFFSET)	Будує плоский структуроутворюючий елемент, що складається з двох точок. Перша точка знаходиться в центрі, а друга точка зміщена від першої на вектор *OFFSET*, який повинен мати дві цілі компоненти.
se=strel('periodicline', P, V)	Будує плоский елемент, що складається з *2*P+1* точок. *V* – двоелементний цілочисельний вектор, в якому записані зміщення по рядку і стовпцю. Одна точка елемента знаходиться в його центрі, а всі інші мають координати *1*V*, *-1*V*, *2*V*, *-2*V*, ..., *P*V* і *-P*V*.
se=strel ('rectangle', MN)	Будує плоский елемент у формі прямокутника, де *MN* позначає його розміри. *MN* – це двоелементний вектор з невід'ємними цілими компонентами. Перший елемент *MN* означає число рядків, а другий – число стовпців.
se = strel('square', W)	Будує квадратний структуроутворюючий елемент ширини *W*, де *W* – невід'ємне ціле.
se = strel('arbitrary', NHOOD) *se = strel (NHOOD)*	Будує елемент довільної форми. *NHOOD* – це матриця з нулів і одиниць, що задає його форму. Друга, більш проста синтаксична форма, виконує ту ж операцію.

strel повертає спеціальну величину, яка називається *strel*-об'єктом.

Ерозія виконується в IPT функцією *imerode*.

```
A = imread('image1.tif');
se = strel('disk', 10);
A1 = imerode(A, se);
imshow(A1)
```

2. Розмикання і замикання

Операції розмикання і замикання реалізовані в пакеті IPT за допомогою функцій *imopen* і *imclose*. Обидві ці функції мають прості форми виклику

C=imopen(A, B)

C=imclose(A, B),

де *A* – це двійкове зображення, а *B* – матриця з *0* і *1*, яка задає структуроутворюючий елемент. *Strel*-об'єкт *se* можна використовувати замість аргументу *B*.

3. Перетворення успіх / невдача

Перетворення успіх / невдача реалізовано в IPT функцією *bwhitmiss*, яка має синтаксис:

C=bwhitmiss(A, Bl, B2),

де *C* – результат операції, *A* – вихідне зображення, *B1* і *B2* – структуроутворюючі елементи, задіяні в перетворенні.

4. Функція bwmorph

Функція *bwmorph* виконує дії, що базуються на операціях дилатації та ерозії. Вона має наступну синтаксичну форму виклику:

g = bwmorph (f, operation, n),

де *f* – вхідне двійкове зображення, *operation* – символьний рядок, що задає конкретну операцію, а *n* позначає, скільки разів необхідно виконати цю операцію. Вхідний аргумент *n* не є обов'язковим. При його відсутності операція виконується тільки один раз. У табл. 8.2 перераховані всі допустимі операції функції *bwmorph*.

Таблиця 8.2

Допустимі операції функції *bwmorph*

Операція	**Опис**
bridge	З'єднання пікселів, розділених проміжком в один піксель.
clean	Видалення ізольованих пікселів переднього плану.
close	Замикання з використанням елемента *3×3*. Для інших структуроутворюючих елементів застосовується операція *imclose*.
diag	Заповнення діагонально пов'язаних пікселів переднього плану.
dilate	Дилатація з використанням елемента *3×3*. Для інших структуроутворюючих елементів застосовується операція *imdilate*.
erode	Ерозія з використанням елемента *3×3*. Для інших структуроутворюючих елементів застосовується операція *imerode*.
fill	Заповнення однопіксельних «дірок» (фонові пікселі, оточені з усіх боків пікселями переднього плану).
majority	Перетворення пікселя *p* у пиксель переднього плану, якщо не менше *5* пікселів в $N_8(p)$ є пікселями переднього плану; в іншому випадку робить піксель *p* фоновим.
open	Розмикання з використанням елемента *3×3*. Для інших структуроутворюючих елементів застосовується операція *imopen*.

Продовження таблиці 8.2

remove	Видалення «внутрішніх» пікселів (пікселів переднього плану, у яких немає жодного фонового сусіда).
shrink	Стискання об'єкти без внутрішніх дірок в точки. Стискання об'єкти з дірками в кільця.
skel	Побудова остову зображення.
spur	Видалення відростків пікселів.
thicken	Потовщення об'єктів без з'єднання незв'язних частин.
thin	Потоншення об'єктів без дірок до мінімальної середньої лінії. Потоншення об'єктів з дірками до кілець.

Операція потоншення скорочує двійкові об'єкти або форми зображення до окремих ліній, які мають товщину всього в один піксель.

Виклик *bwmorph* зі значенням *Inf* (нескінченність) виконує операції доти, поки зображення не припинить змінюватися. Іноді це називають повторенням операції до стабілізації. Наприклад,

```
ginf = bwmorph(f, 'thin', Inf);
imshow(ginf)
```

Побудова остова являє собою інший шлях скорочення об'єктів двійкового зображення для отримання множини тонких ліній, яка несе важливу інформацію про форми вихідних об'єктів.

Процедури побудови остова і стоншення часто будують зайві короткі відростки, які іноді називаються *паразитичними компонентами*. Процес очищення (або видалення) цих компонент називається відсіканням. Для цього можна використовувати функцію *endpoints*. Метод полягає в послідовному виявленні кінцевих точок та їх видаленні. Наступні прості команди здійснюють завершальну обробку збудованого остова *fs* зображення, яка повторюється *5* разів, для видалення кінцевих точок:

```
for k = 1:5
    fs = fs & ~endpoints(fs);
end
```

5. Виділення компонент зв'язності

Функція *bwlabel* з IPT знаходить всі компоненти зв'язності двійкового зображення. Форма її виклику має вигляд:

[L, num]=bwlabel(f, conn),

де *f* – вхідне двійкове зображення, а параметр *conn* позначає бажану зв'язність (по 4- або 8-суміжну). Вихід *L* називається розмічаючою матрицею, а необов'язковий вихідний параметр *num* записується загальна кількість виявлених компонент зв'язності. Якщо параметр *conn* опущений, то він за замовчуванням вважається рівним *8*. Пікселю кожної окремої компоненти зв'язності розмічаючої матриці, обчисленої командою *bwlabel*, присвоюється однакове число від *1* і до загального числа компонент зв'язності. Іншими словами, пікселі, помічені *1*, належать першій компоненті зв'язності, пікселі, помічені *2* – другій компоненті, і т.д. Фонові пікселі позначені *0*.

