НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
БІОРЕСУРСІВ І ПРИРОДОКОРИСТУВАННЯ

Смолій В.В., Савицька Я.А.,
Місюра М.Д., Шкарупило В.В.

СИСТЕМИ ВІЗУАЛІЗАЦІЇ ТА РОЗПІЗНАВАННЯ ОБРАЗІВ

Навчальний посібник

КИЇВ, НУБіП
2020

УДК 004.45(072)
ББК 32,97
П 69

Рекомендовано до видання рішенням вченої ради Національного університету біоресурсів і природокористування України (Протокол № _4_ від 25 листопада 2020 року)

Рецензенти:
Лахно В.А., доктор технічних наук, професор, завідувач кафедри комп'ютерних систем і мереж Національного університету біоресурсів і природокористування України
Чумаченко С.М., доктор технічних наук, професор, завідувач кафедри інформаційних систем Національного університету харчових технологій
Ролік О.І. доктор технічних наук, професор, завідувач кафедри автоматики та управління в технічних системах, Національний технічний університет України "Київський політехнічний інститут імені Ігоря Сікорського"

Навчальний посібник з дисципліни Системи візуалізації та розпізнавання образів [навчальний посібник] / Смолій В.В., Савицька Я.А., Місюра М.Д., Шкарупило В.В. // - К.: ФОП Ямчинський О.В., 2020.- 200 с.

ISBN 978-617-7986-40-9

Навчальний посібник призначений для студентів вищих навчальних закладів ОС «Магістр», які навчаються за спеціальністю 123 – Комп'ютерна інженерія. Розділи посібника охоплюють основні питання зі створення систем здатних відтворювати графічну інформацію, вводити її у комп'ютерні системи та обробляти з метою покращення чи розпізнавання.

Навчальний посібник відповідає програмі дисципліни "Системи візуалізації і розпізнавання образів".

УДК 004.45(072)

ISBN 978-617-7986-40-9

ВІДОМОСТІ ПРО АВТОРІВ

Смолій Віктор Вікторович
Доцент, кандидат технічних наук, викладає ряд профільних дисциплін на кафедрі комп'ютерних систем і мереж Національного університету біоресурсів і природокористування України. Автор більш п'ятдесяти наукових та навчально-методичних праць, 2 з яких монографії. Захистив дисертацію за темою «Методы и средства синтеза и отображения динамических объектов (для центров оперативного управления)».

Наукові напрями роботи:

- геоінформаційні комплекси реального часу;
- геоінформаційні системи аграрного застосування;
- вбудовані мікроконтролерні засоби інтернету речей.

Савицька Яна Артурівна
Доцент, кандидат технічних наук, викладає ряд профільних дисциплін на кафедрі комп'ютерних систем і мереж Національного університету біоресурсів і природокористування України. Є автором понад тридцяти наукових робіт у області інформаційних технологій, з яких 1 монографія, методичних розробок, трьох авторських свідоцтв.

Наукові напрями роботи:

- системи автоматизації освітніх процесів;
- комп'ютерні системи контролю та управління для вугільної промисловості;
- технічні засоби захисту інформації.

Місюра Максим Дмитрович

Кандидат технічних наук, захистив кандидатську дисертацію у листопаді 2010 році за спеціальністю 05.13.07 - автоматизація процесів керування (рішення президії Вищої атестаційної комісії України від 26 січня 2011 року, тема: "Автоматизоване управління технологічним комплексом виробництва пива"). Основні напрями наукових досліджень: комп'ютерні системи, інтелектуальні системи управління, автоматизовані системи управління, комп'ютерні мережі. Викладає дисципліни: Комп'ютерні системи, Технології проектування комп'ютерних систем, Програмування в середовищі сучасних ОС, Технології програмування комп'ютерних систем, Технології проектування систем IoT.

Шкарупило Вадим Вікторович

Доцент, кандидат технічних наук. У 2014 р. захистив дисертацію з технічних наук на тему «Розробка й дослідження моделей і методів специфікації, верифікації і валідації композитних веб-сервісів» (спеціальність 05.13.05 – компютерні системи та компоненти). Викладає дисципліни: “Мікропроцесорні системи керування”, “Системне програмування», “Протоколи передачі даних в IoT системах”. Основні напрямки наукових досліджень:

- Розробка й дослідження моделей і методів специфікації, верифікації і валідації композитних веб-сервісів;
- автоматизовані системи передачі даних в IoT системах;
- програмні засоби в комп'ютерних системах.

ЗМІСТ

ПЕРЕДМОВА

Дисципліна «Системи візуалізації та розпізнавання образів» є різноплановою та включає у себе дуже теоретичних та практичних завдань. Як слідує із назви, вона охоплює широке коло задач, пов'язаних з відтворенням у графічній формі різноманітних користувацьких даних – від звичайних фотографій до складних всесвітів віртуальної реальності, і від поліпшення зображень до створення систем, здатних ідентифікувати та розпізнавати об'єкти у оточенні людини.

Дисципліна є однією з базових дисциплін за спеціальністю 123 «Комп'ютерна інженерія» та включена до переліку дисциплін за вибором ОС «Магістр».

Основні цілі видання полягають у:

- ознайомленні студентів з теоретичними матеріалами з принципів побудови систем візуалізації та розпізнавання образів;
- оволодінні практичними навичками з аналізу вимог та виборі відповідних технологій і програмно-технічних рішень при проведенні проектних робіт;
- отриманні практичних навичок з використання інструментальних засобів мов високого рівня для вирішення задач синтезу і аналізу графічних даних;
- ґрунтовному розумінні конвеєру перетворення графічної інформації у сучасних комп'ютерних системах.

Відповідно до цього, наданий матеріал містить інформацію що пов'язує базові теоретичні поняття та постулати із практичними задачами відтворення зображень у комп'ютерних системах.

Місюра М.Д. виклав передмову, матеріали за темою «Робота з графікою у мові С» викладено Шкарупило В.В., Смолій В.В. є розробником РНП з дисципліни, структури та складу викладеної тематики та сумісно з Савицькою Я.А. автором основної частини роботи.

РОЗДІЛ 1. АЛГОРИТМІЧНЕ ТА ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ СИСТЕМ КОМП'ЮТЕРНОЇ ГРАФІКИ

Тема 1.1. Інструментальні засоби формування зображень у комп'ютерній графіці

Мета: навчитися використовувати програмні інструменти мов високого рівня при вирішенні задач синтезу графічних даних та закріпити розуміння організації роботи програмних компонентів графічних підсистем комп'ютерних засобів.

1.1.1. Загальні теоретичні відомості. Графічні примітиви у мовах високого рівня

У основі роботи усіх інтерактивних систем передбачається процес виведення та вводу інформації, які у сучасних системах будуються графічними засобами. Складність цих процесів обумовлюється кількістю елементів, які потребують обробки та візуалізації, та ступенем інтерактивності чи реалістичності відтворення об'єктів. Але на нижніх рівнях реалізації задач відтворення зображень чи вводу та формування інформації про графічні компоненти зображень, завжди використовують досить незначний набір основних елементів, які називають «графічними примітивами».

Оскільки виведення графічних зображень пов'язано у першу чергу з формуванням у частці оперативної пам'яті комп'ютера яка виділяється для зберігання даних відеоадаптеру та зветься відео-пам’яттю «цифрового еквіваленту» зображення, а вже потім виводиться на екран апаратними засобами відеоадаптеру та монітору, у операційних системах чи системах керування комп'ютерними засобами виділяють відповідну програмну компоненту. Так, наприклад, у операційній системі Windows це «графічне ядро», яке відповідає за усі процеси відтворення візуальних даних – GDI (Graphics Device Interface). Крім нього NTOSKRNL.EXE та WIN*.SYS забезпечують функціонування самої ОС та взаємодію між задачами та

процесами, які виконуються. За необхідності візуалізації, NTOSKRNL через WIN*.SYS звертається до GDI, яка містить компоненти для відтворення графічних примітивів, та виконує це командами, які вже перетворюються в WIN*.SYS за допомогою драйверів пристроїв у конкретні низькорівневі операції з вводу та виводу даних.

У операційних системах типу UNIX/Linux, можна сказати, графічні компоненти взагалі не включені в ядро системи. В них відтворення відбувається високорівневими наборами утиліт (оболонок KDE, GNome та інші), які для візуалізації використовують графічні бібліотеки (найчастіше GTK та QT) та X-Window System (X.Org Server) [1], яка займається будовою найпростіших графічних зображень, та подібно до GDI, здійснює це за допомогою спеціалізованих команд через відповідні інтерфейси бібліотек. У процесі візуалізації використано [2] багатошарову модель «device->device driver->frame buffer->X server->X protocol» взаємодії пристроїв та процесів. При цьому, X-Window Server може взаємодіяти з бібліотеками та застосунками навіть не в рамках однієї системи, використовуючи для взаємодії (і віддаленої також) X-протокол. Кожен додаток, який потребує візуалізації, у такій системі є X-клієнтом (рис. 1.1).

Слід відзначити, що у такій концепції, для графічних оболонок типу KDE кожна компонента у системі повинна мати якийсь опис, що формує візуальну «модель» процесу чи застосунку, а у процесі відтворення цієї моделі відповідні операції повинні виконуватися у певній послідовності, що відповідає структурі «дисплейного файлу» [3]. У склад такої конструкції включають команди, які повинні описати кінцеве зображення у заздалегідь визначених термінах – примітивах.

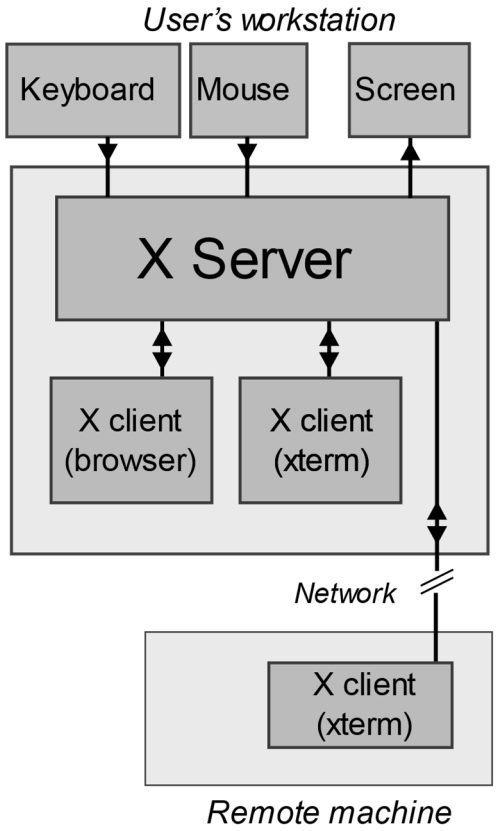


Рис. 1.1. Структура X-Window System

Графічний примітив [4] – це «базовий неділимий графічний елемент для введення або виведення в комп'ютерній графічній системі. Типовими вихідними примітивами є точка, лінія, полілінія, область заповнення.

Відсікання вихідного примітиву не може гарантувати отримання іншого вихідного примітиву. Вихідні примітиви мають такі атрибути, наприклад, як стиль лінії або шаблон.

Типовими вхідними примітивами є локатор, вибір та оцінювач. Вхідні примітиви часто мають стиль відлуння, пов'язаний з ними.»

Архітектурні принципи організації графічних систем обумовлюють відповідність найменшого елементу зображення на екрані деякій сукупності комірок пам'яті у обчислювальній системі. Основною задачею графічної підсистеми у операційних

системах, відповідно, є визначення яку інформацію потрібно занести у ці комірки, чи визначити також і їх адреси, що відповідає задачам «розгортки» графічних примітивів.

На сьогодні, більшість цих процесів є прихованими не тільки від кінцевого користувача, але й для розробника. Він також оперує графічними примітивами для того, щоб створити відповідні елементи інтерфейсів чи «скласти» зображення, використовуючи відповідні елементи мови програмування. Володіння таким інструментарієм є, відповідно, нагальною необхідністю.

Коротко розглянемо концептуальну модель з точки зору трьох основних дій, що виконуються в інтерактивній графіці.

Побудова прикладної моделі. Перш за все, прикладний програміст будує прикладну модель об'єктів, з якими користувач буде працювати і які він буде розглядати. Прикладна модель (надалі просто модель) є сукупністю даних, що представляють об'єкти, і відносин цих даних. Зазвичай зберігається у вигляді прикладної структури даних. Таким чином, модель відображає важливі властивості об'єктів, істотні для цього додатка або для ряду додатків. Ці об'єкти можуть бути конкретними або абстрактними і при візуалізації можуть бути представлені в двовимірному або тривимірному світі. Прикладами можуть служити математична функція, інтегральна схема, план приміщення, зубчата передача, молекула.

Користувач зазвичай бажає мати візуальне уявлення моделі або окремих аспектів моделі. Прикладна графічна програма повинна описати для графічної системи в геометричних поняттях ту частину світу, зображення якого потрібно користувачеві.

Незалежно від змісту бази даних (геометричні або негеометричні дані) вони повинні бути описані для графічної системи у вигляді вихідних графічних примітивів, таких як точки, лінії, полігони, ланцюжки літер і т.д.

Прикладна програма повинна також вказати графічній системі яку частину об'єкта слід показати. Цю частину прийнято називати «Вікном» («Window», рис. 1.2).

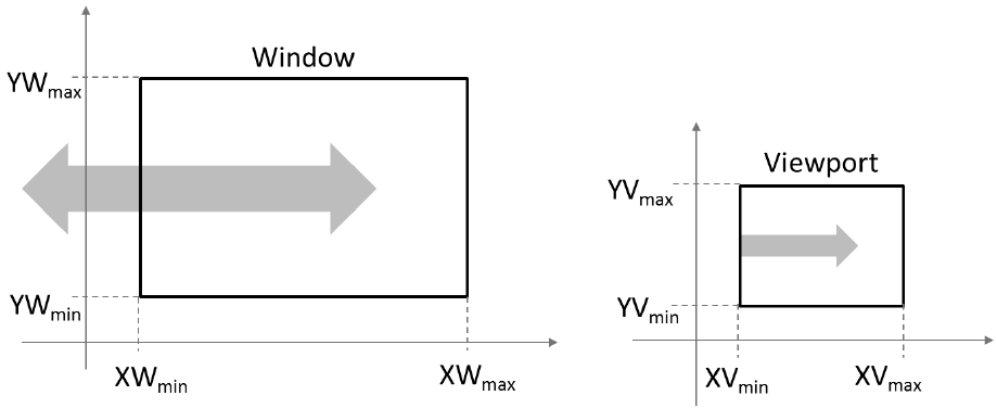


Рис. 1.2. – Принцип дії «Вікна» систем КГ

При будові такого опису потрібно вказати, в якій точці знаходиться спостерігач, звідки направлено освітлення, на якій частині видової поверхні («Viewport») має з'явитися зображення.

Графічну систему зручно розглядати як уявну фотокамеру [5].

Прикладна програма надає цій фотокамері опис сцени, що складається з одного або більше об'єктів у деякому штучному світі – двовимірному або тривимірному.

Потім уявна камера створює вигляд цього об'єкта в цьому світі. Точний вид об'єкта залежить від того, як була налаштована фотокамера, де вона перебувала по відношенню до об'єкту, де знаходилося джерело освітлення. Зроблений знімок відразу проявляється і показується на видовій поверхні. Величина знімка може бути різною і займати всю видову поверхню або її частину.

Координатні простори. Щоб отримати зображення прикладна програма повинна описати для графічної системи різні вихідні примітиви, вказавши їх положення і розміри в прямокутній системі координат. Ці координати за своєю природою є безрозмірними, тому прикладна програма може задавати об'єкти в одиницях, які є природними для цього додатка і для користувача. Так, наприклад, розміри деталі можуть задаватися в міліметрах, графіки зміни температури від часу (секунди, хвилини або години). Ці прикладні координати є

координатами моделі (MC – Modelling Coordinates, рис. 1.3). Оскільки об'єкт потім переноситься у простір з якимось оточенням, яке відображається в двовимірному або тривимірному світі користувача, їх називають світовими координатами (WC – World Coordinates). Стандарт ISO 2382 (ДСТУ ISO/IEC 2382:2017 – «Інформаційні технології. Словник термінів») визначає світові координати як "незалежні від пристрою декартові координати, використовувані в програмах користувача для задання вхідних і вихідних даних". Таким чином, об'єкти можуть бути задані в світовій системі координат як для зберігання в прикладній моделі, так і з метою опису для графічного пакета (ГП). Світові координати задаються зазвичай системою числення з плаваючою комою.

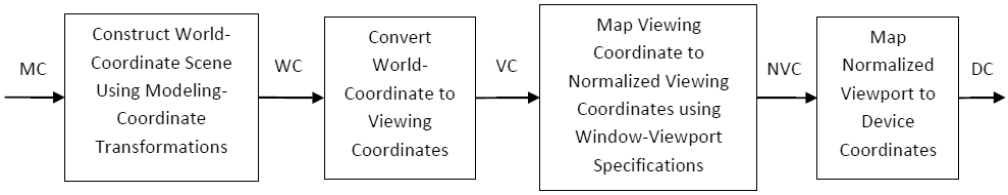


Рис. 1.3. Конвеєр перетворення координат об’єктів

Після того, як об'єкти описані для графічного пакета у вигляді сукупності вихідних примітивів, графічний процесор формує зображення об'єктів на екрані, відображаючи спочатку їх на незалежні від пристрою світові координати на логічній системі координат (Viewing Coordinates, VC), а потім у нормовані координати, в яких видова поверхня нормується дійсними числами від 0 до 1 як по координаті X, так і по координаті Y (Normalized Viewing Coordinates, NVC). За стандартом ISO 2382 це "координати, задані в проміжній незалежній від пристрою системі координат і нормовані щодо деякого діапазону, зазвичай від нуля до одиниці". Зображення, виражене в нормованих координатах, розташовується в одних і тих же відносних позиціях при візуалізації на будь-якому пристрої. Початок координат (X = 0, Y = 0) розташовується в лівому нижньому (верхньому) куті видової поверхні.

І, нарешті, фізичні координати пристрою (Device Coordinates, DC) – це координати, що залежать від пристрою і задаються в елементах зображення – пікселях (дисплей), міліметрах (графічний пристрій) і так далі.

В процесі роботи, користувач має справу з зображенням об'єктів. У найпростішому випадку користувач хоче просто поглянути на об'єкт під іншим кутом (повернути об'єкт). Крім того, він може і перетворити об'єкт, тобто змінити його. Незалежно від його намірів, для повідомлення цих намірів прикладній програмі, повинні використовуватися пристрої введення, а режим роботи з комп'ютерними засобами стає *інтерактивним*. Прикладна програма, розшифрувавши цю інформацію, застосовує її або для управління графічною системою з метою зміни параметрів видової операції (зміна настройки фотокамери), або для зміни моделі в структурі даних. Після модифікації структури даних прикладна програма дає команду графічній системі вивести відредаговане зображення на екран. Користувач може задати переміщення зображення на екрані. Зробити це можна після обробки прикладною програмою запиту введення і подачі сигналу графічної системи на виконання зазначеної дії. Інтерактивний діалог виду "відображення - введення - відображення..." (пінг-понгова модель) між користувачем і прикладною програмою становить *суть інтерактивної графіки*.

Робота прикладної програми полягає в моделюванні та інтерпретації даних, що вводяться користувачем. Графічна система не будує і не змінює модель ні на початку роботи, ні у відповідь на інтерактивні дії користувача. Вона як фотокамера тільки робить знімки.

Прогрес в області виробництва апаратури в значній мірі змінив відношення до систем машинної графіки. Такий, колись дорогий засіб, як графічний дисплей, тепер використовують повсюдно як основний засіб зв'язку людини з ЕОМ. Такий же прогрес спостерігається і в області програмного забезпечення КГ. Процес вдосконалення програмного забезпечення був тривалим і повільним. Було пройдено шлях від апаратно-

залежних пакетів підпрограм низького рівня, що поставляються виробниками разом з конкретними дисплеями, до апаратно-незалежних пакетів високого рівня.

Основна мета апаратно-незалежного пакета, використовуваного в поєднанні з язиками високого рівня (FORTRAN, PASCAL, C) полягає в тому, щоб забезпечити мобільність прикладної програми при переході з однієї ЕОМ на іншу і від одного графічного пристрою до іншого.

Істотний крок вперед зроблений в середині 70-х років з введенням "Графічної системи Core"[5], а потім групи "Графічних стандартів ISO", що мають стандарт ядра GKS [6]. В даний час створено безліч графічних додатків, серед яких можуть бути відзначені такі пакети як COREL DRAW, технологічні AutoCAD чи P-CAD.

