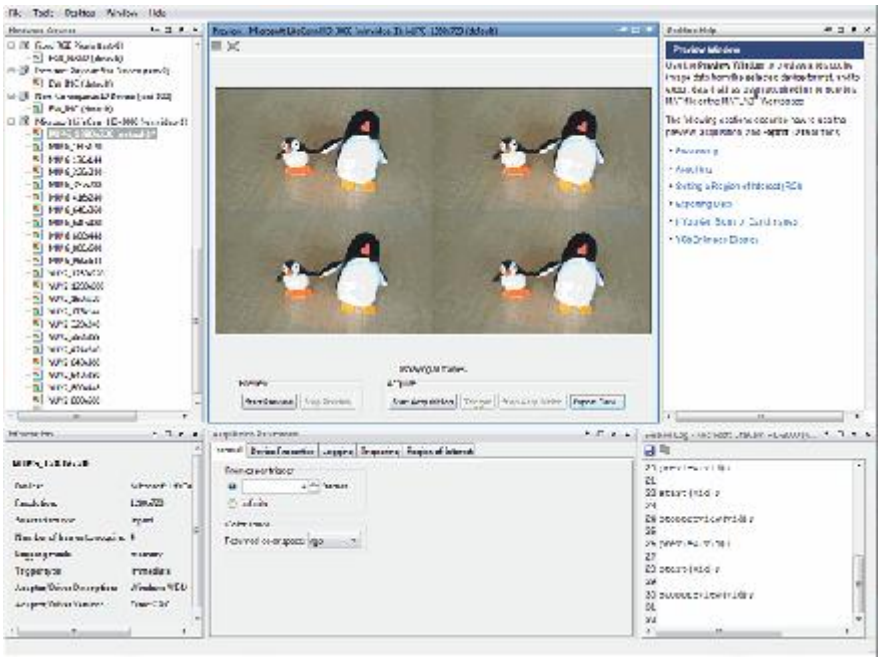
Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЧЕРНІГІВСЬКА ПОЛІТЕХНІКА»

МЕТОДИ ОБРОБКИ ІНФОРМАЦІЇ В СИСТЕМАХ ВІДЕОСПОСТЕРЕЖЕННЯ

Методичні вказівки до виконання лабораторних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення»



Затверджено
на засіданні кафедри електроніки, автоматики, робототехніки та мехатроніки
Протокол № 3 від 27.11.2020 р.

Чернігів – 2020

Методи обробки інформації в системах відеоспостереження. Методичні вказівки до виконання лабораторних робіт для студентів спеціальності 121 «Інженерія програмного забезпечення». – Чернігів: НУ «Чернігівська політехніка», 2020. – 90 с.

Укладач: Войтенко Володимир Павлович, канд. техн. наук, доц.

Відповідальний за випуск – Денисов Юрій Олександрович, докт. техн. наук, проф., завідувач кафедри електроніки, автоматики, робототехніки та мехатроніки

Рецензент: Ревко Анатолій Сергійович, канд. техн. наук, доц., доцент кафедри електроніки, автоматики, робототехніки та мехатроніки Чернігівського національного технологічного університету

Зміст

Перелік умовних скорочень

BMP – Bitmap Picture (формат зберігання растрових зображень).
DNG – Digital Negative Specification (відкритий формат для RAW файлів зображень).
GIF – Graphics Interchange Format (формат для обміну зображеннями).
IPT – Image Processing Toolbox (інструментарій MATLAB для обробки зображень).
JPEG – Joint Photographic Experts Group (назва розробника та один з популярних графічних форматів).
MIME – Multipurpose Internet Mail Extensions (багатоцільові розширення для інтернет-пошти).
PNG – Portable Network Graphics (растровий формат зберігання графічної інформації).
TIFF – Tagged Image File Format (формат зберігання растрових графічних зображень).

Вступ

Дисципліна «Методи обробки інформації в системах відеоспостереження» посідає важливе місце в циклі професійної підготовки бакалаврів спеціальності 121 «Інженерія програмного забезпечення», оскільки успішне засвоєння основ роботи з зображеннями (як статичними, так і динамічними) може стати підґрунтям для подальшої успішної діяльності випускника у таких напрямах суспільного інтересу, як національна, корпоративна та особиста безпека, медицина, інтертейнмент тощо. Суто теоретичне вивчення дисципліни не в змозі дати розуміння складних алгоритмів, застосовуваних під час роботи з зображеннями, тому навчальним планом передбачені також лабораторні заняття, під час яких студент може практично ознайомитися з особливостями методів обробки інформації в системах відеонагляду та набути певних практичних навичок із їхнього застосування.

Дані методичні вказівки призначені для самостійної підготовки студентів до виконання лабораторних робіт і присвячені вивченню та практичному дослідженню характерних особливостей пакетів *Image Processing Toolbox* та *Image Acquisition Toolbox* середовища *Matlab* від фірми *Mathwork* [1]. Більшість завдань до лабораторних робіт можна виконати у вільно розповсюджуваному *Octave* [2].

Процес підготовки до виконання завдань потребує обов’язкової самостійної роботи з рекомендованою літературою та лекційним матеріалом. Після теоретичної частини наведено список контрольних запитань, за допомогою яких оцінюється ступінь опанування теоретичного матеріалу і готовність студента до виконання лабораторної роботи. Далі наводиться перелік завдань, які потрібно вирішити в ході роботи, а також корисні рекомендації щодо виконання завдань.

Результати робіт оформлюються у вигляді звітів на аркушах формату *A4*. Звіт повинен містити таку інформацію:

1) короткі відомості про об’єкт вивчення (*15…20* рядків), в тому числі, – зміст завдання, схеми підключення пристроїв тощо;

2) рукописні відповіді на контрольні запитання;

3) схему, коментований текст програми (скрипта) і стислий опис алгоритму роботи;

4) висновки щодо отриманих результатів.

Пункти 1 та 2 оформлюються і показуються викладачеві на початку заняття, що є умовою допуску до виконання роботи в навчальній лабораторії. Весь звіт надається безпосередньо на захист роботи.

Найбільш ефективним методом створення звіту з лабораторних робіт є використання так званих «Живих скриптів». Про роботу з ними можна дізнатися, зокрема, у розділі допомоги «Create Live Scripts in the Live Editor» *MATLAB*.

1 Лабораторна робота №1. Дослідження основ обробки зображень в MATLAB

Мета роботи: практично дослідити базові можливості пакету розширення Image Processing Toolbox середовища MATLAB із створення, введення-виведення, відображення і обробки зображень.

1.1 Загальна характеристика пакету розширення Image Processing Toolbox середовища MATLAB

Система *MATLAB* і пакет прикладних програм *Image Processing Toolbox* (***IPT***) є корисним інструментом розробки, дослідження і моделювання методів і алгоритмів обробки зображень. Пакет *Image Processing* надає вченим, інженерам і навіть художникам широкий спектр засобів для цифрової обробки і аналізу зображень. Будучи тісно пов'язаним із середовищем розробки додатків *MATLAB*, пакет *Image Processing Toolbox* звільняє від виконання тривалих операцій кодування і налагодження алгоритмів, дозволяючи зосередити зусилля на вирішенні основної наукової або практичної задачі. Основні властивості пакета:

- відновлення і виділення деталей зображень;
- робота з виділеною ділянкою зображення;
- аналіз зображення;
- лінійна фільтрація;
- перетворення зображень;
- геометричні перетворення;
- збільшення контрастності важливих деталей;
- бінарні перетворення;
- обробка зображень і статистика;
- колірні перетворення;
- зміна палітри;
- перетворення типів зображень.

При вирішенні задач обробки зображень пакет *IPT* дозволяє йти двома шляхами. Перший з них полягає у самостійній програмній реалізації методів і алгоритмів. Інший шлях дозволяє моделювати рішення задачі за допомогою готових функцій, які реалізують найбільш відомі методи і алгоритми обробки зображень. І той, і інший спосіб виправданий. Однак для професіоналів в області обробки зображень кращим є другий шлях.

Це пояснюється гнучкістю таких програм, можливістю зміни всіх параметрів, що дуже актуально при дослідженні, розробці, визначенні параметрів регуляризації і т.д. Перш, ніж використовувати для вирішення будь-яких завдань обробки зображень стандартні функції пакета *IPT*, розробник повинен досконало їх дослідити. Для цього він повинен точно знати, який метод і з якими параметрами реалізує та чи інша функція.

Пакет *Image Processing* дає широкі можливості для створення і аналізу графічних зображень в середовищі *MATLAB*. Цей пакет має надзвичайно гнучкий інтерфейс, що дозволяє маніпулювати зображеннями, інтерактив-

но розробляти графічні картини, візуалізувати набори даних і анотувати результати для технічних описів, доповідей і публікацій. Гнучкість, поєднання алгоритмів пакета з такою особливістю *MATLAB*, як матрично-векторний опис, роблять пакет дуже вдало пристосованим для вирішення практично будь-яких завдань з розробки і подання графіки.

Цей пакет дозволяє користувачеві витрачати значно менше часу і сил на створення стандартних графічних зображень і, таким чином, сконцентрувати зусилля на важливих деталях і особливостях зображень.

MATLAB і пакет *Image Processing* максимально пристосовані для розвитку, впровадження нових ідей і методів користувача. Для цього є набір сполучених пакетів, спрямованих на вирішення всіх можливих специфічних завдань, в тому числі, – і завдань у нетрадиційній постановці.

Пакет *Image Processing* на даний час інтенсивно використовується багатьма компаніями та університетами по всьому світу. Є дуже широке коло завдань, які користувачі вирішують за допомогою цього пакету, наприклад, – космічні дослідження, військові розробки, астрономія, медицина, біологія, робототехніка, матеріалознавство, генетика і т.д.

1.2 Основні відомості про графічну систему середовища MATLAB

Для багатьох досліджень і обчислювальних робіт важливим етапом є візуалізація даних, підтримка якої здійснюється в *MATLAB* за допомогою набору потужних графічних команд високого рівня і програмованого інтерфейсу дескрипторної графіки *Handle Graphics*. *Команди високорівневої графіки* реалізують побудову графіків кривих у двовимірному і тривимірному просторах, зображення поверхонь, малювання ліній рівня, гістограм, багатьох видів спеціалізованої графіки і анімацію. При цьому управління кольором, масштабування, нанесення підписів, маркування осей і т.д. здійснюється досить просто, а призначені за замовчуванням режими цілком задовільні для більшості випадків. Є можливість інтерактивного оформлення рисунків. *Об'єктно-орієнтована система Handle Graphics* надає доступ до всіх характеристик графічних об'єктів і дозволяє створювати нові графічні команди.

Криві, поверхні та інші графічні об'єкти будуються в *спеціальному графічному вікні* (*figure*). Перше звернення в сеансі до графічної команди автоматично викликає появу вікна, якому присвоюється перший номер. Щоб організувати нове вікно або перейти від одного вікна до іншого, потрібно вибрати пункт New Figure в меню File командного або графічного вікна. Для того, щоб перейти до наявного вікна з номером N або організувати нове вікно, достатньо в рядку введення ввести команду figure N. Явно викликана всередині програми функція figure ('Name', 'Figure name') дозволяє задати графічному вікну з ім'ям 'Figure name' ті параметри, які більш доцільні, виходячи з логіки подальших дій. Одночасно може бути відкрито кілька графічних вікон. В *MATLAB* для збереження рисунка є великий ви-

бір форматів графічних файлів, причому рисунок можна записати, використовуючи меню графічного вікна або виконавши команду в програмі.

Основна команда двовимірної графіки plot, яка будує графіки кривих з абсцисами x1, x2, ... і ординатами y1, y2, ..., має наступний формат: p1ot (x1, y1, s1, x2, y2, s2, ...). Найбільш типовий варіант її використання виглядає наступним чином: p1ot (x, y). Тут x і y – вектори однакової довжини, що задають відповідні координати точок, виведених на графік. За замовчуванням точки з'єднуються суцільними лініями синього кольору. Розширені можливості, представлені рядками s1, s2, ..., дозволяють задати тип, товщину і колір кривої, що малюється, а також форму і розмір маркера.

При побудові графіка вибір масштабу і побудова осей відбуваються автоматично, а для зміни масштабів застосовується команда axis. Щоб визначити інтервали зміни координат самостійно, потрібно виконати команду: axis ([xmin, xmax, ymin, ymax]). Тут числа xmin і xmax задають інтервал зміни горизонтальної координати (мінімальне і максимальне значення), а ymin і ymax, відповідно, – інтервал зміни вертикальної координати. Якщо потрібно зберегти автоматичне масштабування з якої-небудь осі, то в якості значення слід поставити ідентифікатор Inf (або -Inf). Характерні застосування команди axis наведені в таблиці 1.1.

Таблиця 1.1 – Характерні застосування команди axis

Команда	**Дія**
axis(axis)	Використання поточного масштабу для наступних графіків (фіксація поточних призначень)
axis(auto)	Відновлення режиму автоматичне масштабування
v = axis	Отримання вектора з поточними значеннями масштабу
axis('ij')	Розміщення початку відліку в лівому верхньому кутку (матрична система координат)
axis('xy')	Розміщення початку відліку в лівому нижньому кутку (декартова система координат)
axis off	Відключення позначення осей і насічок
axis on	Відновлення осей і насічок

Для корекції розмірів рисунка використовуються стилі масштабування, що дозволяють встановити однаковий масштаб за обома змінними і узгодити область виведення. Для вказування потрібного стилю STYLE потрібно виконати команду: axis STYLE. Стилі масштабування для команди axis наведені в таблиці 1.2.

Таблиця 1.2 – Стилі масштабування для команди axis

Ім'я	**Опис**
square	Область виводу – квадрат
normal	Масштабування за умовчанням. *MATLAB* вибирає діапазон і розмітку за власним алгоритмом
equal	Однаковий масштаб по осях

Ім'я	Опис
tight	Зображувані дані займають усю область виведення
auto	Вмикання автомасштабування
manual	Масштабування вручну

Функції системи дескрипторної графіки *Handle Graphics* дозволяють ефективно працювати з графічними об'єктами (лініями, поверхнями та іншими об'єктами). Основними тут є вихідні одинадцять ***примітивів***, перераховані в таблиці 1.3.

Таблиця 1.3 – Примітиви дескрипторної графіки

Об'єкт	Призначення
Root	Кореневий об'єкт, який відповідає розміру екрана і створюваний *MATLAB* автоматично на початку сеансу
Figure	Графічне вікно на екрані
Uicontrol	Інтерфейс (кнопки, прокрутки і т.д.)
Axes	Фізична область у вікні
Uimenu	Меню користувача
Image	Двовимірний образ, що задається масивом
Line	Базисний примітив лінії
Patch	Базисний примітив закрашеного багатокутника
Surface	Базисний примітив поверхні
Text	Рядки символів
Light	Джерела світла, що діють на об'єкти в межах області, яку визначену Axes

Наприклад, об'єкт Line потребує розмірів області відображення, які містяться в Axes. У свою чергу, для Axes потрібні дані про об'єкт Figure.

Кожен графічний об'єкт при його створенні отримує деяке значення (***дескриптор***), яке присвоюється ідентифікатору і надалі може бути використано для зміни властивостей об'єкта. Якщо графік сформований з декількох об'єктів, то кожен об'єкт характеризують своїм дескриптором.

Значення дескриптора для кореневого об'єкта дорівнює нулю, а для вікна (Figure) збігається з його номером. Для інших об'єктів дескриптори задаються дійсними числами, що містять інформацію, яка використовується *MATLAB*. Для використання дескриптора графічного об'єкта необхідно присвоїти результат виконання графічної команди деякій змінній. Ця змінна є дескриптором і надає доступ до властивостей графічного об'єкта. Інформацію про властивості графічного об'єкта за допомогою дескриптора дозволяє отримати функція get, а функція set служить для перевизначення властивостей об'єкта.

Щоб перевизначити властивість об'єкта за допомогою функції set, досить вказати кілька перших букв, які специфікують властивість об'єкта. Щоб отримати перелік можливих значень якої-небудь властивості Prop (англ. *Property*), треба виконати команду set(h, 'Prop').

У таблиці 1.4 наведені функції, що надають доступ до часто використовуваних дескрипторів. Значення дескрипторів можуть змінюватися залежно від платформи і сеансу. Ці функції можна використовувати в якості аргументів інших функцій. Наприклад, видалити графічне вікно можна за командою delete(gcf). Для очищення поточного вікна досить виконати команду clf, а для осей – cla.

Таблиця 1.4 – Функції доступу до дескрипторів

Ім'я	**Призначення**
gcf	Отримання дескриптора поточного вікна (figure)
gca	Отримання дескриптора осей (axes) для поточного вікна (figure)
gco	Отримання дескриптора поточного об'єкта для поточного вікна (figure)

Кожен рисунок (вікно Figure) має свою палітру, задану матрицею з трьома стовпцями і числом рядків, які відповідають кількості заданих кольорів. Три числа на одному рядку, які можуть набувати значень від 0 до 1, визначають, відповідно, інтенсивності червоного, зеленого і синього компонент (RGB).

Для завдання і зміни палітри використовується команда colormap. За замовчуванням стандартною є палітра jet, що містить 64 кольори. Щоб призначити нову палітру за допомогою масиву CC, потрібно виконати команду colormap(CC), а повернути стандартну палітру можна за допомогою команди colormap('default'). Звернення до команди colormap без параметрів виводить матрицю з поточною палітрою.

Використовуючи стандартні операції з масивами, можна приготувати свою палітру. У той же час є набір приготованих палітр, зокрема: hsv (веселка), hot (комбінації чорного, червоного, жовтого і білого), cool (фіолетово-блакитна палітра), summer (жовто-зелена палітра), bone (сіро-синя палітра), gray (відтінки сірого). При зверненні до графічної функції вказується параметр, що задає число кольорів. Якщо параметр не вказано, формується палітра з тим же числом кольорів, що і поточна. Наприклад, по команді colormap(gray(4)) сформується палітра з чотирьох відтінків сірого кольору.

Палітри необхідні при побудові графічних об'єктів: поверхонь за допомогою команд mesh, surf та ін., контурних рисунків (за допомогою сімейства команд contour), і т.д. Діапазон зміни кольора для поточної палітри можна звузити за допомогою команди caxis([cmin cmax]). Тут cmin і cmax визначають діапазон палітри (відповідно початковий і кінцевий кольори), який буде використовуватися для фарбування. Команда caxis без параметрів виводить поточні значення cmin і cmax, команда caxis('auto') дозволяє повернутися до автоматичного призначення кольорів.

1.3 Типи зображень і їх подання в MATLAB

Зображення бувають *векторними* і *растровими*. ***Векторним*** назива-

ється зображення, описане у вигляді набору графічних примітивів. ***Растрові*** ж зображення являють собою двовимірний масив, елементи якого (пікселі) містять інформацію про їх яскравість і колір. Зазвичай *елементом зображення* називається мінімальна деталь зображення, всередині якої яскравість і колір вважаються постійними, тобто всередині елемента нерівномірність яскравості і кольору вже не будуть відрізнятися оком. У цьому випадку інформація про зображення подається у вигляді значень характеристик кожної точки зображення – *пікселя* (*pixel* – ***pic****ture* ***el****ement*) – найменшої структурної одиниці зображення. В якості таких характеристик виступають яскравість і колір пікселя, а його порядковий номер в кадрі дає інформацію про його координату на екрані.

Піксель – найменший логічний елемент двовимірного цифрового зображення в растровій графіці. Піксель є неподільним об'єктом прямокутної, звичайно квадратної, або круглої форми, що має певний колір і яскравість. Растрове комп'ютерне зображення складається з пікселів, розташованих по рядках і стовпцях. У цифровій обробці використовуються зазвичай растрові зображення. Вони, в свою чергу, діляться на типи: бінарні (чорно-білі), напівтонові (у вигляді відтінків сірого, *grayscale*), палітрові (індексовані), повнокольорові.

Елементи ***бінарного зображення*** можуть приймати тільки два значення – 0 або 1. Природа походження таких зображень може бути найрізноманітнішою. Але в більшості випадків вони виходять в результаті обробки напівтонових, палітрових або повнокольорових зображень методами бінаризації з фіксованим або адаптивним порогом. Бінарні зображення мають ту перевагу, що вони дуже зручні при передачі даних.

Півтонове зображення складається з елементів, які можуть приймати одне із значень інтенсивності якого-небудь одного кольору. Зазвичай розглядається суміш трьох основних кольорів, і в цьому випадку зображення набирає вигляду відтінків сірого (*grayscale*). Інформація про яскравість елементів зображення більш пріоритетна, тому таке подання зображення найбільш поширене, і воно частіше застосовується в різних дослідженнях. У більшості випадків інформація про яскравість пікселя кодується 8 бітами, що визначає 256 рівнів яскравості.

У ***палітрових зображеннях*** значення пікселів є посиланням на клітинку карти кольорів (палітри). Палітра представляє собою двовимірний масив, в трьох стовпчиках якого розташовані інтенсивності колірних складових для певного індексу.

Елементи ***повнокольорових зображень*** безпосередньо зберігають інформацію про яскравості колірних складових (RGB). Для зберігання повнокольорових зображень потрібен тривимірний масив, третій індекс якого визначає номер кольору для пікселя, що визначається першими двома індексами.

Вибір типу зображення залежить від розв'язуваної задачі, від того, наскільки повно і без втрат потрібна інформація може бути представлена

даним типом зображення. Використання повнокольорових зображень вимагає великих обчислювальних витрат.

За замовчуванням *MATLAB* працює з числами подвійної точності (вісім байт для зберігання числа типу *double*), а для роботи з зображеннями і скорочення необхідної пам'яті реалізовано зберігання даних також у вигляді однобайтових цілих без знака (клас *uint8*). Залежно від типу зображення вони по-різному представляються в різних форматах. Інформація про це наведена в таблиці 1.5.

Таблиця 1.5 – Формати чисел для роботи з зображеннями

Тип зображення	***double***	***uint8***
Бінарне	0 та 1	0 та 1
Напівтонове	[0, 1]	[0, 255]
Палітрове	[1, розмір палітри], де 1 – перший рядок палітри	[0, 255], де 0 – індекс першого рядка палітри
Повнокольорове	[0, 1]	[0, 255]

Тип *uint8* спочатку не призначався для арифметичних операцій і не всі функції і команди можна з ним використовувати. Тому, якщо з зображенням необхідно провести якісь чисельні дії, спочатку потрібно перетворити масив за допомогою команди A = double (A) + l.

В *MATLAB* зображення являє собою таблицю чисел, де значення кожного елемента відповідає певному рівню квантування його енергетичної характеристики (*яскравості*). Це – так звана піксельна система координат. Вона застосовується в більшості функцій пакета *IPT*. Існує також просторова система координат, де зображення представляється безперервним числовим полем квадратів з одиничною величиною. Кількість квадратів збігається з числом пікселів. Значення інтенсивності елемента в центрі квадрата збігається зі значенням відповідного пікселя в піксельній системі координат. При вирішенні практичних завдань, пов'язаних з вимірюваннями реальних геометричних розмірів об'єктів на зображенні, зручно використовувати просторову систему координат, так як вона дозволяє враховувати роздільну здатність (кількість пікселів на метр).

1.4 Кольорові системи і їх перетворення

Існує кілька колірних систем, які використовуються для подання повно кольорової графічної інформації.

Перш за все, – це найбільш поширена система RGB (*red*, *green*, *blue*). Інша часто використовувана система HSV (*hue* – колірний тон, *saturation* – насиченість, *value* – яскравість). Система HSV відповідає особливостям людського ока краще, ніж система RGB. Тому система HSV відповідає особливостям підбору кольорів, які використовуються художниками. Близькою до системи HSV є система HLS (*hue, lightness, saturation*).

Колірній системі HLS подібна колірна система YIQ, яка використовується в стандартній для США, Японії і ряду інших країн телевізійній системі NTSC. В системі YIQ використовуються три складові колірного ко-

дування: Y – складова яскравості, I – колірний тон і Q – насиченість. Перетворення з колірної системи RGB в колірну систему YIQ здійснюється за допомогою наступного матричного виразу:

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

Зворотне перетворення з системи YIQ в колірну систему RGB виконується за допомогою формули:

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \cdot \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}.$$

Інші кольорові аналогові і телевізійні системи відрізняються від розглянутих вище вибором дещо інших коефіцієнтів в матрицях перетворень. Це відноситься і до колірної системи YCbCr, яка найбільш часто використовується в цифровому відео та є близькою до телевізійної системи SECAM, використовуваної в телемовленні Франції, СНД і ряду інших країн. Ця система має наступні компоненти кольору: Y – складова яскравості; Cb і Cr – складові кольорової різниці. Перетворення з системи RGB, у якій складові трьох кольорів лежать в інтервалі від 0 до 1, в систему YCbCr здійснюється у відповідності до наступного матричного виразу:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 125.553 & 24.996 \\ -37.797 & -74.203 & -112 \\ 112 & -96.786 & -18.214 \end{bmatrix} \cdot \begin{bmatrix} R \\ G \\ B \end{bmatrix}.$$

1.5 Базові можливості середовища MATLAB по роботі з зображеннями

Базовий набір функцій системи *MATLAB* має розвинені засоби для роботи з растровими об'єктами (*зображеннями*), включаючи підготовку растрових графічних зображень, запис їх у файл, зчитування з файлу зображень, створених іншими програмами. Ці команди представлені в таблиці 1.6.

Таблиця 1.6 – Команди для роботи з растровими об'єктами

Команда	Призначення
image	Виведення графічного образу
imfinfo	Інформація про графічний файл
imread	Читання зображення з графічного файлу
imwrite	Запис зображення у графічний файл

Для виведення на екран графічного об'єкта в базовому наборі засобів *MATLAB* існує функція image. Команда image(C) виводить двовимірний або тривимірний масив C, як графічний образ. Нехай розмір масиву є MxN або MxNx3, тоді число M визначає кількість прямокутників по горизонталі, а N

– по вертикалі. Якщо C – двовимірний масив, то кожен елемент C розглядається, як значення індексу для масиву, що визначає поточну палітру (команда colormap), і відповідний цьому елементу C прямокутник фарбується в цей колір. Цей спосіб завдання зображення називається *Indexed image* (індексовані зображення). У випадку тривимірного масиву C колір точки (m, n) визначають елементи C (m, n, 1: 3), що дають, відповідно, частки червоного, зеленого і синього кольорів. При такому способі побудови об'єкта image отримують зображення з числом кольорів до 16 мільйонів (truecolor image). В цьому випадку таблиця кольорів не використовується.

Звернення image(X, Y, C), де X і Y – вектори, визначає розміщення пікселя C (1,1) в точці з координатами {X (1), Y (1)} і пікселя C (M, N), відповідно, в точці {X (end), Y (end)}. За замовчуванням *MATLAB* масштабує виведене зображення, тому піксель зазвичай представляється у вигляді прямокутника. Щоб скасувати масштабування, потрібно явно вказати розміри.

Для запису растрового зображення (масив A) в файл FILE в графічному форматі TYPE застосовується команда imwrite(A, FILE, TYPE). Команда зчитування зображення з файлу FILE в разі індексованого зображення має наступний вигляд: [A, M] = imread (FILE, TYPE) або A = imread (FILE, TYPE). У масив A заносяться дані про зображення (кольори пікселів порядково), а масив M буде містити таблицю кольорів, якщо масив A – двовимірний (*indexed image*). Для зчитування зображення *truecolor image* досить одного вихідного параметра. При читанні і записі в якості TYPE виступають наступні графічні формати: jpg (jpeg), tif (tiff), bmp, png, pcx, hdf, pcx, xmd. Щоб дізнатися тип зображення в файлі, можна використовувати команду imfinfo(FILE).

За допомогою команди друку або меню графічного вікна можна роздрукувати зображення, записати рисунок у власному форматі (fig) і зберегти його в декількох графічних форматах.