Для знаходження центрів мас компонент зв'язності, насамперед, слід скористатися функцією *bwlabel* для обчислення 8-зв'язкових компонент:

```
f = imread('image.tif');
[L, n] = bwlabel(f);
```

Функція *find* може бути корисною при знаходженні центру мас об'єкта. Наприклад, наступний виклик функції *find* повертає індекси рядків і стовпців всіх пікселів, що належать третіому об'єкту:

```
[r, c] = find(L == 3);
```

Функція *mean* з виходами *r* і *c* обчислює центр мас цього об'єкта.

```
rbar = mean(r);
cbar = mean(c);
```

Для знаходження центрів мас усіх об'єктів зображення можна скористатися циклом. Щоб зробити центри мас об'єктів видимими на зображенні, можна позначити їх символом «*» білого кольору і розмістити його у центрі чорного кола:

```
imshow(f)
hold on
for k = 1:n
    [r, c] = find(L == k);
    rbar = mean(r);
    cbar = mean(c);
    plot(cbar, rbar, 'Market', 'o', 'MarkerEdgeColor', 'k',...
    'MarkerFaceColor', 'k', 'MarkerSize', 10)
    plot(cbar, rbar, 'Market', '*', 'MarkerEdgeColor', 'w')
end
```

6. Напівтонова морфологія

Плоскі структуроутворюючі елементи для напівтонових зображень будуються з використанням функції *strel* за тими ж правилами, що і для двійкових зображень. Наприклад, наступні команди показують, як здійснювати дилатацію зображення f за допомогою плоского елемента розміром 3x3:

```
se = strel('square', 3);
gd = imdilate(f, se);
```

Дилатацію і ерозію можна поєднувати для досягнення різних ефектів. Наприклад, віднімання результату ерозії зображення від результату дилатації задає 'морфологічний градієнт', який являє собою міру локальної напівтонової варіації зображень.

Постановка завдання

1. Відкрити зображення *'1.tif'*.
2. Виконати операцію ерозії структуроутворюючим елементом:

№ комп'ютера	Форма структуроутворюючого елементу
1	Ромб
2	Коло
3	Лінія
4	Шестикутник
5	Дві точки
6	Послідовність точок

№ комп'ютера	Форма структуроутворюючого елементу
7	Прямокутник
8	Квадрат
9	Елемент довільної форми
10	Послідовність точок
11	Елемент довільної форми
12	Лінія

3. Застосувати операцію дилатації до отриманого результату. По черзі застосовуючи операції дилатації та ерозії, досягти згладжування контурів і видалення дрібних деталей.

4. Відкрити зображення (*'2.tif'* – для парних номерів комп'ютерів, *'3.tif'* – для непарних). Виконати операцію морфологічного розмикання. Застосувати операцію морфологічного замикання до отриманого результату. Форма і розмір структуроутворюючого елемента вибирається довільно у відповідності із забезпеченням найкращого візуального сприйняття результуючого зображення.

5. Відкрити покращене зображення, отримане на кроці 4. Використовуючи функцію *bwmorph*, виконати наступні дії:

№ комп'ютера	Дія
1	З'єднати пікселі, розділені проміжком в один піксель
2	Видалити ізольовані пікселі переднього плану
3	Виконати операцію замикання з використанням елемента *3×3*
4	Виконати заповнення областей навколо діагонально пов'язаних пікселів переднього плану
5	Виконати дилатацію з використанням елемента *3×3*
6	Виконати ерозію з використанням елемента *3×3*
7	Заповнити однопіксельні «дірки»
8	Видалити «внутрішні» пікселі
9	Стиснути об'єкти в точки або в кільця
10	Виконати потовщення об'єктів
11	Виконати потоншення об'єктів
12	Побудувати остов зображення

6. Відкрити зображення (*'4.tif'*). Виконати операції напівтонової дилатації та ерозії. Обчислити морфологічний градієнт. Розмір і форму структуроутворюючого елемента вибрати самостійно.

Питання для підготовки до захисту лабораторної роботи

1. Яку дію виконує команда *strel('diamond', 5)*?

2. Наведіть функції пакету IPT для виконання морфо метричної обробки зображень.

3. Яку дію виконує операція *dilate* функції *bwmorph*?

4. Які функції пакету IPT використовуються для пошуку компонент зв'язності двійкового зображення?

5. Що являє собою 'морфологічний градієнт'?

Лабораторна робота 9
Сегментація зображень

Об'єкт – растрові зображення.

Предмет – методи сегментації зображень пакету MATLAB.

Мета – ознайомитися з можливостями пакету MATLAB щодо виконання сегментації растрових зображень за морфологічними вододілами.

Теоретичні положення

1. Сегментація перетворенням вододілу

У географії термін *вододіл* позначає умовну лінію, яка розділяє області водозборів різних річкових систем. Водозбірний басейн визначає географічну область, вода з якої збирається в одну річку або водосховище. Перетворення вододілу застосовує цю ідею до обробки монохромних зображень при вирішенні різних типів задач сегментації.

Щоб зрозуміти перетворення вододілу, необхідно уявити собі напівтонове зображення у вигляді поверхні рівнів, на якій величини *f(x, y)* інтерпретуються як висоти. Якщо уявити собі дощ, що йде над цією поверхнею, то, очевидно, вода буде збиратися в областях, позначених як водозбірні басейни. Краплі дощу, які падають точно на лінію, позначену як лінія вододілу, будуть з однаковою ймовірністю збиратися і в лівий, і в правий басейн. Перетворення вододілу знаходить водозбірні басейни і будує лінію вододілу на напівтоновому зображенні. Якщо використовувати термінологію по сегментації, то ключовим моментом є зміна початкового зображення і перетворення його в таке зображення, щоб водозбірними басейнами на ньому були області та об'єкти, які слід сегментувати (рис. 9.1).