У мови високого рівня включені спеціальні графічні модулі, які можуть бути інструментом як для створення прикладних графічних пакетів, так і для створення окремих графічних підпрограм. Далі розглянемо ці компоненти та їх властивості більш детально, але усі вони мають деякі основні властивості, характерні для більшості графічних пакетів і графічних модулів.

Більшість мов програмування високого рівня на сьогодні має відповідні компоненти, об'єднані у спеціалізовані бібліотеки.

1.1.2. Робота з графікою у мові С

Для здійснення графічного виводу, потрібно мати елемент, який дозволяє це зробити, тобто має зв'язок із частиною кадрової оперативної пам'яті. У мові С це «графічний контекст» [7], при чому, принципи використання є однаковими у ОС Windows та UNIX/Linux. Для того, щоб використовувати графічні можливості системи, потрібно активувати цей режим. Усі необхідні дії та доступні операції реалізовані відповідними процедурами та функціями з бібліотечного модуля graphics. Відповідно, потрібно встановити та підключити цю бібліотеку.

У ОС Windows це можна здійснити скачавши архів бібліотеки (наприклад, WinBGIM для Windows 7), розгорнувши його та прописавши відповідні шляхи у системі або параметрами для компілятора. У ОС Linux це можна зробити, виконавши дії, наведені у наступному переліку:

```
>sudo apt-get install build-essential
>sudo apt-get install libsdl-image1.2 libsdl-image1.2-dev guile-2.0
\ guile-2.0-dev libsdl1.2debian libart-2.0-dev libaudiofile-dev
\ libesd0-dev libdirectfb-dev libdirectfb-extra libfreetype6-dev
\ libxext-dev x11proto-xext-dev libfreetype6 libaa1 libaa1-dev
\ libslang2-dev libasound2 libasound2-dev
>sudo make install
sudo cp /usr/local/lib/libgraph.* /usr/lib
```

Приклад використання графічних операцій та порядку їх виконання може бути наступним:

```
#include <graphics.h>
int main()
{
  int graphcontext = DETECT, graphmode;
  int x = 120, y = 40, rdius;

  initgraph(&graphcontext, & graphmode, "C:\\TC\\BGI");

  for ( rdius = 25; rdius <= 125 ; rdius += 5)
    circle(x, y, rdius);

  getch();
  closegraph();
  return 0;
}
```

Розглянемо детальніше цей код. Функція *“initgraph”* використовується для ініціалізації дисплея в графічному режимі. Ця функція описана у файлі заголовку “*graphics.h*”. Отже, цей файл включений до програми перед виконанням функції “*initgraph*”. Синтаксис функції наступний:

initgraph(&driver, &mode, “path”);

де *driver* представляє графічний драйвер, встановлений на комп'ютері. Це може бути ціла змінна або цілочисельний ідентифікатор константи, наприклад, CGA, EGA, SVGA тощо. Графічний драйвер також може бути автоматично виявлений за допомогою ключового слова “DETECT”. Компілятору дозволяється виявляти графічний відомий драйвер автоматично. Якщо драйвер повинен бути автоматично виявлений, драйвер змінної оголошується цілим числом і йому присвоюється значення DETECT, як показано далі:

```
int driver, mode;
driver = DETECT;
```

Ця конструкція повинна бути розміщена перед функцією “*initgraph*”. Коли виконуються вищезазначені оператори, комп'ютер автоматично визначає графічний драйвер та графічний режим.

Змінна *mode* представляє вихідну розподільчу здатність на екрані комп'ютера. Звичайним режимом для VGA є VGAHI. Це дає змогу використовувати найвищу роздільну здатність.

Елемент *path* представляє шлях графічних драйверів. Це каталог, де знаходяться файли бібліотеки та драйверів. Припустимо, файли бібліотеки BGI (Borland Graphics Interface) зберігаються в “C:\TC\BGI”, тоді повний шлях записується як:

initgraph (VGA, VGAHI, “C:\TC\\BGI”);

Слід зазначити використання подвійної косої риски «\». Одна зворотна коса риса використовується як символ екранування, а інша - для шляху до каталогу. Якщо файли BGI знаходяться в поточному каталозі, тоді шлях дозволяється записувати як:

initgraph(VGA, VGAHI, “”);

Якщо драйвер увімкнено автоматично, його використання не є обов’язковим. Комп’ютер автоматично визначає драйвер і режим також.

Далі, у циклі за допомогою процедури *circle(x, y, rdius)* виводиться 21 коло, центр яких співпадає та розташований у точці з координатами *(x,y)=(120,40)* та радіусом *rdius*, який змінюється від 25 до 125 пікселів.

Процедура *closegraph()* завершує роботу програми у графічному режимі та повертає пристрій до початкового стану.

Подібна послідовність дій повинна бути виконана у любій програмі, яка передбачає графічний вивід.

Для виводу інших графічних елементів використовують інші відповідні процедури. Для більш детального знайомства з можливостями графічного виводу рекомендується ознайомитися з прикладом, наведеним у [8] у якому створюється графічний редактор типу Paint у ОС Windows.

1.1.3. Робота з графікою у ОС Windows за допомогою бібліотеки MFC

У бібліотеці MFC (Microsoft Foundation Classes) існує об'єкт класу CDC (Context Display Class), який має нащадків у вигляді класів CClientDC, CMetaFileDC, CPaintDC, CWindowDC (рис. 1.4) які призначені для створення графічного контенту.

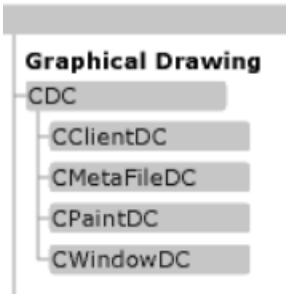


Рис. 1.4. Графічні компоненти MFC

Базовий клас CDC має функції (включаючи деякі віртуальні функції), які потрібні для малювання. За винятком класу CMetaFileDC, похідні класи відрізняються лише своїми конструкторами та деструкторами. Якщо ви (або фреймворк програми) створюєте об'єкт похідного класу контексту пристрою, ви можете передати вказівник CDC такій функції, як *OnDraw()*.

Для простого відображення похідними класами є *CClientDC* та *CWindowDC*. Для інших пристроїв, таких як принтери або буфери пам'яті, створюються об'єкти базового класу CDC.

Область вікна клієнта виключає границі, панель заголовку та панель меню. Якщо створюється об'єкт *CClientDC*, є контекст пристрою, який відображається лише в клієнтській області - ви не можете малювати поза ним. Точка (0, 0) зазвичай відноситься до верхнього лівого кута області клієнта.

Інший об'єкт MFC – *CView* відповідає дочірньому вікну, яке міститься всередині окремого вікна кадру, часто разом з панеллю інструментів, рядком стану та смугами прокрутки. У такому разі, клієнтська область виведення також не включає ці елементи. Якщо вікно містить закріплену панель інструментів у верхній частині, наприклад, (0, 0) відноситься до точки безпосередньо під лівим краєм панелі інструментів.

Якщо створюється об'єкт класу *CWindowDC*, точка (0, 0) знаходиться у верхньому лівому куті некліентської області вікна. За допомогою цього контексту пристрою цілого вікна можна малювати на межі вікна, в області заголовку тощо.

Для створення графічних зображень окрім того «на чому» можна це робити, потрібно мати відповідні інструменти – «чим» це робити. До таких інструментів належать об'єкти класів, представлених на рис. 1.5.

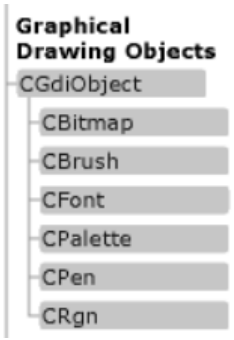


Рис. 1.5. Основні класи інструментів відображення

Призначення цих об'єктів наведено у табл.1.1.

Вивід графіки здійснюється за допомогою GDI-процедур. Загальна кількість GDI-процедур, які дозволяють створювати та маніпулювати графічним контентом – 195 [9].

Windows GDI пропонує апаратно-незалежний інтерфейс для кольорів. Програма надає "абсолютний" колірний код, і GDI відображає цей код у відповідний колір або поєднання кольорів на відеодисплеї вашого комп'ютера. У більшості програмісти для Windows намагаються оптимізувати кольоровий набір своїх програм для кількох поширених категорій відеокарт.

Взагалі, сьогодні існує велика кількість [10] різноманітних GUI-бібліотек, у тому числі й платформо незалежних, доступних для розробників не тільки мовою С, але й для інших мов.

З загальними принципами та особливостями використання бібліотеки MFC для роботи з графікою та не тільки з нею, можна більш детально ознайомитися по матеріалах [11].

Можливості та особливості нового інтерфейсу GDI+ представлені у [12] та “ Getting Started with Graphics Programming ” [13].

Таблиця 1.1

Призначення об'єктів GDI

Класс	Призначення
CBitmap	Растрове зображення - це масив бітів, в якому один або кілька бітів відповідають кожному пікселю відображення. Використовують растрові зображення для представлення існуючих зображень, або для створення пензлів.
CBrush	Пензель визначає растровий малюнок з пікселів, який використовується для заповнення площинних ділянок.
CFont	Шрифт - це повна колекція властивостей символів певного шрифту – вид, стиль, колір та таке інше. Зазвичай шрифти зберігаються на диску як ресурси, а деякі специфічні - для пристрою.
CPalette	Палітра - реалізує інтерфейс відображення кольорів, який дозволяє програмі використовувати переваги кольорів вихідного пристрою, не втручаючись в роботу інших програм.
CPen	Перо - це інструмент для малювання ліній та границь фігур. Описує колір і товщину пера, а також вид креслення - суцільні, пунктирні та інші види ліній.
CRgn	Регіон - це область, форма якої - багатокутник, еліпс або поєднання багатокутників і еліпсів. Використовують регіони для заповнення, відсікання та перевірки на натискання миші.

1.1.4. Графічні примітиви у мові Object Pascal

Особливістю графічних елементів є те, що вони, як правило, повинні бути спадкоємцями візуальних компонентів, тобто, компонентів, що мають інтерфейс для відтворення на екрані. Дана властивість визначається великою мірою тим, що

візуальні компоненти мають властивість Canvas, яка є спадкоємцем класу TCanvas, який безпосередньо бере участь у процесі візуалізації. Даним властивістю повинен володіти будь-який елемент, який підлягає візуалізації.

Об'єкти класу TCanvas. Назва об'єктів даного класу перекладається як «полотно». Структура основних властивостей і методів елементів даного класу представлена нижче:

– Pen - об'єкт «Перо» для малювання лінійних елементів і кордонів елементів;

– Brush - об'єкт «Кисть» для зафарбовування об'єктів;

– Font - властивість «Шрифт», що визначає характеристики для виведення текстової інформації;

– PenPos - властивість, що визначає поточне положення вказівника для рисування;

– Pixels - масив кольорових точок, що представляє зображення канви в пам'яті;

– ClipRect - прямокутна область, яка визначає межі поля виводу графічної інформації на канві;

– CopyMode - режим копіювання зображень графічних об'єктів на канву, що виконується в процедурах Draw і StretchDraw.

Деякі особливості малювання графічних об'єктів пов'язані з таким специфічним поняттям, як поточний стан пера. Будь-яка процедура виведення графічної інформації (малювання), призводить до зміни поточного стану пера. Так, наприклад, після виведення ламаної (процедура Polyline), поточний стан буде відповідати останній виведеній вершині. Якщо далі необхідно вивести лінію з точки, яка не співпадає з поточним становищем, то його потрібно змінити за допомогою процедури MoveTo, а потім вивести лінію за допомогою процедури LineTo.

Змінити колір окремої точки на канві можна скориставшись властивістю Pixels канви, що представляє з себе двовимірний масив елементів типу TColor. Перейти до опису кольору з трибайтового подання базовими кольорами в формат TColor можна побудувавши конструкцію типу $00BBGGRR, де b, g і r - відповідно синя, зелена і червона складові. наприклад:

ImageObject.Canvas.Pixels[125,37]:=$0023FFCD;

Оскільки властивість Pixels є «двонаправленим», то скориставшись ним, можна прочитати поточне значення кольору в заданій точці, наприклад:

Var CurrColor:TColor;
CurrColor:= ImageObject.Canvas. Pixels[x,y];

У наведених прикладах об'єкт ImageObject є візуальний компонент, що володіє властивістю Canvas.

Процедури Draw, StretchDraw використовуються для малювання графічних об'єктів на канву. При цьому, друга процедура «натягує» вихідне зображення на заданий прямокутник, автоматично визначаючи коефіцієнти масштабування.

Процедура TextOut виводить заданий текст у вказану позицію, використовуючи для цього характеристики шрифту, встановлені у властивості Font канви.

ImageObject.Canvas.Pixels[125,37]:=$0023FFCD;

Оскільки властивість Pixels є «дво-направленою», то скориставшись ним, можна прочитати поточне значення кольору в заданій точці, наприклад:

Var CurrColor:TColor;
CurrColor:= ImageObject.Canvas. Pixels[x,y];

У наведених прикладах об'єкт ImageObject є візуальний компонент, що володіє властивістю Canvas.

Процедури Draw, StretchDraw використовуються для малювання графічних об'єктів на канву. При цьому, друга процедура «натягує» вихідне зображення на заданий

прямокутник, автоматично визначаючи коефіцієнти масштабування.

Процедура TextOut виводить заданий текст у вказану позицію, використовуючи для цього характеристики шрифту, встановлені у властивості Font канви.

Об'єкти класу TBrush представляють собою набір властивостей для визначення зафарбовування майданних об'єктів (зафарбовані еліпс, прямокутник, багатокутник), точніше як буде виглядати внутрішня область даних об'єктів. Структура налаштувань ілюструється рис. 1.6.

Рис. 1.6. Структура властивостей інструменту "Пензель"

Для визначення властивостей кисті може бути використаний бітовий образ (Bitmap) або пара властивостей стиль і колір заливки. Якщо перед виведенням площинного об'єкту або виконання операції Fill визначається колір (Color) і стиль (Style), то призначене властивість Bitmap перестає діяти. Призначення властивості Bitmap скасовує поточні установки пари Color-Style. Властивість стиль визначає як буде зарисована внутрішня область відповідно до стандартними значеннями, наприклад, bsSolid визначає «суцільну» заливку, bsVertical - замальовку вертикальними лініями і т.д.

Об'єкти класу TPen відповідають за те, як на канві будуть відображені лінійні об'єкти і кордони площинних об'єктів. Структура налаштувань властивостей об'єктів даного класу представлена на рис. 1.7.

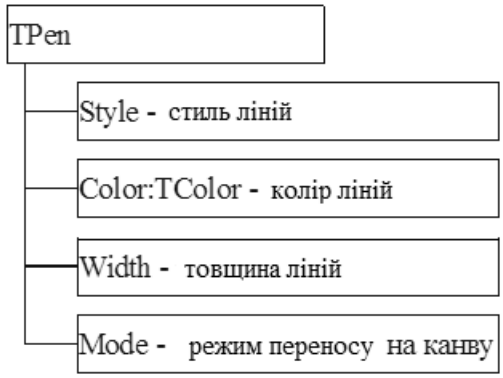


Рис. 1.7. Структура налаштувань інструменту "Перо"

У даній структурі властивість Style відповідає за тип виведеної на екран лінії:

- psSoliod - суцільна лінія;
- psDash - пунктирна;
- psDot - точками і т.д.

Властивість Mode визначає як при малюванні лінії її колір буде взаємодіяти з кольором канви, що дозволяє домагатися різних ефектів.

Об'єкти класу TImage. Об'єкти даного класу є інтегрованими контейнерами для роботи з графічними образами, дозволяючи формувати, зберігати і читати зображення. Дані об'єкти є візуальними компонентами. Для їх використання в інтерфейсі, на панелі інструментів необхідно вибрати закладку "Additional" і вибрати елемент Image. Після цього, клацнувши на елементі-контейнері в вікні дизайнера форми, здійснюється його додавання в проект (рис. 1.8).

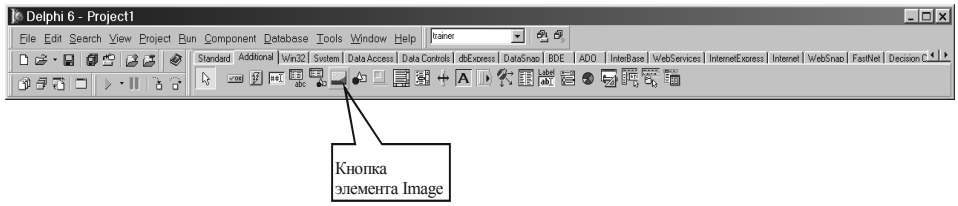


Рис. 1.8. Вид панелі інструментів

Після цього, можна реалізовувати модулі, що виконують операції з канвою даного об'єкта.

1.1.5. Графічні примітиви у мові Java

У мові Java, як і у мові C, для відтворення графічного контенту потрібні елементи з «*графічним контекстом*» [14]. У Java рисування виконується за допомогою класу *java.awt.Graphics*, який управляє графічним контекстом та забезпечує набір незалежних від пристрою методів відтворення текстів, малюнків та зображень на екрані на різних платформах.

Java.awt.Graphics - це абстрактний клас, оскільки фактичний акт малювання залежить від системи та пристрою. Кожна операційна платформа забезпечить підклас *Graphics* для виконання власного відтворення відповідно до своєї платформи, але згідно специфікації, визначеній у *Graphics*.

Структура цього класу представлена на рис. 1.9.

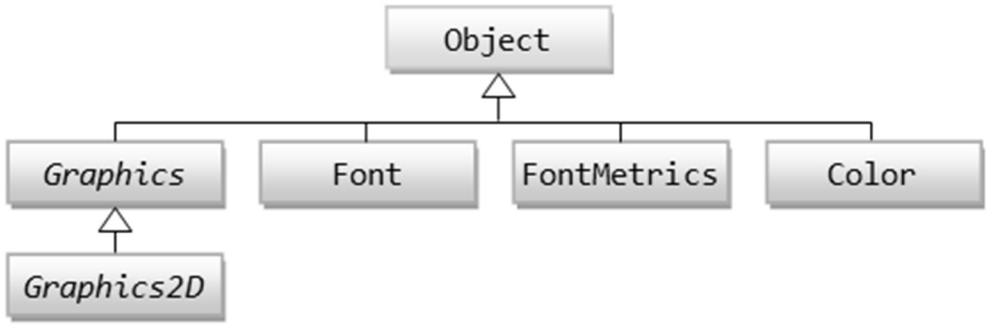


Рис. 1.9. Структура абстрактного класу *Java.awt.Graphics*

Клас *Graphics* забезпечує методи візуалізації трьох типів графічних об'єктів:

- Текст: за допомогою методу *drawString()*. Стандартна функція System.out.println () здійснює вивід у системну консоль, а не на графічному екрані.
- Векторно-графічні примітиви та фігури: за допомогою методів *drawXxx()* та *fillXxx()*, де Xxx може бути Line, Rect, Oval, Arc, PolyLine, RoundRect або 3DRect.

– Растрові зображення: за допомогою методу *drawImage()*.

При відтворенні примітивів потрібно задавати набори координат, які відповідно до фігури представлені на рис. 1.10.