Команда друку зазвичай має додаткові параметри, які вказують графічний формат (таблиця 1.7) і деякі особливості. Так, для додавання рисунка до існуючого файлу досить набрати: print -append file.

Таблиця 1.7 – Додаткові параметри команди друку

Ім'я	**Призначення**
print	Збереження рисунка у файлі
printout	Завдання параметрів команди print
prtsc	Друк рисунка

В MATLAB використовується багато графічних форматів (таблиця 1.8), повну інформацію про які можна отримати, звернувшись до довідки help print. Крім того, зберегти зображення в будь-якому форматі можна за допомогою пункту Export... з меню File.

Таблиця 1.8 – Графічні формати для друку

Параметр	Призначення
-dwin	Чорно-білий друк
-dwinc	Кольоровий друк
-dmeta	Збереження в буфері в форматі *emf*
-dbitmap	Збереження в буфері в форматі *bmp*
-dps	Чорно-білий друк (рисунок) в стандарті *PostScript* рівня 1
-dpsc	Кольоровий друк в стандарті *PostScript* рівня 1
-dps2	Чорно-білий друк в стандарті *PostScript* рівня 2
-dpsc2	Кольоровий друк в стандарті *PostScript* рівня 2
-deps	Чорно-білий друк в стандарті *Encapsulated PostScript* рівня 1
-deosc	Кольоровий друк в стандарті *Encapsulated PostScript* рівня 1
-deps2	Чорно-білий друк в стандарті *Encapsulated PostScript* рівня 2
-depsc2	Кольоровий друк в стандарті *Encapsulated PostScript* рівня 2
-dill	Графічний формат *Adobe Illustrator 88*
-djpeg<nn>	*jpeg*-формат, якість визначається параметром *nn* (за замовчуванням *nn* дорівнює 75)
-dtiff	Графічний формат *tiff*

Призначення можна змінити прямо в команді print, наприклад, для запису в форматі *eps* рисунка 2 (*Figure No.* 2) в файл *abc* досить виконати наступну команду: print -deps -f2 abc. Параметри сторінки можна визначити за допомогою команди orient TYPE з параметром TYPE, який приймає значення portrait (портрет), landscape (ландшафт), tall (друк на всю сторінку). Також можна явно вказати графічне вікно і параметр orient(N, TYPE). Того ж самого можна досягти за допомогою спеціального вікна *Page Setup*, що викликається при виборі однойменного пункту меню *File* графічного вікна.

1.6 Основні формати зберігання растрових зображень

BMP (*Bitmap Picture*) – формат зберігання растрових зображень. Спочатку формат міг зберігати тільки апаратно-залежні растри (англ. *Device Dependent Bitmap*, ***DDB***), але з розвитком технологій відображення графічних даних формат BMP став переважно зберігати апаратно-незалежні растри (англ. *Device Independent Bitmap*, ***DIB***). З форматом BMP працює величезна кількість програм, оскільки його підтримка інтегрована до операційних систем Windows і OS/2. Файли формату BMP можуть мати розширення .bmp, .dib і .rle. Крім того, дані цього формату включаються в двійкові файли ресурсів RES і в PE-файли (Portable Executable). Глибина кольору в даному форматі може бути від 1 до 48 біт на піксель, максимальні розміри зображення 65535×65535 пікселів. У форматі BMP є підтримка стиснення за алгоритмом RLE, однак існують формати з більш сильним стисненням, і через великий обсяг BMP рідко використовується в Інтернеті, де для стиснення без втрат використовуються PNG і старіший GIF.

BMP-файл складається з 4 частин:

1) заголовок файлу (BITMAPFILEHEADER);

2) заголовок зображення (BITMAPINFOHEADER, може бути відсутнім). BITMAPV4HEADER (Win95, NT4.0) BITMAPV5HEADER (Win98 / Me, 2000 / XP);

3) палітра (може бути відсутньою);

4) саме зображення.

Структура BITMAPFILEHEADER містить інформацію про тип, розмір і подання даних у файлі. Палітра може містити послідовність чотирибайтових полів за числом доступних кольорів (256 для 8-бітного зображення). Три молодших байти кожного поля визначають інтенсивність червоної, зеленої і синьої компоненти кольору, старший байт не використовується. Кожен піксель зображення описаний в такому випадку одним байтом, що містить номер поля палітри, в якому збережений колір цього пікселя. Якщо піксель зображення описується 16-бітовим числом, палітра може зберігати три двобайтових значення, кожне з яких визначає маску для вилучення з 16-бітного пікселя червоної, зеленої і синьої компонент кольору. Файл BMP може не містити палітри, якщо в ньому зберігається нестиснене повнокольорове зображення.

Дані зображення *bmp* – це послідовність пікселів, записаних в тому чи іншому вигляді. Пікселі зберігаються порядково, від низу до верху. Кожен рядок зображення доповнюється нулями до довжини, кратної чотирьом байтам. У *bmp* файлах з глибиною кольору 24 біта байти кольору кожного пікселя зберігаються в порядку BGR (Blue, Green, Red). В *bmp* файлах з глибиною кольору 32 біта байти кольору кожного пікселя зберігаються в порядку BGRA (Blue, Green, Red, Alpha). При кількості 1, 2, 4 або 8 біт на кожен піксель може використовуватися спеціальний індексований колір. У цьому випадку число, відповідне до кожного пікселя, вказує не на колір, а на номер кольору в палітрі. Завдяки використанню палітри є можливість адаптувати зображення до кольорів, присутніх на зображенні. У такому випадку зображення обмежено не заданими кольорами, а максимальною кількістю одночасно використовуваних кольорів.

JPEG (*Joint Photographic Experts Group* – назва розробника) – один з популярних графічних форматів, застосовуваний для зберігання фотозображень і подібних до них зображень. Файли даних JPEG зазвичай мають розширення .jpeg, .jfif, .jpg, .JPG, або .JPE. MIME-типом є image/jpeg. Алгоритм JPEG є алгоритмом стиснення даних з втратами. Файл JPEG містить послідовність маркерів, кожен з яких починається з байта 0xFF, що свідчить про початок маркера, і байта-ідентифікатора.

Алгоритм JPEG найбільшою мірою придатний для стиснення фотографій і картин, що містять реалістичні сцени з плавними переходами яскравості і кольору. Найбільшого поширення JPEG отримав в цифровій фотографії, а також для зберігання і передачі зображень з використанням мережі Інтернет. З іншого боку, JPEG малопридатний для стиснення крес-

лень, текстової та знакової графіки, де різкий контраст між сусідніми пікселями приводить до появи помітних артефактів. Такі зображення доцільно зберігати в форматах без втрат, таких як TIFF, GIF, PNG або RAW.

JPEG (як і інші методи спотворюючого стиснення) не підходить для стиснення зображень за багатоступінчастої обробки, оскільки спотворення зображення будуть вноситися кожен раз при збереженні проміжних результатів обробки. JPEG не повинен використовуватися і в тих випадках, коли неприпустимі навіть мінімальні втрати, наприклад, при стисненні астрономічних або медичних зображень. У таких випадках може бути рекомендований передбачений стандартом JPEG режим стиснення *Lossless JPEG* або стандарт стиснення *JPEG-LS*.

RAW (*сирий*) – формат даних, що містить необроблені (або оброблені в мінімальному ступені) дані, що дозволяє уникнути втрат інформації, і не має чіткої специфікації. У таких файлах міститься повна інформація про зберігаємий сигнал, і вона може бути нестиснутою, стиснутою без втрат, або стиснутою з втратами. В RAW-файлах цифрових фотоапаратів зазвичай містяться: "сирі" дані з матриці; метадані – ідентифікація камери; метадані – технічний опис умов зйомки; метадані – параметри обробки за замовчуванням; один або кілька варіантів стандартного графічного представлення ("превью", зазвичай JPEG середньої якості), оброблені за замовчуванням.

Як правило, зображення у форматі RAW можна отримати, використовуючи цифрові дзеркальні фотокамери або напівпрофесійні фотокамери з незмінною оптикою. Однак деякі фотокамери можуть створювати такі файли, перебуваючи в недокументованому відлагоджувальному режимі або зі зміненою прошивкою. У файл записується стан кожного елемента світлочутливої матриці. Дані RAW містять набагато більше інформації в порівнянні з JPEG. Як правило, такі файли мають розрядність 12 або 14 біт на піксель в порівнянні з 8 бітами у JPEG. Тому файли формату RAW мають більший обсяг, ніж JPEG, але менший, ніж, наприклад, TIFF.

RAW часто називають "цифровим негативом", тим самим підкреслюючи, що інформація в ньому не призначена безпосередньо для розглядання, вона одночасно надлишкова (по числу біт на піксель, наприклад) і недостатня (через дії масиву кольорових фільтрів); кінцевий нормально сприймаємий універсальний графічний файл – це "відбиток" з "негативу", умови створення якого (як і при друку в плівковій фотографії) можна змінювати, отримуючи з одного і того ж "негативу" різні відбитки. Обробка RAW-файлу дозволяє змінювати параметри кадру (такі, як експозиція, яскравість, контраст, баланс білого, контурна різкість, насиченість) безпосередньо перед конвертацією, так, якщо б їх робили перед зйомкою. Це дозволяє отримати кінцеве зображення, не втративши при цьому занадто затемнені або занадто освітлені ділянки одного кадру, наприклад, пейзажу з темним лісом і яскравим небом або контрастного об'єкта.

DNG (*Digital Negative Specification*) – відкритий формат для RAW

файлів зображень, використовуваний в цифровій фотографії. Розроблено компанією *Adobe* з метою створити стандартний формат для RAW файлів зображень замість безлічі різних форматів різних виробників фотокамер. DNG є розширенням формату TIFF 6.0 і сумісний зі стандартом TIFF-EP. Файл формату DNG може зберігати в собі одне головне зображення, кілька зображень меншої роздільної здатності для попереднього перегляду і метадані. Рекомендоване розширення для файлів .dng.

GIF (*Graphics Interchange Format*) – формат для обміну зображеннями) – формат зберігання графічних зображень. Формат GIF здатний зберігати стиснуті дані без втрати якості у форматі не більше 256 кольорів. Незалежний від апаратного забезпечення формат GIF був розроблений в 1987 році (GIF87a) фірмою *CompuServe* для передачі растрових зображень по мережах. У 1989 р. формат був модифікований (GIF89a), були додані підтримка прозорості і анімації. GIF використовує LZW-компресію, що дозволяє непогано стискати файли, в яких багато однорідних заливок (логотипи, написи, схеми).

Зображення в форматі GIF зберігається порядково, підтримується тільки формат з індексованою палітрою кольорів. Стандарт розроблявся для підтримки 256-кольорової палітри. Формат GIF підтримує анімаційні зображення. Фрагменти представляють собою послідовності декількох статичних кадрів, а також інформацію про те, скільки часу кожен кадр буде показаний на екрані. Анімація може бути закольцована, тоді після останнього кадру буде знову показаний перший і так далі.

TIFF (*Tagged Image File Format*) – формат зберігання растрових графічних зображень. Спочатку був розроблений компанією *Aldus* у співпраці з *Microsoft* для використання з ***PostScript***. TIFF став популярним форматом для зберігання зображень з великою глибиною кольору. Він використовується при скануванні, надсиланні факсимільних повідомлень, розпізнаванні тексту, в поліграфії, широко підтримується графічними додатками. TIFF був обраний в якості основного графічного формату операційної системи NeXTStep, і з неї підтримка цього формату перейшла в Mac OS X. Компанія-власник специфікацій *Aldus Corporation* згодом об'єдналася з *Adobe*, що володіє на даний час авторським правом на ці специфікації. Файли формату TIFF, як правило, мають розширення .tiff або .tif.

Структура формату гнучка і дозволяє зберігати зображення в режимі кольорів з палітрою, а також в різних колірних просторах: бінарному (двокольоровому, іноді неправильно званому чорно-білим); напівтоновому; з індексованою палітрою; RGB; CMYK; YCbCr; CIE Lab. Підтримуються режими 8, 16, 32 і 64 біт на канал у цілочисельному форматі, а також 32 і 64 біт на канал при поданні значення пікселя числами з плаваючою комою. TIFF є теговим форматом, що дозволяє різну неоднорідну інформацію в файлі розділяти за допомогою тегів.

Є можливість зберігати зображення у файлі формату TIFF із стисненням і без стиснення. Ступень стиснення залежить від особливостей са-

мого зображення, що зберігається, а також від використовуваного алгоритму. Формат TIFF дозволяє використовувати такі алгоритми стиснення: PackBits (RLE), Lempel-Ziv-Welch (LZW), LZ77, ZIP, JBIG, JPEG, CCITT Group 3, CCITT Group 4. При цьому JPEG є просто інкапсуляцією формату JPEG у формат TIFF. Формат TIFF дозволяє зберігати зображення, стиснуті за стандартом JPEG, без втрат даних (JPEG-LS).

Алгоритми CCITT Group 3 і 4 призначені для кодування бінарних растрових зображень. Спочатку вони були розроблені для мереж факсимільного зв'язку (тому іноді їх називають Fax 3, Fax 4). На даний момент вони також використовуються в поліграфії, системах цифрової картографії та географічних інформаційних системах. Алгоритм Group 3 нагадує RLE, так як кодує лінійні послідовності пікселів, а Group 4 – двовимірні поля пікселів.

PCX (*PCExchange*) – стандарт представлення графічної інформації. Використовувався графічною програмою *ZSoft PC Paintbrush* (однієї з перших популярних графічних програм) для MS-DOS компанії Microsoft, текстових редакторів і настільних видавничих систем типу *Microsoft Word* і *Ventura Publisher*. Не настільки популярний аналог BMP, хоча підтримується специфічними графічними редакторами, такими, як Adobe Photoshop, Corel Draw та ін. В даний час витіснений форматами, які підтримують краще стиснення: GIF, JPEG і PNG. Тип формату – растровий. Більшість файлів такого типу використовує стандартну палітру кольорів, але формат був розширений з розрахунку на зберігання 24-бітних зображень. PCX – апаратно-залежний формат. Призначений для зберігання інформації в файлі в такому ж вигляді, як і в відеоадаптері. Алгоритм такого стиснення дуже швидкий і займає невеликий обсяг пам'яті, проте не дуже ефективний, непрактичний для стиснення фотографій і більш детальної комп'ютерної графіки.

PNG (*Portable Network Graphics*) – растровий формат зберігання графічної інформації, що використовує стиснення без втрат за алгоритмом ***Deflate***. PNG був створений, як вільний формат для заміни GIF, тому в Інтернеті з'явився рекурсивний акронім "PNG's Not GIF". PNG підтримує три основних типи растрових зображень: півтонування (з глибиною кольору 16 біт), кольорові індексовані зображення (палітра 8 біт для кольору глибиною 24 біт), повнокольорові зображення (з глибиною кольору 48 біт). Формат PNG зберігає графічну інформацію в стислому вигляді. Причому це стиснення проводиться без втрат, на відміну, наприклад, від JPEG з втратами. PNG має наступні основні переваги перед GIF: практично необмежену кількість кольорів в зображенні (GIF використовує в кращому разі 8-бітний колір); опциональна підтримка альфа-каналу; можливість гамма-корекції; двовимірна черезстрочна розгортка; можливість розширення формату для користувача блоками (на цьому заснований, зокрема, APNG). PNG є хорошим форматом для редагування зображень, навіть для зберігання проміжних стадій редагування, оскільки відновлення і збереження

зображення проходять без втрат в якості. Також, на відміну, наприклад, від TIFF, специфікація PNG не дозволяє авторам реалізацій вибирати, які можливості вони збираються реалізувати. Тому будь-яке збережене зображення PNG може бути прочитано в будь-якому іншому додатку, що підтримує PNG.

1.7 Виведення зображення на екран засобами Image Processing Toolbox

Для виведення зображення на екран існує спеціальна функція imshow зі складу пакета *Image Processing Toolbox*. Функція imshow(I,n) виводить на екран півтонове зображення I, використовуючи при виведенні n рівнів сірого. Якщо при виконанні функції опустити параметр n, то, коли *MATLAB* запущений в графічному режимі *TrueColor*, для виведення півтонового зображення використовується 256 градацій сірого або 64 градації сірого, коли *MATLAB* запущений в графічному режимі з меншою кількістю кольорів.

Функція imshow(I,[low high]) виводить на екран півтонове зображення I, додатково контрастуючи виведене зображення. Пікселі зображення I, яскравість яких менше або дорівнює low, відображаються чорним кольором. Пікселі, яскравість яких більше або дорівнює high, відображаються білим кольором. Пікселі, яскравість яких має значення між low і high, відображаються сірим кольором. Всі рівні сірого рівномірно розподілені від low до high. Якщо викликати функцію imshow(I, []), вказавши другим аргументом порожній масив, то low буде присвоєно мінімальне значення в I (low = min (I (:))), а high буде присвоєно максимальне значення в I (max = max (I (:))).

Функція imshow (BW) виводить на екран бінарне зображення BW. Пікселі, значення яких дорівнює 0, відображаються чорним кольором. Пікселі, значення яких дорівнює 1, відображаються білим кольором.

Функція imshow (X, map) виводить на екран палітровое зображення X з палітрою map.

Функція imshow (RGB) виводить па екран повнокольорове зображення RGB.

1.8 Контрольні запитання

1. В чому полягають основні переваги пакету *Image Processing Toolbox*?
2. Які основні задачі дозволяє вирішувати пакет *Image Processing Toolbox*?
3. Які концептуальні шляхи вирішення прикладних задач обробки зображень передбачає пакет *Image Processing Toolbox*?
4. В яких сферах наразі використовується пакет *Image Processing Toolbox*?
5. Яка основна команда виводу двовимірної графіки існує в *MATLAB*? Схарактеризуйте її формат і наведіть приклад використання.
6. Яким чином можна збільшити вдвічі масштаб графіка в *MATLAB*?

7. Що треба зробити для використання дескриптора графічного об'єкта?
8. Які функції за допомогою дескриптора дозволяють отримати інформацію про властивості графічного об'єкта?
9. Що таке палітра? Яким чином її можна задати та змінити?
10. Якими бувають типи зображень? Що таке елемент зображення? Які значення можуть приймати пікселі в різних видах зображень?
11. За допомогою якої команди потрібно перетворити масив, якщо з зображенням необхідно провести якісь чисельні дії?
12. Перерахуйте відомі Вам колірні системи, які використовуються для подання повнокольорової графічної інформації.
13. Перерахуйте базовий набір функцій системи *MATLAB* для роботи з растровими об'єктами (зображеннями).
14. Перерахуйте відомі Вам основні формати зберігання растрових зображень.
15. Дайте характеристику функції зі складу пакета *Image Processing Toolbox* для виведення зображення на екран. Наведіть приклади використання.

1.9 Хід роботи

1.9.1 Підготовчі стадії

Поза навчальною лабораторією треба зробити наступне:

1. Користуючись підрозділами 1.1 – 1.7, а також рекомендованою літературою, ознайомитися із пакетом розширення *Image Processing Toolbox* середовища *MATLAB*, складом та особливостями його функцій.

2. Оформити першу частину звіту з виконання лабораторної роботи, в якій зазначити прізвище студента, мету лабораторних досліджень та дати письмові відповіді на контрольні запитання (п. 1.8).

1.9.2 Завдання до роботи

Для того, щоб освоїти базові можливості середовища MATLAB з введення-виведення, відображення і обробки зображень, необхідно виконати наступну послідовність дій.

1. Згенерувати зображення відповідно до заданої функції розподілу. В якості такої використати гауссіану $F(x, y) = A \cdot e^{-(x^2+y^2)/2\sigma^2}$ з параметрами σ = 4, A = 256. Зображення згенерувати, як матрицю $\boldsymbol{M}$ розміру $N \times N$ (N = 200). Центр гауссіани (x = 0, y = 0) має бути в центрі зображення, а її область визначення: $x \in [-10,10]$, $y \in [-10,10]$.

1.1. Відобразити отриману матрицю, як індексоване зображення (що використовує палітру ‘Jet’) за допомогою базової функції image з *Image Processing Toolbox*.

1.2. Відобразити отриману матрицю, як зображення у вигляді відтінків сірого (*grayscale*) за допомогою функції subimage.

1.3. Відобразити отриману матрицю, як зображення у вигляді відтінків

сірого (*grayscale*) за допомогою функції imshow.

1.4. За допомогою функції imshow відобразити результат перетворення матриці як індексованого зображення для палітри 'hsv' у вигляді відтінків сірого.

1.5. За допомогою функції imshow відобразити результат перетворення матриці як індексованого зображення для палітри 'hot' в чорно-біле зображення.

1.6. За допомогою функції imshow відобразити кольорове RGB зображення, у якого червоний колір представлений значеннями матриці $\boldsymbol{M}$, зелений – матриці 256–$\boldsymbol{M}$ і синій – матриці $2^8 \cos^2 2\pi \ln M$.

Для порівняння всі зображення повинні бути відображені в одному графічному вікні. Кожне з зображень повинно мати відповідну назву.

2. Створити зображення 200×200 пікселів у вигляді трьох квадратів основних кольорів (розміру 100×100), зміщених один відносно іншого (рисунок 1.1). Відобразити отриману матрицю як зображення.

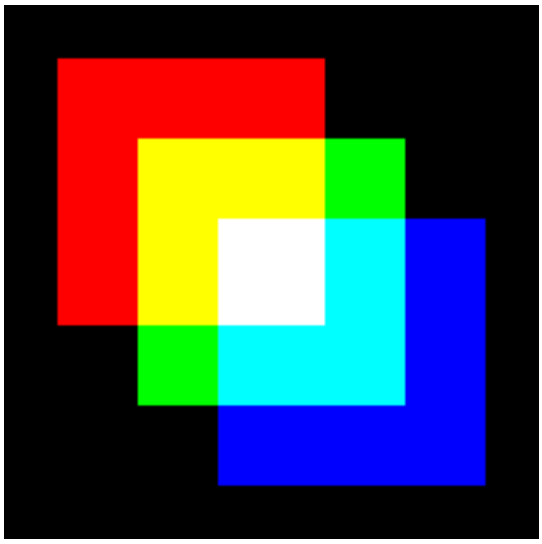


Рисунок 1.1 – Зображення у вигляді трьох квадратів

3. Ввести кольорове зображення з файлу із зображенням, яке можна однозначно асоціювати із автором лабораторної роботи, у вигляді інформації про три основні кольори (RGB-формат) в середовище *MATLAB*.

3.1. Відобразити введений файл засобами середовища *MATLAB* на екрані як зображення.

3.2. Перетворити вхідне зображення в зображення у вигляді інформації про відтінки сірого.

3.3. Перетворити вхідне зображення в зображення у вигляді чорно-білого зображення.

3.4. Відобразити розподіл інформації про червоний основний колір вхідного зображення як картинку з сірою шкалою.

3.5. Відобразити розподіл інформації про зелений основний колір вхідного зображення як картинку з сірою шкалою.

3.6. Відобразити розподіл інформації про синій основний колір вхідного зображення як картинку з сірою шкалою.

3.7. Вивести з файлу із зображенням інформацію про зображення і відобразити її в командному вікні *MATLAB*.

Відобразити отримані зображення в одному вікні поряд з вхідним кольоровим зображенням. Кожне з зображень повинно мати відповідну назву.

4. На базі вхідного кольорового зображення побудувати три нових, у кожного з яких інтенсивність однієї з трьох складових кольору більше істиної в 1,5 рази. Початкове зображення і три його модифікації відобразити поруч.

5. Створити зображення (200×200) у вигляді трьох зміщених по діагоналі одне щодо іншого кіл різних кольорів (рисунок 1.2). Діаметр одного кола повинен становити 100 пікселів, відстань між краєм зображення і кругом має бути не менше 20 пікселів. Зберегти отримане зображення у вигляді файлу *'Circles.tiff'* в форматі TIFF (*Tagged Image File Format*).

5.1. Виконати кадрування отриманого зображення, обрізавши його по краях (зверху, знизу, зліва і справа) на 10% (тобто прибрати поля).

5.2. Відобразити початкове і кадроване зображення поруч.

5.3. Зменшити обсяг пам'яті, що виділяється під кадроване зображення, шляхом проріджування (децимації) елементів матриці, що зберігає зображення, і відобразити зменшене зображення поряд з вхідним.

5.4. Збільшити розмір децимованого зображення в 2 рази і відобразити поруч з вхідним.

5.5. Повернути за допомогою операцій з матрицями кадроване зображення на 90º

5.6. Виконати поворот кадрованого зображення на 45º.

Усі отримані зображення відобразити поруч з вхідним. Кожне з зображень повинно мати відповідну назву.

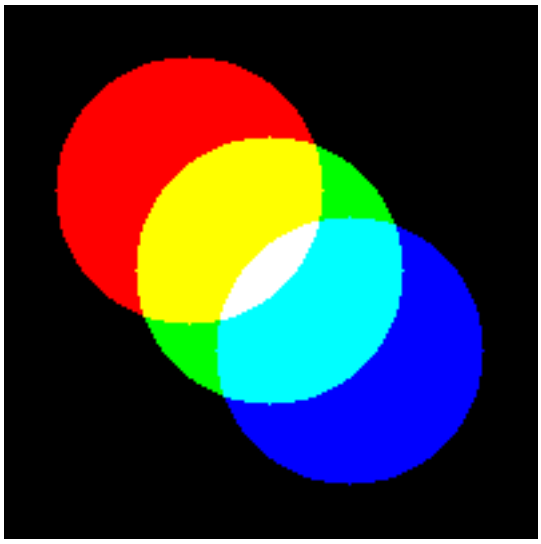


Рисунок 1.2 – Зображення у вигляді трьох кіл

1.9.3 Пояснення до виконання завдань

Для виконання всіх дій, зазначених у завданні, необхідно створити скрипт з назвою, що відповідає номеру роботи та завдання, наприклад 'Lab1_1.m', і розмістити його у відповідній папці. Для забезпечення можливості запуску скрипта за допомогою контекстного меню мишки рекомендується його перший рядок оформити як коментар із зазначенням імені скрипта:

```
%Lab1_1
```

Для забезпечення можливості багаторазового виконання скрипта протягом одного сеансу роботи необхідно в першому виконуваному рядку навести команду очищення робочої пам'яті ***Workspace*** і закриття всіх відкритих графічних вікон:

```
clear; close all
```

Перед виконанням **пункту 1** завдання необхідно попередньо визначити використовувані в подальшому параметри і задати значення змінної, що індексується, i:

```
s = 4;
A = 256;
N = 200;
i = 1:N;
```

Для обчислення двовимірного гаусовського розподілу можна скористатися попередньо розрахованим одновимірним гаусовським розподілом $f(x) = \exp(-x^2/2\sigma^2)$. Це дозволить згенерувати зображення, як матрицю наступного виду $M_{i,j} = A \cdot f(r_i) f(r_j)$, де r_i = 0,1 (i – N/2). Для цього в скрипті необхідно створити відповідні вектори-рядки:

```
r = 0.1*(i-N/2);
f = exp(-r.^2/2/s^2);
```

Щоб за допомогою вектора-рядка створити матрицю, необхідно її транспоновану копію (вектор-стовпець) помножити на цей вектор-рядок:

```
M = A * (f' * f);
```

Для відображення зображень рекомендується попередньо створювати *графічне вікно*, в якому будуть реалізовуватися операції з візуалізації зображень (див. "1.2 Основні відомості про графічну систему середовища *MATLAB*"). Ця дія не є обов'язковою, оскільки будь-яка функція *MATLAB*, що створює графічний образ, автоматично попередньо створює графічне вікно в якості контейнера для створюваного графічного образу, задаючи при цьому його параметри значеннями за замовчуванням. Явно викликана функція створення графічного вікна figure (...) дозволяє задати йому ті параметри, які є кращими для контексту конкретних дій.