а) б) в)

Рис. 9.1 Приклад сегментації за морфологічними вододілами: а) вихідне зображення; б) рельєфне представлення; в) лінії вододілів.

2. Сегментація по вододілах за допомогою перетворення відстані

Інструмент, який часто використовується при сегментації по вододілу, називається *перетворенням відстані*. Перетворення відстані двійкового зображення визначає відстань від кожного пікселя до найближчого пікселя з ненульовим значенням. Ненульові пікселі мають нульові значення перетворення відстані. Це перетворення обчислюється функцією *bwdist*, яка має наступний синтаксис виклику:

D=bwdist(f)

Приклад. Сегментація двійкового зображення по вододілу за допомогою перетворення відстані.

Двійкове зображення					**Перетворення відстані**				
1	1	0	0	0	0,00	0,00	1,00	2,00	3,00
1	1	0	0	0	0,00	0,00	1,00	2,00	3,00
0	0	0	0	0	1,00	1,00	1,41	2,00	2,24
0	0	0	0	0	1,41	1,00	1,00	1,00	1,41
0	1	1	1	0	1,00	0,00	0,00	0,00	1,00

Для виконання сегментації необхідно спочатку перетворити зображення в двійковий формат за допомогою функцій *im2bw* і *graythresh*.

g=im2bw(f, graythresh(f));

Наступний крок полягає в отриманні негативного (додаткового) зображення, обчисленні перетворення відстані і знаходженні перетворення вододілу від зображення, отриманого за допомогою функції *watershed*. Ця функція викликається командою

L=watershed(f),

де *L* – матриця розмітки. Додатні цілі значення *L* відповідають водозбірним басейнам, а нульові значення позначають пікселі ліній вододілів.

```
gc = ~g;
D = bwdist(gc);
L = watershed (-D);
w = L == 0;
```

Оскільки нульові значення в *L* позначають лінії вододілів, останній командний рядок будує двійкове зображення *w*, що виокремлює тільки ці пікселі.

Логічне *AND* вихідного двійкового зображення з доповненням до *w* дає повну сегментацію:

```
g2 = g & ~w;
```

3. Сегментація по вододілах за допомогою градієнтів

Модуль градієнта часто використовується під час попередньої обробки напівтонових зображень перед сегментацією по вододілах. Пікселі зображення

градієнта з великими значеннями розташовуються поблизу границь об'єктів, а іншим ділянкам відповідають нульові значення пікселів. Виконуючи потім перетворення вододілу, можна отримати лінії вододілів уздовж границь об'єктів.

Обчислення модуля градієнта, використовуючи метод лінійної фільтрації:

```
h = fspecial('sobel');
fd = double(f);
g = sqrt(imfilter(fd, h, 'replicate').^2 + ...
    imfilter(fd, h', 'replicate').^2);
```

Обчислення перетворення по вододілах від градієнта і знаходження границь вододілів:

```
L = watershed(g);
wr = L == 0;
```

Проте в результаті сегментації є багато ліній вододілів, які не оточують об'єкти інтересу. Це випадок надлишкової сегментації. Можна спробувати впоратися з цією проблемою, згладивши зображення градієнта перед оптимізацією вододілів. Для цього застосовується замикання розмиканням.

```
g2 = imclose(imopen(g, ones(3,3)), ones(3,3));
L2 = watershed(g2);
wr2 = L2 == 0;
f2 = f;
f2(wr2) = 255;
```

Останні два рядки програмного коду поміщають зображення *wr2* ліній вододілів поверх вихідного зображення.

4. Використання маркерів при сегментації по вододілах

Безпосереднє застосування алгоритму сегментації по вододілах в тому вигляді, який описувався в попередньому пункті, зазвичай призводить до надлишкової сегментації, викликаної шумом або іншими локальними неоднорідностями на градієнтному зображенні. Надлишкова сегментація може бути настільки значною, що зробить результат обробки практично марним. Це означає величезне число областей, виділених при сегментації. Практичне розв'язання цієї проблеми полягає в тому, щоб обмежити допустиму кількість областей шляхом включення до складу процедури кроку попередньої обробки, яка слугує для привнесення додаткової інформації в процедуру сегментації.

Підхід, який застосовується для управління надлишковою сегментацією, заснований на ідеї маркерів. Маркер являє собою компоненту зв'язності, яка належить зображенню. Розрізняють внутрішні маркери, що відносяться до об'єктів інтересу, і зовнішні маркери, які відповідають фону зображення. Потім ці маркери використовуються для поправки градієнтного зображення при виконанні процедури, описаної нижче. Розроблено різні методи для побудови внутрішніх і зовнішніх маркерів, багато з яких використовують лінійну або нелінійну фільтрацію, а також морфологічну обробку.

```
h = fspecial('sobel');
fd = double(f);
g = sqrt(imfilter(fd, h, 'replicate').^2 + ...
    imfilter(fd, h', 'replicate').^2);
L = watershed(g);
wr = L == 0;
```

На результуючому зображенні спостерігається надлишкова сегментація, причина якої полягає у великому числі локальних мінімумів. Функція *imregionalmin* визначає всі локальні мінімуми зображення. Вона викликається у вигляді:

rm=imregionalmin(f),

де *f* – півтонове зображення, а *rm* позначає двійкове зображення, в якому одиницею відзначають положення локальних мінімумів.

Більшість локальних мінімумів є дуже дрібними. Вони відображають маленькі деталі, які не відіграють ролі в задачі сегментації. Щоб виключити такі додаткові мінімуми, використовується функція з IPT *imextendedmin*, яка знаходить «низькі» плями на зображенні, що лежать глибше деякого заданого порогового рівня в порівнянні з їх найближчим оточенням. Ця функція має форму виклику:

im=imextendedmin(f, h),

де *f* – півтонове зображення, *h* – пороговий рівень, а *im* позначає двійкове зображення, на якому пікселі переднього плану відзначають положення глибоких мінімумів. Тепер можна застосувати функцію *imextendedmin*, яка задасть множину внутрішніх маркерів:

```
im = imextendedmin(f, 2);
fim = f ;
fim(im) = 175;
```

Останні два рядки поміщають положення розширеної множини мінімумів у вигляді сіруватих плям на вихідне зображення.