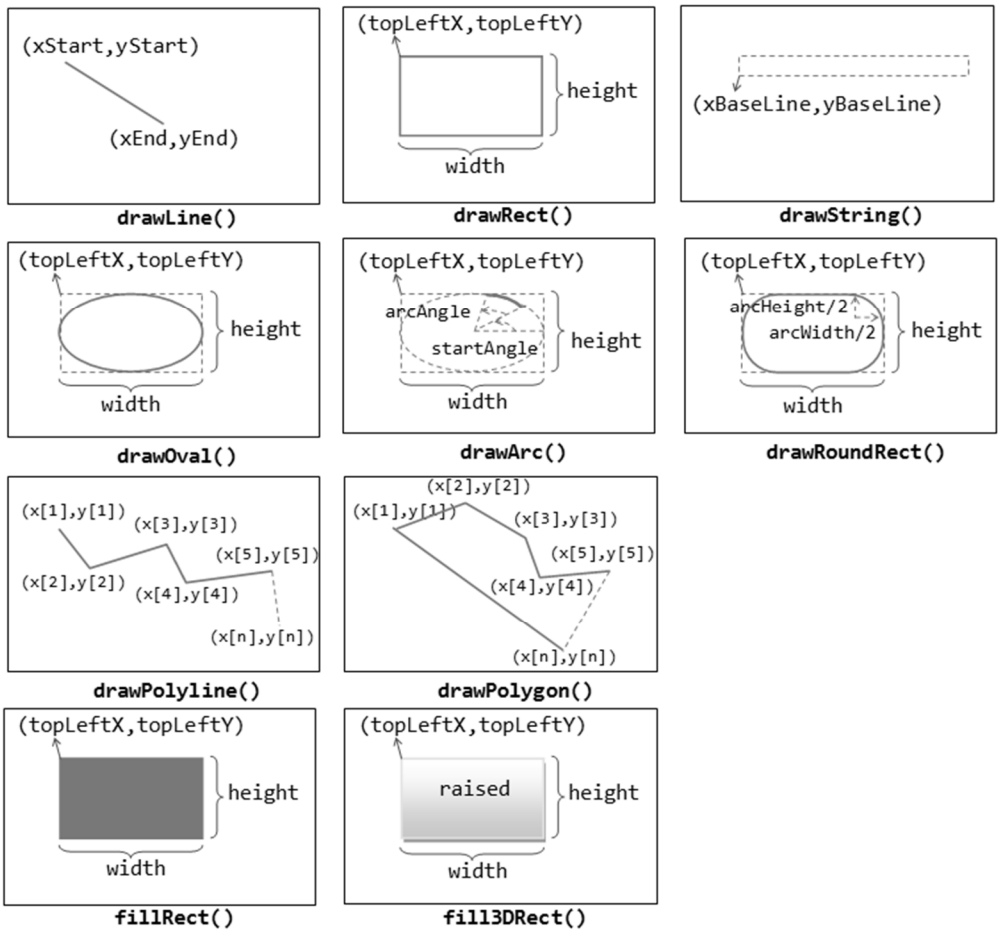


Рис. 1.10. Стандартні примітиви у класі *Graphics*

Графічний контекст підтримує такі властивості (або атрибути), як поточний колір, поточний шрифт для виводу тексту та поточна прямокутна область малювання (так звана clip, аналог Viewport у Ci). Можна використовувати методи *getColor(), setColor(), getFont(), setFont(), getClipBounds(), setClip()*, щоб отримати або встановити колір, шрифт та область виводу. Будь-яке зображення поза областю виводу ігнорується.

У підсистемі вікон Java (як і у більшості 2D-графічних систем) початок системи координат (0,0) знаходиться у верхньому лівому куті. Кожний компонент (графічний контейнер) має власну систему координат, яка варіюється від (0,0) до (*width*-1, *height*-1).

Можна використовувати методи *getWidth()* та *getHeight()*, щоб отримати ширину та висоту компонента. Використання методів *getX()* або *getY()*, дозволяє визначити положення верхнього лівого кута *(x, y)* щодо його батьківського елемента (у системі координат батьківського елемента).

При використанні AWT власний вивід виконується розширенням *java.awt.Canvas* та реалізацією власного методу *paint(Graphics g)* у додатку класу *java.awt.Frame*. Так само, можна явно викликати метод *repaint()* для оновлення контенту.

Клас *java.awt.Color* забезпечує 13 стандартних кольорів як іменовані константи. Це: *Color.RED, GREEN, BLUE, MAGENTA, CYAN, YELLOW, BLACK, WHITE, GRAY, DARK_GRAY, LIGHT_GRAY, ORANGE, ma PINK*. У JDK 1.1 ці імена констант мають нижній регістр, наприклад, *red*. Це порушує конвенцію іменування Java для констант. У JDK 1.2 додаються імена з великими літерами, але імена нижнього регістру не видаляються для зворотної сумісності.

Можна використовувати, також, значення *RGB* або значення *RGBA* (*A* для альфа – прозорість) для побудови власного кольору за допомогою конструкторів:

Color(int r, int g, int b);
Color(float r, float g, float b);
Color(int r, int g, int b, int alpha);
Color(float r, float g, float b, float alpha);

Значення альфа від 0 до 255 (від 0,0f до 1,0f) визначаються як: 0 (0,0f) для абсолютно прозорого, 255 (або 1,0f) для абсолютно непрозорого. Альфа за замовчуванням приймає значення – 255 (або 1.0f).

Окрім цих класів та згаданих методів є безліч інших, які детально описані у «The Java™ Tutorials. Trail: 2D Graphics» [15].

Слід зазначити, що клас *Java.awt.Graphics* реалізує графічні функції на початковому рівні. Для спрощення створення графічного інтерфейсу та реалізації додаткових можливостей є інші бібліотеки, наприклад Swing [16].

1.1.6. Завдання для самостійного виконання

Основне завдання полягає у наступному: реалізувати інтерфейс прикладної програми, що передбачає функції редагування, збереження і завантаження зображення в/із файлу, вибору кольору лінії і фону, вибору шрифту для виведення тексту, зміни режимів малювання (точка, лінія, еліпс, коло, прямокутник, багатокутник, ламана, текст). Для цього потрібно:

1) обрати мову для створення додатку та встановити відповідні компілятор та бібліотеки;
2) завантажити та встановити середовище для редагування та налагоджування програмного коду;
3) розробити структури даних для збереження опису моделі відтворюваної сцени;
4) розробити алгоритми функціонування додатку;
5) розробити та налагодити код програми додатку;
6) оформити звіт, представити програму викладачеві і пояснити отримані результати;
7) відповісти на контрольні запитання.

1.1.7. Питання до самоперевірки

1) Вкажіть набір графічних примітивів, які використовуються в мові програмування Object Pascal.
2) Чи можна виводити графічну інформацію на будь-який візуальний компонент?
3) Вкажіть, як здійснюється адресація точок у віконному інтерфейсі.

4) Запропонуйте способи отримання поточних координат покажчика миші.
5) Скільки байт відводиться для завдання кольору і правила формування колірної константи?
6) Вкажіть, яким чином в програмі реалізується зміна режиму малювання.
7) Вкажіть, як змінюється поточне положення кисті при виведенні графічних примітивів.

Тема 1.2. Алгоритми зафарбовування

Мета: сформувати уявлення про структуру обчислювальних процесів при виконанні задач фотореалістичного фарбування поверхонь об'єктів які підлягають візуалізації та навчитися визначати потрібні ресурси.

Після того, як відбулися зміни у моделі, яка є базою для побудови зображення, постає задача її растрового розгортання – отримання цифрового еквіваленту зображення у пам'яті. У цьому процесі можна виділити кілька основних етапів, або дій. Для кожного не елементарного об'єкту повинно бути візуалізовано його границі та виконано зафарбовування внутрішньої частини між визначеними границями. У основі процесу зафарбовування лежать закони фізики та оптики, що пояснюють, також, загальний механізм сприйняття об'єктів зовнішнього світу через зорові сенсори.

Світловий потік, що потрапляє на поверхню об'єкта може поглинатися, відображатися або бути пропущеним. Відомо, що об'єкт може бути побачений тільки в тому випадку, якщо він відображає або пропускає світлову енергію; якщо він цілком її поглинає, то він невидимий і називається абсолютно чорним тілом.

Кількість поглиненої, відбитої або пропущеної енергії залежить від довжини хвилі світла. Якщо об'єкт освітлюється світлом з білим кольором і рівномірно поглинає всі кольори, то в залежності від ступеня поглинання об'єкт може виглядати

білим, сірим або чорним. Якщо поглинаються тільки певні довжини хвиль, об'єкт буде виглядати кольоровим – колір об'єкта визначається поглиненими довжинами хвиль.

Властивості відбитого світла залежать від будови, напряму та форми джерела світла, від орієнтації і властивостей поверхні. Відбите від об'єкта світло може бути дифузним або дзеркальним. Дифузійне відбиття світла відбувається, коли світло ніби проникає під поверхню об'єкта, поглинається і знову випускається об'єктом. При цьому, розташування спостерігача не має суттєвого значення, оскільки дифузно відбите світло розсіюється рівномірно в усіх напрямках усією поверхнею цього об’єкту. Дзеркальне відображення походить від зовнішньої поверхні об'єкта.

1.2.1. Моделі освітлення у комп’ютерній графіці

У комп'ютерній графіці рівняння, яке використовується для розрахунку освітленості об'єкта, часто називається функцією зафарбовування і застосовується для розрахунку інтенсивності пікселів зображення на основі трьох компонент – розсіяного світла (ambient lighting), дифузного (diffuse lighting) та дзеркально відбитого (specular lighting):

$$I = I_a k_a + [I_\ell (k_d \cos\theta + k_S \cos^n \alpha)] / (d + k), \qquad (1.1)$$

де I_a - інтенсивність розсіяного світла;

k_a - коефіцієнт дифузного відбиття розсіяного світла;

k_s - коефіцієнт дзеркального відображення світла;

n - ступінь апроксимації просторового розподілу дзеркально відбитого світла;

d - відстань від центру проекції до об'єкта;

k - константа;

θ - кут падіння променю від джерела освітлення;

α - кут між відбитим променем і вектором до спостерігача.

Це рівняння відповідає геометричній моделі, представленій на рис. 1.11.

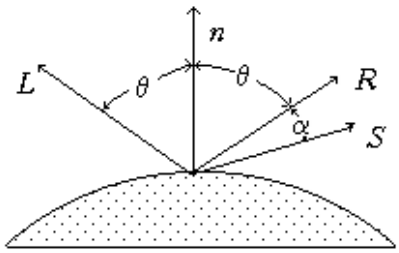


Рис. 1.11. Геометрична модель освітлення

Для паралельних проекцій величина d визначається положенням об'єкта, найближчого до точки спостереження. Це означає, що найближчий об'єкт освітлюється з повною інтенсивністю джерела, а більш далекі - зі зменшеною. Для кольорових поверхонь модель освітлення застосовується до кожного з трьох основних кольорів.

За наявності кількох джерел освітлення, їх ефекти підсумовуються. У цьому випадку модель освітлення визначається як:

$$I = I_a k_a + \sum_{j=1}^{m} (I_{\ell j} \cos\theta_j + k_S \cos\alpha_j^n) / (d_j + k) \qquad (1.2)$$

Існує, так само, кілька методів визначення зафарбовування для точок поверхні об'єкта в залежності від того, на скільки реалістичним має виглядати зображення. Найбільш простим методом зафарбовування є монотонне зафарбування.

Однак, відповідно до алгоритмів зафарбовування, можна виділити кілька основних завдань, що підлягають вирішенню - це визначення нормалі до поверхні, знаходження кута θ (косинуса кута θ), знаходження кута α (косинуса кута α).

1.2.2. Визначення нормалі до поверхні

При виконанні роботи можуть бути використані такі припущення - вісь z перпендикулярна площині екрану, тип проекції - паралельна, тобто, спостерігач знаходиться на осі z в нескінченності, джерело освітлення знаходиться на осі z.

Відповідно до цих припущеннями, одиничні вектори до джерела освітлення і спостерігачеві рівні $\overline{(0,0,1)}$.

Значення вектора нормалі до поверхні може бути знайдено декількома способами, зокрема для граней, що лежать в математичній площині $ax + by + cz + d = 0$ (плоских граней), воно дорівнює:

$$\vec{n} = a\vec{i} + b\vec{j} + c\vec{k} \tag{1.3}$$

де *i, j, k* - одиничні вектори осей x, y, z відповідно.

Значення коефіцієнтів a, b, c можуть бути знайдені за співвідношенням:

$$\begin{aligned} a &= y_1(z_2 - z_3) + y_2(z_3 - z_1) + y_3(z_1 - z_2) \\ b &= x_1(z_2 - z_3) + x_2(z_3 - z_1) + x_3(z_1 - z_2) \\ c &= x_1(y_2 - y_3) + x_2(y_3 - y_1) + x_3(y_1 - y_2) \\ d &= x_1(y_2 z_3 - y_3 z_2) + x_2(y_3 z_1 - y_1 z_3) + x_3(y_1 z_2 - y_2 z_1) \end{aligned} \tag{1.4}$$

Після визначення вектору нормалі необхідно знайти абсолютну величину нормалі:

$$\left|\overline{n}\right| = \left|a^2 + b^2 + c^2\right|^{1/2}, \tag{1.5}$$

і визначити одиничну нормаль як:

$$\overline{n_1} = \frac{\overline{n}}{\left|\overline{n}\right|}. \tag{1.6}$$

1.2.3. Визначення косинуса кута між векторами

Значення cos (v1, v2), де v1 і v2 два довільні вектори, можна обчислити як:

$$\cos(v_1, v_2) = \frac{v_{1x}\,v_{2x} + v_{1y}\,v_{2y} + v_{1z}\,v_{2z}}{\sqrt{v_{1x}^2 + v_{1y}^2 + v_{1z}^2}\,\sqrt{v_{2x}^2 + v_{2y}^2 + v_{2z}^2}} \quad (1.7)$$

1.2.4. Особливості визначення косинуса кута між векторами відбитого променю і вектором до спостерігача

Для моделі освітлення надзвичайно важливо правильно задавати напрямок векторів відображення.

Згідно із законом відображення вектор падаючого світла, нормаль до поверхні і вектор відображення лежать в одній площині, причому на цій площині кут падіння дорівнює куту відбиття.

Фонг вивів просте рішення для випадку, коли світло падає уздовж осі Z:

$$\hat{R}_z \cos 2\theta = 2\cos^2\theta - 1 = 2\hat{n}_z^2 - 1 \quad (1.8)$$

$$\hat{R}_y = 2\hat{n}_z\hat{n}_y \quad (1.9)$$

$$\hat{R}_x = 2\hat{n}_z\hat{n}_x \quad (1.10)$$

де $\hat{R}_x, \hat{R}_y, \hat{R}_z$ - складові одиничних векторів відбиття;

$\hat{n}_x, \hat{n}_y, \hat{n}_z$ - складові одиничних векторів нормалі.

За наявності кількох джерел освітлення, застосування даних виразів неможливо. У цьому випадку може бути використано умову, що одинична нормаль, одиничний вектор падіння і одиничний вектор відбиття лежать в одній площині, що записується за допомогою їх векторних добутків. Рівність кутів

падіння і відбиття виражається через скалярні добутки цих векторів.

1.2.5. Монотонне фарбування – Flat Shading

При монотонному фарбуванні обчислюється один рівень інтенсивності, який використовується для фарбування усього багатокутника (рис. 1.12).

Рис. 1.12. Приклад застосування монотонного фарбування

При цьому робляться такі припущення:

- джерело освітлення розташований в нескінченності, тому добуток L•N постійний по всій площі полігональну;
- спостерігач знаходиться в нескінченності, тому добуток N•S постійний по всій площі полігональну;
- багатокутник є реальною поверхнею яка моделюється, а не апроксимацію криволінійної поверхні.

Якщо будь-яка з перших двох припущень виявляється неприйнятним, доцільно скористатися усередненими значеннями L і S, обчисленими в центрі багатокутника.

Третє припущення як правило не виконується, але надає на отримуєте-моє зображення істотно більший вплив ніж два

перших. Вплив полягає в тому, що кожна з видимих полігональних граней апроксимованої поверхні можна дуже відрізнити від інших, оскільки інтенсивність кожної з цих граней відрізняється від інтенсивності сусідніх.

Відмінність в фарбуванні сусідніх граней помітна внаслідок ефекту смуг Маха, відкритого Махом в 1865 році і докладно описаного в літературі. Цей ефект є однією з причин занадто різкого перепаду інтенсивності на всіх граничних ребрах, на яких виникає порушення безперервності зміни самої величини інтенсивності або її похідної. На рис. 1.13 показані дійсна та удавана зміни інтенсивності вздовж поверхні, викликане літеральним гальмуванням рецепторів ока.

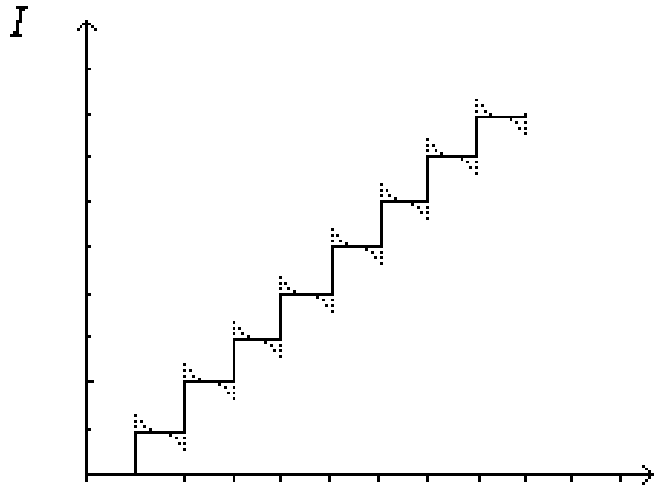


Рис. 1.13. Принцип виникнення смуг Маха

1.2.6. Фарбування за методом Гуро – Gouraud Shading

Процес фарбування по методу Гуро [17] здійснюється в чотири етапи. На першому етапі обчислюються нормалі до поверхні, на другому визначаються нормалі в вершинах шляхом усереднення нормалей за всіма полігональним гранях, яким належить вершина, на третьому етапі обчислюються значення інтенсивності в вершинах. На четвертому етапі кожен багатокутник фарбується за методом білінійної інтерполяції.

Фарбування кожного пікселя здійснюється шляхом попіксельного сканування зображення скануючимм рядком і обчислення інтенсивності зафарбовування кожного пікселя.

Розглянемо інтерполяцію на прикладі (рис. 1.14). Значення інтенсивності в точці P визначається лінійною інтерполяцією інтенсивності в точках D і E. Інтенсивність в точці D визначається лінійною інтерполяцією інтенсивності в точках A і B.

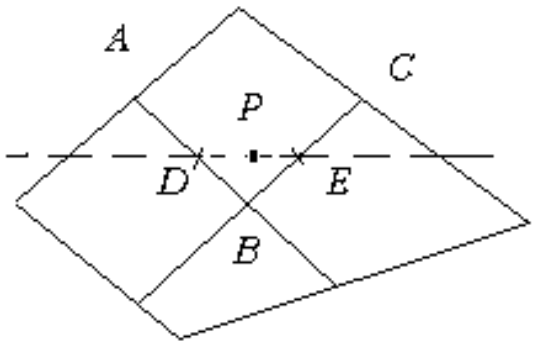


Рис. 1.14. Принцип білінійної інтерполяції

Обчислення виконують за наступними співвідношеннями:

$$I_D = U I_A + (1 - U) I_B \qquad 0 < U < 1 \quad (1.11)$$

де $U = BD/AB$.

$$I_E = W I_B + (1 - W) I_C, \; 0<w<1 \qquad (1.12)$$

де $w = EC/BC$.

$$I_P = m I_D + (1 - m) I_E, \; 0<m<1, \qquad (1.13)$$

де m = EP/DE.

Значення інтенсивності вздовж скануючої строки можна обчислювати інкрементно. Для двох пікселів P1 і P2 на скануючому рядку виконуються співвідношення:

$$I_{P1} = m_1 I_D + (1 - m_1) I_E , I_{P2} = m_2 I_D + (1 - m_2) I_E \quad (1.14)$$

Звідкіля отримуємо:

$$I_{P2} = I_{P1} + (I_D - I_E)(m_2 - m_1) = I_{P1} + \Delta I \Delta m \quad (1.15)$$

При однакових рівних похідних, фарбування за методом Гуро дає більш якісні зображення (рис. 1.15 [18])

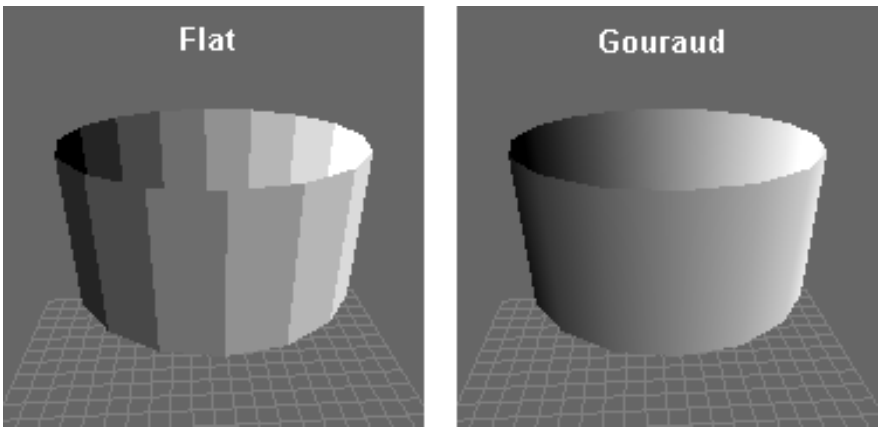


Рис. 1.15. Зафарбовування за методом Фонга – Phong Shading

Цей метод дозволяє отримати реалістичне освітлення у сценах які підлягають візуалізації. Реалістичність досягається за рахунок обчислення освітлення для кожної точки замість множини багатокутників. Кожен піксель отримує свій власний колір на основі моделі освітлення, спрямованій на цей піксель. Цей метод вимагає більш інтенсивних обчислень, ніж метод Гуро.