Для реалізації можливості зіставлення поруч кількох варіантів відображення матриці ***M***, як зображення, необхідно відповідною функцією створити підобласть виведення графічного образу всередині графічного вікна.

Зображення в *MATLAB* може бути представлено різними способами (див. "1.3 Типи зображень і їх подання в *MATLAB*"). Для відображення матриці ***M***, як індексованого зображення, в якому значення кожного пікселя визначає номер кольору з деякої заздалегідь визначеної палітри кольорів, досить викликати базову функцію image (див. "1.5 Базові можливості середовища *MATLAB* по роботі з зображеннями").

У базовому варіанті виклику функції image матриця ***M*** буде розглядатися, як індексоване зображення, що використовує палітру за умовчанням ('Jet'). Для того, щоб пропорції створюваного зображення визначалися не пропорціями місця, виділеного під нього, а пропорціями самого зображення (а воно повинно бути квадратним, оскільки матриця ***M*** є квадратною), необхідно застосувати відповідну команду *з IPT*.

Для того, щоб відрізняти створене зображення від інших, необхідно йому дати заголовок, наприклад, 'image (M)' відповідною командою *з IPT*.

Щоб не вплинути на колірну палітру сформованого першого відображення матриці ***M***, наступне її відображення, як зображення у вигляді відтінків сірого (*grayscale*) в тому ж графічному вікні, слід виконувати за допомогою функції subimage зі складу *Image Processing Toolbox*. Її призначення – ізолювати використовувану при поточному формуванні зображення палітру від раніше використаної. Якщо поточне формування індексованого зображення в графічному вікні, в якому вже є раніше сформовані зображення, зробити за допомогою функцій image або imshow, використовуючи при цьому якусь нову палітру, то це призведе до зміни палітри і в раніше сформованих зображеннях. Функція subimage дозволяє формувати своє власне зображення без конфлікту з колірними схемами раніше сформованих зображень. Використання функції subimage, як і функції imshow, тільки з одним параметром у вигляді імені відображуємої матриці без посилання на необхідну палітру призведе до відображення матриці у вигляді відтінків сірого, коли значення кожного елемента матриці буде інтерпретуватися, як інформація про яскравість відповідного пікселя. Причому, якщо тип матриці буде *double*, то мінімальна яскравість відповідає нульовому значенню її елемента, а максимальна – одиничному. А, якщо тип матриці буде *uint8* або *uint16*, то яскравість буде визначатися цілочисельними значеннями в діапазоні від 0 до 255. Тому, для коректного відображення даної матриці ***M***, слід перетворити її до типу *uint8* за допомогою функції uint8. Застосування відповідних команд *з IPT* дозволить коректно відобразити матрицю ***M*** у вигляді відтінків сірого.

Щоб не виконувати перетворення типу матриці ***M*** при її відображенні у вигляді відтінків сірого, можна скористатися функцією imshow зі складу *Image Processing Toolbox* (див. "1.7 Виведення зображення на екран засобами Image Processing Toolbox"). Якщо її викликати у вигляді imshow (M, [low high]), то вона буде відображати у вигляді відтінків сірого значення матриці з діапазону [low high]. Якщо замість другого параметра викори-

стовувати порожню матрицю [], то це буде еквівалентно виклику даної функції у вигляді imshow (M, [min (M (:)) max (M (:))]), який забезпечує автоматичне масштабування відображуваних значень матриці ***M***. Врешті-решт застосування функції imshow, а також інших відповідних команд *з IPT* дозволить коректно відобразити матрицю ***M*** у вигляді відтінків сірого.

При виконанні функції imshow можна використати явне завдання параметру відображення InitialMagnification значення fit, що дозволить узгодити розмір зображення з розміром виділеної під нього області.

Для перетворення матриці ***M***, як індексованого зображення в зображення у вигляді відтінків сірого, слід використовувати функцію ind2gray, першим параметром якої повинна бути створена матриця (попередньо перетворена в тип *uint8*, щоб значення її елементів інтерпретувалися як відповідні індекси), а другим – використовувана палітра.

Для перетворення матриці ***M***, як індексованого зображення в чорно-біле зображення, слід використовувати функцію im2bw, першим параметром якої повинна бути матриця, перетворена в тип *uint8*, другим – використовувана палітра, третім – значення рівня в діапазоні від 0 до 1, що розділяє пікселі по яскравості на чорний чи білий.

Кольорове RGB зображення в *MATLAB* має бути представлено тривимірним масивом даних, у якого перші два індекси використовуються для адресації пікселя, а третій індекс визначає номер одного з трьох кольорів: 1 – червоний, 2 – зелений і 3 – синій. При наявності трьох матриць $\boldsymbol{M}_{\mathrm{R}}$, $\boldsymbol{M}_{\mathrm{G}}$ та $\boldsymbol{M}_{\mathrm{B}}$, що визначають інтенсивність колірних складових формуємого зображення, тривимірний масив зображення можна отримати за допомогою функції cat, реалізуючи її виклик у вигляді cat(3,M_{R},M_{G},M_{B}), що означає об'єднання трьох матриць за третьою розмірністю. Кольорові компоненти формованого зображення відповідно до завдання можуть бути отримані за допомогою таких виразів: $\boldsymbol{M}_{\mathrm{R}} = \boldsymbol{M}$, $\boldsymbol{M}_{\mathrm{G}} = 256 - \boldsymbol{M}$ и $\boldsymbol{M}_{\mathrm{B}} = 2^8 \cos^2 2\pi \ln M$. Це дозволить скласти фрагмент програми, що відображає кольорове RGB зображення.

Виконання **пункту 2** завдання доцільно почати з формування масиву *Squares*, який буде зберігати зображення трьох квадратів різного кольору. Оскільки, відповідно до рисунку 1.1, колір фону формованого зображення повинен бути чорним, то створюваний масив необхідно ініціалізувати нулями. Зробити це можна за допомогою функції zeros.

Для економії пам'яті сформований масив *Squares* рекомендується перетворити до типу *uint8*.

Розмір квадрата кожного кольору становить половину (50%) від розміру формованого зображення, тому діапазон індексів, що відводяться на кожен квадрат, можна задати індексованою змінною j = 0:N/2. Зсув відносно лівого і верхнього країв зображення, що реалізує зсув квадратів по діагоналі, можна вибрати такий: 10% для червоного квадрата, 25% для зеленого квадрата і 40% для синього квадрата. Задати його можна вектором

N0=[0.1 0.25 0.4]*N. Використовуючи ці змінні, квадрат кожного кольору можна визначити за допомогою присвоювання відповідним пікселям максимального значення інтенсивності певного кольору, наприклад, для червоного квадрату – за допомогою виразу Squares(N0(1)+j,N0(1)+j,1) = uint8(255). Всі три квадрати можна задати в циклі.

Відображення сформованого масиву Squares у вигляді зображення можна реалізувати так само, як при виконанні п. 1.

Для зчитування зображення з файлу в *MATLAB*, як того потребує **пункт 3** завдання, існує команда imread, а зчитування інформації про зображення – команда imfinfo (див. "1.5 Базові можливості середовища *MATLAB* по роботі з зображеннями").

Виведення вмісту структури Info у командне вікно *MATLAB* для перегляду інформації про зображення реалізується командою disp(Info), яка може бути використана для виведення на дисплей значення будь-якого об'єкта *MATLAB*. Однак структура Info може містити безліч порожніх полів, які не містять інформації. Це призведе до виводу на дисплей великого числа порожніх рядків, що ускладнює сприйняття інформації про зображення в цілому. Для очищення цієї структури від порожніх рядків (за допомогою команди rmfield для видалення поля і функції isempty для перевірки порожнечі поля) і, потім, виведення її вмісту на екран, можна скористатися навідповідними командами *IPT*.

Інформація про кольорове зображення зберігається в тривимірному масиві I, інформацію про який (розмір, кількість байт і т.п.) можна переглянути у вікні ***Workspace*** робочого столу *MATLAB*. Якщо в цьому вікні для даного масиву виконати команду Plot all columns / More Plots ... / image (для цього використовувати панель команд або контекстне меню), то масив відобразиться, як зображення в поточному графічному вікні. Інформація про червону складову вхідного кольорового зображення зберігається в двовимірній матриці I(:,:,1), зелену – в I(:,:,2), синю – в I(:,:,3).

Тривимірний масив, який зберігає інформацію про кольорове RGB зображення, може бути перетворений в двовимірну матрицю, що зберігає інформацію про яскравість пікселів у вигляді відтінків сірого, або в матрицю чорно-білого зображення (детальніше див. "1.3 Типи зображень і їх подання в MATLAB"). Перетворення кольорового RGB зображення в зображення у вигляді відтінків сірого виконується функцією rgb2gray, а в чорно-біле зображення – функцією im2bw(I,level), у якій другий параметр level визначає поріг бінаризації (вище level – 1, нижче level – 0).

Остаточно, відображення всіх отриманих варіантів початкового зображення може бути реалізовано так само, як і в попередніх пунктах.

Для того, щоб на базі вхідного кольорового зображення побудувати три нових, у кожного з яких інтенсивність однієї з трьох складових кольору більше істинної в 1,5 рази, як того вимагає **пункт 4** завдання, рекомендується скористатися функцією cat(dim,A,B), що об'єднує масиви A і B

вздовж розмірності dim.

Відображення всіх отриманих варіантів разом з вхідним зображенням може бути реалізовано так само, як і в попередніх пунктах.

Виконання **пункту 5** завдання, як і другого пункту, доцільно почати з формування тривимірного масиву Circles, який буде зберігати вихідне зображення трьох кіл різного кольору. Оскільки, відповідно до рисунку 1.2, колір фону формуємого зображення повинен бути чорним, то створюваний масив необхідно ініціалізувати нулями. Зробити це можна за допомогою функції zeros, перетворюючи при цьому сформований масив до типу *uint8* для економії пам'яті:

```
Circles = uint8(zeros(N,N,3));
```

Для формування кругової області в масиві Circles, заповненої одним числом, необхідно скористатися логічною операцією перевірки приналежності точки колу, виконуваної над операндами у вигляді матриць, які зберігають координати розглянутої області. Результатом такої операції буде матриця, заповнена нулями або одиницями, в залежності від результату операції ("брехня" або "істина") для даного матричного елемента. Радіус кола визначається значенням N/4, центр кола – значенням N/4+N0(k), де k – номер кольору. Сітку з координатами розглянутої області з початком відліку, розташованим в центрі кола, можна задати за допомогою спеціально призначеної для цього функції meshgrid і введеної раніше індексованої змінної i, яка зберегає діапазон значень координат: [x, y] = meshgrid(i - N/4-N0(k)). Для введених таким чином координат умова приналежності кругової області набуде вигляду: x.^2+y.^2<=(N/4)^2. Виходячи з цього, формування кругових областей різного кольору у вихідній матриці Circles можна реалізувати у вигляді циклу.

Оскільки передбачається поряд з отриманим зображенням зіставити його модифікації, графічне вікно, в якому буде відображатися зображення, слід поділити на 6 підобластей, а масив Circles відобразити в першій підобласті.

Щоб зберегти сформоване таким чином зображення в одному із стандартних графічних форматів (більш докладно про формат TIFF див. "1.6 Основні формати зберігання растрових зображень"), необхідно скористатися функцією imwrite (див. "1.5 Базові можливості середовища *MATLAB* по роботі з зображеннями").

Кадрування зображення в *Image Processing Toolbox* виконується функцією imcrop(I, rect), де I – зображення, яке обробляється, rect = [xmin ymin width height] – вектор, який визначає місце розташування і розмір кадруючого прямокутника. Тому, для того, щоб виконати кадрування отриманого зображення, обрізавши його по краях (зверху, знизу, зліва і справа) на 10%, необхідно реалізувати відповідний виклик цієї функції.

Децимація (*decimatio* – десять) – це зменшення частоти дискретизації дискретного в часі сигналу шляхом видалення його відліків. Вико-

ристання децимації дозволяє зменшити обсяг пам'яті для зберігання зображення (природно, – за рахунок огрубіння дискретного образу зображення). В *MATLAB* децимація реалізується дуже просто за допомогою перерахування тільки непарних значень перших двох індексів масиву Circles (конструкція n:m:k задає діапазон значень від n до k з кроком m, замість максимального значення індексу можна вказати зарезервоване ім'я end).

Збільшення розміру зображення передбачає вирішення завдання інтерполяції при вставці нових точок між вже існуючими. В *Image Processing Toolbox* для цього є спеціальна функція imresize(A, scale, method), яка створює нове зображення, змінюючи розміри вихідного зображення A в scale разів. Для зміни розмірів використовується один із зумовлених методів інтерполяції, який задається у вхідному параметрі method у вигляді одного з наступних рядків: 'nearest' (інтерполяція значеннями найближчих сусідів), 'bilinear' (інтерполяція за допомогою двох лінійних функцій), 'bicubic' (інтерполяція за допомогою двох кубічних функцій). Таким чином, збільшення розміру децімованого зображення в 2 рази можна зробити за допомогою відповідної команди.

Функція imrotate(A,angle,method) з *Image Processing Toolbox* здійснює поворот зображення, заданого в масиві A, на заданий кут angle проти годинникової стрілки, використовуючи один з перерахованих вище методів інтерполяції method. Виходячи з цього, можна реалізувати поворот кадрованого зображення на 45º.

При повороті за допомогою матричних операцій необхідно пам'ятати, що масив, який зберігає кольорове зображення, по суті, складається з трьох матриць, і вказаний поворот необхідно здійснити для кожної з них. Поворот на 90º можна здійснити за допомогою функції rot90(A) з базового набору *MATLAB*. Оскільки передбачається здійснювати поворот для кожної матриці кольорів окремо, то для зберігання результатів повороту необхідно створити масив, який буде зберігати результат повороту, а потім в циклі здійснити сам поворот.

Отримані нові зображення можна відобразити поруч з вхідним так само, як і в попередніх пунктах.

1.10 Вимоги до звіту по роботі

Звіт про виконання роботи повинен містити:

1. Прізвище та групу студента, а також назву, мету та завдання лабораторної роботи.

2. Письмові відповіді на контрольні питання.

3. Таблицю з переліком функцій за категоріями, які підтримуються пакетом розширення *Image Processing Toolbox*.

4. Текст розробленого скрипта з коментарями.

5. Скрін-шоти роботи розробленого скрипта.

6. Висновки, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно дозволяє (перерахувати по пунктах) ро-

бити з зображеннями пакет розширення *Image Processing Toolbox* середовища MATLAB?

1.11 Орієнтовні варіанти завдань

Варіанти *індивідуальних* завдань видаються викладачем після самостійного виконання студентом підготовчої стадії (п.1.9.1) та типового завдання (п.1.9.2).

2 Лабораторна робота №2. Дослідження можливостей обробки відеозображень в MATLAB

Мета роботи: експериментально дослідити можливості захоплення відеосигналу з web-камери і введення відео до MATLAB.

2.1 Загальна характеристика пакету розширення Image Acquisition Toolbox середовища MATLAB

Від якості вхідних даних залежить обсяг витягнутої з них інформації та достовірність прийнятих на цій основі рішень. Завдання захоплення і формування вихідних відеозображень передує обробці відеоданих. Для її вирішення може використовуватися ***Image Acquisition Toolbox***. Цей пакет розширення дозволяє безпосередньо підключати, налаштовувати і управляти засобами формування зображень і потокового відео.

Пакет розширення *Image Acquisition Toolbox* призначений для вирішення проблемно-орієнтованих завдань ***захоплення зображень***. Використовуючи функції цього пакета розширення, можна зв'язати в єдину систему *MATLAB* і ***пристрої*** захоплення зображень. Установка різних властивостей об'єктів дозволяє проводити контроль різноманітних аспектів процесу захоплення, таких як, наприклад, кількість захоплених даних.

Після підключення пристрою захоплення зображень його можна використовувати, як тригерну або пускову систему. В *Image Acquisition Toolbox* підтримуються певні типи запуску, при яких існує можливість контролю важливих параметрів захоплення, наприклад, синхронізація захоплення з зовнішніми пристроями.

Для обробки захоплених даних їх необхідно перенести до робочого простору *MATLAB*. Коли кадр захоплюється, *Image Acquisition Toolbox* запам'ятовує його в буфері пам'яті. Додаток забезпечує кілька шляхів передачі даних одного або декількох кадрів до робочого простору, де з’являється можливість їх обробки у вигляді багатовимірних масивів. Це дозволяє, наприклад, поліпшити захоплені зображення або використовувати їх для створення автоматизованих систем керування.

Пакет розширення *Image Acquisition Toolbox* підтримує адаптери, які працюють з професійними пристроями захоплення зображень, такими, як *Coreco Imaging, Data Translation, Matrox* і ін. Пакет розширення призначений для спільної роботи з цифровими камерами, що підтримують *DCAM*-специфікацію. Ця специфікація описує інтерфейс обміну даними з цифровою камерою по шині **IEEE 1394** (*FireWire*), яка часто використовується в наукових додатках. Крім того, *Image Acquisition Toolbox* за допомогою драйверів **WDM** (*Windows Driver Model*) і **VFW** (*Video for Windows*) підтримує пристрої захоплення зображень з інтерфейсами *USB* і *FireWire, web*-камери; цифрові (*DV*) відеокамери, ТВ-тюнери. Додаток може підтримувати також інші пристрої через адаптери, які були розроблені для цих пристроїв.

Пакет розширення *Image Acquisition Toolbox* являє собою набір функцій, які підтримують широкий діапазон операцій з захоплення зображень, включаючи:

- захоплення зображень з різного роду пристроїв, від професійних пристроїв захоплення до *web*-камер з *USB*-входом;
- перегляд потокового відео;
- управління захопленням (включаючи зовнішні пристрої захоплення);
- вибір потрібної конфігурації функцій;
- передача даних до робочого простору *MATLAB*.

Пакет розширення містить також інтерфейс з *Simulink* у вигляді блоків *Image Acquisition*. Вони служать для моделювання захоплення відеоданих і їх обробки в середовищі *Simulink*.

2.2 Основні процедури захоплення зображень в Image Acquisition Toolbox

Щоб використовувати *Image Acquisition Toolbox* для захоплення зображень, потрібно реалізувати дій, представлені в таблиці 2.1.

Таблиця 2.1 – Послідовність дій для захоплення зображень

Крок	**Дія**
1	Інсталяція та конфігурація пристроїв захоплення зображень
2	Отримання інформації, що ідентифікує пристрої захоплення зображень в *Image Acquisition Toolbox*
3	Створення об'єктів відеозахоплення
4	Попередній перегляд відеопотоку (необов'язково)
5	Формування властивостей об'єктів захопленого зображення (необов'язково)
6	Захоплення даних зображення
7	Очищення даних

2.2.1 Інсталяція пристрою захоплення зображень

Для інсталяції пристрою захоплення зображень слід скористатися інструкцією установки, яка містить наступні типові кроки:

- установка плати захоплення зображень в комп'ютер;
- установка необхідних програмних драйверів для даного пристрою (вони поставляються разом з пристроєм захоплення зображень);
- під'єднання камери до плати захоплення зображень;
- перевірка роботи камери, її характеристик, запуск відповідного програмного забезпечення і перегляд отриманої відеопослідовності.

Типові пристрої захоплення зображень, такі, як *web*-камери або портативні камери, не вимагають установки плати захоплення зображень. Ці пристрої підключаються прямо до комп'ютера через *USB* або *FireWire*-порт.

Після інсталяції і конфігурації пристроїв захоплення зображень запускається система *MATLAB*. Для захоплення зображень немає необхідності виконувати спеціальну конфігурацію *MATLAB*.

2.2.2 Отримання інформації про пристрої захоплення зображень

На цьому етапі необхідно отримати інформацію, потрібну *MATLAB* для однозначної ідентифікації встановленого пристрою захоплення зображень. Ці дані будуть використовуватися для створення *об'єктів відеозахоплення Image Acquisition Toolbox*. Для їх отримання може бути використана функція imaqhwinfo. Вид цих даних наведено в таблиці 2.2.

Таблиця 2.2 – Ідентифікація встановленого пристрою захоплення зображень

Назва адаптера (***Adaptor Name***)	Для адаптера є відповідне програмне забезпечення, яке використовується для з'єднання з пристроями захоплення зображень за допомогою відповідних драйверів. Додаток включає адаптери відомих пристроїв захоплення зображень.
ID пристрою (***device ID***)	Device ID являє собою число, що однозначно визначає кожен пристрій захоплення зображень і його підключення. *Примітка*: визначення device ID не є обов'язковим; додаток за замовчуванням використовує перші доступні ID.
Формат відео (***Video format***)	Формат відео визначає розділову здатність зображення, а також інші аспекти відеопотоку. Типові пристрої захоплення зображень підтримують різні формати відео. *Примітка*: Визначення формату відео не є обов'язковим; додаток за замовчуванням використовує один з підтримуваних форматів.

Назва адаптера. Для визначення назви адаптера слід виконати команду imaqhwinfo без аргументів. Функція imaqhwinfo повертає структуру з потрібними даними, серед яких, наприклад, в полі InstalledAdaptors розміщений список адаптерів, встановлених на комп'ютері. Зазвичай команда imaqhwinfo знаходить два адаптера, встановлених на комп'ютері: 'matrox' і 'winvideo'. Шляхом вибору відповідної назви вибирається той або інший пристрій захоплення зображень.

Якщо встановлених на комп'ютері адаптерів немає, з’явиться попередження:

Warning: No Image Acquisition adaptors found. Image acquisition adaptors may be available as downloadable support packages.
Open Support Package Installer to install additional vendors.

У цьому випадку треба відкрити інсталятор пакетів підтримки Matlab. В ньому буде запропонований вибір одного з кількох пакетів:

Select support package to install

Можливо, допоможе вибір

OS Generic Video Interface.

Після вибору Next зявиться вікно входу до вашого облікового запису MathWorks:

Please log in to your MathWorks account to continue the installation.
Click "Log In" to continue.

Якщо ви не маєте облікового запису MathWorks, доведеться пройти процедуру реєстрації.

Якщо пристрій захоплення встановити з першого разу не вдасться, доведеться поекспериментувати з пакетами для інших адаптерів.

ID пристрою. Для визначення параметра device ID окремих пристроїв захоплення зображень слід використовувати функцію imaqhwinfo, що містить назву адаптера в якості аргументу, наприклад, 'winvideo':

```
info = imaqhwinfo('winvideo')
```

В полі DeviceIDs структури info, яка повертається, міститься кількість усіх пристроїв, доступних через даний адаптер.

Для отримання додаткової інформації про пристрій слід використовувати поле DeviceInfo структури, яка визначає масив, що описує пристрій. Ця інформація є корисною, коли в списку DeviceIDs присутні більше одного пристрою. Для перегляду додаткової інформації про адаптер його DeviceID використовується в якості аргументу у функції imaqhwinfo разом з назвою адаптера.

```
DeviceInfo = imaqhwinfo('winvideo', 1)
```

Формат відео. Щоб визначити, який формат відео підтримує пристрій захоплення зображення, потрібно використовувати поле SupportedFormats в структурі DeviceInfo, яку повертатимуть функцією imaqhwinfo.

2.2.3 Створення об'єктів відеозахоплення

На цьому кроці створюються об'єкти вхідних відеопотоків (***об'єкти відеозахоплення***), які використовуються додатком для подання зв'язку між *MATLAB* і пристроєм захоплення зображення. Використовуючи властивості об'єктів відеозахоплення, можна контролювати багато аспектів процесу захоплення зображень.

Для створення об'єктів відеозахоплення призначена функція videoinput. Функція videoinput використовує такі параметри, як назва адаптера, ID пристрою і формат відео, які необхідно визначити на кроці 2. Назва адаптера є необхідним аргументом, інші параметри (ID пристрою і формат відео) функція videoinput може використовувати за замовчуванням:

```
vid = videoinput ( 'winvideo')
```

Для перегляду короткого списку об'єктів відеозахоплення слід використати змінну, що повертається vid. У ній містяться короткі характеристики об'єктів: кількість захоплених фреймів, тип захоплення і черговий стан об'єктів. Властивості об'єктів відеозахоплення можна використовува-

ти для управління іншими характеристиками.

2.2.4 Попередній перегляд відеопотоку

Після створення об'єктів відеозахоплення система *MATLAB* має доступ до пристроїв захоплення зображень і може почати використовувати захоплені дані. Однак спочатку можна провести попередній перегляд відеозображень, що формуються об'єктами відеозахоплення, і переконатися в прийнятності результату. Наприклад, можна змінити розміщення камери, змінити освітлення, відкоригувати фокус або зробити інші зміни в установках захоплення зображення. Цей крок не є обов'язковим, оскільки попередній перегляд можна проводити в будь-який час після створення вхідних об'єктів відеозахоплення.

Для попереднього перегляду потокового відео необхідно використовувати функцію preview і, як аргумент, поставити описання об'єктів відеозахоплення, які були визначені в попередньому кроці:

```
preview (vid)
```

Функція попереднього перегляду відкриває вікно і відображає розглянуту відеопослідовність. Коли вікно попереднього перегляду відкрито, тоді у властивостях встановлюється значення 'on'. Після зміни характеристик зображення шляхом установки властивостей захоплення зображень на видимій частині зображенні, у вікні попереднього перегляду будуть відображені ці зміни.

Для того щоб закрити вікно попереднього перегляду, слід використовувати кнопку Close в верхньому правому куті вікна або функцію closepreview. Як аргумент, для цієї функції необхідно вказати вхідні об'єкти відеопослідовності:

```
closepreview(vid)
```

2.2.5 Формування властивостей об'єктів захопленого зображення

Після створення об'єктів відеозахоплення і попереднього перегляду потокового відео існує можливість модифікації характеристик зображення або інших аспектів процесу захоплення. Цього можна досягти, змінюючи властивості об'єктів захоплення зображень.

Типи об'єктів захоплення зображень. У додатку використовуються два типи об'єктів, які служать для з'єднання з пристроями захоплення зображень: Video input objects, Video source objects. Об'єкти відеозахоплення (video input objects) представляють зв'язок *MATLAB* з пристроями захоплення зображень на високому рівні. Підтримувані властивості об'єктів відеозахоплення є однаковими для всіх типів пристроїв.

Коли об'єкти відеозахоплення створені, додаток автоматично формує один або кілька об'єктів відеоджерел (video source objects), пов'язаних з об'єктами відеозахоплення. Кожен об'єкт відеоджерела представляє набір фізичних даних, що відображають його суть. Кількість створюваних об'єк-

тів відеоджерел залежить від пристроїв та використовуваного формату відео. У будь-який момент часу тільки один з об'єктів відеоджерел може бути активним. Він використовується при захопленні.

Попередній перегляд властивостей об'єктів. Щоб переглянути список всіх властивостей, підтримуваних об'єктами відеозахоплення або об'єктами відеоджерел, використовується функція get. Список властивостей об'єктів відео створюється на кроці 3, і командою get(vid) він виводиться в командне вікно *MATLAB*. Функція get виводить список всіх властивостей об'єктів і їх поточні значення.

Для перегляду властивостей обраних об'єктів відеоджерел, пов'язаних з відповідними об'єктами відеозахоплення, разом з функцією get використовується функція getselectedsource. Функція getselectedsource повертає активне відеоджерело. Список властивостей обраних об'єктів відеоджерел, які пов'язані з об'єктами відеозахоплення, створюється на кроці 3 і використовується при виклику наступним чином: get(getselectedsource(vid)).

Установка властивостей об'єктів. Для установки значень властивостей об'єктів відеозахоплення або значень властивостей об'єктів відеоджерел можна використовувати функцію set або використовувати посилання на відповідне поле в структурі властивостей об'єкта. Деякі властивості можуть бути тільки прочитані без зміни їх значень. Ці властивості дають інформацію про стан об'єкта. Інші властивості можуть бути прочитані тільки в тому випадку, коли об'єкт запущений. Щоб переглянути список всіх властивостей, необхідно використовувати функцію set, описавши об'єкт в якості аргументу. Наприклад, для безперервного захоплення зображень в команді set властивості TriggerRepeat потрібно поставити значення Inf:

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

2.2.6 Захоплення даних зображення

Після створення об'єктів відеозахоплення і формування їх властивостей можна приступати безпосередньо до захоплення даних. Типові етапи захоплення зображень наступні:

1) старт об'єкта відеозахоплення;

2) запуск захоплення відеозображень;

3) передача захоплених даних в робочий простір *MATLAB*.