Тепер необхідно побудувати зовнішні маркери або пікселі, що належать фону зображення. Фон відзначається пікселями, які розташовані точно посередині між внутрішніми маркерами. Для цього потрібно обчислити перетворення вододілу від перетворення відстані зображення *im* внутрішніх маркерів:

```
Lim = watershed(bwdist(im));
em = Lim == 0;
```

Внутрішні і зовнішні маркери використовуються для модифікування градієнтного зображення за допомогою процедури, яка називається *мінімальним підйомом*. Техніка мінімального підйому модифікує півтонування так, що локальні мінімуми досягаються тільки у зазначених положеннях. Інші величини пікселів підвищуються для зникнення всіх інших точок локального мінімуму. Функція IPT *imimposemin* реалізує цей підхід. Вона викликається командою:

mp = imimposemin (f, mask),

де *f* – це вихідне півтонове зображення, а *mask* – двійкове зображення, пікселі переднього плану якого відзначають бажане положення локальних мінімумів на вихідному зображенні. Модіфіцікація градієнтного зображення шляхом збільшення локальних мінімумів в положеннях внутрішніх і зовнішніх маркерів:

g2=imimposemin(g, im|em);

Перетворення вододілу зазначеного та модифікованого градієнтного зображення:

```
L2 = watershed(g2);
f2 = f;
f2(L2 == 0) = 255;
```

Постановка завдання

Сегментація по вододілах за допомогою перетворення відстані

1. Відкрити зображення *1.tif (a)*:

```
f = imread('V:\1.tif');
a = im2double(f);
```

2. Отримати зображення-негатив зображення *a (b)*:

```
b = ~a;
```

3. Виконати перетворення відстані для зображення *b*, візуалізувати отриману матрицю при діапазоні значень її елементів від *0* до *1 (c)*:

```
c=bwdist(b);
```

4. Побудувати лінії вододілів на основі матриці відстані *(d)*:

```
d=watershed(-c);
```

5. Поєднати зображення *a* і *d*:

```
e = a + ~d;
figure, imshow(e);
```

Сегментація по вододілах за допомогою градієнтів

1. Відкрити зображення *2.tif (a)*:

```
f = imread('V:\2.tif');
a = im2double(f);
```

2. Використовуючи маску оператора Собела, обчислити модуль градієнта зображення *a (b)*:

```
h=fspecial('sobel');
b=sqrt(imfilter(a, h, 'replicate').^2 + imfilter(a, h', 'replicate').^2);
```

3. Виконати перетворення вододілу зображення *b (c)*:

```
c = watershed(bwdist(b));
```

4. Поєднати зображення *a* і *c*, візуалізувати результат:

```
d = a + ~c;
figure, imshow(d);
```

5. Виконати розмикання-замикання зображення модуля градієнта *b* за примітивом, який представляє собою матрицю розміру *3×3*, заповнену одиницями *(d)*:

```
d=imclose(imopen(b, ones(3,3)), ones(3,3));
```

6. Виконати перетворення вододілу зображення *d (e)*:

```
e=watershed(d);
```

7. Поєднати зображення *a* і *e*, візуалізувати результат:

```
f= a + ~e;
figure, imshow(mat2gray(f));
```

Використання маркерів при сегментації по вододілах

1. Відкрити зображення *3.tif (a)*:

```
f = imread('V:\3.tif');
a = im2double(f);
```

2. Використовуючи маску оператора Собела, обчислити модуль градієнта зображення *a (b)*:

```
h=fspecial('sobel');
b=sqrt(imfilter(a, h, 'replicate').^2 + imfilter(a, h', 'replicate').^2);
```

3. Виділити локальні мінімуми зображення *a*, виконавши перетворення розширеного мінімуму *(c)*:

```
c = imextendedmin(a, 0.1);
```

4. Виконати перетворення вододілу від перетворення відстані зображення *c (d)*:

```
d = watershed(bwdist(c));
```

5. Створити зображення *e*, значення елементів якого рівні *1* у випадку, якщо значення елементів зображення *d* рівні *0*, наприклад:

```
e = d == 0;
```

6. Виконати процедуру мінімального підйому:

```
f = imimposemin(b, c|e);
```

7. Провести перетворення вододілу для зображення *f (g)*:

```
g = watershed(f);
```

8. Виділити контури вододілів у відповідності з виразом:

```
k = a + ~g;
```

9. Візуалізувати зображення *g*:

```
figure, imshow(mat2gray(k));
```

Питання для підготовки до захисту лабораторної роботи

1. Поясніть принцип сегментації зображень перетворенням вододілу.

2. Що визначає перетворення відстані двійкового зображення?

3. Яка функція пакету IPT дозволяє виконувати перетворення вододілу від зображення?

4. На чому полягає використання маркерів при сегментації по вододілах?

5. Яку дію виконує команда *im2bw(f, graythresh(f))*?

Лабораторна робота 10
Обробка зображень у середовищі Borland Delphi

Об'єкт – растрові зображення.

Предмет – методи обробки зображень у середовищі Borland Delphi.

Мета – створити найпростіший інтерфейс користувача для роботи з зображеннями. Розробити функціональний інструментарій для інвертування палітри кольорів та подання повнокольорового зображення в градаціях сірого. Реалізувати процедури тематичної обробки зображень.

Теоретичні положення

1. Створення графічного інтерфейсу користувача

Основними складовими графічного інтерфейсу користувача є графічне вікно і меню відкриття, збереження зображень (рис. 10.1).