Якщо при фарбуванні Гуро уздовж рядку інтерполюють значення інтенсивності, то при фарбуванні Фонга інтерполюють вектор нормалі.

Потім він використовується в моделі освітлення для обчислення інтенсивності пікселя. При цьому, досягають найкращу локальну апроксимацію кривизни поверхні і більш

реалістичне зображення. Зокрема, правдоподібніше виглядають дзеркальні відблиски.

Отже, при фарбуванні Фонга:

- апроксимують нормалі в вершинах багатокутників;
- білінійною інтерполяцією апроксимують нормалі в кожному пікселі;
- за моделлю освітлення обчислюється інтенсивність в кожному пікселі.

Метод Фонга усуває більшість недоліків методу Гуро (рис. 1.16), він теж ґрунтується на лінійній інтерполяції. Тому, в місцях розриву першої похідної інтенсивності помітний ефект смуг Маха, хоча і не такий сильний як при фарбуванні Гуро.

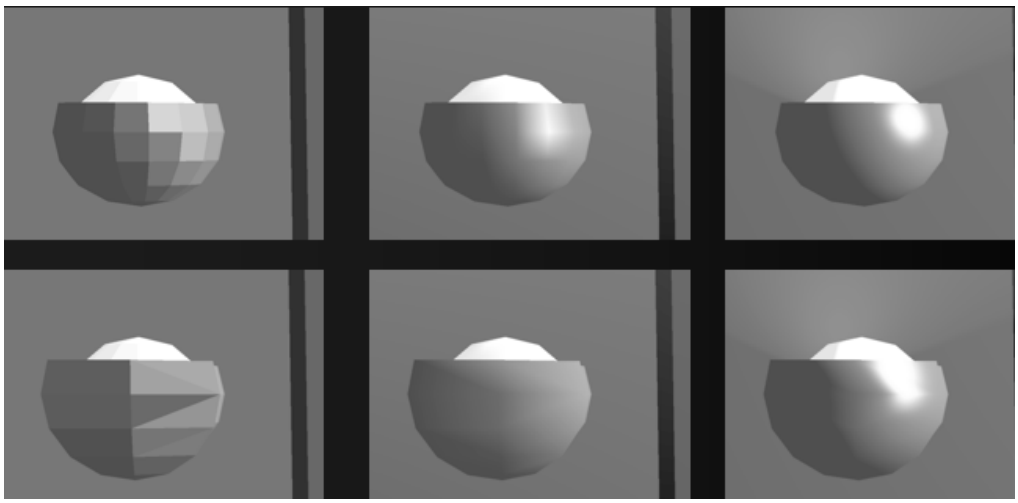


Рис. 1.16. Порівняння методів фарбування: монотонне, Гуро та Фонга

1.2.7. Індивідуальні завдання до виконання

Згідно до варіанту, наведеному у табл. 1.2, виконати розрахунки освітлення та фарбування для трикутника, координати якого задано. Фарбування виконати за методами монотонного фарбування та вказаного у таблиці. Порівняти отримані результати та пояснити їх.

Таблиця 1.2

Варіанти вихідних даних

Варіант	**Координати вершин об’єкта**	**Метод фарбування**
0	V1(-2,1,-2), V2(-1,1,-2), V3(-1,0.5,-1)	Гуро
1	V1(-1,1,-2), V2(-1,0.5,-1),V3(-2,0,0)	Фонга
2	V1(-2,1,-2), V2(0,1,-2), V3(-1,0.5,-1)	Гуро
3	V1(0,1,-2), V2(-1,0.5,-1), V3(-2, 0.5, -1)	Фонга
4	V1(-2, 0.5, -1), V2(0, 0.5, -1), V3(0, 0, 0)	Гуро
5	V1(0, 0.5, -1), V2(0, 0, 0), V3(-2, 0, 0)	Фонга
6	V1(0.5, 1.5, -1), V2(1.5, 1.5, -0.5), V3(1.5, 1.5, 0)	Гуро
7	V1(1.5, 1.5, -0.5), V2(1.5, 1.5, 0), V3(1, 1.5, 0.5)	Фонга
8	V1(0, 0, -2), V2(2, 0, -1), V3(2, 0, 0)	Гуро
9	V1(0.5, 1.5, -1), V2(1.5, 1.5, -0.5), V3(2, 0, -1)	Фонга

1.2.8. Порядок виконання роботи

1) Запустити комп'ютер і завантажити ОС Windows.
2) Завантажити середовище програмування.
3) Розробити програмне забезпечення, яке здійснює візуалізацію заданого об'єкта з розрахунком моделі освітлення і передбачає можливість зміни яскравості джерел точкового і розсіяного освітлення, зміна положення точкового джерела освітлення або самого об'єкта.
4) Відкомпілювати програму.
5) Зафіксувати результати роботи для кількох варіантів освітлення та розташування джерела освітлення.
6) Оформити звіт, представити програму викладачеві і пояснити отримані результати.

1.2.9. Контрольні запитання

1. Вкажіть, якими способами можна визначити значення нормалі до поверхні.
2. Вкажіть, в чому полягає особливість методу Ньюела для визначення нормалі до поверхні.
3. У чому полягає відмінність між нормаллю до поверхні і одиничною нормаллю до поверхні, навіщо потрібно їх знаходити в алгоритмах КГ.
4. Вкажіть на основні відмінності в моделях зафарбовування.
5. В чому полягає суть методу білінійної інтерполяції для розрахунку яскравості.
6. У чому полягає суть методу білінійної інтерполяції для розрахунку вектору нормалі до точки поверхні.

Як впливає ступінь апроксимації розподілу дзеркально відбитого світла на представлення зображення в різних моделях фарбовування.

Тема 1.3. Апаратні засоби у процесі візуалізації

Мета: сформувати розуміння зв'язку логічних процесорів та алгоритмів візуалізації з апаратними компонентами, які супроводжують процес візуалізації графічної інформації у комп'ютерних системах, їх структуру та вимоги обчислювальних та часових характеристик.

1.3.1. Архітектури апаратних засобів

У п.1.1 було описано загальну структуру процесорів перетворення в графічній системі і склад процесорів перетворення зображень. Частина логічних процесорів перетворення реалізується апаратно, інша частина емулюється будь-яким фізичним пристроєм. Природньо, що програмна емуляція – процес більш повільний, ніж апаратна реалізація. З іншої сторони, апаратна реалізація вимагає більш високої

вартості. Два цих фактори в значній мірі і визначають архітектуру побудови графічних пристроїв.

У зв'язку зі значним прогресом в технології мікроелектронних засобів розвиток архітектури графічних засобів, як і, в загальному, архітектури обчислювальних засобів характеризується все більш повним використанням апаратних способів реалізації логічного конвеєра обробки зображень.

Зробимо оцінку необхідного швидкодії конвеєра. Будь-які геометричні перетворення можуть бути виражені результуючої матрицею (детально розглядається у 2 розділі):

$$M=\begin{vmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix} \qquad (1.16)$$

Якщо прийняти, що X, Y, Z – координати точки в світовому просторі, а X1, Y1, Z1 – екранні координати, то використовуючи матрицю перетворення, можна записати:

$$\begin{aligned} X1 &= X \cdot r_{11} + Y \cdot r_{21} + Z \cdot r_{31} + T_X \\ Y1 &= X \cdot r_{12} + Y \cdot r_{22} + Z \cdot r_{32} + T_y \\ Z1 &= X \cdot r_{13} + Y \cdot r_{23} + Z \cdot r_{33} + T_z \end{aligned} \qquad (1.17)$$

Кожне таке перетворення вимагає виконання 18 операцій з плаваючою комою для кожної тривимірної вершини або 12 операцій для двомірної вершини. Для прикладу розглянемо наступну ситуацію.

Прийнявши, що типова сцена містить 5000 багатокутників і багато багатокутників мають ряд спільних вершин. Визначимо середню кількість вершин для багатокутника як 3, а загальну кількість вершин для сцени – 15000.

Найбільш важкий режим перетворення – динамічна сцена. Для даного випадку це буде означати, що 15000 вершин повинні бути відпрацьовані відповідно до 30 раз в секунду.

При цьому для перетворення всієї сцени потрібно виконати 8.1 млн операцій з плаваючою комою в секунду (МФлоп). Якщо додати до цього реалістичне зафарбовування навіть з використанням найбільш простого алгоритму зафарбовування (наприклад, Гуро – п.1.2.6) ця цифра зросте до величини ~ 15 МФлоп. Очевидно, що багато стандартних мікропроцесорів не володіють продуктивністю, необхідною для вирішення такого завдання.

У своєму розвитку графічні пристрої пройшли великий шлях – від найпростішого пристрою виведення, повністю працює під управлінням ЕОМ – до автономної графічної робочої станції.

Високі темпи зростання СОЗ (Системи Обробки Зображень) і КГ привели до появи великої кількості надвеликих інтегральних схем, орієнтованих на машинну графіку і обробку зображень, а також до великої кількості стандартних програмних пакетів.

Стандарти типу GKS, GKS -3D, Core, PHIGS, CGI, CGM прийняті і впроваджені в багатьох країнах світу. Розроблено операційну систему Unix, яка широко використовується в більшості зарубіжних графічних робочих станціях.

Якщо в період свого становлення графічні засоби відображення інформації були найчастіше тільки допоміжним інструментом користувача, то сьогодні – це, в більшості випадків – центр, навколо якого групуються засоби обробки, реєстрації, введення і т.п.

Структура дисплейних терміналів безперервно ускладнюється. Від дисплейного процесора, що реалізує найпростіші функції управління променем електронно-променевої трубки (ЕПТ) по командам верхньої ЕОМ, до дисплейної багатофункціональної системи з декількома спеціалізованими процесорами – ось шлях, пройдений відео терміналами за два десятиліття. Кажучи про графічні засоби не можна не відзначити таке явище, як графічні робочі станції (WORKSTATION). Термін "робоча станція" зазвичай використовують майже для будь-яких систем, які об'єднують

кілька рівнів обчислювальних ресурсів, дисплейний термінал, клавіатуру, ОЗП, пристрої введення-виведення.

Робоча станція (РС) має як характеристики міні ЕОМ, так і ПЕОМ. Типова РС включає блок центрального процесора (ЦП), графічний дисплей з середньою або високою роздільною здатністю, пристрої графічного введення (ПГВ), пристрої реєстрації або виведення графічних даних, один або кілька пристроїв зовнішньої пам'яті, інтерфейси зв'язку, інтерфейси для включення в мережу. Робочі станції можуть, також, підтримувати режим роботи з багатьма користувачами.

За роздільною здатністю екранів відео дисплейні системи можуть бути розділені на наступні категорії:

– дисплейні системи з низькою роздільною здатністю – до 512x512 пікселів – основне застосування направлене на вбудовані системи;

– дисплейні системи з середньою роздільною здатністю – від 512x512 крапок до 1024x768, призначені для мобільних термінальних систем;

– дисплейні системи з високою роздільною здатністю – від 1024x800 крапок до 1280x1024 пікселів – для реалізації графічних систем взаємодії з користувачем у обмеженому колі задач;

– дисплейні системи з надвисокою роздільною здатністю – понад 1280x1024 пікселів.

Застосування такого параметричного ряду відео дисплейної системи цілком виправдано як з чисто технічних, так і економічних міркувань.

У світовій практиці прийнято ділити відео термінали і робочі станції на наступні типи по застосуванню:

- CAD/CAM/CAE (computer aided design/computer aided manufacturing/computer aided engineering) – використовуються для автоматизованого проектування, автоматизованого виготовлення або автоматизації інженерних робіт.
- Ділові та адміністративні РС (business or administrative workstation).

– Промислові системи управління і диспетчерські функції.
– Обробка зображень.

Відрізняючись функціями, складом устаткування і конструктивним виконанням, програмним забезпеченням, ці пристрої, проте, мають багато спільного. Розглянемо структуру відео терміналів і робочих станцій різних рівнів.

Умовно розділимо термінали по реалізованим функціям на прості, середньої складності і великої складності. Для стислості надалі будемо використовувати термін "термінал типу А", "термінал типу В", "термінал типу С" відповідно. У табл. 1.3 ці функції обробки розділені між терміналами та ЕОМ.

Блок-схеми терміналів наведені на рис. 1.17 Розглянемо особливості роботи відео терміналів.

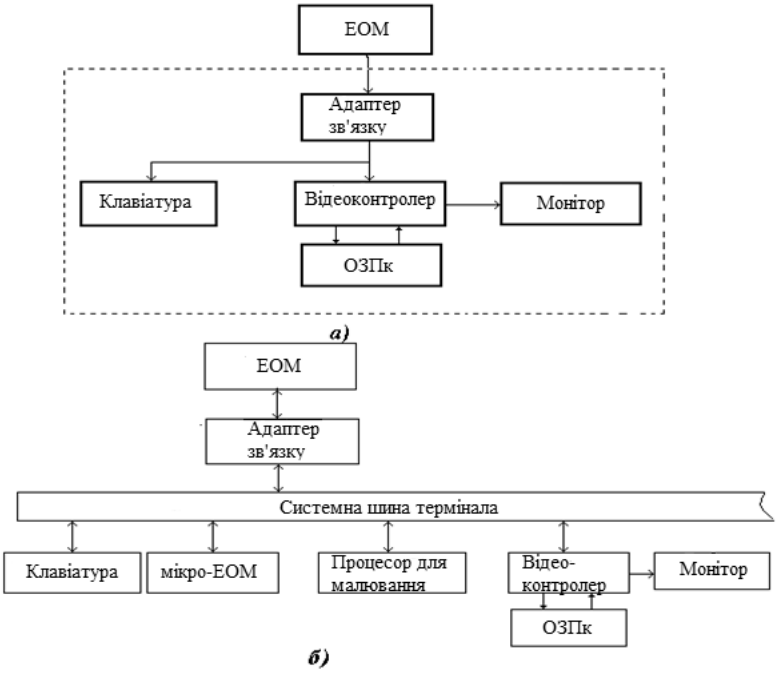


Рис. 1.17. Архітектури графічних систем типу А та В

Таблиця 1.3

Співвідношення розподілу функцій графічної системи

Реалізована функція	Розподіл функцій					
	A	ВМ	B	ВМ	C	ВМ
Моделювання та перетворення	-	+	-	+	-	+
Відсікання у вікні	-	+	-	+	+	-
Перетворення ОМК ПВнк	-	+	-	+	+	-
Геометричні перетворення в НК	-	+	-	+	+	-
Перетворення ПВнк ПВфк	-	+	-	+	+	-
Растрова розгортка примітивів	-	+	+	-	+	-
Візуалізація	+	-	+	-	+	-

Термінал "А" – найбільш простий вид терміналів. По суті тут на термінал покладено завдання запису інформації в кадрове ЗУ, зчитування інформації з ОЗП і формування вхідного сигналу на відео монітор. Всі інші операції, включаючи розрахунок адрес кадрового ЗУ при розгортці векторів, здійснює ЕОМ.

Такі рішення сьогодні використовують для реалізації вбудованих та мобільних систем, наприклад, на основі TFT ЖК-терміналів фірми NEXTION (рис. 1.18) [19].Розміри екрану становлять від 2.4" до 10.1" по діагоналі. Взаємодія з комп'ютерною системою здійснюється по послідовному інтерфейсу у форматі команда-відповідь.

Рис. 1.18. HMI-панель (термінал) Nextion

У концепції фірми-розробника термінал реалізує Human Machine Interface （HMI, який поєднує вбудований процесор та сенсорний дисплей зі своєю оперативною пам'яттю (кадрове ОЗП) із підтримкою програмного забезпеченням Nextion Editor для розробки графічного інтерфейсу HMI.

Використовуючи програмне забезпечення NEXTION Editor, можна у режимі графічного редактора, перетягуючи компоненти (графіка, текст, кнопки, повзунки тощо) створити фнтерфейс та реалізувати режими текстової взаємодії на основі ASCII для кодування інформації на стороні дисплея.

Дисплей Nextion HMI підключається до периферійного MCU через послідовний TTL (5V, TX, RX, GND), щоб надавати

сповіщення про події, на які відповідає MCU, який оновлює стан дисплею Nextion.

Термінальні функції підтримуються реалізацією екранної клавіатури та функціями Touch-screen.

Продукти Nextion Intelligent Series мають більш потужне апаратне забезпечення з точки зору MCU, флеш-пам'яті та SRAM порівняно з Basic Series та Enhanced Series – функції відтворення аудіо, відео та анімації. Intelligent Series підтримує розширені програмні функції та функції, такі як прозорий компонент, ефект завантаження сторінки, компонент Move and Drag та інші.

Термінал "B" – має вбудовану мікроЕОМ, яка керує роботою всіх пристроїв, що входять до складу терміналу. Залежно від потужності мікро ЕОМ завдання растрової розгортки примітивів і заповнення кадрового буфера виконується або спеціальним малювальним процесором, або самою ЕОМ. Як і в попередньому випадку використовується спеціальний відео контролер, основним завданням якого є управління кадровим буфером і формування сигналів для керування відео монітором.

ЕОМ розвантажена від розгортки примітивів управління клавіатурою. Значно зменшується обсяг інформації, що передається між ЕОМ і терміналом, тому що інформація передається не в растровому вигляді, як в терміналі "А", а у вигляді адрес кінцевих точок примітивів.

Термінал "С" (рис. 1.19) – являє собою "інтелектуальний" термінал. На такий термінал можуть бути покладені практично всі завдання графічної системи, за винятком завдань моделювання, зберігання і формування геометричних моделей об'єктів.

На дисплейний процесор покладаються завдання управління всім потоком обробки і управління роботою пристроїв, що входять до складу терміналу. Цей варіант терміналу зручний для багато-термінальних систем проектування, коли центральна ЕОМ зберігає загальну базу даних, яку використовують ряд користувачів. Ресурси

центральної ЕОМ залучаються також для складних завдань геометричного моделювання.

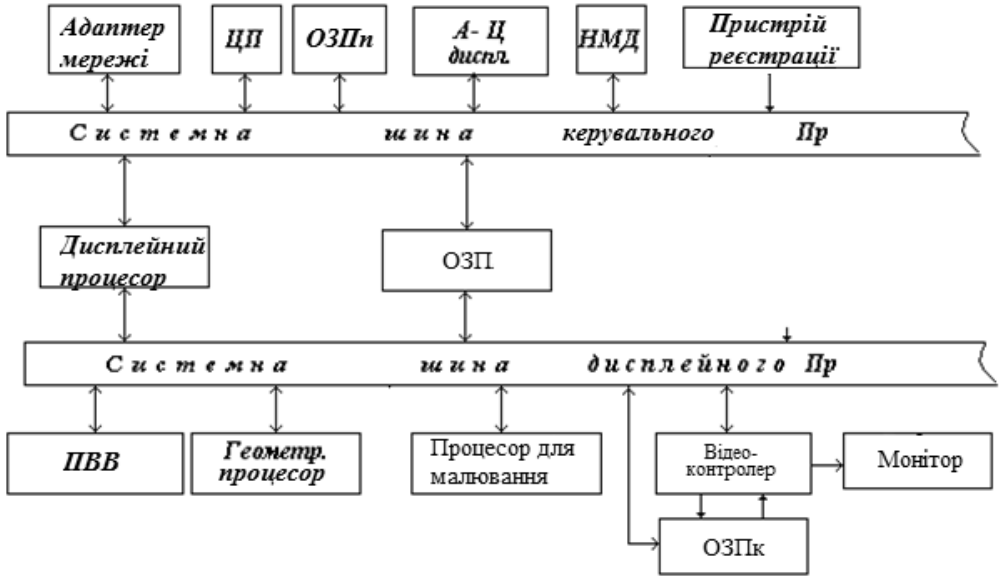


Рис. 1.19. Архітектура терміналу класу С

1.3.2. Кадровий запам'ятовувальний пристрій

Кадровий запам’ятовуючий пристрій є однією з основних компонент графічної системи, який багато в чому визначає принцип роботи як відео контролерів, так і графічних співпроцесорів.

Перш за все, зупинимося на обсязі і розрядності ОЗП. Як правило, обсяг кадрового ОЗП визначається інформаційною ємністю екрана, яка визначається як добуток кількості адресованих позицій екрану по кожній координаті – стовпців та строк.

У стандарті ICO 2382/13-84 визначено наступні терміни:

- Адресованість – число адресованих позицій по кожній осі координат фізичного простору.
- Адресна позиція – будь-яка точка фізичного простору, яка може бути задана координатами.

Найчастіше обсяг ОЗП зручно робити дещо більшого обсягу – це дозволить легко реалізувати операції панорамування (горизонтальний і вертикальний скролінг).