Старт об'єкта відеозахоплення. Для запуску об'єкта відеозахоплення необхідно використовувати функцію start. Вона готує об'єкт до захоплення відеозображень. Наприклад, після старту об'єкта всі його властивості фіксуються, як незмінні з атрибутом read only, і їх не можна змінити, не зупиняючи об'єкт. Але старт об'єкта не означає початок безпосереднього захоплення кадрів відеозображень. Ініціація процесу захоплення залежить від маніпуляцій з тригером.

Запуск захоплення відеозображень. Для прийому даних об'єкт відеозахоплення повинен запустити спусковий механізм (***тригер***). Перемикання може бути реалізовано кількома шляхами, в залежності від значення, встановленого у властивості TriggerType. Наприклад, якщо задати невідкладне включення (immediate trigger), об'єкт виконує тригер автоматично, відразу після запуску. Якщо вказати ручний запуск (manual trigger), об'єкт очікує виклику функції trigger перед початком прийому даних.

Передача захоплених даних в робочий простір MATLAB. Пакет розширення *Image Acquisition Toolbox* розміщує захоплені дані або в буфері пам'яті, або у файлі на диску, або там і там, в залежності від значення властивості LoggingMode. Для того щоб працювати з цими даними в середовищі *MATLAB*, їх необхідно туди перемістити. Для перенесення певного числа кадрів можна використовувати функцію getdata. Після перенесення даних в робочий простір *MATLAB* з ними можна виконувати будь-які дії, як з будь-якими іншими даними.

Приклад захоплення відеозображень. Приклад починається зі створення об'єкта відеозахоплення, його конфігурації і запуску функцією start. Оскільки в прикладі властивість TriggerType має за замовчуванням значення 'immediate', а властивості TriggerRepeat встановлено значення Inf, що означає автоматичний запуск тригера і безперервне сприйняття даних, приклад необхідно закінчити функцією stop. У прикладі описано кількість фреймів, які будуть захоплені. Кожна ітерація призводить до переміщення двох фреймів в робочий простір *MATLAB*. Для визначення руху в прикладі один фрейм віднімається з іншого, створюється різницеве зображення diff_im і потім візуалізується. Якщо пікселі в захоплених фреймах змінили своє значення, то на різницевому зображенні відповідні пікселі будуть мати відмінне від нуля значення.

```
vFormat = getfield( imaqhwinfo('winvideo', 'DeviceInfo'), 'SupportedFormats' );
vid = videoinput( 'winvideo', 1, char(vFormat(end)) );
set(vid, 'TriggerRepeat' ,Inf, 'FrameGrabInterval', 5, 'ReturnedColorSpace',
'grayscale');
figure('Name', 'Motion detection');
start(vid)
while(vid.FramesAcquired <= 100)
   data = getdata(vid, 2);
   diff_im = imabsdiff( data(:,:,:,1), data(:,:,:,2) );
   imshow(diff_im);
end
stop(vid)
```

Функція getdata переміщує 2 кадри (*фрейми*) з буфера пам'яті в робочий простір *MATLAB* і робить це регулярно в циклі, інакше пам'ять системи дуже швидко б переповнилася. У графічному вікні різницеве зображення diff_im має відображати наявність руху. В даному прикладі функція getdata повертає дані зображення у вигляді 4-мірного масиву в форматі *uint8*. Висота і ширина зображення залежить від розділової здатності та

формату відео. Тому при визначенні *області інтересу* (*region-of-interest*, ***ROI***) має бути використано властивість ROIPosition і враховано роздільову здатність.

2.2.7 Очищення даних

Після завершення процедури захоплення зображень дані з пам'яті переміщаються в робочий простір *MATLAB*, а процедуру треба завершити командами: delete(vid); clear; close(gcf).

2.3 Підключення пристроїв захоплення зображень до MATLAB

2.3.1 Отримання інформації про пристрої

Для під'єднання пристрою захоплення зображення до системи *MATLAB* необхідно створити ***об'єкт відеозахоплення*** (*video input object*). Цей об'єкт забезпечує зв'язок між системою *MATLAB* і пристроєм. Властивості цього об'єкта можна використовувати для контролю різних параметрів захоплення. Перед створенням об'єкта користувач повинен мати деяку інформацію про пристрій і спосіб його приєднання.

Для доступу до пристроїв захоплення зображень *MATLAB* вимагає деяку інформацію: назва адаптера для підключення пристрою захоплення зображень; ID пристрою для організації доступу (необов'язково); формат відеопотоку (необов'язково). Для отримання цієї інформації може використовуватися функція imaqhwinfo.

Адаптер являє собою програмне забезпечення, яке використовується для зв'язку з пристроями захоплення зображень через їх драйвери. Пакет розширення *Image Acquisition Toolbox* включає адаптери декількох виробників обладнання захоплення зображень і декількох спеціальних пристроїв захоплення. У таблиці 2.3 наведено список адаптерів, які включені в пакет розширення *Image Acquisition Toolbox*.

Таблиця 2.3 – Список адаптерів, які включені в пакет розширення *Image Acquisition Toolbox*

Назва адаптера	**Опис**
'coreco'	Адаптер для пристроїв захоплення зображень, розроблений фірмою *Coreco Imaging, Inc.*
'dcam'	Адаптер IEEE 1394 (*FireWire*) для пристроїв захоплення зображень, які працюють з *DCAM*-камерами, розроблений робочою групою Торгової асоціації інструментального та індустріального контролю.
'dt'	Адаптер для пристроїв захоплення зображень, розроблений фірмою *Data Translation, Inc.*
'matrox'	Адаптер для пристроїв захоплення зображень, розроблений фірмою *Matrox Electronic Systems, Ltd.*
'winvideo'	Адаптер для пристроїв, які забезпечуються *WDM*- або

	VFW-драйверами, включаючи пристрої з *USB* і *FireWire* інтерфейсом.

Для визначення того, який адаптер вбудований в систему, необхідно використати функцію imaqhwinfo. Функція imaqhwinfo без параметрів повертає інформацію (у вигляді структури) про використане програмне забезпечення і список вбудованих в системі адаптерів в полі InstalledAdaptors.

Адаптер ***встановлює ідентифікаційний номер*** (***ID***) кожному пристрою, з яким він пов'язаний. Адаптер встановлює для першого пристрою номер ID 1, для другого пристрою – ID 2 і т.д. Для пошуку ID індивідуальних пристроїв захоплення зображень використовується функція imaqhwinfo з назвою адаптера в якості аргументу:

```
info = imaqhwinfo('winvideo');
```

При використанні цього синтаксису функція imaqhwinfo повертає структуру, яка містить інформацію про всі пристрої, які доступні через цей адаптер. Поля в структурі info, яка повертається функцією imaqhwinfo, містять інформацію, опис якої представлено в таблиці 2.4.

Таблиця 2.4 – Опис полів в структурі info

Поле	Опис
AdaptorDllName	Текстовий рядок, що ідентифікує назву DLL адаптера.
AdaptorDllVersion	Інформація про версію DLL адаптера.
AdaptorName	Назва адаптера.
DeviceIDs	Масив, що містить ID всіх пристроїв, доступних через цей адаптер.
DeviceInfo	Структура з інформацією про пристрій.

Поле DeviceInfo структури info повертає більш детальну інформацію про пристрої захоплення, доступні через даний адаптер. Для її отримання можна використовувати функцію imaqhwinfo із зазначенням ID пристрою в якості другого аргументу:

```
dev_info = imaqhwinfo('winvideo', 1);
```

Поля в структурі dev_info містять інформацію про пристрої. Їх опис представлено в таблиці 2.5.

Таблиця 2.5 – Опис полів в структурі dev_info

Поле	Опис
DefaultFormat	Текстовий рядок, що визначає формат відео, використовуваний за замовчуванням.
DeviceFileSupported	Якщо прапор виставлений в 1, то пристрій підтримується конфігураційним файлом (відомим, як *camera file*); в інших випадках повинен бути встановлений 0.
DeviceName	Текстовий рядок, що визначається адаптером для ідентифікації пристрою.

DeviceID	ID пристрою, присвоєне йому адаптером.
ObjectConstructor	Синтаксис, який може використовуватися для створення об'єкта відеозахоплення, використовуваного середовищем *MATLAB* для подання пристрою.
SupportedFormats	Масив, який ідентифікує формати відео, підтримувані цим пристроєм.

Формат відео визначає характеристики зображень у відеопотоці, такі, як роздільна здатність (ширина і висота), розмірність і т.д. У більшості випадків пристрої захоплення зображень підтримують кілька форматів відео. Існує можливість задати формат відео, бажаний для використання, при створенні об'єкта відеозахоплення, що представляє зв'язок з пристроєм. Опис формату відео не є обов'язковим, оскільки в цьому випадку буде використовуватися один з підтримуваних форматів за замовчуванням. Формати відео, що підтримуються цим пристроєм захоплення зображень, зберігаються в полі SupportedFormats структури dev_info, яку повертатиме функція imaqhwinfo при використанні ID пристрою в якості другого аргументу.

2.3.2 Створення об'єктів захоплення зображень

Після отримання інформації про пристрої захоплення зображень можна встановити зв'язок з пристроєм за допомогою створення об'єктів захоплення зображень. Використовується два типи об'єктів захоплення зображень: *об'єкти відеозахоплення* (video input objects) і *об'єкти відеоджерела* (video source objects).

Об'єкти відеозахоплення представляють зв'язок між системою *MATLAB* і пристроями відеозахоплення на високому рівні. Вони створюються за допомогою функції videoinput.

При створенні об'єкта відеозахоплення автоматично створюється ***об'єкт відеоджерела***, який пов'язаний з об'єктом вхідного відеосигналу. Кожен об'єкт відеоджерела представляє набір з одного або декількох фізичних джерел даних для обробки їх, як єдиного цілого. Кількість об'єктів відеоджерела (їх можна розуміти, як канали захоплюваного відео) визначається пристроєм захоплення і його форматами відео.

Однак у даний конкретний момент часу тільки один об'єкт відеоджерела може бути активним. Це джерело і використовується для захоплення. Додаток вибирає один об'єкт відеоджерела для використання в режимі "за замовчуванням". Проте цей об'єкт може бути змінений.

Для створення об'єкта відеозахоплення в функції videoinput вказується назва адаптера, ID пристрою і формат відео. Необхідним аргументом є назва адаптера (наприклад, 'winvideo'). Для інших аргументів можуть використовуватися значення за замовчуванням:

```
vid = videoinput('winvideo');
```

Для того, щоб отримати більш докладні характеристики об'єкта відеозахоплення, необхідно їх вивести в командному вікні *MATLAB* за допомогою disp(vid).

При створенні об'єктів відеозахоплення в якості третього аргументу у функції videoinput можна вказувати опис формату. Цей аргумент може представлятися у вигляді двох форм:

- ❖ текстовий рядок, що описує формат відео;
- ❖ ім'я файлу конфігурації пристрою. Також відомий, як файл камери.

Якщо формат відео не описаний, функція videoinput використовує один з форматів відео за замовчуванням, підтримуваних цим пристроєм. Наприклад, для пристроїв *Matrox* і *Data Translation* вибирається формат відео 'RS170'. Для *Windows*-пристроїв використовується формат RGB з найменшою роздільною здатністю зі списку підтримуваних форматів. Якщо ж формат RGB не підтримується, то використовується інший формат, який встановлений за замовчуванням.

При створенні об'єкта відеозахоплення *MATLAB* створює один або більше об'єктів відеоджерел, які пов'язані з об'єктом відеозахоплення. Кількість створених об'єктів відеоджерел залежить від можливостей пристрою захоплення та від використовуваних форматів відео. У властивостях об'єкта відеозахоплення перераховані його об'єкти відеоджерела. Їх список також можна отримати за допомогою команди get(vid, 'Source'). За замовчуванням об'єкт відеозахоплення робить активним (обраним) перший об'єкт відеоджерела з усього списку об'єктів відеоджерел, створених для даного об'єкта відеозахоплення. Для використання іншого відеоджерела, як активного, потрібно змінити значення властивостей SelectedSourceName. Функція getselectedsource повертає об'єкти відеоджерела, які обрані (активізовані) на момент виклику функції.

2.3.3 Конфігурація об'єктів захоплення зображень

Об'єкти відеозахоплення і об'єкти відеоджерела визначаються їх властивостями, які дають можливість контролювати і змінювати характеристики їх роботи і захопленого відеозображення.

Властивості об'єктів відеозахоплення контролюють різні аспекти захоплення, які є загальними для всіх пристроїв захоплення зображень. Наприклад, можна використовувати властивість FramesPerTrigger для опису кількості даних, які необхідно захопити за один раз спрацювання пристрою захоплення.

Властивості об'єктів відеоджерела контролюють аспекти захоплення, які пов'язані з окремими відеоджерелами. Установка властивостей, підтримуваних об'єктами відеоджерела, різна для кожного пристрою захоплення. Наприклад, деякі пристрої захоплення зображень підтримують властивості, які дають можливість контролювати якість елементів зображення, наприклад, яскравість, колір, насиченість.

Для конфігурації будь-яких об'єктів в *Image Acquisition Toolbox* передбачається виконання такого порядку дій:

- ✓ перегляд списку всіх властивостей, підтримуваних даним об'єктом, і їх значень;
- ✓ більш детальний перегляд значень окремих властивостей;
- ✓ отримання інформації про властивості командою get;
- ✓ установки значень властивостей командою set.

Для перегляду всіх властивостей об'єктів захоплення зображень і їх поточних значень може бути використана функція get. Також можна використовувати іншу функцію для перегляду списку властивостей об'єктів у вікні *Property Inspector*, де можна також редагувати встановлені значення. Для перегляду властивостей, підтримуваних об'єктом відеоджерела, створеного об'єктом відеозахоплення, використовується функція getselectedsource. Для перегляду значень властивостей об'єктів захоплення зображень використовується функція get з ім'ям властивості як аргумент, наприклад, get(vid, 'Previewing'). Також існує можливість безпосереднього виведення значення властивості об'єкта, наприклад, disp(vid.Previewing).

Для отримання інформації про певні властивості можна скористатися довідковою інформацією *Image Acquisition Toolbox* в розділі *Property Reference*. Також існує можливість отримати інформацію про деякі властивості в командному рядку шляхом використання функцій propinfo або imaqhelp.

Функція propinfo повертає структуру, яка містить інформацію про такі властивості, як тип даних, значення, що встановлюються за замовчуванням і список усіх варіантів значень, які підтримуються даними властивостями. Наприклад, для отримання інформації про властивості LoggingMode слід реалізувати виклик propinfo(vid, 'LoggingMode').

Для установки значення властивості об'єкта захоплення зображень може бути використана функція set із зазначенням найменування властивості, як аргументу. Також існує можливість безпосереднього присвоєння значення властивості, використовуючи для цього присвоювання однойменному полю структури *MATLAB*. Наприклад, для установки значення властивості LoggingMode, можна або реалізувати виклик функції set(vid, 'LoggingMode', 'disk&memory'), або використовувати оператор присвоювання vid.LoggingMode = 'disk'. Оскільки деякі властивості відеооб'єктів доступні тільки для читання, встановити значення властивостей можна не для всіх, а тільки для певної їх підмножини.

Щоб переглянути список всіх властивостей об'єкта відеозахоплення або об'єкта відеоджерела, які можуть бути встановлені, можна скористатися наступним викликом функції: set(vid).

Значення, які визначають властивості запуску TriggerType, TriggerCondition і TriggerSource, є взаємопов'язаними. Наприклад, деяке значення TriggerCondition є дійсним при відповідному описі властивості

TriggerType. Для того щоб застрахуватися, що встановлена комбінація значень матиме сенс, необхідно використовувати дві функції: функцію triggerinfo, яка повертає всі дійсні комбінації значень для опису властивостей відеооб'єктів, і функцію triggerconfig, яка встановлює значення цих властивостей.

2.3.4 Запуск і зупинка об'єкта захоплення зображень

Створення об'єкта відеозахоплення встановлює зв'язок між системою *MATLAB* і пристроєм захоплення зображень. Однак перш, ніж робити захоплення даних з пристрою, необхідно запустити об'єкт за допомогою функції start:

```
start (vid);
```

При запуску об'єкта пристрій резервується для використання відповідно до встановленої конфігурації параметрів, причому значення деяких властивостей можуть бути прочитані тільки при запуску.

Об'єкт захоплення зображення буде *зупинений* при виконанні однієї з перерахованих нижче умов:

- запитувана кількість фреймів вже захоплена, що визначається умовою

```
FramesAcquired = FramesPerTrigger * (TriggerRepeat + 1);
```

де параметри FramesAcquired, FramesPerTrigger і TriggerRepeat є властивостями об'єкта відеозахоплення;

- виникла помилка періоду виконання;
- закінчився час захоплення;
- була використана функція stop.

Коли об'єкт запущений, властивості об'єкта Running встановлюється значення 'on'. Якщо об'єкт не запущений, то ця властивість має мати значення 'off'. Цей стан називається *зупиненим*. На рисунку 2.1 показано значення властивостей об'єктів в залежності від стану запуску або зупинки.

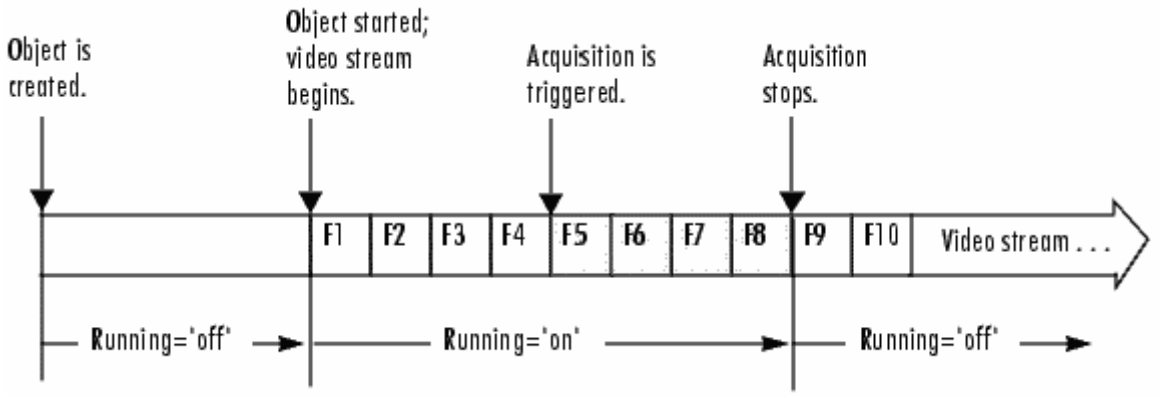


Рисунок 2.1 – Перехід об'єкта захоплення зі стану запуску в стан зупинки

Розглянемо приклад захоплення зображень за допомогою *web*-камери. Спочатку створюється об'єкт відеозахоплення, який буде пов'язаний з web-камерою:

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

Потім, якщо виконати перевірку поточного стану об'єкта відеозахоплення за допомогою виклику функції isrunning(vid), вона повинна повернути нульове значення, оскільки об'єкт ще не запущений. Для проведення аналізу стану об'єкта відеозахоплення його властивість TriggerType встановимо в стан 'Manual'. Для цього можна скористатися функцією triggerconfig:

```
triggerconfig(vid, 'Manual');
```

Для того щоб захоплення зайняло кілька секунд (щоб можна було встигнути поспостерігати за роботою пристрою захоплення в його запущеному стані), слід встановити:

```
vid.FramesPerTrigger = 100;
```

Потім слід виконати запуск об'єкта захоплення зображень відповідною однойменною функцією:

```
start(vid);
```

Тепер при перевірці стану запущеного об'єкта виклик функції isrunning(vid) буде повертати 1, а виклик функції islogging(vid) буде повертати 0. Ручний (manual) запуск здійснюється за допомогою функції trigger:

```
trigger(vid);
```

Тепер, якщо перевірити стан об'єкта захоплення, виклик функції islogging(vid) поверне 1, а після захоплення заданого числа кадрів, коли процес захоплення завершиться, і об'єкт відеозахоплення зупиниться, виклик функції isrunning(vid) буде знову повертати 0.

При завершенні використання об'єктів захоплення зображень існує можливість їх видалення за допомогою функції delete. Після видалення об'єктів відеозахоплення для очищення робочого простору *MATLAB* від змінних, які пов'язані з цими об'єктами, слід використовувати функцію clear.

Тобто робота по захопленню зображень в *MATLAB* повинна бути завершена видаленням об'єкта відеозахоплення і очищенням пам'яті.

```
delete(vid);
clear vid;
```

2.4 Управління захопленням зображень в MATLAB

Ключовим моментом будь-якого захоплення зображень є отримання даних від відповідного пристрою. ***Вмикання*** (*trigger*) є подією, яке ініціалізує захоплення фреймів зображення. Цей процес називається записом відеоданих (*logging*). Вмикання пристрою захоплення відбувається за певних умов. Для деяких типів тригера умова вмикання може бути визначена функціями *Image Acquisition Toolbox*. Для інших типів тригера (включення) такою умовою може бути сигнал із зовнішнього джерела, який контролює пристрій захоплення зображень.

2.4.1 Процес запису відео

Коли відбувається вмикання пристрою захоплення, *Image Acquisition Toolbox* встановлює властивість об'єкта відеозахоплення Logging в стан '*on*' і запускає розміщення захоплених кадрів в буфері пам'яті, у файлі на диску або одночасно і в пам'яті, і на диску. При зупиненні захоплення *Image Acquisition Toolbox* встановлює цю властивість у стан '*off*'. Коли об'єкти захоплення знаходяться в запущеному стані (*running state*) і коли переходять в стан запису (*logging stat*e) демонструє рисунок 2.2.

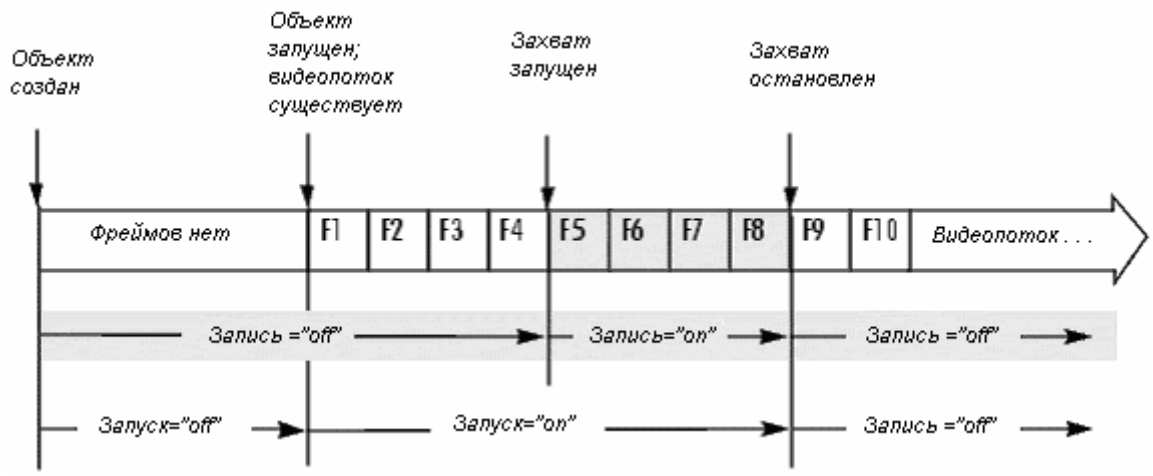


Рисунок 2.2 – Перехід до стану запису

Після установки властивості Logging в стан '*off*' не виключено, що об'єкти відеозахоплення все ще будуть записувати дані на диск. Для визначення закінчення процесу запису захоплених даних на диск необхідно використовувати властивість об'єкта відеозахоплення DiskLoggerFrameCount. На рисунку 2.3 показаний процес захоплення групи фреймів з відеопотоку і їх подальший запис на диск.

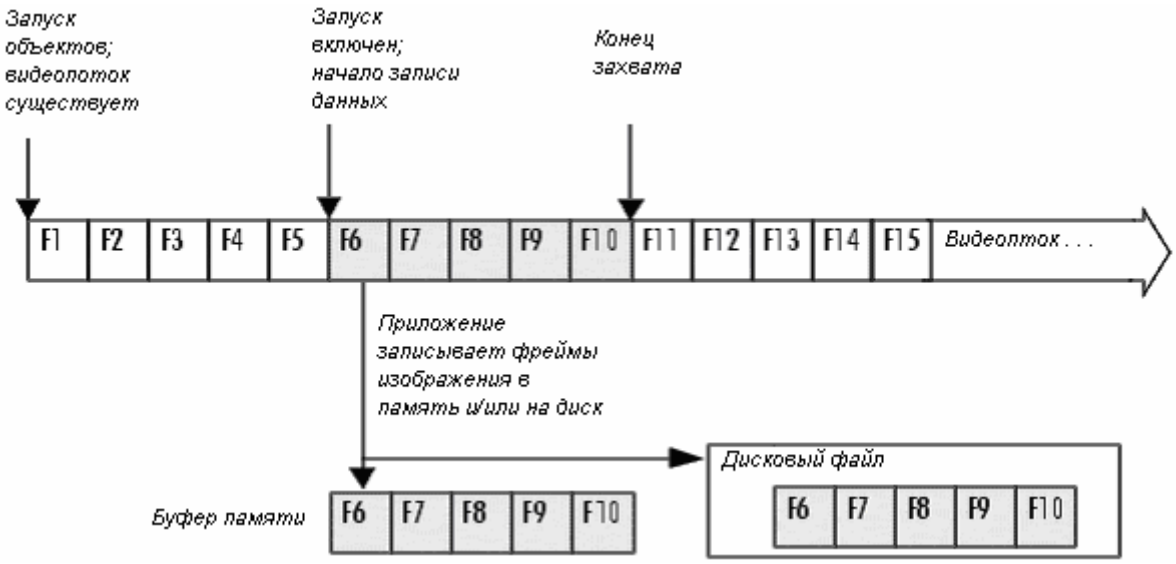


Рисунок 2.3 – Процес захоплення кадрів з відеопотоку та їх запис на диск

2.4.2 Властивості тригерів і керування ними

Об'єкти відеозахоплення підтримують властивості, які використовуються для управління різними аспектами запуску захоплення відеоданих. Деякі з цих властивостей повертають інформацію про запуск. Наприклад, у разі першого запуску інформативним є значення властивості InitialTriggerTime. Інші властивості дають можливість контролювати інші аспекти поведінки пристрою захоплення при запуску. Опис усіх властивостей запуску захоплення даних наведено в таблиці 2.6.

Таблиця 2.6 – Опис властивостей запуску захоплення даних

Властивості	Опис властивостей тригерів захоплення даних
InitialTriggerTime	Значення моментів часу першого запуску
TriggerCondition	Опис умов, необхідних для запуску захоплення. Цю властивість завжди встановлено в стан '*none*' для безпосереднього (*immediate*) і ручного (*manual*) включення захоплення
TriggerFcn	Специфікація функції обробки події "запуск захоплення"
TriggerFrameDelay	Кількість кадрів, що пропускаються перед записом даних в пам'ять або на диск
TriggerRepeat	Кількість додаткових включень тригера. Коли значення TriggerRepeat дорівнює 0, вмикання захоплення здійснюється, але не повторюється додатково
TriggersExecuted	Опис числа включень, які можуть бути здійснені
TriggerSource	Опис джерела перевірки умов запуску захоплення. Цю властивість завжди встановлено в стан *'none'* для безпосереднього (*immediate*) і ручного (*manual*) включення захоплення
TriggerType	Опис типу запуску захоплення: *'immediate'*, *'manual'* або *'hardware'*

Майже всі властивості запуску захоплення можуть бути встановлені за допомогою тих же методів, які використовуються при установці властивостей будь-яких об'єктів *MATLAB*, наприклад, за допомогою функції set. Щоб задати значення властивості TriggerRepeat, наприклад, можна скористатися наступним викликом функції set:

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

Тут змінна vid є об'єктом відеозахоплення, який був створений попередньо за допомогою функції videoinput.