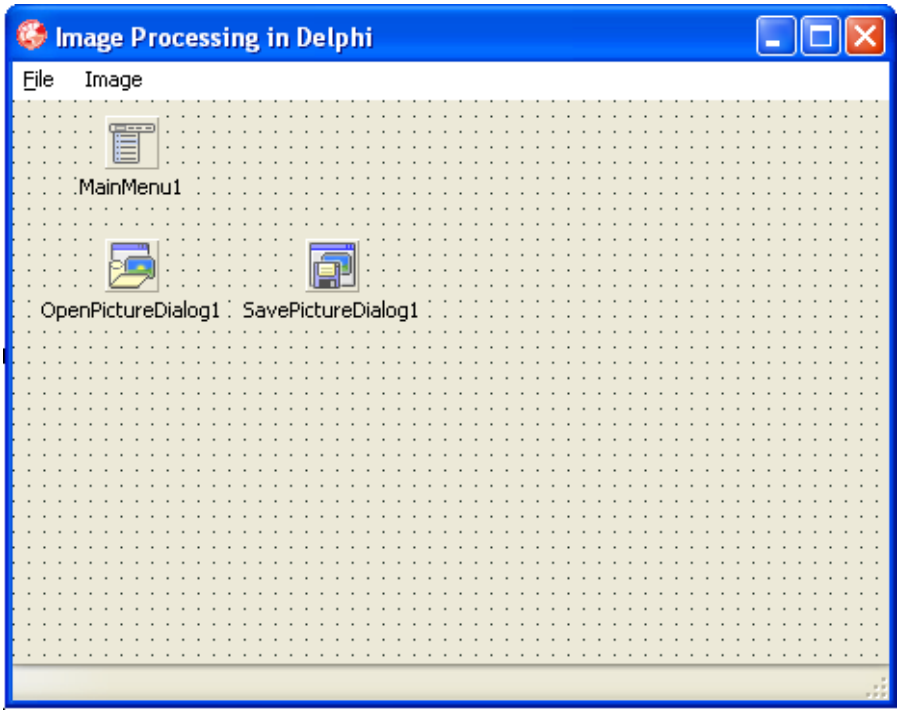


Рис. 10.1 Складові найпростішого інтерфейсу користувача для роботи з зображеннями у середовищі Borland Delphi

Етапи виконання роботи:

1. Створіть новий проект (*File->New->Application*).

2. Для характеристики форми *FormStyle* задайте значення *fsMDIForm* (*MDI – Multiple Document Interface*, багатодокументний інтерфейс. В додатках з *MDI*, в основному (батьківському) вікні можна відкрити більше одного дочірнього вікна).

3. Задайте ім'я форми: *MainForm*.

4. Додайте на форму компонент *Main Menu* (панель *Standard*), додайте основний пункт меню *File* і підпункти *Open*, *Save*, *Close*, *Exit*.

5. Додайте на форму компоненти *OpenPictureDialog* і *SavePictureDialog* (панель *Dialogs Component Pallet*).

6. Для компонента *SavePictureDialog* задайте властивість *Option.ofOverwritePrompt=true* (у випадку, якщо при збереженні файлу користувач написав ім'я існуючого файлу, з'являється зауваження, що файл з таким ім'ям існує, і запитується бажання користувача переписати існуючий файл).

7. Додайте на форму компонент *Status Bar* (панель *Win32*).

8. Збережіть основну форму проекту, призначте їй осмислену назву.

9. Створіть нову форму для відображення зображення в дочірньому вікні основної форми (*File->New->Form*).

10. Для властивості форми *FormStyle* задайте значення *fsMDIChild*.

11. Задайте ім'я форми: *ImageForm*.

12. Додайте на форму компонент *Image* (панель *Additional*).

13. Для компонента *Image* задайте властивість *Align=alClient* (компонент *Image* займе весь доступний простір компонента *ImageForm*).

14. Для компонента *Image* задайте властивість *Stretch=true* (дана властивість дозволяє підганяти зображення під розмір компонента).

15. Для компонента *Image* задайте властивість *Proportional=true* (дозволяє автоматично масштабувати зображення без спотворення. Для виконання масштабування, значення властивості *AutoSize* має бути рівним *False*).

16. Збережіть форму для візуалізації зображень і проект в цілому.

Після виконання перерахованих етапів і компіляції проекту, при запуску виконуваного файлу дочірня форма буде автоматично показана на екрані. Для коректного відображення дочірньої форми виключно при відкритті файлу зображення необхідно відключити автоматичне створення форми: *Project-> Options->Form*. У вікні перемістіть дочірню форму зі списку *Auto-create forms* в список *Available forms*.

Обробник події *OnClose* для *ImageForm*.

При закритті форми, призначеної для візуалізації зображення, необхідно звільнити використані нею ресурси пам'яті. Для цього в обробнику події *OnClose* форми *ImageForm* необхідно прописати:

```
procedure TImageForm.FormClose(Sender: TObject; var Action: TCloseAction);
begin
  Action := caFree;
end;
```

Завантаження зображення з файлу

Оскільки для візуалізації зображення використовується форма *ImageForm*, необхідно це явно вказати в основному юніті. Натисніть на *MainForm*, виберіть меню *File->Use Unit* і *ImageUnit*. Натисніть *OK*:

```
implementation

uses ImageUnit;
```

Створіть обробник події MainMenu1-> File-> Open:

```
procedure TMainForm.Open1Click(Sender: TObject);
var
 formatInfo:string;
begin
 if OpenPictureDialog1.Execute then
 begin
 Application.CreateForm(TImageForm, ImageForm);
  ImageForm.Image1.Picture.LoadFromFile(
    OpenPictureDialog1.FileName);
  ImageForm.ClientHeight:=
    ImageForm.Image1.Picture.Height;
  ImageForm.ClientWidth:=
    ImageForm.Image1.Picture.Width;
  case (ImageForm.Image1.Picture.Bitmap.PixelFormat) of
    pf1bit : formatInfo:='Binary';
    pf8bit : formatInfo:='Gray scale';
    pf24bit: formatInfo:='True color';
    end;
  StatusBar1.SimpleText:= OpenPictureDialog1.FileName +' '+
    IntToStr(ImageForm.Image1.Picture.Width) + 'x'+
    IntToStr(ImageForm.Image1.Picture.Height) + ' '+
    formatInfo;
 end;
end;
```

Запис зображення в файл

Створіть обробник події *MainMenu1->File->Save*:

```
procedure TMainForm.Save1Click(Sender: TObject);
begin
try
begin
  if SavePictureDialog1.Execute then
    TImageForm(ActiveMDIChild).Image1.Picture.SaveToFile(
    SavePictureDialog1.FileName);
end
except
  MessageDlg('Cannot complete the operation!', mtError, [mbOK], 1);
end;
end;
```