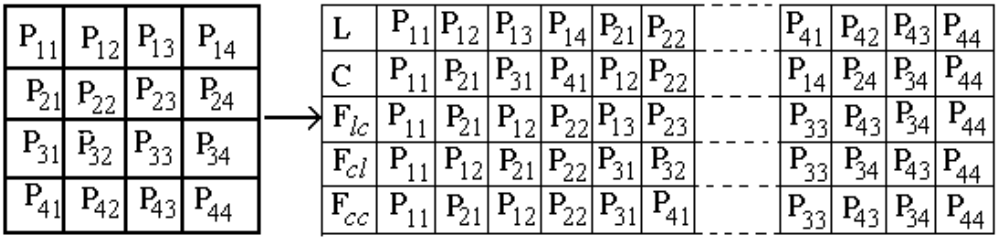
Глибина ОЗП, тобто його розрядність, визначається необхідною кількістю рівнів яскравості монохромної шкали (одноколірні зображення) і кількістю кольорових відтінків.

В якості початкового уявлення приймемо, що пам'ять регенерації організована у вигляді тривимірного масиву: елемент, що знаходиться на перетині конкретного рядка і стовпця, зберігає значення яскравості або кольору відповідної точки екрана. Механізм цієї простої взаємно-однозначної відповідності для двовимірного масиву (однобітова площина) – позиція екрану і елемент в пам'яті адресуються X-координатою від 0 до M-1 і Y-координатою, що змінюється від 0 до N-1. Верхній рядок масиву відповідає верхньому рядку растра і т.д. Регенерація зображення здійснюється послідовним скануванням буфера по рядках растра.

Простий буфер регенерації містить по одному біту на піксель (чорно-біле зображення) – часто використовується термін "бітова площина". Для півтонової картинки використовується кілька бітів на один елемент зображення (глибину пам'яті будемо розглядати як третій вимір).

Розглянемо цифрове зображення P, представлене у вигляді двовимірного масиву чисел. У послідовних обчислювальних процесах, в залежності від алгоритмів обробки, використовуються різні форми послідовного доступу до елементів цього масиву. На рис 1.20 показані три основні варіанти відображення масиву на лінійну пам'ять: в формі L – по рядках, C – по стовпцях, F – за фрагментами.

В останньому випадку можливі різні види упорядкування даних при переході від фрагмента і всередині кожного з них: F_{ll} – фрагменти впорядковані в напрямку рядків і елементи в кожному фрагменті впорядковані по рядках; F_{lc} – те ж саме, але елементи по стовпцях; F_{cl} – фрагменти впорядковані в напрямку стовпців і елементи в кожному фрагменті – по рядках; F_{cc} – теж, але елементи по стовпцях.



Ри. 1.20. Варіанти відображення масиву зображення на лінійну пам'ять

Розглянуті для даного прикладу зображення 4x4 способи організації даних в форматі F є вичерпними, але при збільшенні розміру масиву кількість варіантів швидко зростає.

Структура пристроїв пам'яті в більшості сучасних ЕОМ передбачає адресний доступ до елементів даних. При цьому проблема перетворення з однієї форми організації в іншу вирішується шляхом обчислення відповідних адрес елементів.

Кожен елемент цифрового зображення P представляється деякою послідовністю одиниць і нулів. На самому детальному рівні інформація про зображення – це тривимірна бітова структура (рис. 1.21).

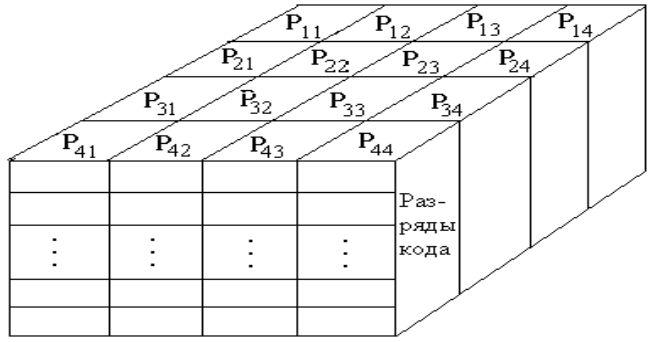


Рис. 1.21. Бітова структура масиву відео-ОЗП

Додавання ще однієї координати Q дозволяє ввести в розгляд додаткові форми організації даних [20]. Перш за все це розрядний зріз РСР і розрядний шар РСЛ цифрового

зображення. РСЛ утворюється набором РСР. Форми РСЛ і РСР мають самостійне значення, оскільки існує самостійний клас алгоритмів обробки зображень, які працюють з такими структурами даних.

Зупинимося тепер на проблемі побудови пам'яті, в якій могли б бути реалізовані можливості доступу до даних у всіх описаних вище формах. Розглянемо багатоблокову пам'ять, складену з блоків (модулів). Цей блок, орієнтований в площині РСР, дозволяє зберігати К біт РСР, а пакет з k блоків зберігає зображення в тому вигляді, як це показано на рис 1.22. При такій організації пам'яті доступ до елементу зображення (числу) забезпечується можливістю одночасного звернення до всіх блоків.

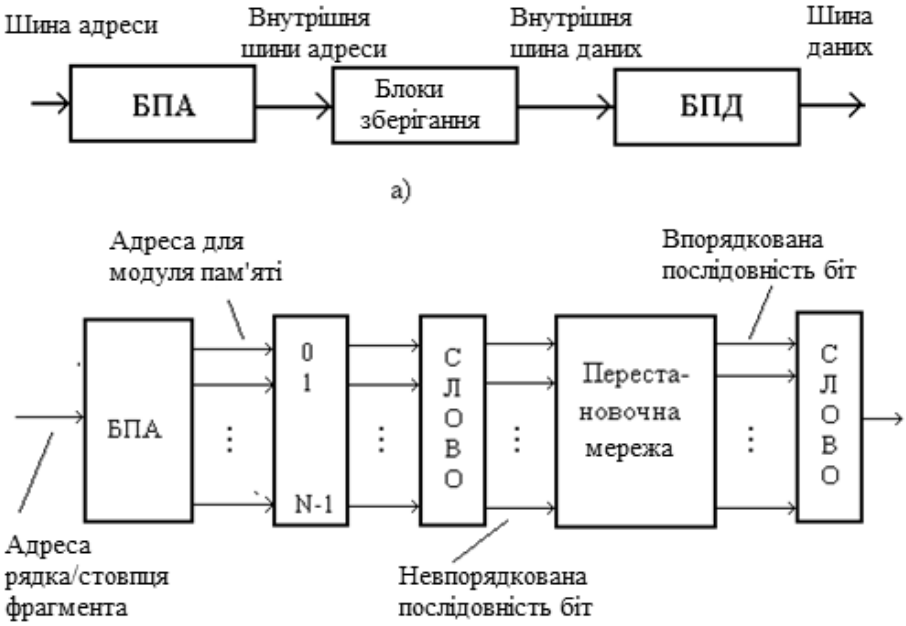


Рис. 1.22. Структура багатомодульної (багатоблочної) пам'яті

При побудові блоку пам'яті з паралельним (L, C) і багатовимірним (F) доступом до даних особливу роль відіграють блоки перетворення адрес і даних БПА і БПД. Вони забезпечують правильну адресацію і впорядкованість даних при використанні в блоці зберігання розподілу елементів РСР по

модулях пам'яті. На основі описаних принципів організації блоків пам'яті (n•m•1) може бути побудований пристрій пам'яті зображень, що забезпечує дуже гнучкі можливості роботи з даними.

Розглянемо деякі технічні проблеми, що виникають при проектуванні блоків пам'яті.

Отже, обсяг блоку пам'яті визначається (1.18) роздільною здатністю екрану:

$$Q > R_l \times R_c \times r \text{ [бит]}, \qquad (1.18)$$

де Q – обсяг блоку пам'яті;
R_l – роздільна здатність по горизонталі;
R_c – роздільна здатність по вертикалі;
r – розрядність.

Кількість розрядів визначимо (1.19) з вимог до кількості рівнів яскравості сірої шкали (монохромний монітор) або до кількості кольорів, що одночасно виводяться на екран:

$$r > \log 2\ M, \qquad (1.19)$$

де M – кількість рівнів сірої шкали
(кількість одночасно виведених кольорів).

Часові параметри блоку визначимо наступним чином. Якщо частота регенерації інформації на екрані дорівнює $F_в$, тоді час розгортки по вертикалі (час формування одного кадру зображення) складе:

$$T_в = 1/ F_в \qquad (1.20)$$

Оскільки при використанні ЕПТ частину часу повинна використовуватися для зворотного ходу по кадру, визначимо $T_в$ як:

$$T_в = T_{вп} + T_{во} \quad (1.21)$$

Де $T_{вп}$ – час прямого ходу по кадру (активна частина розгортки);
$T_{во}$ – час зворотного ходу по кадру.

Зазвичай обирають $T_{во} = (0,1 - 0,2)T_с$. Час на індикацію одного рядка визначимо як:

$$Tc = T_{вп}/ R_с \quad (1.22)$$

Як і в попередньому випадку частина часу повинна відводитися для зворотного ходу по рядку:

$$Tc = T_{сп} + T_{со} \quad (1.23)$$

де Тсп – час прямого ходу по рядку (активна частина розгортки);
T_{CO} – час зворотного ходу по рядку.

Приймемо, що $T_{CO} = (0,1 - 0,2)\ T_с$. Зазначені співвідношення для $T_{во}$ і T_{CO} справедливі тільки для випадку застосування в якості пристрою індикації ЕПТ. При використанні відеомоніторів на газорозрядних панелях, рідкокристалічних індикаторах, світлодіодних панелях і інших приймають $T_{во} = T_{со} = 0$.

У такому разі час, що відводиться на індикацію одного пікселя визначимо виразом:

$$T_п = T_{сп}/R_с \quad (1.24)$$

Приклад розрахунку характеристик кадрового ОЗП.
Задані: $F_в = 60$ HZ , $R_l = 640$, $R_с = 512$.
Визначити Тп для випадку використання ЕПТ.
1. Визначимо $T_в = 1/60 = 16,6 \cdot 10^{-3}$ с.
2. Визначимо $T_{во}$ як
$T_{во} = 0,15\ T_в = 0,15$ х $16,6 \bullet 10^{-3} = 2,4 \bullet 10^{-3}$ с,

звідки $T_{вп} = T_в - T_{во} = 14{,}2 \bullet 10^{-3}$ с.

3. Визначимо $T_с$:

$T_с = T_{вп} / P_с = 14{,}2 \bullet 10^{-6} / 512 = 27{,}7 \bullet 10^{-6}$ с.

$T_{со} = 0{,}1\ T_с = 2{,}77 \bullet 10^{-6}$ с.

$T_{сп} = T_с - T_{со} = 27{,}7 - 2{,}77 \bullet 10^{-6} = 24{,}9 \bullet 10^{-6}$ с.

4. Визначимо час індикації одного пікселя:

$T_п = T_{сп} / R_1 = 24{,}9 \bullet 10^{-6} / 640 = 38{,}9 \bullet 10^{-9}$ с.

Вже з наведеного прикладу видно, що час на індикацію одного пікселя малий і з ростом роздільної здатності відео моніторів убуває в квадратичній залежності. Так для екрану з роздільною здатністю 1024x1024 пікселів час на індикацію одного пікселя складає 8-10 нс. Цілком очевидно, що реалізувати ОЗП більшого обсягу із зазначеними часовими характеристиками якщо і можливо, то, по меншій мірі, важко. Тому вдаються або до використання ЗУ з розшаруванням (рис. 1.23), або до паралельної організації зчитування даних (рис. 1.24).

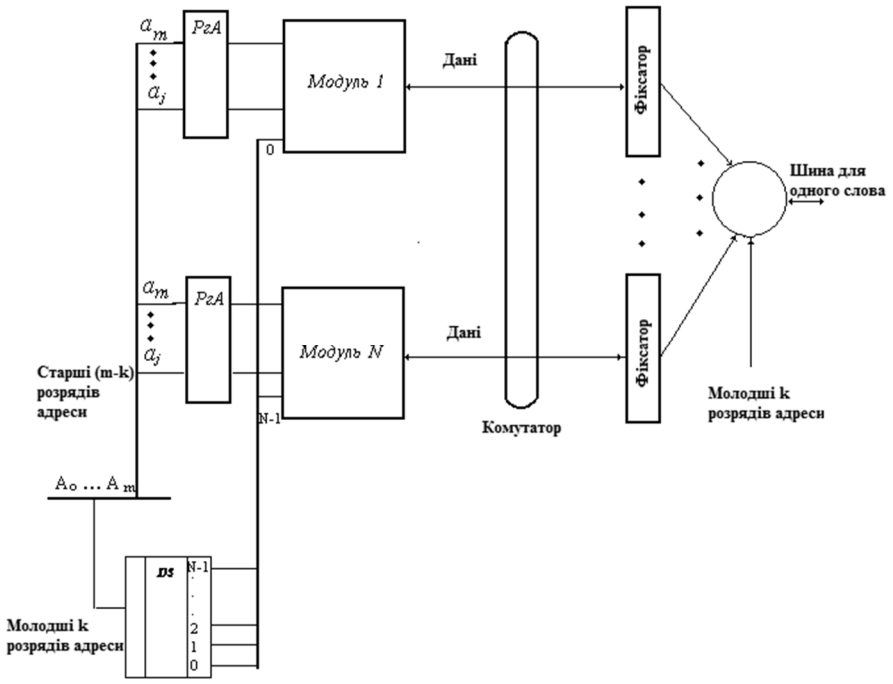


Рис. 1.23. Відео-ОЗП з розшаруванням

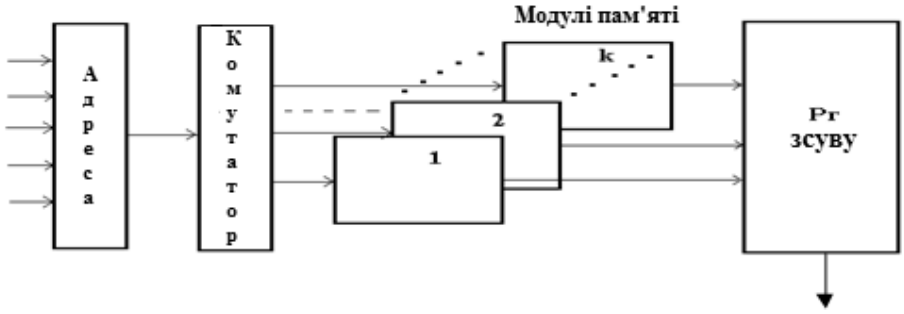


Рис. 1.24. Відео-ОЗП з паралельною організацією

При використанні ЗУ з розшаруванням модулі зазвичай упорядковуються так, щоб N послідовних адрес пам'яті i, i + 1, i + 2,. . . , I + N-1 доводилося на N різних модулів. В i-му модулі знаходяться тільки слова адреси, які мають вигляд кN + i, де 0<к <M–1, а M – число слів в модулі. Така адресація природна для пристроїв відображення, такт як кожний наступний піксель, що видається на індикацію має по відношенню до попереднього адреса збільшення на 1. В цьому випадку досягається збільшення швидкості в N раз:

$$T_{Ц.ЗУ} = T_{Ц.МОД} / N. \qquad (1.25)$$

Розшарування нагадує конвеєр, де "логічною функцією" є доступ до слова в модулі. Оскільки в даному ЗП може адресуватися кожен піксель, але в кожному циклі звернення до ЗП адреси записуваних пікселів можуть не задовольняти зазначеним вище вимогам, то мінімальна швидкість запису становитиме:

$$T_{Ц.ЗУ.ЗАП.МАКС} = T_{Ц.МОД}. \qquad (1.26)$$

Максимальна швидкість запису (наприклад, під час запису горизонтальних векторів):

$$T_{Ц.ЗУ.ЗАП.МИН} = T_{Ц.МОД}/N \qquad (1.27)$$

У ЗП з паралельним доступом в одному циклі звернення до ОЗП зчитується інформація про кількох пікселях. Ці дані записуються паралельно в зсувний регістр і потім послідовно виводяться на індикацію. У цьому випадку цикл звернення до ОЗП становить:

$$Тц.зу \geq k \bullet Тп \qquad (1.28)$$

де k – кількість пікселів, дані про які одночасно зчитуються з ОЗП.

При запису інформації в таке ОЗП формується масив з k слів, який і записується в одному циклі. Якщо під час запису оновлюється тільки один або кілька пікселів, наприклад, n, причому n <k, то запис здійснюється за циклом: зчитування - запис по масці у вхідний регістр даних (модифікація даних) - запис в запам'ятовувальну середу (цикл СМЗ – зчитування – модифікація – запис).

Паралельний (або швидкий) запис можливий тільки для горизонтальних векторів (для варіанту відображення типу L). У той же час при реалізації алгоритмів розгортки, наприклад, алгоритму Брезенхема, коли в кожному такті формується один піксель, причому координати кожного пікселя по X і Y будуть відрізнятися, доцільно використовувати інший принцип організації ОЗП, який отримав найменування "багатовимірний" або "просторовий" ОЗП.

Номер від 1 до 9 в кожній клітинці зображення вказує номер модуля ОЗП, в якому зберігається дана інформація.

Розглянемо принцип роботи багатовимірного ОЗП. Припустимо, що зображення розміром 3x3 пікселя проектується на k двомірних ОЗП. Поставимо задачу в такий спосіб: за один цикл читання або запису необхідно порахувати або записати інформацію в усьому просторі 3x3 елемента зображення. Це означає, що кожен піксель заданого простору (в даному випадку 9 пікселів) повинен проектуватися на своє власне ОЗП (рис. 1.25). У такті запису в усі 9 ОЗП за однією і тією ж адресою записується відповідна інформація про зображення.

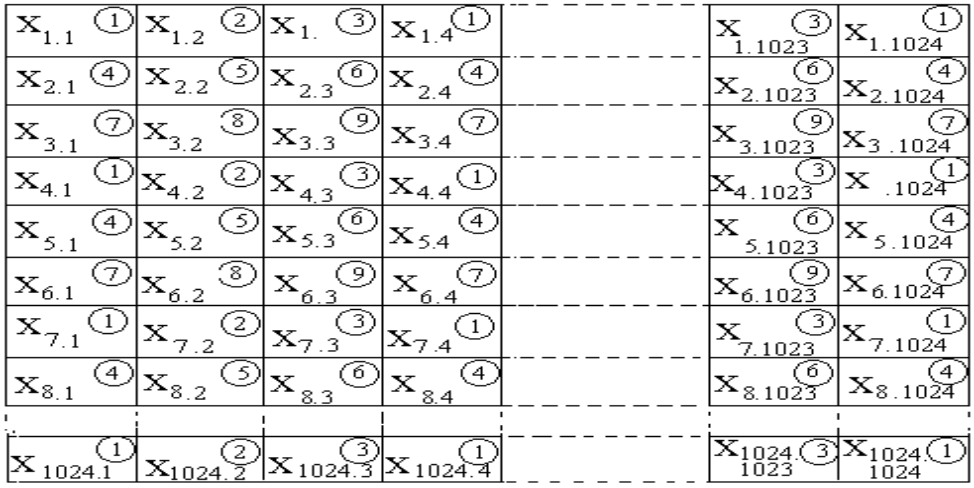


Рис. 1.25. Принцип розподілу інформації у багатовимірному відео-ОЗП

У такті зчитування (відрізняти від такту зчитування для регенерації інформації) за однією і тією ж адресою зчитується інформація з 9 ОЗП. Зчитана інформація записується в паралельний регістр для модифікації і знову записують слово в ОЗП або для пересилання в графічний процесор. При зчитуванні інформації для регенерації (повинен зчитуватися рядок зображення) для наведеного варіанту проектування зображення доцільно зчитувати для першого рядка тільки ОЗП 1 – ОЗП 3, для другого рядка ОЗП 4 – ОЗП 6, для третього ОЗП 7 – ОЗП 9, для четвертого ОЗП 1 – ОЗП 3 і т.д. Як бачимо, в цих режимах необхідна програмна перебудова блоків перетворення адрес і блоків перетворення даних. Зауважимо, що в даному випадку розглянутий тільки принцип роботи багатовимірного ОЗП, а не його дійсна технічна реалізація.

Така структура дозволяє здійснити одночасний запис в ОЗП r пікселів, довільно розташованих в межах заданого вікна. Кількість пікселів r визначається

$$r \le k, \qquad (1.29)$$

де k - кількість ОЗП.

Порівняємо роботу ОЗП з паралельним доступом і багатовимірний на прикладі запису трьох векторів: горизонтального вектора з трьох пікселів, вертикального вектора з трьох пікселів і діагонального вектора з трьох пікселів. В обох випадках вважаємо, що пікселі одно розрядні, ОЗП з паралельним доступом має три площини, багатовимірне ОЗП має 9 площин. Запис здійснюється по циклу: читання–модифікація–запис. У зв'язку з цим, приймемо, що тривалість циклу дорівнює

$$t_ц = 3T \qquad (1.30)$$

де Т – тривалість циклу звернення до ОЗП.