Деякі властивості запуску захоплення (TriggerType, TriggerCondition і TriggerSource) є взаємопов'язаними. Наприклад, деякі значення TriggerCondition матимуть сенс, коли властивості TriggerType буде встановлено значення *'hardware'*. Тому вони вимагають використання функції triggerconfig для установки своїх значень. Для визначення всіх діючих

конфігурацій властивостей TriggerType, TriggerCondition і TriggerSource застосовується функція triggerinfo, яка використовує об'єкт відеозахоплення, як аргумент виклику:

```
config = triggerinfo(vid);
```

Ця функція повертає масив структур, кожна з яких представляє собою діючу комбінацію значень властивостей. Кожна структура в масиві містить три поля, які містять значення цих властивостей запуску. Наприклад, структура для безпосереднього запуску завжди містить значення:

```
TriggerType: 'immediate'
TriggerCondition: 'none'
TriggerSource: 'none'
```

Після отримання потрібної інформації для установки значень властивостей TriggerType, TriggerCondition і TriggerSource слід скористатися функцією triggerconfig зі значеннями властивостей в якості аргументу. Наприклад, для установки значення властивостей апаратного запуску виклик може мати вигляд:

```
triggerconfig(vid, 'hardware', 'risingEdge', 'TTL');
```

При описі ручного запуску в якості аргументу можна вказати тільки значення типу запуску захоплення:

```
triggerconfig(vid, 'manual');
```

Також можна використовувати структуру, що повертається функцією triggerinfo, для установки всіх трьох властивостей відразу:

```
triggerconfig(config(1));
```

2.4.3 Основні типи запуску захоплення і їхня реалізація

Існує три типи запуску захоплення відеоданих:

- автоматичний або безпосередній (*immediate*);
- ручний (*manual*);
- апаратний (*hardware*).

Для опису типу запуску захоплення (типу тригера) необхідно встановити значення властивості TriggerType об'єкта відеозахоплення. Для установки значень цих властивостей можна використовувати функцію triggerconfig. Список всіх типів запуску захоплення, які підтримує *Image Acquisition Toolbox*, з інформацією про те, коли і який використовувати, наведено в таблиці 2.7.

Таблиця 2.7 – Список всіх типів запуску захоплення

Значення типу тригера	**Значення TriggerSource і TriggerCondition**	**Опис типу запуску захоплення**
'immediate'	Завжди '*none*'	Запуск відбувається автоматично, безпосередньо після виклику функції start. Цей тип запуску завжди встановлений за замовчуванням.
'manual'	Завжди '*none*'	Запуск відбувається після використання

Значення типу тригера	Значення TriggerSource і TriggerCondition	Опис типу запуску захоплення
		функції trigger. Ручний контроль забезпечує більш великі можливості управління захопленням зображень. Наприклад, існує можливість за допомогою функції preview контролювати відеопотік при захопленні і виробляти ручний запуск в потрібний момент при настанні певних умов в процесі спостереження.
'hardware'	Залежить від пристрою	Апаратний запуск проводиться зовнішнім сигналом. Цей тип запуску використовується в разі синхронізації з іншими пристроями або, коли необхідно підтримувати встановлену швидкість захоплення зображень.

Для ***автоматичного запуску захоплення відеоданих*** (*immediate trigger*) достатньо створити об'єкт відеозахоплення, а потім його запустити командою start. Прямий запуск є тим типом тригера, який встановлюється за умовчанням для всіх об'єктів відеозахоплення. Об'єкти реалізують безпосередній запуск після використання команди start. На рисунку 2.4 продемонстрована реалізація автоматичного або безпосереднього запуску захоплення відеоданих.

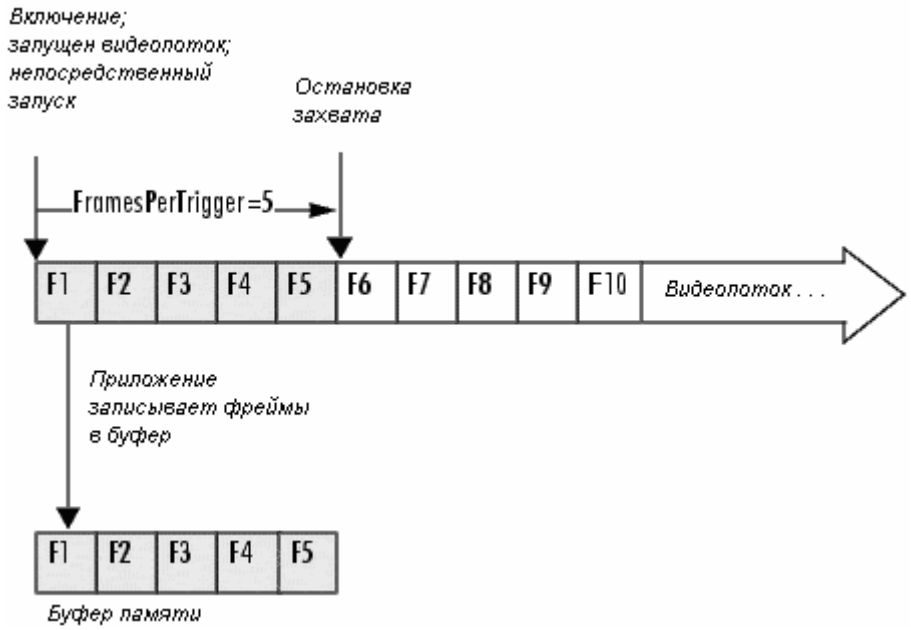


Рисунок 2.4 – Автоматичний або безпосередній запуск захоплення відеоданих

Розглянемо *приклад реалізації автоматичного прямого запуску*. Спочатку необхідно створити об'єкт відеозахоплення, наприклад, для ро-

боти з *web*-камерою:

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

При використанні прямого запуску немає необхідності встановлювати властивості TriggerType, оскільки значення *'immediate'* (автоматичний або безпосередній запуск захоплення) використовується за умовчанням. Для установки властивості FramesPerTrigger значення 5 кадрів (за замовчуванням це значення дорівнює 10), щоб була відповідність прикладу до рисунку 2.4, слід виконати команду:

```
set(vid, 'FramesPerTrigger', 5);
```

Використання функції start призведе до запуску об'єкта захоплення зображень:

```
start(vid);
```

Об'єкт vid буде використовувати безпосередній запуск і захоплення п'яти кадрів (фреймів) даних; запис даних буде проводитися в буфер пам'яті. Після запису заданого числа кадрів відбувається зупинка захоплення зображень. Для контролю виконання захоплення даних необхідно переглянути значення властивості FramesAcquired командою disp(vid.FramesAcquired), яка при нормальній реалізації захоплення повинна вивести в командному вікні значення 5. Для виконання інших прямих запусків необхідно перезапустити об'єкт відеозахоплення. Це призводить до видалення даних, які стосуються першого запуску. Для виконання кількох повторюваних прямих запусків слід використовувати властивість TriggerRepeat. Якщо об'єкт відеозахоплення не передбачається більше використовувати, його необхідно видалити і очистити пам'ять:

```
delete(vid);
clear vid;
```

Для реалізації ***ручного запуску захоплення відеоданих*** (*manual trigger*) створюється об'єкт відеозахоплення і властивості TriggerType встановлюється значення *'manual'*. Об'єкти відеозахоплення виконують ручний запуск з використанням функції trigger. На рисунку 2.5 показаний приклад ручного запуску.

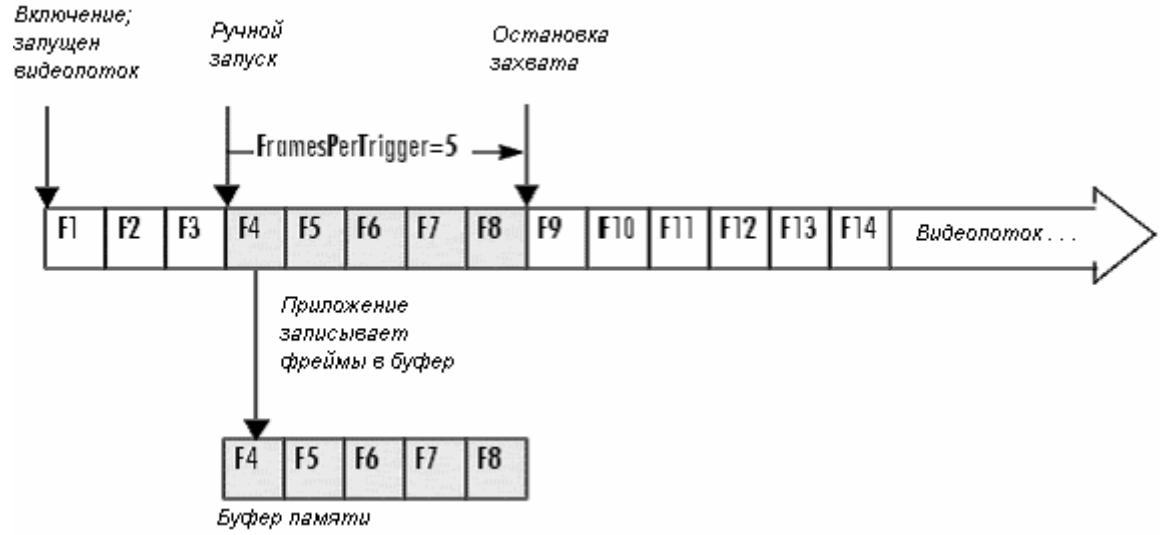


Рисунок 2.5 – Ручний запуск захоплення відеоданих

Розглянемо *приклад реалізації ручного запуску захоплення даних*. Спочатку необхідно створити об'єкт відеозахоплення, наприклад, для роботи з *web*-камерою:

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

Потім властивості TriggerType слід встановити значення *'manual'*. Для установки значень, що визначають властивості запуску захоплення відеоданих, включаючи властивість TriggerType, необхідно використовувати функцію triggerconfig:

```
triggerconfig(vid, 'Manual');
```

Для установки властивості FramesPerTrigger значення 5 кадрів, щоб була відповідність до рисунку 2.5, слід виконати команду:

```
set(vid, 'FramesPerTrigger', 5);
```

Потім слід запустити об'єкт відеозахоплення (перевести його в стан *'running state'*) функцією start:

```
start(vid);
```

Об'єкт захоплення буде перебувати в запущеному стані, але не в стані запису (*logging state*). При ручному тригері відеопотік виникає при запуску об'єкту захоплення, але захоплення кадрів не буде, поки не спрацює тригер. Контроль процесу захоплення викликом get(vid, 'FramesAcquired') поверне значення 0. Для реалізації ручного запуску захоплення слід використовувати функцію trigger:

```
trigger(vid);
```

Об'єкт відеозахоплення ініціалізує захоплення п'яти кадрів, що можна підтвердити викликом get(vid, 'FramesAcquired'), який повинен повернути значення 5. Після захоплення зазначеної кількості кадрів об'єкт відеозахоплення зупиняється (виходить з станів *'logging state'* і *'running state'*). Для виконання таких ручних запусків захоплення відеоданих необхідно спочатку запустити об'єкт відеозахоплення. Але це призводить до видалення даних, отриманих після першого запуску. Для виконання кількох повторюваних ручних запусків слід використовувати властивість TriggerRepeat. Якщо об'єкт відеозахоплення не передбачається більше використовувати, його необхідно видалити і очистити пам'ять: delete(vid); clear vid.

При використанні ***апаратного запуску захоплення даних*** (*hardware trigger*) створюється об'єкт відеозахоплення і його властивості TriggerType встановлюється значення *'hardware'*. Також слід описати джерело апаратного запуску і тип умов. Пристрій захоплення моніторить вказане джерело, чекаючи настання специфікованої умови. На рисунку 2.6 продемонстрований процес апаратного запуску захоплення відеоданих.

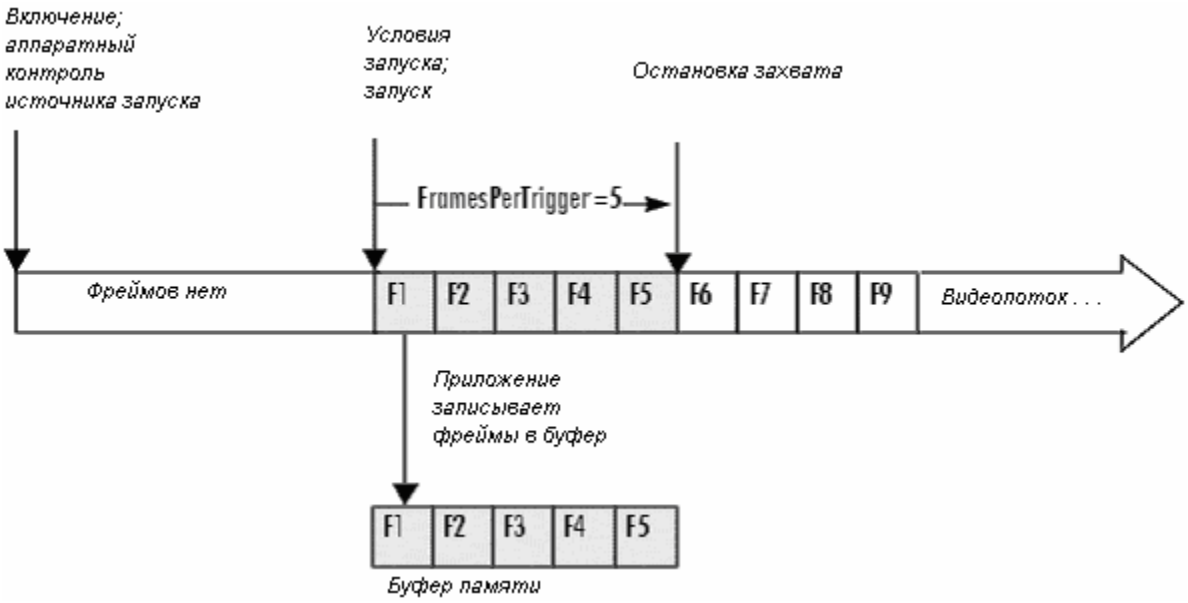


Рисунок 2.6 – Апаратний запуск захоплення відеоданих

При використанні апаратного тригера відеопотік не починається до тих пір, поки тригер (умова включення) не спрацює. Джерело запуску і умови, які контролюють апаратний запуск, залежать від конкретного пристрою захоплення. Для визначення того, чи підтримує пристрій захоплення зображень апаратний запуск і за яких умов, необхідно використовувати функцію triggerinfo.

Розглянемо *приклад реалізації апаратного запуску* для пристрою захоплення відеозображень *Matrox*, яке підтримує цей тип запуску. Для використання цього пристрою необхідно створити об'єкт відеозахоплення такого вигляду:

```
vid = videoinput('matrox', 1);
```

Для визначення того, чи підтримує пристрій захоплення зображень апаратний запуск, слід використовувати функцію triggerinfo. За допомогою цієї функції також проводиться пошук необхідної конфігурації властивостей TriggerSource і TriggerCondition. Нехай функція triggerinfo повертає наступну конфігурацію запуску:

```
triggerinfo(vid)
Valid Trigger Configurations:
TriggerType:        TriggerCondition:    TriggerSource:
'immediate'         'none'               'none'
'manual'            'none'               'none'
'hardware'          'risingEdge'         'TTL'
'hardware'          'fallingEdge'        'TTL'
```

На основі отриманої інформації слід настроїти апаратний тригер функцією triggerconfig наступним чином:

```
triggerconfig(vid, 'hardware','risingEdge', 'TTL');
```

Можливий і альтернативний шлях. Для цього одну зі структур, яка повертається функцією triggerinfo, необхідно підставити в функцію triggerconfig:

```
configs = triggerinfo(vid);
triggerconfig(vid, configs(3));
```

Для установки властивості FramesPerTrigger значення 5 кадрів, щоб була відповідність до рисунку 2.6, слід виконати команду:

```
set(vid, 'FramesPerTrigger', 5);
```

Потім слід запустити об'єкт відеозахоплення (перевести його в стан *'running state'*) функцією start:

```
start(vid);
```

Об'єкт захоплення буде перебувати в запущеному стані, але не в стані запису (*logging state*). При цьому буде здійснюватися апаратний контроль джерела запуску. Для цього повинні бути описані відповідні умови настання моменту запуску. Якщо такі умови формуються джерелом запуску, то пристрій захоплення виконує запуск захоплення відеозображень і доставляє кадри зображення об'єкту відеозахоплення. Число захоплених кадрів буде визначатися значенням FramesPerTrigger. Захоплення можна підтвердити викликом get(vid, 'FramesAcquired'), який повинен повернути значення 5. Після виконання апаратного запуску і захоплення описаного числа кадрів, об'єкт відеозахоплення зупиняється (виходить зі стану *'running state'*). Для виконання подальших апаратних запусків необхідно перезавантажити об'єкт відеозахоплення. Але це призводить до втрати даних, отриманих при першому запуску. Для виконання декількох запусків необхідно встановити відповідне значення властивості TriggerRepeat. Якщо об'єкт відеозахоплення не передбачається більше використовувати, його необхідно видалити і очистити пам'ять: delete(vid); clear vid.

2.5 Контрольні запитання

1. Для чого може бути використаний пакет розширення *Image Acquisition Toolbox*?
2. Наведіть основні процедури захоплення зображень в *Image Acquisition Toolbox*.
3. Опишіть послідовність дій при приєднанні пристроїв захоплення зображень до *MATLAB*.
4. Яким чином реалізується управління захопленням зображень в MATLAB?
5. Дайте визначення таким поняттям *Image Acquisition Toolbox*, як *trigger*, *logging*, *running*.
6. Якими командами треба завершити процедуру захоплення зображень в MATLAB?

2.6 Хід роботи

2.6.1 Завдання до роботи

1. Дослідити можливості пакета розширення *Image Acquisition Toolbox* з введення відеоінформації з *web*-камери в робоче середовище

MATLAB. Скласти таблицю з переліком функцій, наявних у пакеті, згрупувавши їх за категоріями.

2. Відповідними програмними засобами пов'язати пристрій захоплення відеозображення у вигляді *web*-камери з об'єктом відеозахоплення, конфігурувати його і виконати попереднє проглядання відеозображення, що сприймається *web*-камерою.

3. Проаналізувати можливості *web*-камери, як відеоджерела середовища *MATLAB*, задати певну роздільну здатність відеозображень (не найнижчу з усіх можливих варіантів).

4. Встановити півтонове подання (у вигляді відтінків сірого) захопленого відеозображення.

5. Виконати захоплення 100 кадрів відеозображення з *web*-камери, визначити реалізовану на даному апаратному забезпеченні швидкість захоплення відеозображень (кадрів за секунду) і відобразити захоплені кадри засобами *MATLAB* з використанням знайденої частоти кадрів.

2.6.2 Підготовчі стадії

Поза навчальною лабораторією треба зробити наступне:

1. Користуючись підрозділами 2.1 – 2.4, а також рекомендованою літературою, ознайомитися із пакетом розширення *Image Acquisition Toolbox* середовища *MATLAB*, складом та особливостями його функцій.

2. Оформити першу частину звіту з виконання лабораторної роботи, в якій зазначити прізвище студента, мету лабораторних досліджень та дати письмові відповіді на контрольні запитання (п. 2.5).

3. Перш, ніж приступити до виконання завдань, необхідно ознайомитися з основними відомостями про можливості системи MATLAB з захоплення відеозображень, викладеними в розділі "2.1 Загальна характеристика пакету розширення Image Acquisition Toolbox середовища MATLAB".

4. Для виконання поставлених завдань необхідно більш детально розглянути принципи і основні етапи реалізації процедури захоплення зображень засобами *Image Acquisition Toolbox*, які представлені в розділі "2.2 Основні процедури захоплення зображень в Image Acquisition Toolbox".

5. Перш за все, необхідно пов'язати пристрій захоплення зображень із середовищем *MATLAB*. Основні відомості про це викладено в розділі "2.3 Підключення пристроїв захоплення зображень до MATLAB".

2.6.3 Пояснення до виконання завдань

Роботу необхідно почати з отримання інформації про доступні в системі MATLAB (а, відповідно, – і в операційній системі) пристрої захоплення відеозображень. Для цього слід використовувати функцію imaqhwinfo (більш докладно про це можна дізнатися в підрозділі "2.3.1 Отримання інформації про пристрої"):

```
imaqInfo = imaqhwinfo;
```

Функція imaqhwinfo без вхідних аргументів повертає структуру imaqInfo з інформацією про наявні в системі адаптери для захоплення зображень і програмне забезпечення. Для виведення в командне вікно *MATLAB* цієї інформації можна скористатися функцією disp:

```
disp('imaqInfo:'); disp(imaqInfo);
```

З отриманої інформації слід витягти список InstalledAdaptors наявних в системі адаптерів

```
InstalledAdaptors = imaqInfo.InstalledAdaptors;
```

і вивести його в командне вікно *MATLAB* (керуюча esc-послідовність \ b в рядку форматного виведення функції fprintf видаляє попередній порожній рядок у вікні виводу):

```
fprintf('\b InstalledAdaptors:'); disp(InstalledAdaptors);
```

З отриманого списку адаптерів InstalledAdaptors слід вибрати один – з яким буде здійснюватися робота. Не всі виявлені системою *MATLAB* адаптери для захоплення зображень можуть бути представлені реально присутнім пристроєм захоплення зображення. Передбачається, що лабораторна робота виконується на комп'ютері стандартної конфігурації, і з усіх можливих пристроїв захоплення зображення він має тільки одне, наприклад, – у вигляді *web*-камери. Для подальшої роботи необхідно визначити ім'я адаптера, відповідного наявному в системі пристрою захоплення, і ідентифікаційний номер *ID* цього пристрою.

Якщо адаптер не пов'язаний з реально присутнім в системі пристроєм захоплення, інформація про пристрій захоплення, пов'язаний з ним, буде відсутньою, і виклик функції imaqhwinfo з ім'ям адаптера в якості першого аргументу і рядком 'DeviceInfo' в якості другого буде повертати порожній (*empty*) масив. Перевіряючи в циклі по всім адаптерам, чи є значення, що повертається, не порожнім, слід зробити примусове завершення циклу за непорожнього значення. Адаптер, на якому стався вихід з циклу, і є пов'язаним з наявним в комп'ютері пристроєм захвату зображень. Зробити зазначені дії можна за допомогою наступних команд:

```
adaptNum = 0;
for i = 1:length(InstalledAdaptors)
   if ~isempty( imaqhwinfo(InstalledAdaptors{i}, 'DeviceInfo') )
      adaptNum = i;
      break
   end
end
```

Цикл може повторюватися стільки разів, скільки адаптерів виявлено системою. Це визначається довжиною вектора InstalledAdaptors за допомогою функції length(InstalledAdaptors). Функція isempty() перевіряє вхідний параметр на "порожнечу" і повертає 1 для порожнього. Для перевірки "непорожнечі" перед нею стоїть логічне "НЕ", яке інвертує значення, що повертається і дає істину (1) для непорожнього вхідного параметра. В якості вхідного параметра використовується значення, що повертається викликом

функції imaqhwinfo(InstalledAdaptors {i}, 'DeviceInfo'), сенс якого описаний вище. Для визначення порядкового номера адаптера, пов'язаного з присутнім в системі пристроєм захоплення зображень, використовується змінна adaptNum. Перед входом в цикл ця змінна обнуляється, а в циклі вона отримує значення поточної ітерації циклу тільки в разі виконання умови зв'язку адаптера з реальним пристроєм. Тому при виході з циклу в цій змінній буде зберігатися порядковий номер потрібного адаптера. Примусовий вихід з циклу реалізується командою break.

Якщо цикл завершився без примусового виходу з нього, це означає, що жоден з адаптерів не пов'язаний з реально існуючим пристроєм захоплення. Тому продовжувати подальше виконання програми безглуздо, і її слід завершити. Для перевірки цього факту можна використовувати значення змінної adaptNum: якщо не було примусового завершення циклу, її значення так і залишиться нульовим. Передчасне завершення програми без досягнення її кінця можна реалізувати командою return:

```
if adaptNum == 0
   fprintf('Працюючого апаратного адаптера для відеозахоплення не
виявлено.\n')
   return
end
```

Після того, як був обраний адаптер, пов'язаний з реально присутнім в системі пристроєм захоплення, необхідно отримати і проаналізувати інформацію про нього і пов'язаний з ним пристрій. Для цього можна використовувати все ту ж функцію imaqhwinfo з ім'ям адаптера в якості вхідного параметра, яке для знайденого порядкового номера адаптера adaptNum визначається значенням InstalledAdaptors{adaptNum}:

```
hwInfo = imaqhwinfo(InstalledAdaptors{adaptNum});
```

Використовуючи повернуту функцією imaqhwinfo структуру hwInfo, можна отримати ім'я адаптера і ідентифікаційний номер (ID) пристрою захоплення (ці значення знадобляться при створенні об'єкта відеозахоплення):

```
AdaptorName = hwInfo.AdaptorName;
deviceID = hwInfo.DeviceIDs{1};
```

Іншу інформацію про адаптер можна вивести в командне вікно *MATLAB* за допомогою команди disp:

```
fprintf('\bInformation about the adaptor ''%s'' for deviceID №%i:\n', AdaptorName,
deviceID);
disp(hwInfo);
```

У структурі hwInfo, крім інформації про адаптер, у полі DeviceInfo представлена інформація про пов'язаний з адаптером конкретний пристрій захоплення зображень. Для аналізу цієї інформації (для аналізу, перш за все, режимів роботи пристрою захоплення) її теж рекомендується вивести в командне вікно:

```
DeviceInfo = hwInfo.DeviceInfo;
fprintf('\bDeviceInfo:\n');
```

```
disp(DeviceInfo);
```

На основі цієї інформації слід визначитися з форматом відео, який буде використовуватися при захопленні відео з *web*-камери. Якщо не задавати формат відео при створенні об'єкта відеозахоплення, буде використовуватися формат за замовчуванням, який, як правило, має найменшу роздільну здатність. Оскільки до першого запуску цієї програми невідомі позначення підтримуваних пристроєм захоплення формати відео, виберемо останній зі списку всіх підтримуваних форматів, який зберігається в полі SupportedFormats структури DeviceInfo.

```
vFormat = char(DeviceInfo.SupportedFormats(end));
fprintf('\bUsedFormats: %s\n', vFormat);
```

Тут використана функція char() для перетворення елемента масиву осередків SupportedFormats в текстовий рядок. Ключове слово end на місці індексного виразу дозволяє послатися на останній елемент масиву. Функція fprintf () виводить рядок vFormat з кодовим позначенням формату відео в командне вікно *MATLAB* з необхідним форматуванням виведення.