Закриття вікна візуалізації зображення

Створіть обробник події *MainMenu1->File->Close*:

```
procedure TMainForm.CLose1Click(Sender: TObject);
begin
  try
    ActiveMDIChild.Close;
  except
    MessageDlg('Cannot complete the operation!', mtError, [mbOK], 1);
  end;
end;
```

Вихід з програми

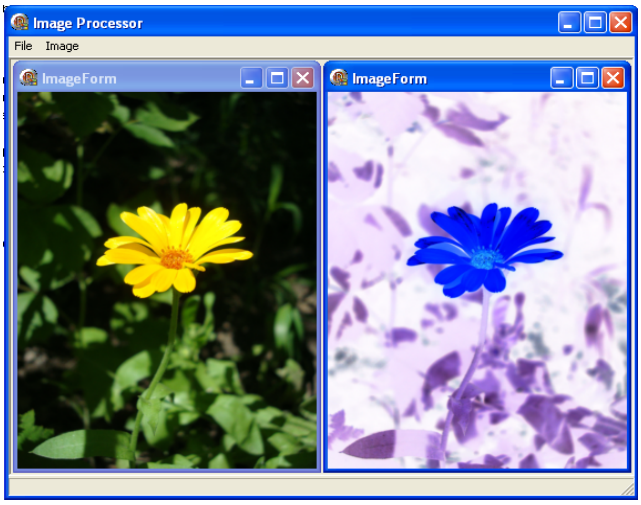
Створіть обробник події *MainMenu1->File->Exit*:

2. Зміна палітри кольорів зображень

1. Для інвертування палітри кольорів зображення створіть підміню *Invert* меню *Image*. Реалізуйте відповідний обробник *onClick*:

```
procedure TMainForm.Invert1Click(Sender: TObject);
var
  i,j:integer;
  ptr:PByteArray;
begin
  try
    ImageForm:=TImageForm(ActiveMDIChild);
    for i:=0 to (ImageForm.Image1.Height-1) do
    begin
      ptr:=ImageForm.Image1.Picture.Bitmap.ScanLine[i];
      for j:=0 to (ImageForm.Image1.Width-1) do
      begin
        if ImageForm.Image1.Picture.Bitmap.PixelFormat
          =pf8bit then ptr[j]:=255-ptr[j];
        if ImageForm.Image1.Picture.Bitmap.PixelFormat
          =pf24bit then
        begin
          ptr[3*j]:=255-ptr[3*j];
          ptr[3*j+1]:=255-ptr[3*j+1];
          ptr[3*j+2]:=255-ptr[3*j+2];
        end;
      end;
      ImageForm.Image1.Refresh;
    end;
    except
      ShowMessage('Cannot complete the operation');
    end
end;
```

Приклад роботи програми:

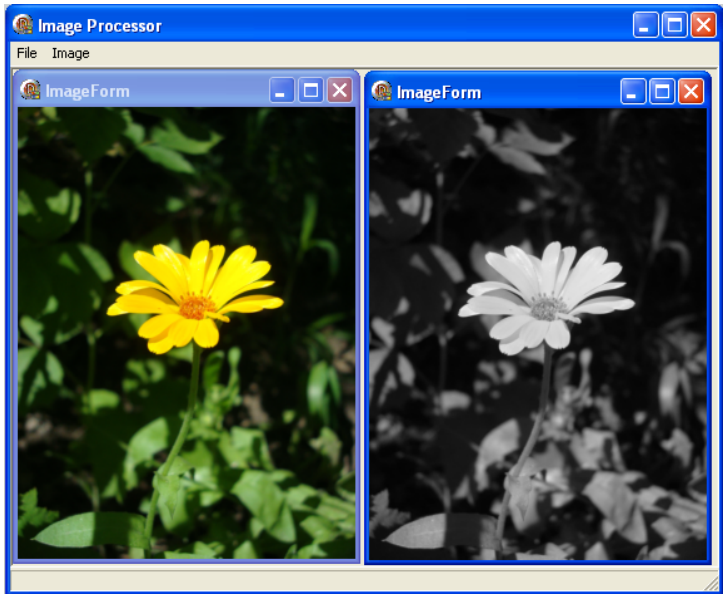


2. Для представлення повнокольорового зображення в градаціях сірого створіть підміню *Convert to Gray Scale* меню *Image*. Реалізуйте відповідний обробник *onClick*:

```
procedure TMainForm.ConverttoGrayScale1Click(Sender: TObject);
var
  i,j:integer;
  ptr:PByteArray;

begin
  try
    ImageForm:=TImageForm(ActiveMDIChild);
    for i:=0 to (ImageForm.Image1.Height-1) do
    begin
      ptr:=ImageForm.Image1.Picture.Bitmap.ScanLine[i];
      for j:=0 to (ImageForm.Image1.Width-1) do
      begin
        if ImageForm.Image1.Picture.Bitmap.PixelFormat
          =pf24bit then
        begin
          ptr[3*j]:=round(0.114* ptr[3*j]
            +0.587*ptr[3*j+1] + 0.299*ptr[3*j+2]);
          ptr[3*j+1]:=ptr[3*j];
          ptr[3*j+2]:=ptr[3*j];
        end;
      end;
      ImageForm.Image1.Refresh;
    end;
    except
      ShowMessage('Cannot complete the operation');
    end
end;
```

Приклад роботи програми:



Індивідуальні завдання

1. Розробити фрагмент програми порівняння бінарних (і напівтонових) зображень. Програма має забезпечувати можливості:

– візуалізації двох вхідних зображень, а також вихідного зображення, що містить знайдені відмінності;

– обчислення коефіцієнта подібності зображень (відсотка ідентичних пікселів);

– розрахунку критеріїв якості зображень, серед яких: пікове відношення сигнал / шум, мінімаксний критерій.

2. Розробити фрагмент програми, що забезпечує можливість виконання матричних перетворень (фільтрації) зображень. Реалізувати процедуру фільтрації з використанням наступних фільтрів:

– усереднюючий фільтр;

– просторові фільтри підвищення різкості з використанням перших похідних (оператори Превітта, Собела, Робертса);

– просторові фільтри підвищення різкості з використанням других похідних (оператор Лапласа);

– фільтр користувача.