Результати порівняння наведені в табл. 1.4.

Таблиця 1.4

Порівняльні характеристики циклу звернення до ОЗП

Тип ОЗП	**Варіант вектора**		
	горизонтальний	**вертикальний**	**діагональний**
з паралельним доступом	$t_{зап} = t_ц$	$t_{зап} = 3\ t_ц$	$t_{зап} = 3\ t_ц$
з просторовим доступом	$t_{зап} = t_ц$	$t_{зап} = t_ц$	$t_{зап} = t_ц$

Узагальнюючи наведений приклад на багатовимірне ОЗП з k модулями ОЗП, отримаємо, що при записі в вікні n пікселів $n \le k$, отримаємо:

$$t_{зап} = t_ц \qquad (1.31)$$

проти $t_{зап} \le kt_ц$ для випадку ОЗП з паралельним доступом. Слід зауважити, що в разі багатовимірного ОЗП збільшується кількість ОЗП (при рівноцінному об’ємі) і ускладняються блоки перетворення адрес і даних.

Кадрове ОЗП може працювати в двох режимах:

- режим роботи з записом в довільний момент;
- режим роботи з записом на зворотних ходах.

У свою чергу перший режим із записом в довільний момент може записуватися наступним чином:

- використання розщепленого циклу ОЗП;
- подвійна буферизація пам'яті;
- запис з перериванням циклу регенерації зображення.

При використанні розщепленого циклу ОЗП, кожен цикл звернення до пам'яті складається з двох підциклів:

- підциклу запису
- підциклу зчитування на екран

Це вимагає збільшення швидкості роботи ОЗП в порівнянні з іншими способами в два рази при використанні ЗП з розшаруванням і в три рази при використанні ЗП з паралельним доступом (підцикл запису: зчитування – модифікація – запис).

При подвійний буферизації пам'яті використовується два ідентичних кадрових ОЗП, які працюють в пушпульному режимі – одне з них працює на екран, друге перезаписується. Після закінчення запису по кадровому синхроімпульсу відбувається перемикання ОЗП.

У третьому випадку запис проводиться в будь-який момент часу шляхом переривання циклу регенерації. Це призводить до "мерехтливого" режиму роботи екрану, так як на час запису виведення на екран даних з ОЗП припиняється і на ньому виникають світлі або темні смуги. Принципово цей недолік може бути усунутий тільки за рахунок використання спеціальних відео ОЗП з двома портами, що забезпечують можливість одночасного доступу до пам'яті з боку порту регенерації і порту запису. До таких ОЗП відносяться мікросхеми пам'яті типу IMS 4161 і MPD 41264.

У режимі запису на зворотних ходах запис інформації здійснюється тільки під час зворотного ходу по рядках і кадру. Можна вважати, що в цьому випадку для запису використовується близько 25% загального часу. Може бути запропонований наступний варіант розрахунку:

а) кількість пікселів, що записуються під час зворотного ходу по рядку:

$$N_c \leq T_{CO} / T_{Ц.ЗУ.ЗАП.МАКС} \quad (1.32)$$

б) кількість пікселів, що записуються під час зворотного ходу по кадру:

$$N_B \leq T_{BO} / T_{Ц.ЗУ.ЗАП.МАКС} \quad (1.33)$$

в) загальна кількість пікселів, що записуються в одному кадрі:

$$N_{\Sigma} = N_B + N_c (R_C - 1) \quad (1.34)$$

Як видно з наведеного матеріалу, у кожному конкретному випадку, який обумовлено індивідуальними характеристиками системи, для досягнення максимальної ефективності потрібно визначати свої окремі структурні рішення та виконувати розрахунки часових характеристик.

1.3.3. Індивідуальні завдання до розрахунку параметрів відео-ОЗП

Визначте параметри відео-ОЗП для заданої розподільчої здатності екрану та використовуваної палітри (табл. 1.5). знайти ємність відео-ОЗП, необхідну тактову частоту для синхронізації ІС-пам'яті, зробити висновок про можливий тип ОЗП та організацію доступу до даних відео-ОЗП.

Таблиця 1.5

Варіанти завдань для розрахунку

Варіант	Розп. здатність екрану, стовпч/строк	Палітра кольорів[1]	Частота оновлення кадрів, Гц
0	320/240	1024 К	15
1	640/480	64 М	56
2	1024/768	32к К	60
3	1024/768	64к К	75
4	1024/768	256 М	60
5	640/480	256 К	80
6	800/600	16к К	100
7	800/600	truecolor	75
8	1024/768	truecolor	100
9	1280/1024	truecolor	85

Література до 1 розділу

1. Computer Grafics & Geometry [Электронный ресурс] : международный научно-образовательный журнал / Московский инженерно-физический институт. – М. : МИФИ, 1999– . – Режим доступу: http://www.cgg-journal.com.
2. – X.Org Foundation [Електронний ресурс]: Офіційний сайт організації. Режим доступу: https://www.x.org/wiki/
3. Unix&Linux: How is the linux graphics stack organised? [Електронний ресурс]: – Режим доступу: https://unix.stackexchange.com/questions/7943/how-is-the-linux-graphics-stack-organised
4. Computer Graphics Basics – Tutorialspoint. [Електронний ресурс]: – Режим доступу:

[1] К – кольорове зображення; М – монохромне

https://www.tutorialspoint.com/computer_graphics/computer_graphics_basics.htm

5. "Graphics primitive." A Dictionary of Computing. Encyclopedia.com. (October 16, 2020). [Електронний ресурс]: – Режим доступу: https://www.encyclopedia.com/computing/dictionaries-thesauruses-pictures-and-press-releases/graphics-primitive

6. Геометрия стереофотосъёмки [Електронний ресурс]: – Режим доступу: https://www.ixbt.com/digimage/stereogeometry.shtml.

7. Delphi 5. Руководство разработчика: в 2-х томах: пер. с англ. уч. пособие.- М.: Изд. Дом «Вильямс», 2000.

8. Graphics in C Programming Language - MYCPLUS - Part 4 [Електронний ресурс]: – Режим доступу: https://www.mycplus.com/tutorials/c-programming-tutorials/graphics-programming/4/

9. Paint program in C. [Електронний ресурс]: – Режим доступу: https://www.programmingsimplified.com/c-paint-program

10. MFC - GDI – Tutorialspoint. [Електронний ресурс]: – Режим доступу: https://www.tutorialspoint.com/mfc/mfc_gdi.htm

11. List of platform-independent GUI libraries – Wikipedia. [Електронний ресурс]: – Режим доступу: https://en.wikipedia.org/wiki/List_of_platform-independent_GUI_libraries

12. The Tenouk's C++ and MFC programming tutorialsPart 1. Train and master yourself the Windows Graphic User Interface, GUIs, controls, forms, system and etc. using MFC C++ library. [Електронний ресурс]: – Режим доступу: https://www.tenouk.com/cplusplusnmfc.html

13. The Basics of GDI+, Markus Egger. [Електронний ресурс]: – Режим доступу: https://www.codemag.com/article/0305031

14. Getting Started with Graphics Programming - Windows Forms .NET Framework | Microsoft Docs [Електронний ресурс]: – Режим доступу: https://docs.microsoft.com/en-

us/dotnet/desktop/winforms/advanced/getting-started-with-graphics-programming?view=netframeworkdesktop-4.8

15. Custom Graphics Programming - Java Programming Tutorial. [Електронний ресурс]: – Режим доступу: https://www3.ntu.edu.sg/home/ehchua/programming/java/J4b_CustomGraphics.html

Trail: 2D Graphics (The Java™ Tutorials) «Painting in AWT and Swing» [Електронний ресурс]: – Режим доступу: https://docs.oracle.com/javase/tutorial/2d/index.html

17. Painting in AWT and Swing. [Електронний ресурс]: – Режим доступу: https://www.oracle.com/java/technologies/painting.html

18. Gouraud, H. (1971). Computer display of curved surfaces. 1-80. UTEC-71-113; UTEC-CSc-71-113

19. Gouraud shading – Wikipedia. [Електронний ресурс]: – Режим доступу: https://en.wikipedia.org/wiki/Gouraud_shading

20. Home – Nextion. [Електронний ресурс]: офіційна сторінка. – Режим доступу: https://nextion.tech/

21. Intel Unveils Neural Compute Engine in Movidius Myriad X VPU to Unleash AI at the Edge [Електронний ресурс]: офіційна сторінка. – Режим доступу: https://newsroom.intel.com/news/intel-unveils-neural-compute-engine-movidius-myriad-x-vpu-unleash-ai-edge/#gs.o3h1bl

РОЗДІЛ 2. ГЕОМЕТРИЧНІ ПЕРЕТВОРЕННЯ У СИСТЕМАХ КОМП'ЮТЕРНОЇ ГРАФІКИ

Підтримка операцій геометричного перетворення є невід’ємною частиною систем комп'ютерної графіки. Більшість подій у світі, що нас оточує, пов'язано із рухом різного типу. Людинне сприйняття оточення через зорові сенсори, у комп'ютерних системах може бути імітоване як раз низкою операцій геометричних перетворень.

Обертання голови (навкруги визначеної вісі), наприклад, може бути реалізоване найпростіше усього – виконанням панарамування, або лінійного зміщення плоского зображення, яке відповідає моделі оточення.

Лінійний рух призводить до цілого комплексу дій, але якщо це розглядати як рух уздовж оптичної вісі ока, то об’єкти, до яких ми рухаємося, вони будуть поступово збільшуватися (операція масштабування).

Таким чином, реалізація геометричних перетворень буде значною мірою впливати на якість та швидкість виконуваних операцій.

Слід зазначити, що спеціальні види геометричних перетворень відіграють значну роль, також, у процесах підготовки зображень до розпізнавання чи ідентифікації. Наприклад, корекція дисторсії у різних проявах – бочкоподібна, подушкоподібна та комплексна, у системах введення на основі оптичних лінз (фотоапарати, телескопи та мікроскопи, камери). Або геометрична корекція зображень у системах розпізнавання тексту.

Особливістю реалізації геометричних перетворень є те, що вони виконуються окремо для кожної вершини об'єкту моделі або для кожної точки растрового зображення.

У першому випадку кількість виконуваних операцій геометричного перетворення значно менша ніж у другому, але у процесі візуалізації потрібно виконувати додатково розгортання примітивів та фарбування поверхонь, що теж потребує великих

витрат ресурсів. При чому, ці затрати будуть змінними у залежності від умов їх виконання.

Геометричні перетворення растрових зображень, з точки зору витрат ресурсів, є більш постійними – кількість оброблюваних точок на растровому зображенні не змінюється, але призводить до появи специфічних дефектів та сполучення оброблюваного зображення.

Відповідно, при проектуванні системи, потрібно вміти оцінювати витрати ресурсів та обирати компроміс на основі великої кількості факторів.

Обробка «модельних» даних тісно пов'язана з умовним розташуванням об'єктів у «віртуальному» просторі світових координат та їх поступовому перетворенню у фізичні координати пристрою візуалізації. Тобто, одна із вирішуваних задач – вибір та операції із системами координат чи координатних просторів.

Іноді, вибір «правильної» системи координат значно зменшує вимоги до обчислювальної потужності. Так, наприклад, вибір полярної системи зменшує витрати на операціях обертання. Використання відносної системи координат дозволяє економити на операціях зсуву (лінійного переміщення) об'єктів і так далі.

Інша задача, яка виникає при цьому – вибір відповідної структури даних для зберігання інформації про вершини, які описують моделі, особливо полігональні моделі. Специфіка синтезу зображень потребує регулярної «регенерації», або перерисовки зображень, які відповідають використовуваній моделі.

Тема 2.1. Моделі використовуваних структур даних

Усі процеси у системах КГ пов'язані з найбільш загальною моделлю, представленою на рис. 2.1.

Структура даних містить опис реальних або абстрактних об'єктів, зображення яких повинні з'являтися на екрані. Тому в структурі даних може зберігатися вся необхідна інформація для

найрізноманітніших об'єктів, наприклад електричні схеми, інженерні споруди, функції, моделі ядерних реакторів і т.п.

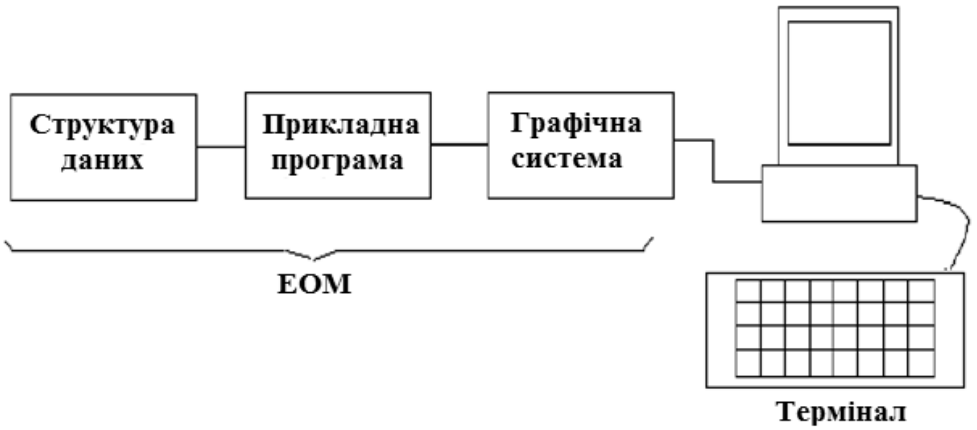


Рис. 2.1. Модель компонентів графічної системи

В опис об'єктів зазвичай включаються геометричні дані про координати, що визначають форму об'єктів або їх складових частин, атрибути об'єкта (колір, фактура поверхні), а також дані про зв'язності і положення. Часто є також негеометрична або текстова інформація про властивості, яка корисна, наприклад, для інтерактивного користувача. Прикладами таких даних можуть служити відомості про ціну і постачальників, характеристики і таке інше.

Прикладна програма описує двовимірну або тривимірну геометрію об'єктів та, що підлягають виведенню на видову поверхню, для графічної системи, яка зазвичай забезпечена комплектом графічних підпрограм виведення, сумісних з такими мовами високого рівня як Фортран і Паскаль. Цей пакет підпрограм керує конкретним пристроєм і забезпечує виведення зображення цим пристроєм зазвичай на основі дисплейного списку, сформованого пакета.

Перш за все, прикладний програміст будує прикладну модель об'єктів, з якими користувач буде працювати і які він буде розглядати. Прикладна модель (надалі просто модель) є сукупністю даних, що представляють об'єкти, і відносин цих даних. Зазвичай зберігається у вигляді прикладної структури

даних. Таким чином, модель відображає важливі властивості об'єктів, істотні для цього додатка або для ряду додатків. Ці об'єкти можуть бути конкретними або абстрактними і при візуалізації можуть бути представлені в двовимірному або тривимірному світі. Прикладами можуть служити математична функція, інтегральна схема, план приміщення, зубчата передача, молекула.

Прикладна графічна програма повинна описати для графічної системи в геометричних поняттях ту частину світу, зображення якого потрібно користувачеві. Незалежно від змісту бази даних (геометричні або негеометричні дані) вони повинні бути описані для графічної системи у вигляді вихідних графічних примітивів, таких як точки, лінії, полігони, ланцюжки літер і т.д.

Прикладна програма повинна також вказати графічній системі яку частину об'єкта слід показати, в якій точці знаходиться спостерігач, звідки направлено освітлення, на якій частині видової поверхні має з'явитися зображення.

Графічну систему зручно розглядати як уявну фотокамеру [1].

Прикладна програма надає цій фотокамері опис сцени, що складається з одного або більше об'єктів у деякому штучному світі – двовимірному або тривимірному.

Потім уявна камера створює вигляд цього об'єкта в цьому світі. Точний вид об'єкта залежить від того, як була налаштована фотокамера, де вона перебувала по відношенню до об'єкту, де знаходилося джерело освітлення. Як і при використанні фотоапарата типу "Polaroid" – зроблений знімок відразу проявляється і показується на видовій поверхні. Величина знімка може бути різною і займати всю видову поверхню або її частину.

Робота прикладної програми полягає в моделюванні та інтерпретації даних, що вводяться користувачем. Графічна система не будує і не змінює модель ні на початку роботи, ні у відповідь на інтерактивні дії користувача. Вона як фотокамера тільки робить знімки.

Необхідність описувати просторові форми з'являється в двох випадках. По-перше, коли необхідно змоделювати вже існуючий об'єкт: моделі літака або автомобіля, деталі, архітектурної споруди. У загальному випадку, об'єкт не може повністю відповідати своїм представленням. Для цього необхідно нескінченне число трійок *(x,y,z)* - по одній на кожну точку об'єкта. У кращому випадку об'єкт вдається представити як комбінацію математичних поверхонь, таких як площини, сфери, циліндри та ін.

Така необхідність виникає також в задачах автоматизованого проектування. У цьому випадку ні об'єкта ні його моделі не існує і конструктор, керуючись своїм уявленнями про проектований об'єкт створює просторову форму, діючи в інтерактивному режимі.

Існують два загальновизнаних способу тривимірного представлення поверхонь в просторі: полігональна сітка і параметричні бікубічні криві.

Полігональної сіткою є сукупність пов'язаних між собою плоских багатокутників (сегментів, граней). Сітка може бути задана декількома різними способами.

Явне завдання багатокутників. Кожен багатокутник можна представити у вигляді списку координат його вершин:

$$P = [(x_1, y_1, z_1), (x_2, y_2, z_2,), \ldots . (x_n, y_n, z_n)] \qquad (2.1)$$

Вершини запам'ятовуються в тому порядку, в якому вони зустрічаються при обході навкруги багатокутника. При цьому всі послідовні вершини багатокутника, а також перша і остання з'єднуються ребрами.

Якщо для окремого багатокутника даний спосіб завдання є ефективним, то для полігональної сітки призводить до великих втрат (рис. 2.2) внаслідок дублювання інформації про координати загальних вершин. Крім того, полігональна сітка зображується шляхом креслення ребер кожного багатокутника, що призведе до повторного викреслювання суміжних ребер.

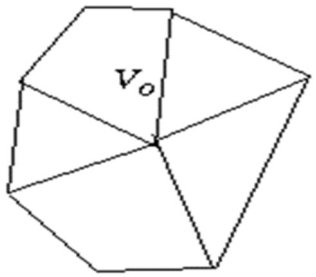


Рис. 2.2. Полігональна сітка для неплоскої поверхні

Завдання багатокутників за допомогою покажчиків в списку вершин.

При цьому поданні кожна вершина запам'ятовується одноразово в списку вершин полігональної сітки.

$$V = [(x_1, y_1, z_1), \ldots . (x_n, y_n, z_n)] \qquad (2.2)$$

Після цього, багатокутник визначається списком покажчиків (індексів) в списку вершин. Наприклад, багатокутник, що складається з вершин V4, V7, V8, V9, представляється як P = (4,7,8,9). На рис. 2.3 наведено приклад такого подання. Воно має ряд переваг в порівнянні з явним завданням, оскільки кожна вершина запам'ятовується в списку вершин тільки один раз. Однак, тут непросто відшукати багатокутники з загальними ребрами.

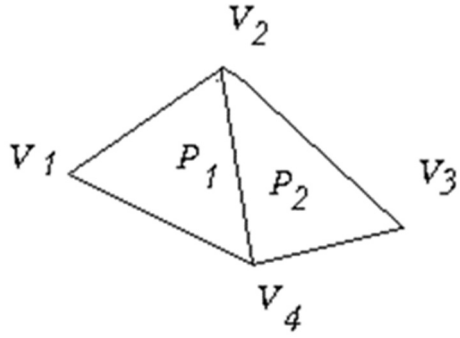
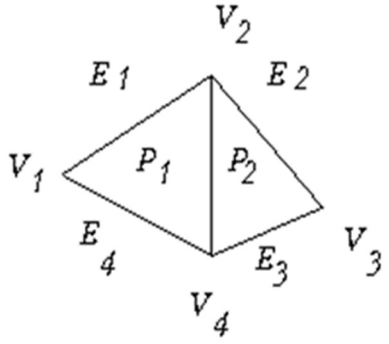


Рис. 2.3. Опис полігональної сітки індексами вершин

Явне завдання ребр. На рис. 2.4 дано представлення полігональної сітки з явним завданням ребр. У цьому варіанті завдання полігональна сітка зображується виведенням не багатокутників, а їх ребр. В результаті вдається уникнути багатократного малювання загальних ребр.