Після того, як отримана вся необхідна інформація, можна приступити до створення об'єкта відеозахоплення середовища *MATLAB*. Інформація про те, як це зробити і що таке об'єкт відеозахоплення (*Video Input Object*), викладена в підрозділі "2.3.2 Створення об'єктів захоплення зображень". Створення об'єкта відеозахоплення встановлює зв'язок між системою *MATLAB* і пристроєм захоплення зображень. Створити об'єкт відеозахоплення можна функцією videoinput(), вхідними параметрами якої є ім'я адаптера AdaptorName, ID пристрою захоплення deviceID і використовуваний формат відео vFormat:

```
vid = videoinput( AdaptorName, deviceID, vFormat );
```

Для перевірки встановленого зв'язку між об'єктом відеозахоплення і пристроєм, представленим *web*-камерою, рекомендується проаналізувати характеристики створеного об'єкта в командному вікні *MATLAB*, куди вони можуть бути виведені командою disp(vid):

```
disp('Video Input Object:'); disp(vid);
```

Одночасно з об'єктом відеозахоплення створюється пов'язаний з ним об'єкт відеоджерела (*Video Source Object*). Доступ до нього можна отримати через поле Source об'єкта відеозахоплення vid:

```
Source = vid.Source;
```

Характеристики об'єкта відеоджерела також рекомендується проаналізувати в командному вікні *MATLAB*, куди вони можуть бути виведені командою disp(Source):

```
disp('Video Source Object:'); disp(Source);
```

Для попереднього перегляду відеопотоку, що формується *web*-камерою в спеціально призначеному для цього графічному вікні, необхідно виконати команду:

```
preview(vid);
```

Функція preview(vid) створює вікно *Video Preview*, в якому в реальному часі відображається відеопотік для об'єкта відеозахоплення vid. Внизу вікна в рядку стану також відображається поточний час, роздільна здатність відеозображення і поточний стан об'єкта відеозахоплення vid. Вікно *Video Preview* відображає відеодані в 100%-масштабі (один піксель екрану представляє один піксель відеозображення). Тому розмір вікна попереднього перегляду визначається розміром зображення, що відображається яке, в свою чергу, визначається значенням властивості ROIPosition об'єкта відеозахоплення vid.

Вікно попереднього перегляду *Video Preview* залишається активним до тих пір, поки не буде зупинено командою stoppreview або закрито командою closepreview. Якщо об'єкт відеозахоплення vid буде видалений командою delete(vid), вікно *Video Preview* зупинить попередній перегляд і закриється автоматично.

Поведінка вікна *Video Preview* залежить від поточного стану об'єкта відеозахоплення і від критичної конфігурації. При стані *об'єкта Running = off* вікно буде в реальному часі відображати відеозображення, що сприймаються пристроєм, для всіх типів тригерів. Зміна конфігурації властивостей об'єкта відеозахоплення буде приводити до зміни виду зображення в вікні *Video Preview*. При переході об'єкта в стан *Running=on* для TriggerType, встановленого в immediate або manual, вікно буде продовжувати оновлювати відображувані зображення. При переході об'єкта в стан запису (*Logging=on*) вікно *Video Preview* може пропускати деякі кадри відеоданих, але це ніяк не буде позначатися на кадрах, що записуються в пам'ять або на диск.

Для установки напівтонового уявлення (у вигляді відтінків сірого) захопленого відеозображення (колірне представлення визначається властивістю ReturnedColorSpace об'єкта відеозахоплення) і розміру захоплення в 100 кадрів (визначається значенням властивості FramesPerTrigger) необхідно відповідним чином конфігурувати об'єкт відеозахоплення vid за допомогою функції set(), яка призначена для установки значень властивостей об'єкта:

```
FrameNum = 100;
set(vid, 'ReturnedColorSpace', 'grayscale', 'FramesPerTrigger',
FrameNum);
```

Попередньо змінній FrameNum, яка в даному прикладі створюється для зберігання розміру захоплення, необхідно присвоїти відповідну кількість кадрів. Для того, щоб дізнатися значення інших властивостей об'єкта відеозахоплення vid, їх можна вивести в командне вікно *MATLAB* командою get(vid).

Перед тим, як приступити до реалізації програмного коду, присвяченого захопленню відеозображень, слід ознайомитися з розділом "2.4 Управління захопленням зображень в *MATLAB*". Реалізувати поставлене завдання із захоплення 100 кадрів з *web*-камери можна за допомогою без-

посереднього (автоматичного) тригера (включення захоплення). Більш докладно про типи тригерів можна ознайомитися в підрозділі "2.4.3 Основні типи запуску захоплення і їх реалізація". Безпосередній (*immediate*) тип включення захоплення є встановленим за замовчуванням, тому немає необхідності в явному завданні типу тригера.

Запуск захоплення відеозображень при безпосередньому (*immediate*) типі тригера відбувається автоматично, безпосередньо після запуску об'єкту відеозахоплення, який реалізується викликом функції start():

```
start(vid);
```

Більш докладно про запуск об'єкта відеозахоплення можна ознайомитися в підрозділі "2.3.4 Запуск і зупинка об'єкта захоплення зображень".

Відповідно до установок для об'єкта відеозахоплення відразу після виконання функції start() буде проведене захоплення 100 кадрів відеозображень, які будуть розміщені в буфері пам'яті (оскільки властивість LoggingMode об'єкта vid має значення *'memory'* за замовчуванням). Якщо є необхідність блокувати виконання наступних дій, поки не закінчиться захоплення (а в даному прикладі вона є, тому що наступний перенос захоплених даних в *Workspace* повинен відбутися тільки після закінчення захоплення), слід використовувати функцію wait().

```
wait(vid, FrameNum, 'logging');
```

Очікування визначається або вичерпанням часу тайм-ауту (заданою в прикладі кількістю кадрів захоплення FrameNum, що справедливо для швидкості захоплення 1 кадр за секунду), або через вихід зі стану запису (*Logging=on*). Можливий і альтернативний підхід (оскільки наступне перенесення даних не почалась, поки не вичерпається час тайм-ауту): задати попередньо час тайм-ауту великим, порівнянним з числом кадрів захоплення, наприклад, set(vid, 'Timeout', FrameNum / 2); .

Для того, щоб працювати з захопленими даними в середовищі *MATLAB*, їх необхідно перемістити до робочого простору *MATLAB* (*Workspace*). Для їх перенесення з тимчасового буфера пам'яті до *MATLAB Workspace* можна використати функцію getdata ():

```
[VideoData, time] = getdata(vid);
```

Після перенесення відеоданих до робочого простору *MATLAB* та подання їх у вигляді масиву VideoData з ними можна виконувати будь-які дії, як з будь-якими іншими даними. Розмір масиву VideoData і його розмірність можна дізнатися за допомогою команди size(VideoData). Змінна time, яка повертається функцією getdata, зберігає часові відліки захоплення кожного кадру. Цю змінну можна використовувати для визначення часу захоплення зображень time (end) -time (1) і його швидкості fps (частоти кадрів за секунду):

```
   fps = FrameNum/(time(end)-time(1));
fprintf('Acquisition time: %g; frames per second: %g.\n', time(end)-time(1),
fps);
```

Для відображення захоплених відеоданих засобами *MATLAB* слід скористатися функцією mplay(), яка входить до складу *Computer Vision System Toolbox* (в старих версіях – в *Video and Image Processing Blockset*) і призначена для запуску ***MPlay GUI***, що дозволяє переглядати відео з робочого простору *MATLAB*:

```
mplay(VideoData, fps);
```

Указання частоти кадрів fps в функції mplay() дозволяє відтворювати відео з тією ж швидкістю, з якою здійснювався захоплення. Вона може бути різною і визначається апаратними можливостями комп'ютера і камери.

Для завершення роботи з об'єктом відеозахоплення, якщо не передбачається більше його використовувати для захоплення відеозображень, об'єкт слід видалити командою delete():

```
delete(vid);
```

При цьому об'єкт не видаляється з робочого простору *MATLAB*, а тільки розривається його зв'язок з конкретним пристроєм захвату. Пам'ять, виділена під об'єкт відеозахоплення, очищається, і сеанс роботи з пристроєм захоплення закінчується. Для повного очищення робочого простору *MATLAB* слід використовувати команду clear vid.

2.7 Вимоги до звіту по роботі

Звіт про виконання роботи повинен містити:

1. Прізвище та групу студента, а також назву, мету та завдання лабораторної роботи.
2. Письмові відповіді на контрольні питання.
3. Таблицю з переліком функцій за категоріями, які підтримуються пакетом розширення *Image Acquisition Toolbox*.
4. Короткий перелік основних дій в процесі захоплення відео.
5. Текст розробленого скрипта з коментарями.
6. Скрін-шоти роботи розробленого скрипта (захоплені кадри).
7. Висновки, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Що саме конкретно дозволяє (перерахувати по пунктах) робити з відеозображеннями пакет розширення *Image Acquisition Toolbox* середовища MATLAB? Які конкретно практичні навички роботи з відео здобуті в ході дослідження?

2.8 Орієнтовні варіанти завдань

Варіанти *індивідуальних* завдань видаються викладачем після самостійного виконання студентом *типового* завдання та підготовчої стадії (п.2.6).

Індивідуальне завдання полягає в демонстрації можливостей функцій *Image Acquisition Toolbox* для роботи з відео. Можна скористатися будь-яким з численних прикладів при роботі з власним відео. При цьому треба обов’язково послатися на першоджерело (назву прикладу та використану версію *Matlab*.

3 Лабораторна робота №3. Дослідження особливостей цифрової обробки сигналу відеозображення

Мета роботи: ознайомлення з особливостями методів аналізу частотного спектра сигналу відеозображення

3.1 Подання відеозображення у вигляді сигналу

Перетворенню відеозображення в сигнал передує розкладання його на елементи. Зробити це необхідно тому, що передавати по каналу зв'язку в кожен момент часу можна тільки одне значення сигналу. Для лінії передачі сигналу характерна одновимірна залежність напруги від часу $s = s(t)$, тому сигнал має бути функцією тільки однієї незалежної змінної – часу.

Математичний опис чорно-білого нерухомого зображення має вигляд розподілу яскравості

$$L = L(x, y). \qquad (3.1)$$

Навіть в цьому простому випадку воно описується двовимірним розподілом і не може бути безпосередньо перетворене в одновимірний сигнал. Якщо розглядати передачу рухомих зображень, що є основним призначенням телебачення, то завдання ще більше ускладнюється – розподіл яскравості в цьому випадку буде функцією трьох змінних.

Для вирішення завдання перетворення тривимірного сигналу в одновимірний використовуються два фундаментальних принципи: ***дискретизація зображення*** і ***його розгортка***.

Просторова дискретизація полягає в розбивці всього поля переданого зображення на кінцеве число дискретних елементів. *Елементом зображення* називається мінімальна деталь зображення, всередині якої яскравість і колір вважаються постійними, тобто всередині елемента нерівномірність яскравості і кольору вже не будуть відрізнятися оком. ***Піксель*** – найменший логічний елемент двовимірного цифрового зображення в растровій графіці.

Проблему каналів зв'язку вирішує інший основний принцип: це *послідовна в часі передача* по каналу зв'язку інформації про яскравість елементів. Процес послідовної, почергової передачі елементів зображення називається ***розгорткою*** (скануванням) зображення. Координати розгорнутих точок зображення є функціями часу:

$$x = \varphi_x(t), \quad y = \varphi_y(t),$$

де $\varphi_x(t)$ і $\varphi_y(t)$ – довільні однозначні функції часу. Тому і передана інформація про розподіл яскравості зображення (3.1) стає функцією часу:

$$L = L(x, y) = L(\varphi_x(t), \varphi_y(t)) = L(t). \qquad (3.2)$$

Отже, процес розгортки вирішує завдання **перетворення зображення в сигнал**. Ця послідовність передачі вибирається в залежності від призначення системи. У телевізійному мовленні прийнята *лінійно-рядкова розгортка* (зліва направо і зверху вниз), тобто передача елемента за елемен-

том з постійним напрямком і швидкістю уздовж рядка і з постійною швидкістю чергування рядків в кадрі.

Лінія, по якій переміщається елемент розгортки по осі *x*, називається ***рядком***. Сукупність видимих рядків на екрані називається ***растром***. Повний цикл обходу пристроєм розгортки всіх елементів зображення називається ***кадром***. При лінійно-рядковій розгортці телевізійну систему зазвичай характеризують числом рядків у кадрі *M* і числом кадрів *n* – повних зображень за секунду.

3.2 Частотний спектр сигналу відеозображення

Якщо середня освітленість кадру відеозображення змінюється незначно від кадру до кадру, то значний внесок в спектр сигналу відеозображення вносить постійна складова, яка визначається середнім значенням освітленості кадру. Якщо білий кадр буде змінюватися чорним, то найнижчою частотою спектра буде частота кадрів $f_к$.

Високі частоти визначають тонку структуру сигналу, тобто відтворення контурів і дрібних деталей зображення. Максимальна частота спектра сигналу відеозображення визначається числом елементів в кадрі і дорівнює $f_{\max} = \kappa M^2 f_к / 2$, де κ – формат растра, *M* – число рядків в кадрі. Наприклад, для $\kappa = 4/3$, $M = 625$, $f_к = 25$ Гц вона дорівнює 6,5 МГц.

Для аналізу частотного спектра сигналу відеозображення розподіл освітленості $E(x, y)$ представляють у вигляді розкладання в ряд Фур'є

$$E(x, y) = \sum_{n=0}^{\infty} \sum_{m=-\infty}^{\infty} E_{mn} \cos\left(\frac{2\pi n}{l} x + \frac{2\pi m}{h} y + \varphi_{mn} \right), \qquad (3.3)$$

де *x* – горизогтальний розмір (ширина) зображення;

де *y* – вертикальний розмір (висота) зображення;

φ_{mn} – початкова фаза.

При постійній швидкості розгортки по рядках v_x і по кадрам v_y координати переданих елементів зображення змінюються пропорційно часу: $x = v_x t = M l f_к t$ і $y = v_y t = h f_к t$. Це дозволяє представити сигнал зображення у вигляді

$$s(t) = \sum_{n=0}^{\infty} \sum_{m=-\infty}^{\infty} E_{mn} \cos\left[2\pi (nM + m) f_к t + \varphi_{mn} \right]. \qquad (3.4)$$

Величина

$$f = (nM + m) f_к = n f_x + m f_y \qquad (3.5)$$

має розмірність частоти і визначає частотний спектр сигналу відеозображення. Тут $f_x = v_x / l = M f_к$ – частота горизонтальної розгортки, $f_y = v_y / h = f_к$ – частота вертикальної розгортки. Спектр сигналу зображення дискретний. Частоти його компонент кратні частотам горизонтальної і вертикальної розгорток. Він містить гармоніки з номерами $0 \le n \le \infty$, кратні частоті горизонтальної розгортки f_x, біля кожної з яких групуються

гармоніки з номерами $-\infty \le m \le +\infty$, кратні частоті вертикальної розгортки. В силу цього бічні спектри компонентів рядкової частоти перекриваються (рисунок 3.1), що призводить до специфічних спотворень зображення.

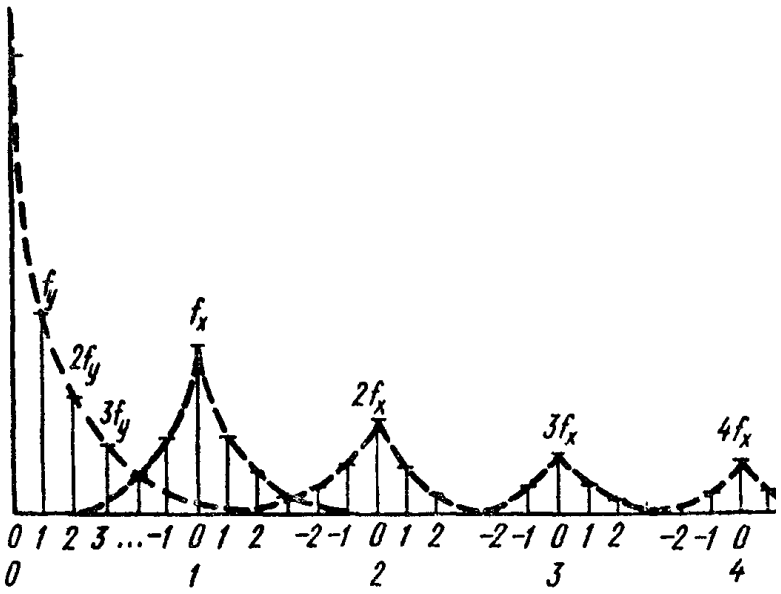


Рисунок 3.1 – Дискретна структура спектра сигналу зображення

В окремому випадку, коли освітленість зображення змінюється *тільки по вертикалі*, в спектрі сигналу зберігаються лише гармоніки частот вертикальної розгортки $f = mf_y$, і спектр сигналу виявляється *вузькосмуговим*. Якщо освітленість зображення змінюється тільки *по горизонталі*, в спектрі зберігаються тільки гармоніки частоти горизонтальної розгортки $f = nf_x = nMf_y$, і спектр виявляється *широкосмуговим*.

Гармоніки рядкової частоти утворюють ***первинний спектр сигналу відеозображення***. Амплітуди основних спектральних складових з ростом частоти зменшуються, швидкість убування визначається характером зміни сигналу уздовж рядка (для меандру амплітуда гармоніки обернено пропорційна її номеру). Біля кожної з основних частот спектру групуються *бічні складові*, обумовлені кадрової розгорткою і рухом деталей зображення. Вони утворюють ***вторинний спектр*** з частотами, кратними частоті кадрової розгортки.

Амплітуди складових вторинного спектра також зменшуються з зростанням їх номера (обернено пропорційно). Співвідношення між амплітудами складових первинного і вторинного спектрів залежить від виду зображення та розташуванням його відносно растра.

У разі, коли освітленість зображення змінюється і по рядках, і по кадрам, складові спектра групуються біля гармонік частоти горизонтальної розгортки, огинаюча яких залежить від розподілу освітленості вздовж рядків. У свою чергу огинаючі бічних спектрів залежать від розподілу освітленості в поперечному (вертикальному) напрямку. Таким чином, в цілому основна енергія відеосигналу зосереджена близько гармонік f_x і утворює

дискретні зони енергії, що несуть інформацію про передане зображення (рисунок 3.1).

У разі черезрядкового розкладання з кратністю *w* (числом полів на кадр) на інтервалі між двома гармоніками частоти рядкової розгортки укладається M + 1/*w* інтервалів вертикальної частоти (частоти полів) і при перекритті сусідніх бічних спектрів їх гармонійні складові перемежовуються. Таким чином, збільшення в *w* раз часу передачі кадру, що містить *w* полів, супроводжується ущільненням в *w* раз частотного спектра сигналу. У разі розкладання на два поля (*w* = 2) гармонійні складові з нижнього бокового спектра (*n*+1)-ої гармоніки частоти рядкової розгортки розташовуються строго посередині між складовими верхнього бічного спектру *n*-ої гармоніки, і частотний спектр сигналу ущільнюється в два рази.

При порядковому розкладанні відношення частоти горизонтальної (рядкової) розгортки до частоти вертикальної (кадрової) розгортки

$$f_x / f_y = M \qquad (3.6)$$

характеризується цілим числом і визначає повне число рядків в кадрі *M*. На інтервалі між двома сусідніми гармоніками частоти рядкової розгортки укладається *M* цілих інтервалів частоти кадрової розгортки, і, отже, при перекритті бічних спектрів частоти їх гармонійних складових збігаються. Підсумовування складових з номерами $n_1 \neq n_2$ і $m_1 \neq m_2$, що належать різним компонентам просторових частот зображення, і їх передача сигналом однієї і тієї ж частоти

$$f = n_1 f_x + m_1 f_y = n_2 f_x + m_2 f_y \qquad (3.7)$$

створюють на зображенні сторонні візерунки – ***муари***. Вони можуть призводити до погіршення якості відеозображення.

3.3 Контрольні запитання

1. Яким чином вирішується завдання перетворення тривимірного сигналу відеозображення в одновимірний?
2. Дайте визначення поняттю елемент зображення.
3. Дайте визначення поняттю растр.
4. Яким чином представляють розподіл освітленості для аналізу частотного спектра сигналу відеозображення?
5. У якому вигляді можна представити сигнал зображення?
6. Яка величина визначає частотний спектр сигналу відеозображення?
7. Що являє собою первинний та вторинний спектри сигналу відеозображення?
8. Що являють собою муари відеозображення? Наведіть приклад створення муара.

3.4 Хід роботи

3.4.1 Підготовчі стадії

Поза навчальною лабораторією треба зробити наступне:

1. Користуючись підрозділами 3.1, 3.2, а також рекомендованою літературою, ознайомитися із особливостями спектра відеосигналу.

2. Оформити першу частину звіту з виконання лабораторної роботи, в якій зазначити прізвище студента, мету лабораторних досліджень та дати письмові відповіді на контрольні запитання (п. 3.3).

3.4.2 Завдання до роботи

1. Провести аналіз частотного спектра одновимірного сигналу, що представляє розгортку зображення розміром 240×320 у вигляді:

1) 16 вертикальних чорних смуг рівної ширини, розділених білими смугами такої ж ширини;

2) 24 горизонтальних чорних смуг рівної ширини, розділених білими смугами такої ж ширини;

3) перетину 24 горизонтальних білих смуг рівної ширини, розділених чорними смугами такої ж ширини, і 16 вертикальних білих смуг рівної ширини, розділених чорними смугами такої ж ширини.

Аналізовані зображення, відповідні їм графіки одновимірних сигналів (або їх більш коротких сегментів) і отримані спектри відобразити графічно. Порівняти отримані спектри, знайти в них спільні елементи, виявити і пояснити відмінності. Обчислити частоти кадрової і рядкової розгорток, частоти зміни яскравості смуг в горизонтальному і вертикальному напрямку; визначити, як пов'язані обчислені значення з частотами гармонік отриманих спектрів.

Проаналізувати, як будуть змінюватися спектри сигналів аналізованих зображень при зміні кількості: 1) вертикальних смуг; 2) горизонтальних смуг; 3) одночасно і вертикальних, і горизонтальних смуг.

2. Проаналізувати, як позначиться на якості зображення його передача каналом зв'язку з одиничним коефіцієнтом передачі у смузі частот: 1) 0-400 кГц; 2) 190-194 кГц; 3) 188-196 кГц; 4) 192-201 кГц; 5) 400-800 кГц (і нульовим коефіцієнтом передачі поза цією смугою). Відобразити графічно зображення, передане по такому каналу зв'язку, і зіставити його вид з вхідним зображенням. Також для проведення порівняльного аналізу на одному графіку відобразити сигнал розкладання на вході і на виході каналу зв'язку (аналогічно вчинити і для спектрів).

3. Отримати частотний спектр сигналу розкладання фотографій: 1) 'blobs.png'; 2) 'cameraman.tif'; 3) 'testpat1.png'; 4) 'text.png' (вони знаходяться в папці *MATLAB \toolbox\images\imdemos*, або *\toolbox\images\imdata*). Дати пояснення виду отриманих спектрів, виходячи із специфіки кожної з аналізованих фотографій. Розглянути можливі спотворення при передачі цих фотографій вузькосмуговими каналами зв'язку з характерис-

тиками, наведеними у попередньому пункті. Зясувати найвужчу полосу каналу, при якій спотворення зображення припустимі.

4. Операції попереднього пункту здійснити над власноруч зробленим фотографічним зображенням. Зясувати найвужчу полосу каналу, при якій спотворення зображення припустимі.

3.4.3 Пояснення до роботи

1. Перед тим, як приступити до створення зображення у вигляді смуг, необхідно задати значення усіх необхідних параметрів зображення та сигналу, що отримується шляхом його розгортки. Перш за все необхідно визначити розмір зображення – задати уількість рядків M, стовпців J і елементів зображення N:

```
M = 240;
J = 320;
N = M*J;
```

Потім треба вивести цю інформацію на друк в командному вікні *MATLAB* (виведення на друк основних параметрів виконуваної програми вважається хорошим стилем програмування):

```
fprintf('К-ть рядків M: %i; К-ть стовпчиків J: %i; К-ть пікселів N: %i.\n', M, J, N);
```

Також необхідно визначити часові параметри формованого шляхом розгортки сигналу зображення. Ключовою є кількість кадрів за секунду, яка визначає тривалість одного кадру і, тим самим, – період сигналу зображення:

```
T = 1e3/50;    % тривалість кадру в мілісекундах
dt = T/N;      % крок дискретизації
t = (0:N-1)*dt;
fprintf('Тривалість кадру: %g мс; крок дискретизації: %5.1e мс.\n', T, dt);
```

Вектор t визначає відліки часової розгортки зображення. Зворотне значення кроку дискретизації в часовій області задає частоту дискретизації fs і визначає крок дискретизації в частотній області (він потрібен, щоб задати вектор дискретних значень частот відліків спектру f):

```
fs = 1/dt;     df = fs/N;     % частотні параметри в кГц
f = (0:N-1)*df;
fprintf('Частота дискретизації: %g кГц; частотний крок: %g кГц.\n', fs, df);
```

За заданими часовими параметрами розгортки можна обчислити частоти кадрової і рядкової розгорток:

```
fy = 1/T;                    % частота кадрової розгортки
fx = M*fy;                   % частота рядкової розгортки
```

Для виведення цих значень на друк в програму слід вставити команду:

```
fprintf('Частота кадрової розгортки fy = %g Гц; частота рядкової
розгортки fx = %g кГц.\n', 1e3*fy, fx);
```

Тепер можна перейти до створення зображень і аналізу їх спектрів. Щоб створити **перше зображення**, необхідно задати число вертикальних

смуг на ньому K і обчислити частоту зміни яскравості цих смуг в горизонтальному напрямку Fx:

```
K = 16;
Fx = K/J/dt;  fprintf('Частота полос по x: Fx = K*fx = %g кГц.\n', Fx);
```

Для формування смуг на зображенні можна скористатися функцією y= pulstran(t,d,'func'), яка генерує сигнал y в дискретних відліках t у вигляді послідовності імпульсів функції 'func', що мають початок в моменти d. За допомогою цієї функції можна задати перший рядок формованого зображення *row* у вигляді меандру (тому тип форми імпульсу 'rectpuls')

```
row = pulstran(0:J-1, (0.25:K)*J/K, 'rectpuls', J/K/2);
```

а потім виконати *M*-кратне дублювання цього рядка командою repmat, щоб отримати зображення у вигляді вертикальних чорних смуг рівної ширини, розділених білими смугами такої ж ширини:

```
Image = repmat(row, M, 1);
```

Отримане зображення *Image* можна відобразити графічно за допомогою команд;

```
figure('Name', 'Image 1');
imshow(Image); title('Image 1');
```

Для формування з зображення *Image* часового сигналу шляхом його порядкового розкладання необхідно перейти до використання наведеного індексу, що дозволяє розглядати матрицю, як вектор. Але зробити цей перехід треба на транспонованій матриці *Image*, оскільки в *MATLAB* матриці зберігаються в пам'яті по стовпцях. Таким чином, для подання отриманого зображення *Image* сигналом розкладання *s* необхідно виконати наступні команди:

```
s = Image';
s = s(:);
```

Отриманий сигнал розкладання можна відобразити графічно, але для кращого візуального сприйняття зробити це рекомендується для невеликої його частини, наприклад, для першої пари рядків зображення (реалізується це завданням меж для відображуваних значень вектора *t* у функції axis):

```
figure('Name', 'Videosignal 1', 'WindowStyle', 'docked');
plot(t, s, '.k-'); grid on
axis([t(1) t(2*J) -inf inf]);
xlabel('t, ms'); ylabel('s(t)');
title('Videosignal 1');
```

Для перетворення часового сигналу в його спектр слід використовувати функцію швидкого перетворення Фур'є, виклик якої в *MATLAB* має вигляд S = fft(s). Після обчислення спектра його необхідно відобразити графічно:

```
figure('Name', 'Spectrum 1', 'WindowStyle', 'docked');
plot(f, abs(S), '.k-'); grid on
axis([f(1) fs/2 -inf inf]);
xlabel('f, кHz'); ylabel('|S(f)|');
title('Spectrum of videosignal 1');
```