3. Розробити фрагмент програми для класифікації зображень. Програма повинна забезпечувати можливість відкриття і візуалізації зображень-еталонів (різних класів), а також одного зображення, що підлягає класифікації. Для еталонних зображень і зображень, що мають бути класифіковані, розраховується набір ознак:

– яскравісні ознаки;

– гістограмні ознаки.

Віднесення зображення до класу виконується за допомогою оцінки його схожості з еталонами (по мінімуму спектральної відстані в просторі ознак) на підставі розрахунку:

– евклидової відстані;

– відстані Махаланобіса;

– відстані по Манхеттену (city-block).

4. Розробити фрагмент програми для сегментації та морфологічної обробки зображень. Програма повинна забезпечувати можливість відкриття і візуалізації вихідного зображення і результату обробки, а також забезпечувати можливості виконання:

– порогової сегментації (реалізувати алгоритм автоматизованого вибору порогового значення);

– морфологічної обробки (реалізувати оператори дилатації, ерозії, замикання, розмикання, морфологічного градієнта).

ПЕРЕЛІК ПОСИЛАНЬ

1. Айвазян С.А. Прикладная статистика: классификация и снижение размерности / С.А. Айвазян, В.М. Бухштабер, И.С. Енюков, Л.Д. Мешалкин // М.: Финансы и статистика, 1989. – 607 с.

2. Вапник В.Н. Теория распознавания образов / В.Н. Вапник, А.Я. Червоненкис // М.: Наука, 1974.

3. Гонсалес Р. Цифровая обработка изображений / Р. Гонсалес, Р. Вудс // М.: Техносфера. – 2005. – 1072 с.

4. Гонсалес Р. Цифровая обработка изображений в среде MATLAB / Р. Гонсалес, Р. Вудс, С. Эддинс // М.: Техносфера, 2006. – 616 с.

5. Грузман И.С. Цифровая обработка изображений в информационных системах: учебное пособие / И.С. Грузман, В.С. Киричув, В.П. Косых, Г.И. Перетягин, А.А. Спектор // Новосибирск: Изд-во НГТУ, 2000. – 168 с.

6. Дуда Р. Распознавание образов и анализ сцен / Р. Дуда, Р. Харт // М.: Мир, 1976. – 512 с.

7. Дьяконов В.П. MATLAB 6.5 SP1/ 7/ 7 SP1/ 7 SP2 Simulink 5/6. Инструменты искусственного интеллекта и биоинформатики / В.П. Дьяконов, В.В. Круглов // М.:СОЛОН-ПРЕСС, 2006. – 456 с.

8. Прэтт У. Цифровая обработка зображений / У. Прэтт // М.: Мир, 1982. – 480 с.

9. Рутковская Д. Нейронные сети, генетические алгоритмы и нечеткие системы / Д. Рутковская, М. Пилиньский, Л. Рутковский // М.: Горячая линия-Телеком, 2007. – 452 с.

10. Ту Д. Принципы распознавания образов / Д. Ту, Р. Гонсалес // М.: Мир, 1978. – 411 с.

11. Фукунага К. Введение в статистическую теорию распознавания образов / К. Фукунага // М.: Наука, 1989. – 368 с.

Хайкин С. Нейронные сети: полный курс, 2-е издание / С. Хайкин // М.: Издательский дом "Вильямс", 2006. – 1104 с.

ДОДАТОК А

ПРИКЛАДИ РОЗРАХУНКУ МІР ВІДСТАНІ МІЖ ОБ'ЄКТАМИ У БАГАТОВИМІРНОМУ ПРОСТОРІ ОЗНАК

Поняття міри відстані дозволяє оцінити ступінь подібності між елементами системи розпізнавання, тобто між об'єктами, класами об'єктів, класом і окремим об'єктом. Залежно від вирішуваної задачі використовується певна міра відстані (подібності). Нижче наведено міри подібності, які найбільш часто застосовуються для вирішення задач розпізнавання. Для прикладу розглядається два вектори ознак у тривимірному просторі ознак:

$$x_r = (0.1 \quad 0.2 \quad 0.7);$$
$$x_s = (0.3 \quad 0.6 \quad 0.8);$$

1. Евклідова відстань.

$$d_{rs} = \sqrt{\sum_{j=1}^{n} (x_{rj} - x_{sj})^2};$$

де x_r, x_s – r і s рядка матриці X об'єктів вихідної множини даних; n – розмірність вектору ознак.

Приклад: $d_{rs} = \sqrt{(0.1-0.3)^2 + (0.2-0.6)^2 + (0.7-0.8)^2} \approx 0.458.$

2. Відстань Махаланобіса.

$$d_{rs} = \sqrt{(x_r - x_s) V^{-1} (x_r - x_s)'},$$

де V – коваріаційна матриця, розрахована за вибіркою X.

3. Відстань по Манхеттену (city-block).

$$d_{rs} = \sum_{j=1}^{n} |x_{rj} - x_{sj}|.$$

Приклад: $d_{rs} = |0.1-0.3| + |0.2-0.6| + |0.7-0.8| = 0.7.$

4. Метрика Мінковського.

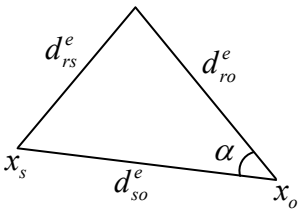
$$d_{rs} = \left[\sum_{j=1}^{n} |x_{rj} - x_{sj}|^p \right]^{\frac{1}{p}}.$$

При p = 1 метрика Мінковського еквівалентна відстані по Манхеттену. При p = 2 метрика Мінковського перетворюється у Евклідову відстань.

Приклад (для p = 3): $d_{rs} = \sqrt[3]{|0.1-0.3|^3 + |0.2-0.6|^3 + |0.7-0.8|^3} \approx 0.418.$

5. Косинусна відстань.