·V=·(V1,V2,V3,V4)=·[(x1,y1,z1),¶
·(x2,y2,z2),(x3,y3,z3),(x4,y4,z4)]¶
·E1=(V1,V2,P1,r)·E2=(V2,V3,P2,r)¶
·E3=(V3,V4,P2,r)·E4=(V4,V2,P1,P2)¶
·E5=(V4,V1,P1,r)¶
·P1=(E1,E4,E5)¶
·P2=(E2,E3,E4)¶
·r·-·означає·"пусто".¶

Рис. 2.4. Опис явним завданням ребр

Далі, згідно з структурною схемою рис. 1.8., вершини, які складають модель, модифікують у залежності від того, що з цим моделями потрібно виконати, але є стійка визначена послідовність базових перетворень, які складають суть комп'ютерної графіки. Таким чином формується конвеєр, представлений на рис. 2.5.

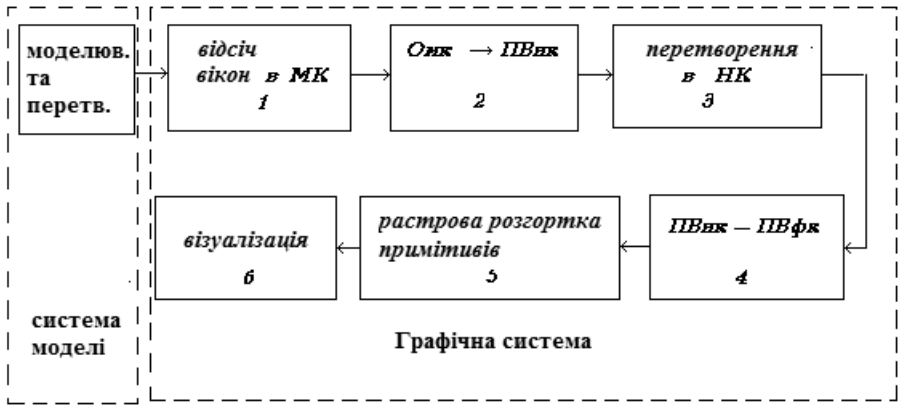


Рис. 2.5. Послідовність базових перетворень у графічних системах

Така схема виконання операцій, сформована ще на початку розвитку засобів комп'ютерної техніки, у сучасних системах має багато різноманітних реалізацій. У найпростіших широковживаних (універсальних) системах операції графічної обробки виконуються, як за правило, програмними методами окрім задач візуалізації. У спеціалізованих графічних системах майже усі задачі знайшли апаратне втілення. Для цього були, також й історичні передпосилки – швидкість виконання програмного коду у перших комп'ютерних системах була дуже низька, і тому розробники велику кількість задач вирішували на основі апаратних розробок. У сьогоденні, ці розробки стали прототипами існуючих графічних співпроцесорів або їх компонентів, які формують загальну структуру графічної системи, представлену на рис. 2.6.

Комутаційний процесор (КП) виконує функції обміну інформацією між системою моделювання і графічною системою.

Основні функції дисплейного процесора:

- управління дисплейними списками (файлами);
- підтримка пристроїв діалогу, тобто організація інтерфейсу користувача;
- адміністрування взаємодії всіх інших процесорів ГС.

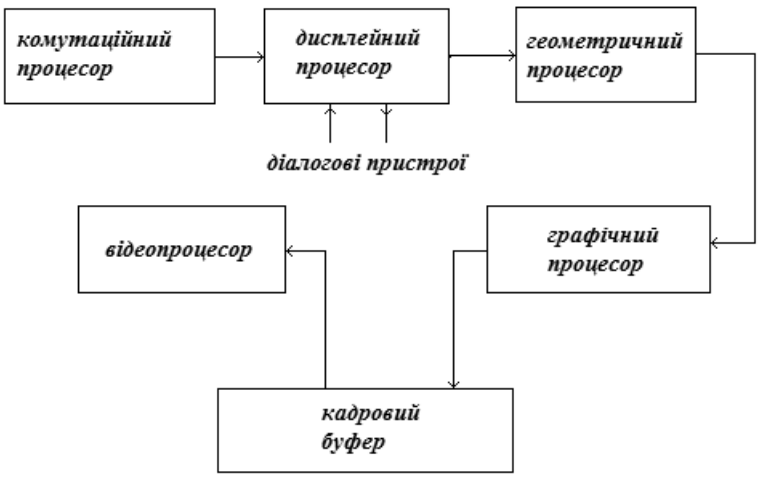


Рис. 2.6. Структура логічних співпроцесорів графічної системи

Дисплейний процесор має статус центрального процесора графічної системи і в разі потреби виконує (емулює) функції інших співпроцесорів.

Геометричний процесор (ГП) виконує дії, спрямовані на зміну геометрії зображення об'єкта - перенесення, поворот, масштабування, проекції, перетворення координат, відсікання примітивів. Як правило, всі перетворення даного типу описуються матрицею перетворювання розмірністю 4x4 для тривимірних і матрицею 3x3 для двовимірних перетворень. Члени матриці - це числа в форматі з плаваючою комою, що задають усі різновиди геометричних змін зображення. Тому, основа геометричного процесора - матричний помножувач, що обчислює добуток всіх координат зображення на матрицю перетворення для отримання нового виду зображення.

Графічний процесор (ГрП) створює в кадровому буфері цифровий еквівалент зображення об'єкта. Він отримує від геометричної підсистеми координати примітивів (символи, відрізки, прямокутники і багатокутники, дуги), що складають зображення об'єкта, викреслює їх в кадровому буфері, тобто, заносить туди значення властивостей усіх точок, що складають растрову розгортку зображення. Розгортання полягає в обчисленні координат проміжних точок відрізків, контурів дуг, внутрішніх точок багатокутників для їхнього зафарбування і

таке інше. Для реалістичної передачі освітленості об'єктів обчислюються також інтенсивності кольорів в кожній точці зображень.

Всі описані функції, виконувані КП, ГП і ГрП прийнято називати відновленням зображення. Регенерацію зображення покладено на відеопроцесор (ВП), який послідовно сканує цифровий еквівалент зображення в кадровому буфері, витягує цифрові значення яскравості і колірних характеристик кожної точки, що становлять зображення, перетворює ці значення в відеосигнали і передає їх у монітор.

Розглянемо далі особливості реалізації графічних перетворень у вказаних логічних співпроцесорах.

2.1.1. Особливості виконання геометричних перетворень.

Першою особливістю є необхідність виконання перетворень для плоских (2-D) та об’ємних (3-D) моделей об'єктів. Іншою – те що координати вершин представляють і перетворення та виконують у однорідних системах координат.

Найбільш часто виникають задачі обрахунку координат вершин при виконанні операцій здвигу (позначимо T - translating, shift), масштабування (позначимо S - scaling) та обертання (позначимо R - rotating).

Операція здвигу (рис. 2.7) задається співвідношенням:

$$x' = x + t_x$$
$$y' = y + t_y \qquad (2.3)$$

Інша операція – масштабування (рис. 2.8). базовим є масштабування відносно центру системи координат. Зміна коефіцієнтів масштабування призводить до ефекту руху точки уздовж радіальної лінії до центру (коефіцієнти менше 1) або від центру (коефіцієнти більші 1). У загальному випадку, масштабування може виконуватись відносно довільної точки $O(x_O,y_O)$. Але тоді ця дія виконується у три етапи – площина

перетворення зміщується таким чином, що точка О співпадає з центром системи координат. Потім, виконується масштабування і на третьому етапі виконується зворотнє зміщення, яке повертає площину у її початкове положення.

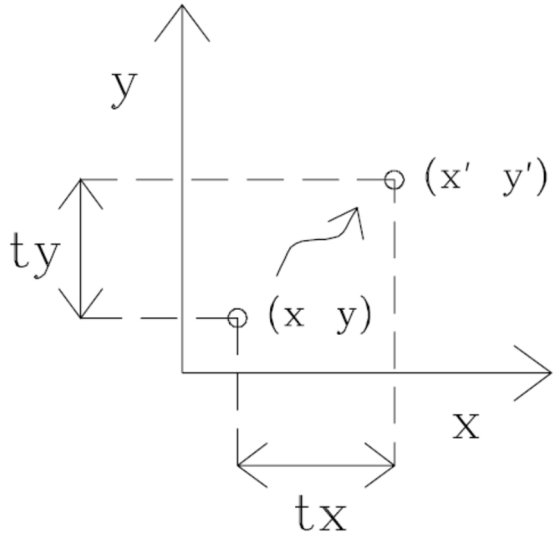


Рис. 2.7. Операція здвигу (переносу)

Операція масштабування задається співвідношенням 2.4:

$$x' = x \cdot s_x$$
$$y' = y \cdot s_y \qquad (2.4)$$

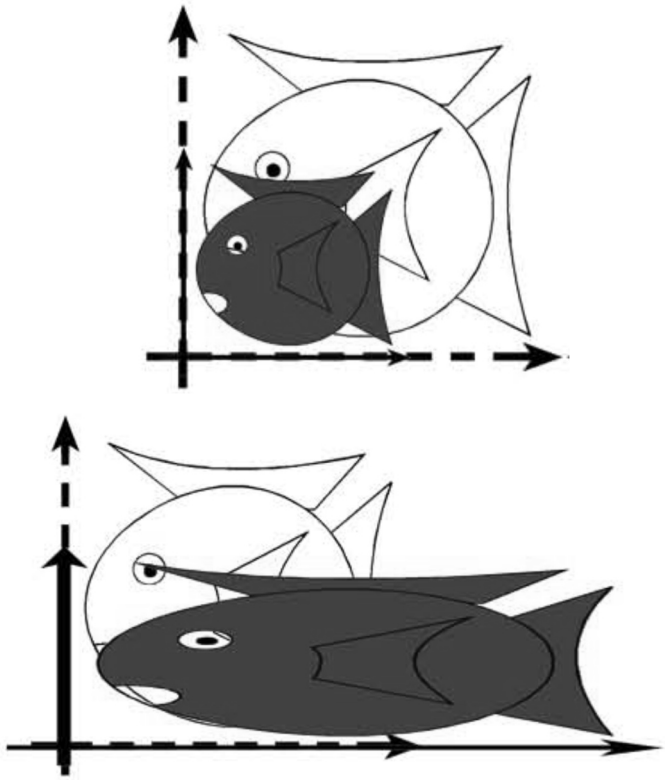


Рис. 2.8. Виконання операції масштабування

Подібно до операції масштабування, операція обертання відносно довільної точки виконується також у три етапи. У найпростішому виді, обертання (рис. 2.9) задається співвідношенням 2.5:

$$\begin{aligned} x' &= x \cdot \cos\alpha - y \cdot \sin\alpha \\ y' &= x \cdot \sin\alpha + y \cdot \cos\alpha \end{aligned} \qquad (2.5)$$

Однак, з точки зору організації процесу обчислення перетворень, їх виконання потребує різні типи обчислень і, відповідно, наявність різних типів операційних модулів. Для уніфікації їх подання та виконання обчислень використовують матричне представлення та однорідні координати.

За допомогою однорідних координат всі три перетворення можна реалізувати за допомогою множення.

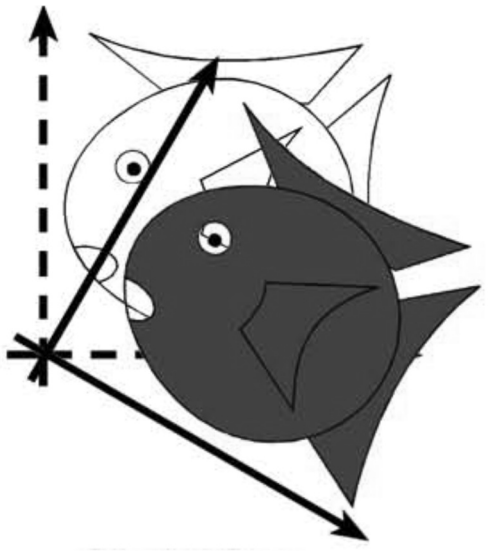


Рис. 2.9. Обертання об'єкту відносно центру системи координат

В однорідних координатах точка *P(x, y)* записується як *(W*x, W*У, W)* для будь-якого масштабного множника W не дорівнює нулю. При цьому, якщо для точки задано її подання в однорідних координатах *P(x, y, w)*, то можна знайти її двовимірні Декартові координати як *x=x/W* і *y= y/W*. Якщо прийняти значення W = 1, операція ділення не потрібна. Однорідні координати можна уявити як вкладення масштабованої з коефіцієнтом W двовимірної площини в площину z = W (z = 1) в тривимірному просторі (2.6).

$$(x, y) \rightarrow (x, y, 1) \tag{2.6}$$

Точки тепер описуються триелементними вектор-рядками, тому матриці перетворень, на які множиться вектор точки, повинні мати розмір 3*3. Рівняння перенесення (2.7) записується у вигляді матриці перетворення однорідних координат:

$$[x'y'1]= [xy1]\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{vmatrix} <=> P' = P\bullet\ T\ (Dx\ ,\ Dy) \qquad (2.7)$$

Де операція зсуву задана матрицею із значеннями зсуву вздовж вісі x – D_x та вісі y – D_y.

$$T(D_x,D_y) = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x & D_y & 1 \end{vmatrix} \qquad (2.8)$$

Що буде, якщо точку P перенести в точку P' на відстань $(Dx_1,\ Dy_1)$, а потім в P'' на відстань $(Dx_2,\ Dy_2)$? Інтуїтивно очікуваний результат являє собою сумарне перенесення на відстань: (Dx_1+Dx_2,Dy_1+Dy_2):

$$P''=P\bullet T(D_{x1},D_{y1})\bullet T(D_{x2},D_{y2}) \qquad (2.9)$$

Матричний добуток $T(D_{x1},D_{y1})\bullet T(D_{x2},D_{y2})$ є:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x1 & D_y1 & 1 \end{vmatrix} * \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x2 & D_y2 & 1 \end{vmatrix} = \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ D_x1 + D_x2 & D_y1 + D_y2 & 1 \end{vmatrix} \qquad (2.10)$$

Рівняння масштабування в матричній формі записуються у вигляді:

$$[x'y'1] = [x\ y\ 1]\begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.11)$$

Визначаючи:

$$S(S_x, S_y) = \begin{vmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{vmatrix}, \qquad (2.12)$$

отримаємо:

$$P' = P \; x \; S(S_x, S_y) \qquad (2.13)$$

Так само як послідовні переноси є адитивними, можна очікувати, що послідовні масштабування будуть мультиплікативними.

Якщо задані $P'=P \bullet S(S_{x1}, S_{y1})$ та $P''=P' \bullet S(S_{x2}, S_{y2})$, отримуємо:

$$P'=P*[S(S_{x1}, S_{y1})*S(S_{x2}, S_{y2})] \qquad (2.14)$$

Матричний добуток $S(Sx1,Sy1) \bullet S(Sx2,Sy2$ є

$$\begin{vmatrix} S_x1 & 0 & 0 \\ 0 & S_y1 & 0 \\ 0 & 0 & 1 \end{vmatrix} \begin{vmatrix} S_x2 & 0 & 0 \\ 0 & S_y2 & 0 \\ 0 & 0 & 1 \end{vmatrix} = \begin{vmatrix} S_x1S_x2 & 0 & 0 \\ 0 & S_y1S_y2 & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.15)$$

Тобто, масштабування мультиплікативні.

Рівняння для виконання повороту можна представити у вигляді:

$$[x'y'1]=[xy1] \begin{vmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.16)$$

Вважаючи $R(\theta)=\begin{vmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix}$ отримаємо:

$$P' = P \bullet R(\theta) \qquad (2.17)$$

Два послідовних повороту є адитивними по кутах:

$$R(\theta 1)\bullet R(\theta 2)=\begin{vmatrix} \cos(\theta 1+\theta 2) & \sin((\theta 1+\theta 2) & 0 \\ -\sin(\theta 1+\theta 2) & \cos(\theta 1+\theta 2) & 0 \\ 0 & 0 & 1 \end{vmatrix} \qquad (2.18)$$

Композиція перетворень. Але найбільш важливою є здатність описати цілу низку послідовних перетворень однією матрицею композиції. Композицією будемо називати добуток матриць. Розглянемо, як можна використовувати композицію перетворень для об'єднання фундаментальних матриць R, S, T з метою отримання бажаних загальних результатів. Перевага об'єднання перетворень полягає в тому, що до точки зручніше застосувати одне результуюче перетворення, ніж ряд перетворень одне за одним.

Розглянемо на прикладі обертання об'єкта щодо деякої довільної точки P1. Раніше розглянуто поворот відносно початку координат. Тому розіб'ємо цю задачу на три етапи:

- перенесення площини так, що точка P1 зсувається в початок координат і відповідно зсувається об'єкт;
- поворот об'єкта щодо центру системи координат;
- перенесення площини у початкове положення і відповідний зсув об'єкту.

Зазначена послідовність приведена на рис. 2.10.

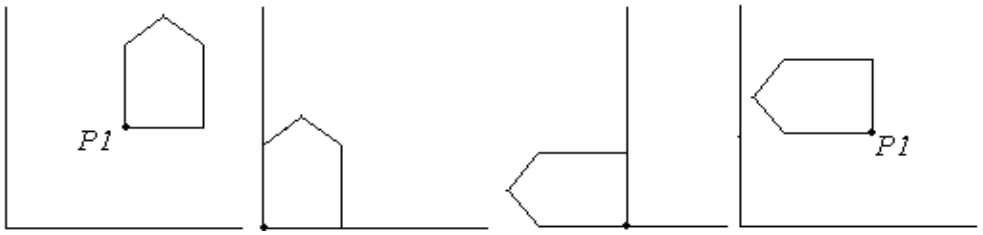


Рис. 2.10. Приклад обертання навколо довільної точки

Відповідно до зазначених умов, матриця композиції перетворень буде визначатися як добуток похідних матриць відповідних складових операцій:

$$\begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x & -y & 1 \end{vmatrix} \text{x} \begin{vmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{vmatrix} \text{x} \begin{vmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x & y & 1 \end{vmatrix} = \begin{vmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ x(1-\cos\theta)+ & y(1-\cos\theta)- & 1 \\ +y\sin\theta & -x\sin\theta & \end{vmatrix} \qquad (2.19)$$

Композиція найбільш загального вигляду, складена з операцій R, S і T, має матрицю:

$$\text{M}= \begin{vmatrix} r_{11} & r_{12} & r_{13} \\ r_{21} & r_{22} & r_{23} \\ t_x & t_y & 1 \end{vmatrix} \qquad (2.20)$$

Її верхня частина розміром 2x2 елементи є об'єднанням матриць повороту і масштабування, в той час як t_x и t_y описують сумарний зсув. Для обчислення координат точки як добутку вектору на матрицю розміром 3x3 потрібно 9 операцій множення і 6 операцій додавання. Структура останнього стовпця матриці у співвідношенні (2.20) дозволяє спростити фактично виконувані дії:

$$\begin{aligned} x' &= x r_{11} + y r_{21} + t_x \\ y' &= x r_{12} + y r_{22} + t_y \end{aligned} \qquad (2.21)$$

Скоротивши процес обчислень до 4 операцій множення і 4 операцій додавання. Це істотно прискорює процес, особливо якщо врахувати, що перетворенням можуть піддаватися сотні і навіть тисячі точок на кожному зображенні. Якщо ж множення матриць виконується апаратно за допомогою спеціальних схем, то питання ефективності перестає бути актуальним.

Однією з областей застосування, в якій важливо швидкодію, є генерація послідовних кадрів-зображень об'єктів, коли кожен наступний кадр відрізняється від попереднього

поворотом об'єкта на малий кут. Якщо кожен наступний кадр будується і виводиться на екран дисплея досить швидко - протягом 30-60 мс - то поворот об'єкту буде здаватися безперервним і плавним. Для досягнення такого ефекту кожну точку об'єкта слід перетворити якомога швидше. У рівняння повороту входить 4 операції множення і 2 операції додавання. Якщо врахувати, що кут повороту дуже малий - кілька градусів - можна прийняти, що $cos\ \theta = 1$. При такій апроксимації рівняння повороту набуває вигляду:

$$x' = x - y*sin\ \theta$$
$$y' = x*cos\ \theta + y \qquad (2.22)$$

Як видно, потрібно тільки 2 операції множення і 2 операції додавання. Тут з'являється небезпека накопичення помилки, що виникає при наближених обчисленнях. В результаті ми можемо отримати замість зображення об'єкта випадковий набір ліній. Це утруднення можна подолати, якщо зберігати вихідне зображення і після кожного повороту на 360 градусів відкидати отримані дані, замінюючи їх вихідними.

Наведений приклад показує, що програміст або розробник графічної системи повинен аналізувати як класи зображень так і способи реалізації графічних перетворень, вибираючи найбільш ефективні.