Аналогічно можна побудувати **друге зображення** у вигляді 24 горизонтальних чорних і білих смуг рівної ширини, розкласти його в одновимірний сигнал і побудувати його спектр. Для цього спочатку необхідно задати число горизонтальних смуг на ньому L і обчислити частоту зміни яскравості цих смуг у вертикальному напрямку Fy:

```
L = 24;
Fy = L/M/J/dt;  fprintf('Частота смуг по y: Fy = L*fy = %g кГц.\n', Fy);
```

Для побудови матриці зображення у вигляді горизонтальних смуг необхідно задати перший стовпець цієї матриці, а потім його *J* раз продублювати по горизонталі. Такий стовпець повинен містити в собі ділянки одиничних і нульових значень, що чергуються. Для його формування, як і в попередньому випадку, підійде функція pulstran, викликана з аналогічними параметрами:

```
col = pulstran(0:M-1, (0.25:L)*M/L, 'rectpuls', M/L/2)';
```

Далі залишається тільки продублювати цей стовпець (завдання вектора *col*, як стовпчика, визначається використанням операції транспонування ' до результату дії функції pulstran):

```
Image = repmat(col, 1, J);
```

Подальші дії по візуалізації отриманого зображення, сигналу його розкладання і відповідного йому спектра аналогічні тим, що були розглянуті вище для першого зображення (але не слід забувати про необхідність внесення змін номера зображення з 1 на 2). Для другого зображення також рекомендується внести зміни в команди, що задають часовий діапазон відображення сигналу розкладання зображення і частотний діапазон відображення його спектра. Часовий діапазон повинен охоплювати тривалість відображення одного білого і одного чорного рядка (число пікселів в одному такому рядку *J*M/L*):

```
axis([t(1) t(2*J*M/L) -inf inf]);
```

Частотний діапазон відображення спектру сигналу розкладання другого зображення краще обмежити частотою *M*fx/L*:

```
axis([f(1) M*fx/L -inf inf]);
```

При побудові **третього зображення** у вигляді перетину 24 горизонтальних і 16 вертикальних білих і чорних смуг можна скористатися вектором рядку *row*, побудованого для першого зображення, і вектором стовпчика *col*, побудованого для другого зображення. Спочатку формується зображення у вигляді *M*-кратного дублювання рядку *row*:

```
Image = repmat(row, M, 1);
```

А потім вектор *col* слід використовувати в якості логічної маски, яка виділяє рядки зображення *Image*, значення яких інвертується операцією ~, змінюючи тим самим чорний колір на білий, а білий – на чорний:

```
Image(~col,:) = ~Image(~col,:);
```

Візуалізація зображення, сигналу його розкладання і відповідного йому спектру аналогічні тим, що були розглянуті вище. Для більшої наочності рекомендується обмежити часовий діапазон відображення сигналу

розкладання значеннями t(J*M/L-J) t(J*M/L+J), а частотний діапазон відображення його спектру – значеннями f(1) fs/2. Також з метою збільшення наочності (для промальовування гармонік спектру з великим розмахом) рекомендується перед виконанням перетворення Фур'є відняти від сигналу розкладання його постійну складову (частотну компоненту, що відповідає нульовій частоті):

```
S = fft(s - mean(s));
```

2. Для аналізу **впливу на зображення усічення його спектра** при передачі його каналом зв'язку з обмеженою шириною смуги частот необхідно, перш за все, визначити передаточну частотну характеристику. Для цього треба задати межі смуги частот, де коефіцієнт передачі одиничний:

```
fMin = 0; fMax = 400;  % Межі смуги частот каналу передачі в кГц
```

Потім потрібно побудувати логічну маску *ind*, що визначає ті дискретні значення спектру, які потрапляють всередину цієї смуги.

```
ind = (fMin<=f & f<=fMax);
```

Щоб одержуваний шляхом усічення спектра сигнал був дійсним, а не аналітичним, крім частотних компонент всередині смуги частот [fMin; fMax] дана логічна маска також повинна включати відповідні компоненти негативних частот, тобто тих, що належать смузі [-fMax; -fMin]. Ця смуга через дискретність сигналу відображається в смугу fs- [fMin; fMax]. Тому для включення відліків, що відповідають негативним частотам, необхідно отриману логічну маску об'єднати з її копією, взятою в зворотному порядку і без відліку, відповідному нульовій частоті (першого відліку):

```
ind(2:end) = ind(2:end) | ind(end:-1:2);
```

Тепер для побудови усіченого спектру досить по створеній логічній масці *ind* в заздалегідь підготовлений вектор з нульовими значеннями *Sp* скопіювати значення вихідного спектру *S*:

```
Sp = zeros(size(S));
Sp(ind) = S(ind);
```

Зворотне перетворення Фур'є, прикладене до створеного вектора зі спектральними відліками, дозволить отримати сигнал, який визначається цим спектром: sp = ifft(Sp). Якщо вихідний спектр виходив з сигналу, з якого була вилучена постійна складова mean(s), то при зворотному перетворенні її необхідно повернути:

```
sp = ifft(Sp) + mean(s);
```

Для формування з сигналу розкладання матриці зображення можна скористатися функцією реконфігурації масивів reshape, не забуваючи при цьому, що матриці в *MATLAB* розташовуються в пам'яті по стовпцях, а зображення розкладається по рядках. Тому реконфігуровану матрицю треба транспонувати:

```
pImage = reshape( sp, J, M )';
```

Щоб зіставити отримане зображення з вхідним, його треба вивести в графічному вікні, що можна зробити в такий спосіб:

```
figure('Name', 'Зображення після фільтрації');
imshow(pImage); title('Фільтроване зображення');
```

Для зіставлення на одному графіку сигналу розкладання на вході і на виході каналу зв'язку рекомендується використовувати наступну послідовність команд:

```
figure('Name', 'Фільтрований відеосигнал', 'WindowStyle', 'docked');
subplot(2,1,1); plot(t, s, '.b-', t, sp, '.r-'); grid on;
axis([t(J*M/L-5.5*J) t(J*M/L-2.5*J) -inf inf]);
legend('Оригінал','Фільтрований'); xlabel('t, мс'); ylabel('s(t)');
title('Середина першого рядка оригінального та фільтрованого
відеосигналу');
subplot(2,1,2); plot(t, s, '.b-', t, sp, '.r-'); grid on;
axis([t(J*M/L-2.5*J) t(J*M/L+0.5*J) -inf inf]);
legend('Оригінал','Фільтрований'); xlabel('t, мс'); ylabel('s(t)');
title('Кінець першого рядка оригінального та фільтрованого
відеосигналу ');
```

Аналогічно для зіставлення на одному графіку вхідного і вихідного спектрів сигналу розкладання зображення можна скористатися командами:

```
figure('Name', 'Фільтрований спектр', 'WindowStyle', 'docked');
plot(f, abs(S), '.b-', f, abs(Sp), '.r-'); grid on;
axis([f(1) fs/4 -inf inf]);
legend('Оригінал','Фільтрований');
xlabel('f, кГц'); ylabel('|S(f)|');
title('Оригінальний та фільтрований спектри');
```

3. Для отримання **частотного спектра сигналу розкладання фотографії** необхідно, перш за все, завантажити фотографію і відобразити її в графічному вікні. Наприклад, для фотографії 'cameraman.tif' це можна зробити такою послідовністю команд:

```
figure('Name', 'Photo');
P = imread('cameraman.tif');
imshow(P); title('Photo');
```

Перед тим, як отримати сигнал розкладання фотографії, необхідно визначити число рядків, стовпців і пікселів в фотографії:

```
[Mp, Jp] = size(P);
Np = Mp*Jp;
```

Рекомендується вивести ці значення на друк за допомогою команди:

```
fprintf('Число рядків фотографії M: %i; число стовпців фотографії J: %i;  число пікселів фотографії N: %i.\n', Mp, Jp, Np).
```

Використовуючи ці значення, можна сформувати вектор з відліками часової розгортки зображення, представленого на фотографії:

```
tp = (0:Np-1)*T/Np;
```

Також необхідно визначити частотні параметри, вивести їх на друк і задати по них вектор дискретних значень частот відліків спектру *fP*:

```
fsP = Np/T;        dfP = fsP/Np;
fprintf('Частота дискретизації фотографії: %g кГц; частотний крок: %g кГц.\n', fsP, dfP);
fprintf('Частота рядкової розгортки фотографії fx: %g кГц.\n', Mp*fy);
fP = (0:Np-1)*dfP;
```

Для подання фотографії *P* сигналом розкладання *p* необхідно виконати наступні команди:

```
p = P';
p = p(:);
```

Для візуального аналізу отриманий сигнал розкладання фотографії слід відобразити в графічному вікні (краще сигнал розбити на частини, наприклад, які містять по 50 рядків, і відобразити кілька перших частин):

```
figure('Name', 'Photosignal', 'WindowStyle', 'docked');
subplot(2,1,1); plot(tp, p, '.k-'); axis([tp(1) tp(50*Jp) -inf inf]);
xlabel('t, ms'); ylabel('s(t)'); title('First 50 row of photosignal'); grid on
subplot(2,1,2); plot(tp, p, '.k-'); axis([tp(51*Jp) tp(100*Jp) -inf inf]);
xlabel('t, ms'); ylabel('s(t)'); title('Second 50 row of photosignal'); grid on
```

Для отримання спектру сигналу розкладання фотографії *Q* і відображення його в графічному вікні можна скористатися такою послідовністю команд (для більш точного виконання операції перетворення Фур'є необхідно попередньо перетворити сигнал розкладання до типу double):

```
Q = fft(double(p) - mean(p));
figure('Name', 'Spectrum of photo', 'WindowStyle', 'docked');
plot(fP, abs(Q), '.k-'); axis([fP(1) fsP/4 -inf inf]); grid on
xlabel('f, кHz'); ylabel('|S(f)|'); title('Spectrum of photo');
```

Для аналізу характеру спотворень під час передачі цієї фотографії вузькосмуговим каналом зв'язку необхідно вчинити аналогічно тому, як це було описано вище. Для цього потрібно задати межі смуги частот [fMin; fMax], де коефіцієнт передачі ненульовий, і по ним побудувати логічну маску ind, що визначає дискретні значення спектра, які передаються без спотворень (в наведеному нижче прикладі обрана смуга частот від 0 до 100 кГц):

```
fMin = 0; fMax = 100; % Межі смуги частот каналу передачі в кГц
ind = (fMin<=fP & fP<=fMax);
ind(2:end) = ind(2:end) | ind(end:-1:2);
```

За створеною логічною маскою *ind* в заздалегідь підготовлений вектор з нульовими значеннями *Qp* скопіювати значення спектра неспотвореної фотографії *Q*:

```
Qp = zeros(size(Q));
Qp(ind) = Q(ind);
```

Для отримання переданого сигналу розкладання фотографії *pp* необхідно виконати зворотне перетворення Фур'є від "обрізаного" спектра *Qp* і, якщо треба, додати до нього значення видаленої постійної складової:

```
pp = ifft(Qp) + mean(p);
```

За отриманим сигналом розкладання *pp* вид переданої фотографії *Pp* можна визначити, застосувавши до нього конвертацію до типу uint8 і операцію переконфігурування в матрицю необхідного розміру:

```
Pp = reshape( uint8(pp), Jp, Mp )';
```

Відображення отриманого зображення *Pp* можна виконати за допомогою команд:

```
figure('Name', 'Passed photo');
imshow(Pp); title('Passed photo');
```

3.5 Вимоги до звіту по роботі

Звіт про виконання роботи повинен містити:

1. Прізвище та групу студента, а також назву, мету та завдання лабораторної роботи.

2. Письмові відповіді на контрольні питання.

3. Текст розробленого скрипта з коментарями.

4. Скрін-шоти роботи розробленого скрипта.

5. Висновки, в яких треба зазначити, чи досягнута мета даної лабораторної роботи. Які саме конкретно (перерахувати по пунктах) особливості методів аналізу частотного спектра сигналу відеозображення у середовищі MATLAB вдалося виявити?

3.6 Орієнтовні варіанти завдань

Варіанти *індивідуальних* завдань видаються викладачем після самостійного виконання студентом *типового* завдання та підготовчої стадії (п.3.4).

Індивідуальне завдання полягає в демонстрації можливостей спектральної обробки власних відеозображень за допомогою фільтрів з різними частотними характеристиками.

4 Лабораторна робота №4. Дослідження прямого та оберненого перетворення Радона

Мета роботи: Дослідження прямого та оберненого перетворення Радона та практичне ознайомлення з основними особливостями обробки зображень з використанням перетворення Радона в середовищі MatLAB.

4.1 Поняття про перетворення Радона

У 1917 році австрійський математик Йоганн Радон запропонував метод відновлення (реконструкції) багатовимірних функцій за їхніми інтегральними характеристиками, тобто метод розв'язання оберненої задачі інтегральної геометрії.

Перетворення Радона $R(k, b)$ безперервної функції двох змінних (зображення) $f(x, y)$ обчислюється шляхом інтегрування (складання) значень f вздовж прямої лінії, як показано на рисунку 4.1.

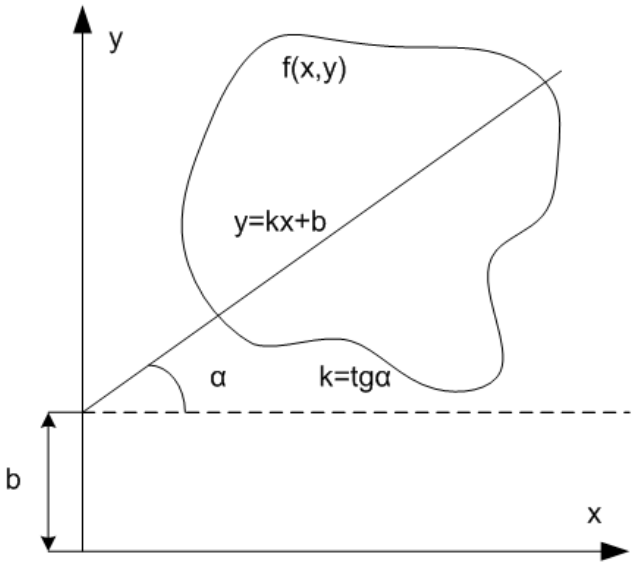


Рисунок 4.1 – Перетворення Радона

Можна записати вираз для інтегралу від функції вздовж прямої – прямого перетворення Радона:

$$R(k,b) = \int_{-\infty}^{\infty} f(x,y)dx = \int_{-\infty}^{\infty} f(x,kx+b)dx. \qquad (4.1)$$

З використанням дельта-функції Дірака це ж можна записати:

$$R(k,b) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x,y)\delta(y-kx-b)dxdy.$$

Загальний вигляд зворотного перетворення Радона:

$$f\ \ x,y\ = \frac{1}{2\pi}\int_{-\infty}^{+\infty}\frac{d}{dy}H\left[R\ \ k,y-kb\ \right]dk,$$

де H – перетворення Гільберта.

4.2 Використання перетворення Радона в комп'ютерній томографії

В основі роботи більшості томографів лежить ідея, що внутрішню структуру об'єкта можна зобразити, отримавши ряд паралельних поперцевих перетинів. Тому головне завдання комп'ютерної томографії полягає в отриманні (реконструкції) двовимірного (плаского) зображення поперечного перерізу досліджуваного об'єкту.

Метод отримання двовимірного томографічного зображення включає два етапи. На першому формуються проекційні дані, а на другому за проекційними даними відновлюється двовимірне зображення поперечного перерізу.

Методи реконструкції зображень можна розділити на три класи:

- аналітичні;
- алгебраїчні (засновані на розкладанні в ряд);
- статистичні.

Найбільше розповсюдження отримали аналітичні алгоритми реконструкції внаслідок своєї чисельної стійкості і високої швидкості розрахунків.

Для отримання інформації про внутрішню структуру об'єкта будується двовимірний розподіл щільності розподілу речовини $f(x, y)$ в тонкому плоскому перерізі об'єкта. Для цього використовується випромінювання, проникаюче крізь об'єкт. Досліджуваний об'єкт в межах тонкого поперечного шару просвічується паралельним пучком сфокусованих рентгенівських променів. Напрямок променів становить деякий кут φ з віссю. Промені, які проходять, послаблюються речовиною, що знаходиться всередині об'єкта, пропорційно до її щільності у тонкій площині томографічного зрізу (Рисунок 4.2).

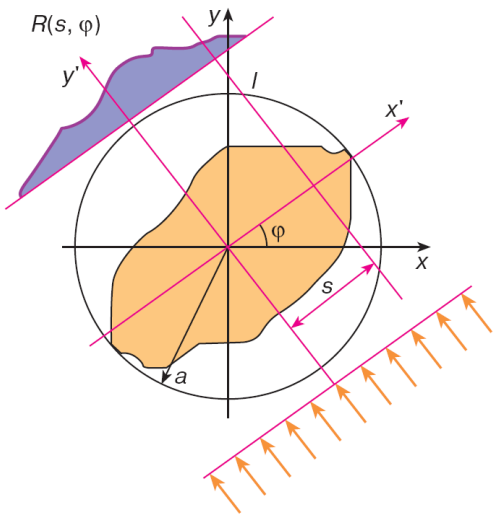


Рисунок 4.2 – Переріз об’єкту, промені та реєстрація інтенсивності

З протилежного боку об'єкту розташована лінійка датчиків випромінювання, що реєструє інтенсивність променів, які пройшли крізь об'єкт. За

просвічування об'єкта інтенсивність променя на виході дорівнює інтегралу функції розподілу щільності речовини вздовж траєкторії променя.

Задамо пряму на площині двома параметрами: s – відстань від прямої до початку координат, φ – кут між зовнішньою нормаллю до прямої і віссю абсцис. Рівняння прямої записується через параметри наступним чином:

$$x\cos\varphi + y\sin\varphi = s\,. \tag{4.2}$$

Тут x, y – координати точки на зображенні. Вираз (4.2) називається нормальним параметричним рівнянням прямої на площині. Він описує будь-яку пряму лінію за допомогою завдання кута до нормалі до цієї лінії і відстані від прямої до початку координат.

З урахуванням параметричного рівняння прямої (4.2) вираз для прямого перетворення Радона (4.1) буде мати вигляд:

$$R'\ s,\varphi\ = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f\ \ x,y\ \delta\ \ s - x\cos\varphi - y\sin\varphi\ \ dxdy\,. \tag{4.3}$$

Тут кожний вимір комп'ютерної томографії відповідає інтегралу від функції $f(x, y)$ вздовж одного з променів, а вся множина проекційних даних дорівнюватиме $R'\ s,\varphi\ $. Обмежимо кут межами від 0 до π, оскільки в інших випадках просвічування відбуватиметься в протилежному напрямку.

За допомогою перетворення Радона зображення представляється у вигляді набору проекцій уздовж різних напрямків. При цьому проекція функції двох змінних $f(x, y)$ є інтегралом у визначеному напрямку. Наприклад, інтеграл від $f(x, y)$ у вертикальному напрямку є проекцією $f(x, y)$ на вісь x; інтеграл в горизонтальному напрямку є проекцією на вісь y (Рисунок 4.3).

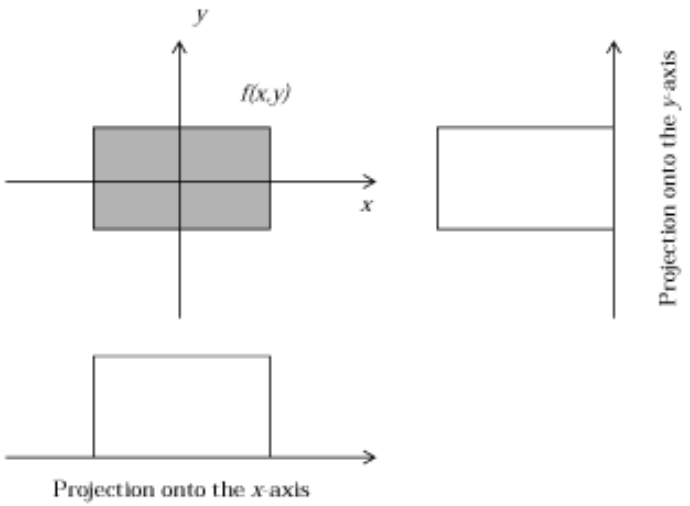


Рисунок 4.3 – Представлення зображення у вигляді набору проекцій

Проекції можуть бути обчислені вздовж прямої, що проходить під любим кутом φ. Так, проекція функції двох змінних $f(x, y)$ на вісь x' зада-

ється інтегралом

$$R\ \ s,\varphi\ \ = \int_{-\infty}^{\infty} f\,(x'\cos\varphi - y'\sin\varphi, x'\sin\varphi - y'\cos\varphi)dy'\,.$$

де осі x' і y' задаються поворотом проти годинникової стрілки на кут φ з використанням наступного виразу:

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\varphi & \sin\varphi \\ -\sin\varphi & \cos\varphi \end{bmatrix} \cdot \begin{bmatrix} x \\ y \end{bmatrix}$$

Геометричне уявлення перетворення Радона наведено на рисунку 4.4.

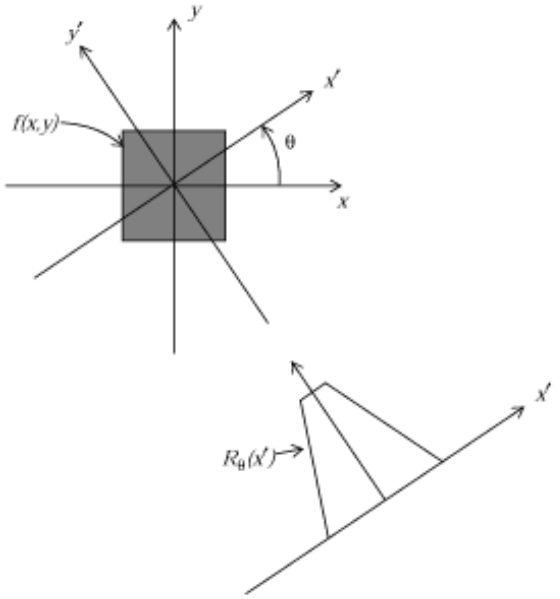


Рисунок 4.4 – Геометричне уявлення перетворення Радона

Перетворення Радона для великої кількості кутів найчастіше відображається у вигляді зображення-сінограмми. Наприклад, перетворення Радона для прямокутника при зміні φ від 0 до 180° з кроком 1° має вигляд, представлений на Рисунок 4.5.

Реконструкція зображення по проекціям еквівалентна находженню зворотного перетворення Радона. Реєстроване випромінювання (радоновський образ або проекція), обчислене під різними кутами, дозволяє за допомогою зворотного перетворення Радона відновити зображення поперечного перерізу об'єкта.

Основним алгоритмом відновлення вихідного зображення з сінограмми є алгоритм зворотної проекції з фільтрацією. Він складається з двох етапів: зворотного проектування і подальшої фільтрації.

Нехай $G\ \ \omega\ \ $ позначає перетворення Фур'є від деякої функції g :

$$G\ \ \omega\ \ = \int_{-\infty}^{+\infty} g\ \ x\ \ e^{-j2\pi\omega x} dx\,.$$

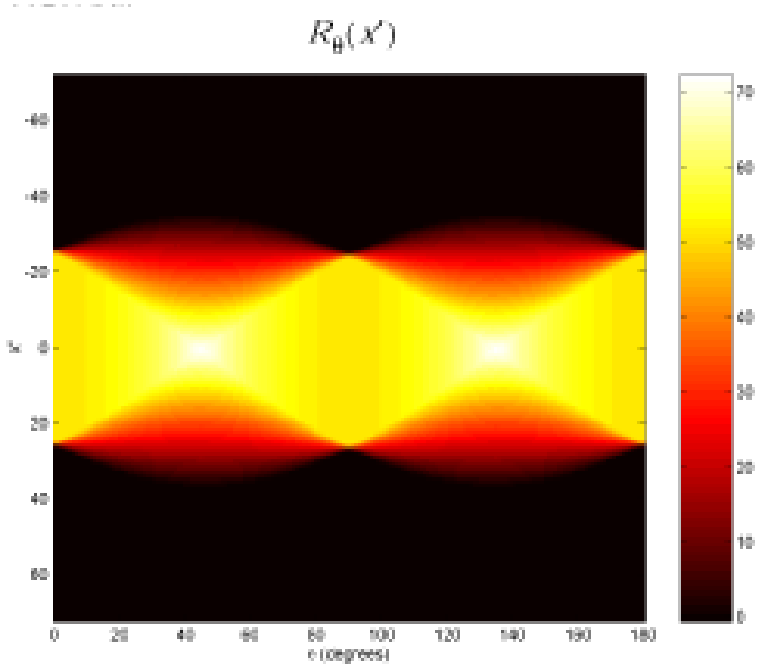


Рисунок 4.5 – Перетворення Радона для прямокутника

Зворотне перетворення Фур'є запишеться:

$$g\ \ x\ \ = \int_{-\infty}^{+\infty} G\ \ \omega\ \ e^{j2\pi\omega x} d\omega\,.$$

Двовимірне перетворення Фур'є деякої функції $g\ \ x,y\ \ $ має такий вигляд:

$$F\ \ \omega_x,\omega_y\ \ = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} g\ \ x,y\ e^{-j2\pi\ \omega_x x+\omega_y y}\ dxdy\,. \qquad (4.4)$$

На першому етапі знайдемо перетворення Фур'є від прямого перетворення Радона (4.3) по координаті s:

$$P\ \ \varphi,\omega\ \ = \int_{-\infty}^{+\infty} R'\ \ s,\varphi\ \ e^{-j2\pi\omega s} ds = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f\ \ x,y\ e^{-j2\pi\omega\ x\cos\varphi+y\sin\varphi}\ dxdy\,. \qquad (4.5)$$

Порівнюючи вирази (1.5) і (1.4) отримаємо наступну тотожність:

$$F\ \ \omega\cos\varphi,\omega\sin\varphi\ \ = P\ \ \varphi,\omega\ \ . \qquad (4.6)$$

відоме як **проекційна теорема**.

Геометричний сенс цієї теореми полягає в тому, що одновимірне перетворення Фур'є проекційних даних по змінній s еквівалентно двовимірному перетворенню Фур'є функції об'єкту, вираженої через полярні координати:

$$\omega_x, \omega_y = \omega\cos\varphi, \omega\sin\varphi \ .$$

Зворотне двовимірне перетворення Фур'є від F повинно давати відразу необхідне реконструйоване зображення. Однак на практиці такий прямий **алгоритм Фур'є** призводить до труднощів, пов'язаних з дискретизацією і спотвореннями в просторової області. В результаті на зображенні з'являються артефакти.

Для отримання двовимірного перетворення Фур'є від (4.6) щодо $\omega_x = \omega\cos\varphi$ і $\omega_y = \omega\sin\varphi$, перейдемо до змінних інтегрування ω і φ:

$$f\ x,y = \int_0^{\pi} d\varphi \int_{-\infty}^{+\infty} |\omega| P\ \varphi,\omega\ e^{2\pi j\omega s} d\omega \Big|_{s=x\cos\varphi+y\sin\varphi} . \qquad (4.7)$$

З формули (4.7) видно, що внутрішній інтеграл по $d\omega$ являє собою добуток двох множників, а саме: $|\omega|$ і фур'є-образу проекцій $P\ \varphi,\omega$. В просторової області це еквівалентно згортці двох функцій. Тому можна записати остаточний вираз для отримання реконструйованого зображення:

$$f\ x,y = \int_0^{\pi} P\ \varphi,\omega\ * k\ s\ d\varphi \Big|_{s=x\cos\varphi+y\sin\varphi} . \qquad (4.8)$$

де ядро виражається за формулою:

$$k\ s = \int |\omega| e^{2\pi j\omega s} d\omega = \frac{-1}{2\pi^2 s^2} .$$

З (4.8) випливає, що проекційні дані спочатку слід згорнути з ядром реконструкції *k*(*s*) (фільтром). Значення відновлюваної функції в точці (*x*, *y*) знаходять на другому кроці процедури, який полягає в інтегруванні результату згортки $p = p * k$ уздовж синусоїдальної кривої $s\ \varphi = x\cos\varphi + y\sin\varphi$ по куту φ в діапазоні від 0 до 180˚.