Для виведення міри косинусної відстані введемо додатковий вектор x_o – початок координат n- мірного простору ознак. За теоремою косинуса, де d^e – Евклідова відстань:



$$\cos\alpha = \frac{\left(d^e_{ro}\right)^2 + \left(d^e_{so}\right)^2 - \left(d^e_{rs}\right)^2}{2d^e_{ro}d^e_{so}};$$

$$d_{rs} = 1 - \cos\alpha.$$

Приклад:

$$d_{rs} = 1 - \frac{\left(0.1^2 + 0.2^2 + 0.7^2\right) + \left(0.3^2 + 0.6^2 + 0.8^2\right) - 0.458^2}{2 \cdot \sqrt{0.1^2 + 0.2^2 + 0.7^2} \cdot \sqrt{0.3^2 + 0.6^2 + 0.8^2}} = 1 - \frac{0.54 + 1.09 - 0.21}{2 \cdot 0.73 \cdot 1.04} = 0.074.$$

6. Кореляційна відстань.

$$d_{rs} = 1 - \frac{\sum_{j=1}^{n}\left(x_{rj} - \overline{x}_{rj}\right)\left(x_{sj} - \overline{x}_{sj}\right)}{\sqrt{\sum_{j=1}^{n}\left(x_{rj} - \overline{x}_{rj}\right)^2}\sqrt{\sum_{j=1}^{n}\left(x_{sj} - \overline{x}_{sj}\right)^2}}.$$

де $\overline{x}_r = \frac{1}{n}\sum_{j=1}^{n} x_{rj}$, $\overline{x}_s = \frac{1}{n}\sum_{j=1}^{n} x_{sj}$.

Визначається як одиниця мінус вибірковий коефіцієнт кореляції між значеннями ознак багатовимірної випадкової величини. Вектори спостережень трактуються як вибірки.

Приклад:

$$\overline{x}_r = \frac{0.1 + 0.2 + 0.7}{3} = 0.333;$$

$$\overline{x}_s = \frac{0.3 + 0.6 + 0.8}{3} = 0.567;$$

$$x_r - \overline{x}_r = \left(-0.233 \quad -0.133 \quad 0.367\right);$$

$$x_s - \overline{x}_s = \left(-0.267 \quad 0.033 \quad 0.233\right);$$

$$d_{rs} = 1 - \frac{-0.233 \cdot \left(-0.267\right) + \left(-0.133\right) \cdot 0.033 + 0.367 \cdot 0.233}{\sqrt{\left(-0.233\right)^2 + \left(-0.133\right)^2 + 0.367^2}\sqrt{\left(-0.267\right)^2 + 0.033^2 + 0.233^2}} = 1 - \frac{0.143}{0.162} = 1 - 0.883 = 0.117.$$

7. Квадратна кореляційна відстань.

$$d_{rs} = 1 - cor\left(x_r, x_s\right)^2 = 1 - \left[\frac{\sum_{j=1}^{n} \left(x_{rj} - \overline{x}_{rj}\right)\left(x_{sj} - \overline{x}_{sj}\right)}{\sqrt{\sum_{j=1}^{n} \left(x_{rj} - \overline{x}_{rj}\right)^2} \sqrt{\sum_{j=1}^{n} \left(x_{sj} - \overline{x}_{sj}\right)^2}} \right]^2.$$

где $\overline{x}_r = \frac{1}{n}\sum_{j=1}^{n} x_{rj}$, $\overline{x}_s = \frac{1}{n}\sum_{j=1}^{n} x_{sj}$.

Визначається як одиниця мінус квадрат вибіркового коефіцієнта кореляції між значеннями ознак багатовимірної випадкової величини.

Приклад: $d_{rs} = 1 - 0.883^2 = 0.22.$

8. Відстань Хеммінга.

$$d_{rs} = \left(\frac{\#\left(x_{rj} \neq x_{sj}\right)}{n} \right).$$

Визначається як процент відмінних координат від їх загального числа.

Приклад:

x_r =("Синій", "Червоний", "Білий", "Фіолетовий", "Помаранчевий").

x_s =("Червоний", "Зелений", "Білий", "Синій", "Синій").

$d_{rs} = \frac{4}{5} = 0.8.$

ДОДАТОК Б

ВИМОГИ ДО ОФОРМЛЕННЯ ЗВІТУ З ЛАБОРАТОРНОЇ РОБОТИ

Звіт оформлюється на аркушах формату А4 і повинен мати:

– титульну сторінку з назвою лабораторної роботи, прізвищем студента, номером групи та номером варіанта для робіт, в яких завдання по варіантах;

– мету роботи;

– постановку завдання лабораторної роботи;

– код програми з докладними коментарями (для робіт № 1-3);

– скріншоти основних результатів виконання кожного пункту завдання;

– висновки по роботі.

Звіт оформлюєтьсґя відповідно до стандарту ДСТУ 3008 – 95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення.

ДОДАТОК В

КРИТЕРІЇ ОЦІНЮВАННЯ ЛАБОРАТОРНИХ РОБІТ

Максимальну оцінку виставляють за дотримання таких умов:

- повна відповідність звіту про виконання лабораторної роботи методичним рекомендаціям;
- володіння теоретичними відомостями, на яких базується предмет досліджень;
- загальна та професійна грамотність, лаконізм і логічна послідовність викладу матеріалу;
- відповідність оформлення звіту чинним стандартам (ДСТУ 3008 – 95. Документація. Звіти у сфері науки і техніки. Структура і правила оформлення) та наведеним вимогам.

Сергєєва Катерина Леонідівна

РОЗПІЗНАВАННЯ ОБРАЗІВ ТА ОБРОБКА ЗОБРАЖЕНЬ

МЕТОДИЧНІ РЕКОМЕНДАЦІЇ ДО ЛАБОРАТОРНИХ ЗАНЯТЬ З ДИСЦИПЛІНИ

для студентів спеціальностей
122 Комп'ютерні науки та інформаційні технології
(спеціалізація "Комп'ютерний еколого-економічний моніторинг"),
193 Геодезія та землеустрій
(спеціалізація "Геоінформаційні системи і технології")

Видано в редакції автора

Підписано до друку 20.10.2016. Формат 30×42/4.
Папір офсет. Ризографія. Ум. друк. арк. 3,9.
Обл.-вид. арк. 3,9. Тираж 20 пр. Зам. №___

Державний вищий навчальний заклад
«Національний гірничий університет».
49005, м. Дніпро, просп. Д. Яворницького, 19