На практиці замість того, щоб вибирати точку (*x*, *y*), а потім брати інтеграл по всім проекціям φ, діють навпаки. Спочатку фіксують кут φ і для цієї проекції виконують розрахунки в циклі по всім пикселам (*x*, *y*), причому для кожної трійки (*x*, *y*, φ) обчислюють координату *s*(*x*, *y*, φ), що відповідає положенню детектора. Результати розрахунків використовують для знаходження необхідного значення згортки $p\ \varphi, s$, яке потім додається до пикселу з поточними координатами x, y . Цей процес називають зворотним проектуванням.

Реконструкція зображення відповідно до (4.8) складається з фільтрації і обчислення зворотної проекції, що і дало назву алгоритму.

4.3 Реалізація перетворення Радона в середовищі Matlab

4.3.1 Пряме перетворення Радона

Функція *R=radon*(*I, theta*) виконує перетворення Радона напівтонового зображення I і поміщає результат в матрицю проекцій *R*. Перетворення Радона є обчислення проекцій зображення на осі, що задаються кутами в градусах відносно горизонталі проти годинникової стрілки. Ці кути передаються в параметрі *theta*. Якщо *theta* – скаляр, то *R* є вектор-стовпець, що містить перетворення Радона для кута *theta*. Якщо *theta* є вектором, то *R* є матрицею, в якій кожен стовпець є перетворенням Радона для одного з кутів, що містяться в векторі *theta*. Якщо при виконанні функції параметр *theta* опущений, то в *theta* записуються значення кутів від 0 до 179° з кроком в 1°.

Функція *R=radon*(*I, theta, n*) виконує перетворення Радона півтонового зображення *I*. Значення кожної проекції обчислюються в *n* точках, і матриця *R* має *n* рядків.

Якщо додатково визначити вихідний параметр *xp*: [*R, xp*] = *radon* (...), то в нього поміщаються значення координат, в яких обчислювалися значення проекції. Значення в *k*-му рядку *R* відповідають координаті *xp* (*k*).

Матриця проекцій *R* може розглядатися як напівтонове або палітрове зображення і має формат представлення даних *double*.

Початкове півтонове зображення *I* розглядається як двовимірна функція. Початок координат в системі *x'y'* відповідає центральному пікселю зображення *I* в піксельній системі координат.

4.3.2 Зворотне перетворення Радона

Функція *I = iradon* (*P, theta*) виконує реконструкцію зображення *I* по його проекційним даним, які містяться в масиві *P*. Рядки *P* є даними точок проекцій.

Параметр *theta* описує кути (в градусах), під якими отримана кожна проекція. Цей параметр може являти собою вектор, що містить кути або скаляр, що описує кут між проекціями.

Функція *iradon* використовує алгоритм фільтрації зворотних проекцій для виконання інверсного перетворення Радона. Фільтр проектується безпосередньо в частотній області і множиться на функцію перетворення Фур'є проекцій. Для прискорення обчислень функції перетворення Фур'є проводяться спеціальні перетворення над проекціями.

Функція *I = iradon* (*P, theta, interp, filter, d, n*) містить опис параметрів, які використовуються при інверсних перетвореннях Фур'є. Існує також можливість точного визначення деяких комбінацій останніх чотирьох аргументів. Для втрачених параметрів функція *iradon* за замовчуванням встановлює деякі значення.

Параметр *interp* визначає тип інтерполяції. Наведемо список доступних опцій:

- '*nearest*' – інтерполяція по найближчій околиці;
- '*linear*' – лінійна інтерполяція (за замовчуванням);
- '*spline*' – сплайнова інтерполяція.

Параметр *filter* описує, який тип фільтра використовується для частотної фільтрації. Параметр *filter* є рядком, в якому описані кілька стандартних фільтрів:

- • '*Ram-Lak*' – усічений фільтр Рама-Лаку (встановлюється за умовчанням). АЧХ цього фільтра дорівнює | *f* |. Одним з недоліків фільтра Рама-Лаку є те, що він чутливий до шуму на проекціях. Тому він використовується в комбінаціях з іншими фільтрами.
- '*Shepp-Logan*' – фільтр Шепа-Логана, помножений на фільтр Рама-Лаку через фазову функцію.
- '*Cosine*' – косінусний фільтр, помножений на фільтр Рама-Лаку через косинусну функцію.
- '*Hamming*' – фільтр Хеммінга, помножений на фільтр Рама-Лаку через вікно Хеммінга.
- '*Hann*' – фільтр Ханна, помножений на фільтр Рама-Лаку через вікно Ханна.

Параметр *d* являє собою скаляр в діапазоні (0, 1] і служить для модифікації фільтра в плані масштабування по частотній осі. За замовчуванням він дорівнює 1. Коли *d* менше 1, тоді фільтр стискає частотний діапазон до [0, *d*], нормує частоти; всі частоти, які більше значення *d*, прирівнюються до 0.

Параметр n являє собою скаляр, що описує кількість рядків і стовпців у відновленому зображенні. Коли параметр n не описаний, тоді розміри визначаються, виходячи з довжини проекцій.

Після визначення параметра *n*, функція *iradon* відновлює зображення, не змінюючи масштаб даних. Коли проекції були обчислені за допомогою функції *radon*, тоді розміри відновленого і вихідного зображень можуть не збігатися.

Функція *iradon* використовує алгоритм фільтрації зворотних проекцій для реалізації зворотного перетворення Радона. Фільтр проектується для роботи в частотної області і далі множиться на функцію перетворення Фур'є проекційних даних. Для прискорення перетворення Фур'є з проекційними даними проводиться деяка попередня обробка.

4.3.3 Корисні функції

radon, iradon, imshow, colorbar, colormap, imagesc, imread, imwrite, phantom, etime, clock, tic, toc

4.4 Контрольні запитання

1. Визначити поняття прямого перетворення Радона.
2. Визначити поняття зворотного перетворення Радона.
3. Визначити поняття перетворення Гільберта.

4. Записати вираз для прямого перетворення Радона з урахуванням параметричного рівняння прямої.
5. Пояснити принцип використання перетворення Радона в комп'ютерній томографії.
6. Показати, яким чином можна обчислити проекції вздовж прямої, що проходить під любим кутом.
7. Навести геометричне уявлення перетворення Радона та пояснити поняття сінограмми.
8. Записати вирази для прямого та зворотного перетворення Фур'є для одно- та двовимірної функції.
9. Записати вираз для перетворення Фур'є від прямого перетворення Радона за координатою *s*.
10. Пояснити поняття «проекційна теорема».
11. Записати вираз для отримання реконструйованого зображення.
12. Що таке зворотне проектування? З якою метою застосовується частотна фільтрація?
13. Які засоби середовища *Matlab* застосовують для прямого та зворотного перетворення Радона?
14. З'ясувати можливості корисних функцій та коротко описати їхнє призначення та параметри. Навести приклади використання.

4.5 Порядок виконання роботи

4.5.1 Підготовча стадія

1. Уважно вивчити теоретичну інформацію про перетворення Радона та його реалізацію в середовищі *Matlab*, користаючись матеріалами перших підрозділів цієї роботи та рекомендованими джерелами.

2. Підготувати проект звіту з роботи в складі титульного аркуша, та письмових відповідей на контрольні запитання.

4.5.2 Дослідження перетворень Радона на моделі томограми головного мозку

1. Виконати пряме перетворення Радона моделі томографічного зрізу (рисунок 4.6), вибравши кути проекцій від 0 до 180 градусів.

2. Відновити зображення за допомогою зворотного перетворювання Радона за всіма значеннями проекцій. Порівняти отримане зображення з вхідним.

3. Побудувати залежність часу відновлення зображення від кількості використаних проекцій.

4. Дослідити вплив параметрів зворотного перетворення Радона на схожість відновленого зображення з оригіналом для п'яти комбінацій виду інтерполяції і типу фільтра по семи значенням похибки для випадку використання максимальної кількості проекцій.

5. Побудувати залежності значень однієї з похибок відновлення від

кількості проекцій для п'яти комбінацій виду інтерполяції і типу фільтра.

6. * Побудувати графік залежності похибки відновлення від кількості проекцій і кількості пікселів зображення для трьох комбінацій виду інтерполяції і типу фільтра.

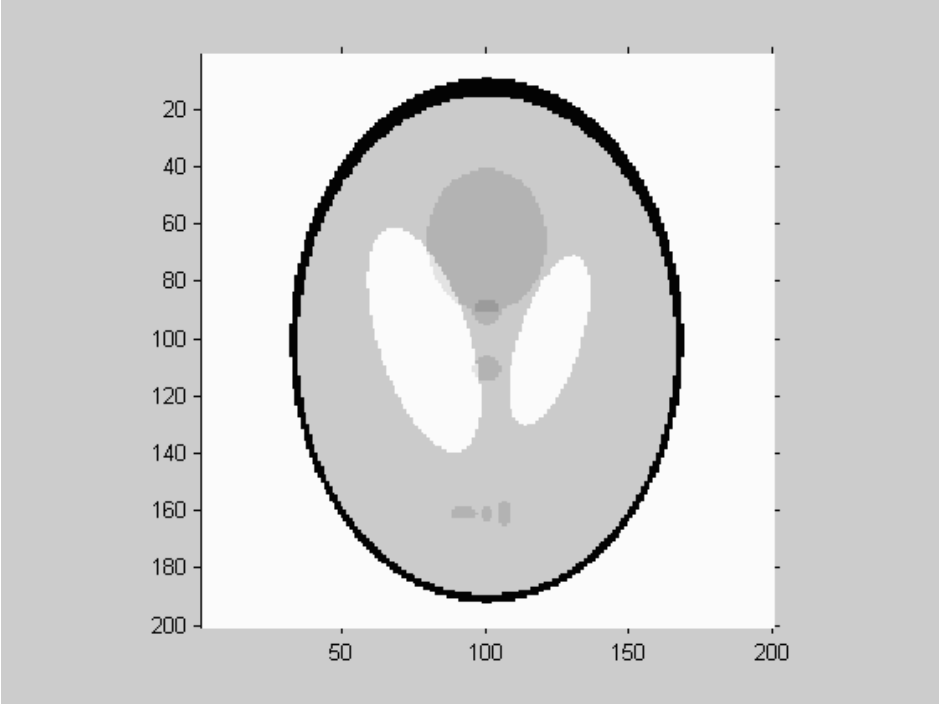


Рисунок 4.6 – Початкове зображення «фантом», що імітує зріз головного мозку

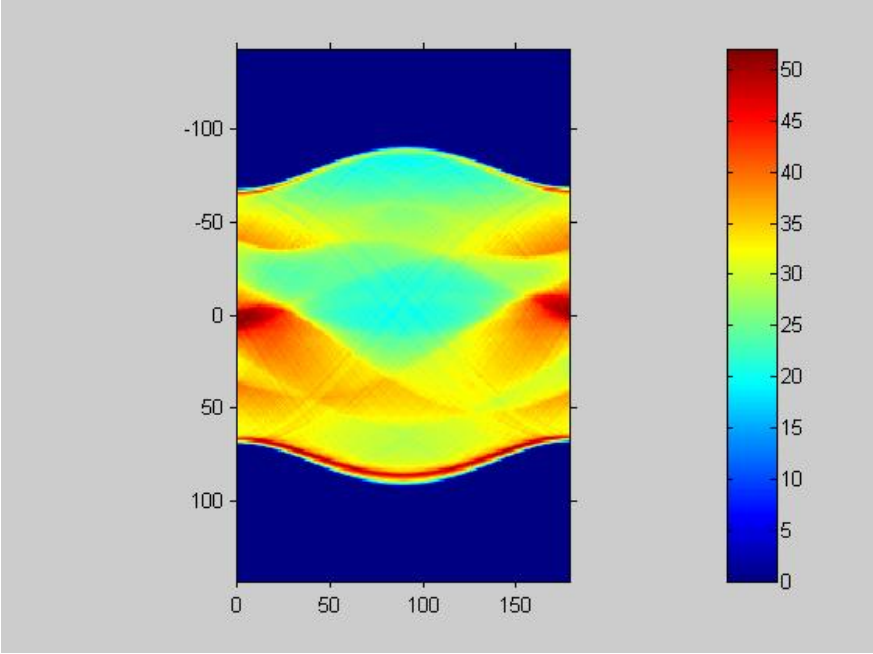


Рисунок 4.7 – Зображення після виконання прямого перетворення Радона з проекціями через 1º

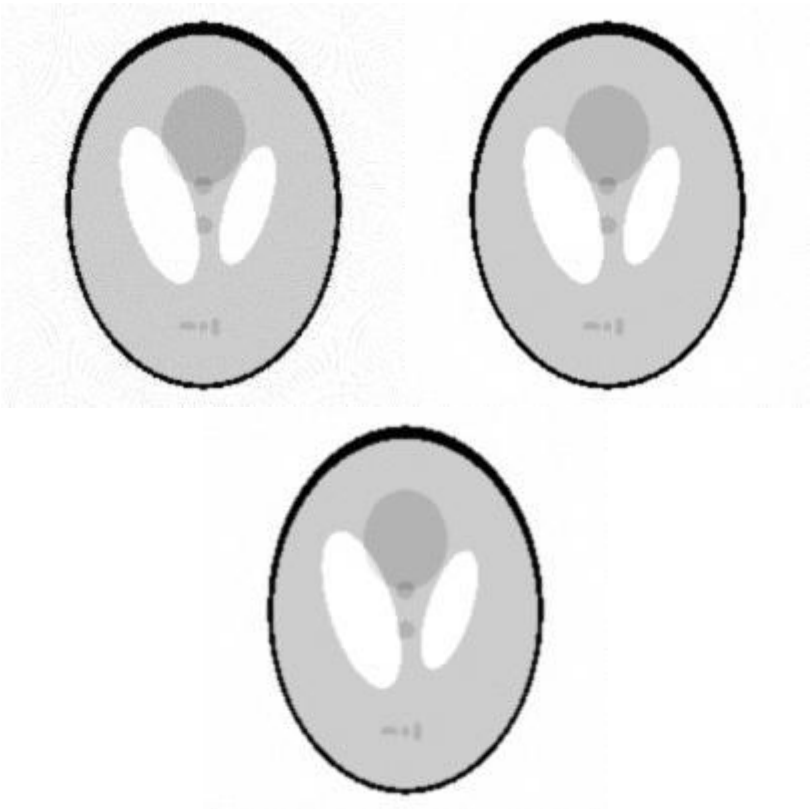


Рисунок 4.8 – Відновлені зображення

4.5.3 Перетворення реального біомедичного зображення

1. Виконати пряме і зворотне перетворення Радона, використовуючи реальне самостійно знайдене біомедичне зображення, при необхідності попередньо перетворивши його в напівтонове. Використовувати п'ять різних комбінацій фільтрів і методів інтерполяції. Зробити висновки про часові витрати і погрішності відновлення зображення.

Наприклад, в якості реального зображення можна взяти перетин головного мозку людини, отриманий неподалік від середньої точки (Рисунок 4.9). При виконанні прямого перетворення Радона формується набір даних, (в графічному вигляді він відображений на Рисунок 4.10), одержуваний при скануванні голови людини за допомогою томографа. Далі, використовуючи найкращі способи і параметри, можна відновити зображення так, як його відновлює апаратура комп'ютерного томографа (Рисунок 4.11).

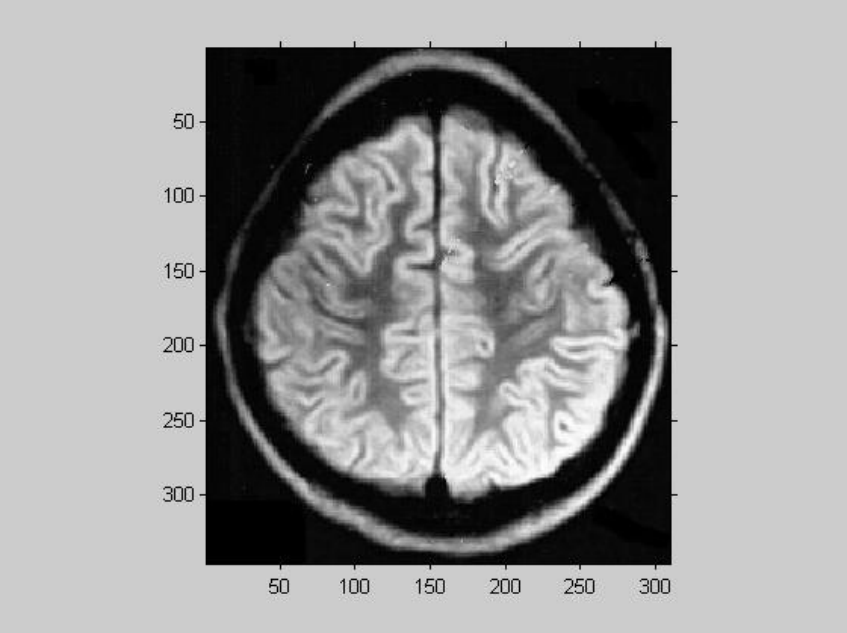


Рисунок 4.9 – Початкове зображення зрізу головного мозку

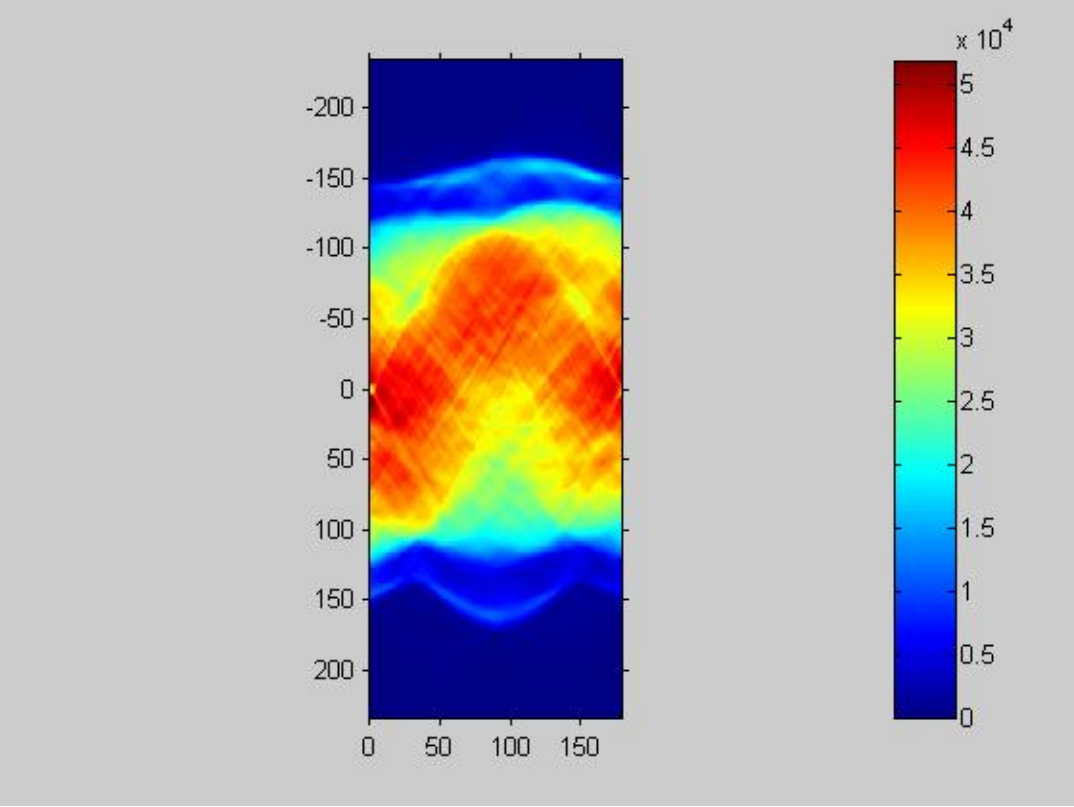


Рисунок 4.10 – Набір даних в графічному вигляді, що отримується при КТ

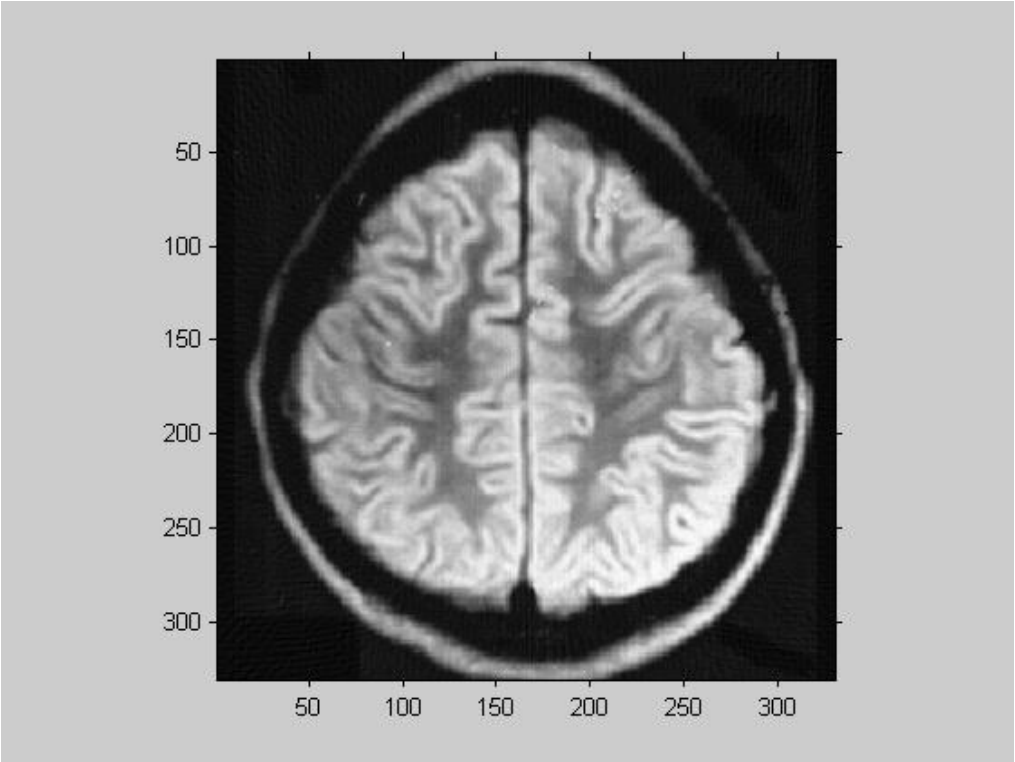


Рисунок 4.11 – Відновлене зображення зрізу головного мозку

4.5.4 Оцінка похибки відновлення зображення

1 Середньоквадратична відносна похибка

$$\delta_1 = \left| \frac{\sum_{n=1}^{M} (\mu_n - \hat{\mu}_n)^2}{\sum_{n=1}^{M} (\mu_n)^2} \right|^{1/2} \times 100\%$$

2 Нормалізована середньоквадратична похибка

$$\delta_2 = \left| \frac{\sum_{n=1}^{M} (\mu_n - \hat{\mu}_n)^2}{\sum_{n=1}^{M} (\mu_n - \mu_{cp})^2} \right|^{1/2} \times 100\%$$

3 Максимальна похибка

$$\delta_3 = \max|\mu_n - \hat{\mu}_n|$$

4 Максимальна відносна похибка

$$\delta_4 = \frac{\max|\mu_n - \hat{\mu}_n|}{\max(\mu_n)} \times 100\%$$

5 Нормалізована абсолютна середня похибка

$$\delta_5 = \frac{\sum_{n=1}^{M} |\mu_n - \hat{\mu}_n|}{\sum_{n=1}^{M} |\mu_n|} \times 100\%$$

6 Абсолютна середня похибка

$$\Delta = \sum_{n=1}^{M} |\mu_n - \hat{\mu}_m|$$

7 Середньоквадратична абсолютна похибка

$$\delta_{cp} = \sqrt{\sum_{n=1}^{M} (\mu_n - \hat{\mu}_n)^2}$$

Рекомендована література

1. The MathWorks Support [Електронний ресурс]. – Режим доступу: http://www.mathworks.com/support/

2. GNU Octave. Scientific Programming Language [Електронний ресурс]. – Режим доступу: http://www.gnu.org/software/octave/

3. Image Processing Toolbox [Електронний ресурс]. – Режим доступу : https://www.mathworks.com/help/images/index.html

4. Гонсалес Р., Вудс Р. Цифровая обработка изображений. М.: Техносфера, 2012. – 1072 с.

5. Гонсалес Р., Вудс Р., Эддинс С. Цифровая обработка изображений в среде MATLAB. М.: Техносфера, 2006. – 621 с.

6. Gonzalez R. C., Woods R. E. Digital Image Processing. Prentice Hall, 2002. – 813 p.

7. Уэбб. Физика визуализации медицинских изображений. Т.1, т.2. – Мир, 1991.

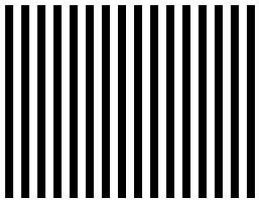
Додатки

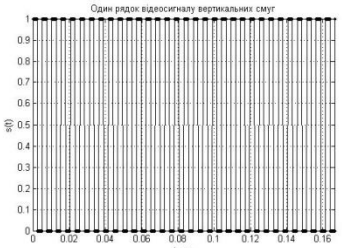
Додаток А
Палітри кольорів на рисунках у Matlab

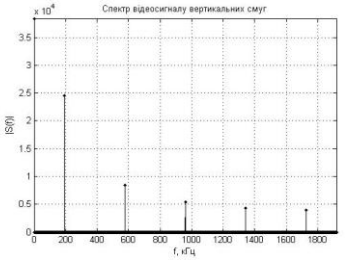
Colormap Name	Color Scale
parula	
jet	
hsv	
hot	
cool	
spring	
summer	
autumn	
winter	
gray	
bone	
copper	
pink	
lines	
colorcube	
prism	
flag	
white	

Додаток Б
Приклади скрін-шотів до лабораторної роботи №3

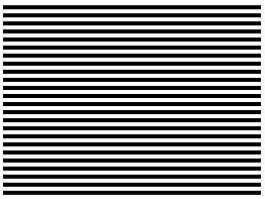
Вертикальні смуги

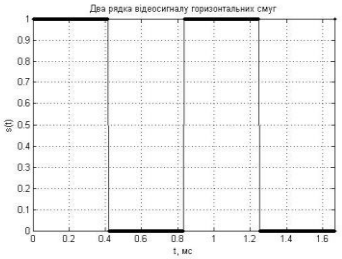


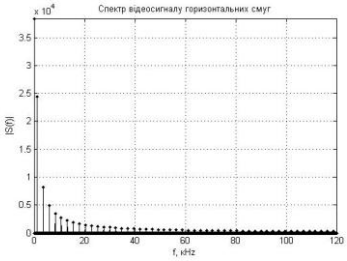




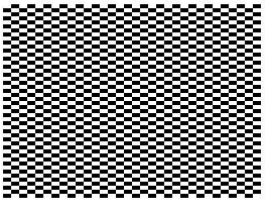
Горизонтальні смуги

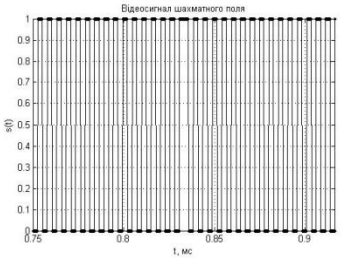


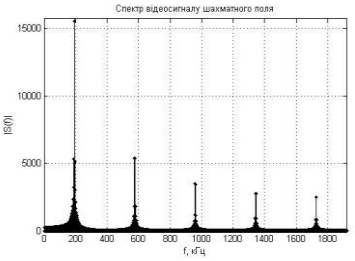




Шахматне поле







Фільтроване зображення