Розглянуті співвідношення, як видно, відповідають процесам перетворень у площині (2D системи та перетворення). Але велика частина задач відображення пов'язана або з моделями просторових даних, або з багатовимірними розподіленнями.

Для відображення моделей, пов'язаних зі світовим простором, використовують 3D системи координат.

Існує дві основні тривимірні системи координат - правостороння і лівостороння. Приймемо, що позитивними будемо вважати такі повороти, при яких (якщо дивитися з кінця позитивної півосі в напрямку початку координат) поворот проти годинникової стрілки на 90 градусів буде переводити одну

позитивну піввісь в іншу. На основі цієї угоди побудуємо таблицю, яка справедлива як для лівосторонньої так і правобічної систем координат:

Вісь обертання	Позитивний напрямок
X	від Y до Z
У	від Z до X
Z	від X до Y

У КГ частіше буває зручна лівостороння система, так як вона більше відповідає поверхні екрану дисплею. У такій системі природно інтерпретується той факт, що точки з великим значенням z знаходяться далі від спостерігача. В лівосторонній системі координат позитивними будуть повороти за годинниковою стрілкою, а в правосторонній - проти годинникової стрілки, якщо дивитися з кінця позитивної півосі в напрямку початку системи координат.

В окремих випадках зручно використовувати сферичні координати (рис. 2.11), які характеризуються довжиною вектору від центра системи координат до поточного положення точки та двома кутами (уклін та азимут, як приклад).

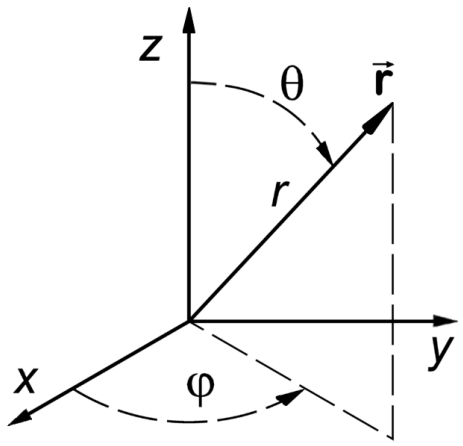


Рис. 2.11. Сферична система координат

Координати Декартової системи координат та сферичної системи координат пов'язані співвідношеннями (2.23) та (2.24):

$$\begin{aligned} x &= r\sin\theta\cos\phi \\ y &= r\sin\theta\sin\phi \\ z &= r\cos\theta \end{aligned} \qquad (2.23)$$

$$r = \sqrt{x^2 + y^2 + z^2}$$

$$\theta = \arccos(z/r), \quad r \neq 0$$

$$r' = r\sin\theta = \sqrt{x^2 + y^2}$$

$$\phi = \begin{cases} \arccos(x/r') & \text{if} \quad y \geq 0, \quad r' \neq 0 \\ 360^0 - \arccos(x/r') & \text{if} \quad y < 0, \quad r' \neq 0 \end{cases} \qquad (2.24)$$

Перехід від тривимірної системи координат до однорідної відбувається за правилами, подібними до двовимірної системи координат:

(x ,y, z) <=> (Wx, Wy, Wz, W), де $W \neq 0$. (2.25)

А базові операції геометричних перетворень є розширенням відповідних операцій зсуву, масштабування та обертання, але останнє може відбуватися навкруги трьох вісей. Відповідні матриці перетворення представлені нижче.

$$T(Dx,Dy,Dz) = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ D_x & D_y & D_z & 1 \end{vmatrix} \qquad (2.26)$$

$$S(Sx,Sy,Sz) = \begin{vmatrix} S_x & 0 & 0 & 0 \\ 0 & S_y & 0 & 0 \\ 0 & 0 & S_z & 0 \\ 0 & 0 & 0 & 1 \end{vmatrix} \qquad (2.27)$$

$$R_x(\alpha) = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & \cos\alpha & -\sin\alpha & 0 \\ 0 & \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.28)$$

$$R_y(\alpha) = \begin{bmatrix} \cos\alpha & 0 & \sin\alpha & 0 \\ 0 & 1 & 0 & 0 \\ -\sin\alpha & 0 & \cos\alpha & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.29)$$

$$R_z(\alpha) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 & 0 \\ \sin\alpha & \cos\alpha & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (2.30)$$

Усі інші операції можуть бути представлені як композиція базових перетворень. Так, наприклад, обертання навколо довільної вісі, що проходить через дві точки A та B, може бути представлена наступною послідовністю елементарних (базових) дій:

1) зміщення простору з параметрами, які суміщують, наприклад, точку A з центром системи координат;
2) послідовність обертань, яка суміщує трансформовану вісь AB з якоюсь з основних вісей;
3) обертання з заданим кутом;
4) зворотня до п.2 послідовність обертань;
5) зворотній до п.1 зсув.

Після виконання цих дій буде отримано кінцеве перетворення.

При довільному добутку матриць перенесення, повороту, масштабування результуюча матриця композиції тривімірного перетворення матиме вигляд:

$$M = \begin{vmatrix} r_{11} & r_{12} & r_{13} & 0 \\ r_{21} & r_{22} & r_{23} & 0 \\ r_{31} & r_{32} & r_{33} & 0 \\ t_x & t_y & t_z & 1 \end{vmatrix} \quad (2.31)$$

Верхня ліва підматриця розміром 3x3 буде описувати сумарний поворот і масштабування, підматриця T задаватиме сумарний зсув.

Для реалізації операції за допомогою матриці потрібно 9 операцій множення і 9 операцій додавання з плаваючою комою.

Сьогодні, усі базові програмні засоби, за допомогою яких здійснюється обробка зображень чи їх синтез, мають у своєму складі компоненти, які призначені для виконання геометричних перетворень та робот з матрицями перетворень.

Наприклад, у крос-платформному пакеті OpenCV [https://www.tutorialspoint.com/opencv/index.htm], перетворення здійснюється за допомогою метода *warpAffine()* класу *imgproc*.

Для здійснення перетворення потрібно знати параметри перетворення, які у OpenCV можуть бути отримані двома способами – безпосеретнім завданням, якщо вони відомі, або завданням положення трьох точок у початковому та кінцевому розташуванні. Наведений нижче код відповідає безпосередньому завданню параметрів перетворення (обертання) та їх виконання.

```
Point center = Point( warp_dst.cols/2, warp_dst.rows/2 );
double angle = -50.0;
double scale = 0.6;

Mat rot_mat = getRotationMatrix2D( center, angle, scale );

Mat warp_rotate_dst;
warpAffine( warp_dst, warp_rotate_dst, rot_mat, warp_dst.size() );

imshow( "Source image", src );
imshow( "Warp", warp_dst );
imshow( "Warp + Rotate", warp_rotate_dst );
```

Строка «*Mat rotationMatrix = Imgproc.getRotationMatrix2D(point, angle, scale);*» генерує відповідну матрицю перетворення, яка приймає у якості аргументів значення центру обертання (*point*), кут оберту (*angle*) та коефіцієнт масштабування (*scale*).

Базова операція у форматі «*Imgproc.warpAffine(src, dst, rotationMatrix, size);*» вказує на виконання перетворення початкового зображення *src* у результуюче зображення *dst* на основі матриці перетворення *rotationMatrix*.

На рис. 2.12 представлено похідне зображення, а на рис. 2.13 – результат перетворення згідно наведеного коду.

Подібні та процедури є і у інших пакетах. Так, наприклад, бібліотека для роботи з графікою OpenGL. Так, OpenGL підтримує матрицю “modelview”, яка утримує поточну трансформацію M. Матриця Modelview застосовується до точок (зазвичай вершин багатокутників) перед виведенням. Вона модифікується за допомогою команд, наведених на рис. 2.14.

Рис. 2.12. Похідне зображення

Рис. 2.13. Результат перетворення зображення 2.12

`glLoadIdentity()` **M ← I**
– set **M** to identity

`glTranslatef(`t_x`, `t_y`, `t_z`)` **M ← MT**
– translate by (t_x, t_y, t_z)

`glRotatef(θ, x, y, z)` **M ← MR**
– rotate by angle θ about axis (x, y, z)

`glScalef(`s_x`, `s_y`, `s_z`)` **M ← MS**
– scale by (s_x, s_y, s_z)

Рис. 2.14. Команди модифікації матриці перетворень OpenGL

Більш детально з особливостями використання системи координат OpenGL та роботи з моделями і їх візуалізації можна ознайомитися, наприклад, у [https://learnopengl.com/Getting-started/Coordinate-Systems].

2.1.2. Завдання для самостійного виконання

Основне завдання цієї роботи полягає у реалізації системи геометричних перетворень для моделей об'єктів та їх зображень, створених у попередній роботі. Згідно наданого матеріалу, потрібно спочатку визначити матрицю композиції перетворень, а потім застосувати її до усіх точок у моделі сцени. Після того, відтворити сцену на екрані. Завдання виконуються за індивідуальними варіантами, наведеними у табл. 2.3 та 2.4.

Таблиця 2.1

Склад геометричних перетворень

Варіант	**Послідовність та етапи перетворення**
0	T1R2S1
1	T2S2R2
2	T2R3S2
3	S1R2T3
4	S2T2R1
5	S2T1R3
6	R3T2S3
7	R1S3T1
8	R2S1T2
9	R3S2T1

Таблиця 2.2

Параметри операцій перетворення

Тип операції	**Позначення**	**Координати «центру»**		**Параметри перетворення**
		X	Y	
перенос	T1	0	0	$d_x = -50$; $d_y = 17$
перенос	T2	0	0	$d_x = 12$; $d_y = -3$
поворот	R1	13	-5	$\varphi = 33°$
поворот	R2	-3	9	$\varphi = -15°$
поворот	R3	-10	15	$\varphi = -74°$

Продовження табл. 2.2

масштабування	S1	0	1	$S_x = 0.1$; $S_y = 0.3$
масштабування	S2	-5	-5	$S_x = 1.7$; $S_y = 0.7$
масштабування	S3	5	0	$S_x = 0.6$; $S_y = 2.5$

2.1.3. Порядок виконання роботи

1) Запустити комп'ютер і завантажити ОС Windows.
2) Завантажити середовище розробки ПЗ.
3) Для програмного забезпечення, розробленого у попередній роботі, яке дозволяє малювати (задавати) геометричні об'єкти, реалізувати функції геометричного перетворення відповідно до заданого варіантом для об'єктів типу точка, лінія, прямокутник, ламана.
4) Відкомпілювати програму.
5) Оформити звіт, представити програму викладачеві і пояснити отримані результати.

2.1.4. Питання для самоперевірки

1) Вкажіть, які переваги з точки зору реалізації обчислювальних систем має використання для геометричних перетворень однорідної системи координат.
2) Вкажіть, яким чином здійснюється переклад координат в однорідну систему.
3) Чому, на Вашу думку, система координат зветься «однорідної».
4) Вкажіть, яким чином здійснюються операції масштабування і повороту щодо довільної точки.
5) Чи можна в якості альтернативи для операції перенесення точки використовувати операцію масштабування? Якщо так, визначте параметри такого перетворення.

6) Чи можна в якості альтернативи для операції перенесення відрізка використовувати операцію масштабування? Якщо так, визначте параметри такого перетворення.

7) Поясніть призначення елементів в матриці композиції перетворень.

Поясніть грає роль послідовність виконуваних дій на отримання кінцевого результату або процес отримання матриці композиції перетворень не є комутативним.

РОЗДІЛ 3. МАТЕМАТИЧНІ ОСНОВИ ФОРМУВАННЯ ЦИФРОВИХ ЗОБРАЖЕНЬ

У попередніх розділах розглянуто задачі, пов'язані з візуалізацією даних, які представлено просторовими моделями, або відображають стан елементів у комп'ютерній системі. Але існує низка практичних задач, пов'язаних із введенням та обробкою зображень об'єктів з реального оточуючого людину світу. Особливостями цих задач є те, що системи введення графічної інформації будувалися, чи намагалися будувати, на основі принципів роботи зору людини для чого проводилися відповідні дослідження. Як окремий результат цієї роботи, виникли, наприклад, такі два напрямки як інженерна психологія та когнітивна графіка.

Взагалі, задача введення зображень характерна не тільки для задач пов'язаних з людиною. Розвиток новітніх технологій у напрямку створення інтелектуальних автономних систем, спроможних орієнтуватися у просторі, задачі виробничого характеру які потребують систем «технічного зору», окремі компоненти систем безпеки – вимагає вирішення задач пов'язаних із введенням, обробкою та розпізнаванням образів. Але найперша задача яка з'являється – це введення зображення у комп’ютерну систему та формування «цифрового образу» об'єктів.

Оскільки перші задачі з обробки зображень були пов'язані з «допомогою» людині, відповідно системи розроблялися у відповідності до характеристик зору людини. Отже, потрібно розуміти які це характеристики перед тим, як використовувати їх у практичних задачах.

У загальному випадку можна говорити про те, що обробка зображень може бути віднесена до однієї з наступних категорій:

– квантування і ефективне кодування зображень, тобто перетворення безперервного зображення в цифрову форму стосовно завдань її зберігання і передачі по цифрових каналах зв'язку.

– Поліпшення зображень - комплекс перетворень з метою поліпшення візуального сприйняття або отримання форми більш зручною для візуального або машинного аналізу. У тому випадку, якщо поліпшення проводиться для усунення раніше внесених спотворень, кажуть про реставрацію зображень.
– Розпізнавання образів. Під цим терміном мається на увазі як саме виявлення образів, так і виділення ознак, за якими може бути вироблено розпізнавання об'єкта.

Виділяють наступні основні етапи вирішення завдань обробки зображень (рис. 3.1):

Рис. 3.1. Етапи вирішення завдань обробки зображень

- введення та формування цифрового представлення зображень. Цей етап включає введення зображення в ЕОМ і його первинне перетворення для зберігання або подальшої обробки;

- попередня обробка зображення. Включає в себе перетворення, що не торкаються форму і структуру опису вихідного зображення;

- формування графічного препарату зображення - опис найбільш істотних деталей вихідного зображення. При вирішенні багатьох завдань значну частину інформації про об'єкт містить його контур. Тому перехід до бінарного контурного зображення об'єкта дозволяє зберегти більшу частину інформації про об'єкт;

- виділення об'єктів і їх класифікація. Зазвичай це заключний етап аналізу зображень, що безпосередньо примикає до завдань штучного інтелекту.

З безлічі областей, пов'язаних з завданнями обробки зображень, відзначимо наступні:

- обробка аерокосмічних знімків;
- біомедицина;
- обробка даних мікроскопічних досліджень;
- промисловий контроль і робототехніка;
- наукові дослідження.

Кожна із зазначених областей в свою чергу включає безліч застосувань. Наприклад, обробка аерокосмічних знімків ведеться стосовно завдань сільського господарства, картографії, охорони навколишнього середовища, контролю поновлюваних і невідновлюваних природних ресурсів, метеорології, військової розвідки і т.п.

Розглянемо більш докладно змісту кожного етапу.

Квантування і введення зображення. Як правило, будь-яке природне зображення є безперервним. Для обробки в ЕОМ необхідно дискретне зображення, тому при введенні зображення вирішується завдання його квантування і формування. Основні технічні засоби, що використовуються для формування зображень можна розділити на наступні категорії:

- оптичні та теплові;

- з використанням проникаючого випромінювання;
- радіолокаційні;
- комбіновані (наприклад, рентгенографія і машинна графіка).

Попередня обробка зображень. Графічний процесор супроводжується його спотвореннями за рахунок недосконалості технічних засобів, наявності шумів і перешкод. Попередня обробка служить для підвищення якості зображення перед подальшим аналізом. Зазвичай в неї включаються такі операції:

- корекція яскравості і контрастності по всьому полю зображення;
- колірні перетворення;
- придушення шумів;
- перетворення зображення (підкреслення перепадів).

Препарація зображень. У великій кількості випадків можна говорити про те, що основна інформація про об'єкт міститься в його контурі, остові (кістяку), фактурі поверхні, зв'язності областей, контурів та ін. Завданням етапу препарації і є виділення цих основних ознак. Відзначимо наступні основні операції цього етапу:

- виділення перепадів;
- бінарізація зображення;
- операції стиснення і розширення;
- виділення скелета (кістяка) зображення;
- виділення контуру;
- виділення зв'язкових областей;
- сегментація зображення.

Аналіз зображень. Завдання цього етапу - ідентифікація та класифікація зображень, тобто, розпізнавання образів.

Суттєвою особливістю завдань опрацювання зображень на відміну від завдань передачі зображень є те, що у процесі обробки зображення виступають з теоретико – інформаційної точки зору не як повідомлення, а як сигнали. Повідомленнями ж на зображенні є випадкові параметри окремих деталей зображення або його загальної структури, визначення яких і є,

власне, кінцевою метою опрацювання. Конкретного сенс цих параметрів, а також критерії точності їх оцінки визначаються одержувачем.

Можна виділити три види одержувачів повідомлень, які полягають в зображенні:

- колективний, наприклад, в телебаченні, кінематографі, поліграфії і т.д;
- індивідуальний - людина-оператор;
- автомат.

Колективний одержувач характеризується усередненими властивостями зору. Суттєвою особливістю індивідуального одержувача є неформальний характер критерію обробки. У разі одержувача-автомата, зазвичай, повинен існувати точний критерій обробки.

За відсутності формального критерію якості обробки для індивідуального одержувача випливає необхідність діалогового режиму обробки під управлінням користувача.

Зображення в задачах опрацювання слід розглядати як елементи статистичного ансамблю. Вміщуваний сенс випадкових факторів породжує даний ансамбль, визначається тим, якого роду інформацію для одержувача містить зображення.

В загальному випадку, можна записати ряд рівнянь, досить загальних для аналізу багатьох практичних систем візуалізації та опрацювання, що володіють певною повнотою, і що робить їх засобом для вивчення принципів, що лежать в основі методів візуалізації. По суті, ці рівняння створюють основу для розуміння таких питань, як причини і характер зниження якості зображення, а також для створення методів обробки зображень .

Тема 3.1. Принципи дискретизації цифрових зображень

3.1.1. Поняття об'єкту та його зображення

Будемо розуміти під терміном «об'єкт» "деякий багатовимірний розподіл, який підлягає вимірюванню, а під терміном «зображення» – виміряне розподілення, яке було отримано і яке вважається найкращим поданням розподілу, створюваного об'єктом. Загально прийнято, що об'єкт позначається через *f*, а зображення через *g*. Зі сказаного випливає, що ідеально відтворююча система - це така система, для якої в будь-якій точці простору виконується рівність $f = g$. У загальному випадку розподіли *f* і *g* є тривимірними, однак, в більшості випадків можна обмежитися розглядом двовимірних розподілів.

Розглянемо найпростішу оптичну систему отримання фотографій (в площині зображення) картини, відтворену на двомірному екрані (в площині об'єкту). Якщо ввести відповідні системи координат, то двовимірний об'єкт запишеться як $f(a, b)$, а фотографія як $g(x, y)$. У цьому прикладі передбачається, що розподіли *f* і *g* мають одну і ту ж розмірність, оскільки вони є просторовим розподілом інтенсивності світла або кольору в площині.

Фотографія формується квантами світла, відбитого від картини, що пройшли через оптичну систему фото-системи і потрапили на фотосенсор. Зрозуміло, що таке формування зображення призводить до втрати його якості за рахунок спотворень і недосконалостей фото-системи, а отже *f* і *g* не рівні один одному.

У більш складних випадках (наприклад, томографія) зображення *g* є якесь уявленням (описом) об'єкта *f*, яке має відмінні від об'єкта розміри, але розташовується в тому ж місці.

Розумно вважати, що об'єкт і зображення фізично збігаються і пов'язані один з одним співвідношеннями, що

характеризують конкретний метод візуалізації, хоча в ряді випадків можуть мати відмінні розміри.

3.1.2. Співвідношення, що зв'язують об'єкт і зображення

Будемо позначати розподіл об'єкту, що цілком знаходиться в об'єктній площині, через *f(a, b)*, а розподіл зображення, також повністю займає площину зображення, - через *g(x, y)*.

У загальному випадку не існує ідеальної відповідності між інформацією, що міститься в будь-якій точці з координатами *(a', b')* і інформацією, що відповідає точці *(x', y')*, так як інформацію від кожної точки об'єкта можна розсіяти по всіх точках зображення. Однак в будь-якому корисному методі візуалізації головний внесок в кожну точку *(a', b')* буде вкладати окрема конкретна точка *(x', y')*. Інші сусідні точки будуть вносити меншу кількість інформації, причому зменшення зазначеного вкладу відбувається досить різко в міру віддалення основної точки з координатами *(x', y')*. Ці висновки відомі як принцип "близькості".

Який зв'язок існує між просторами об'єкта і зображення? Концептуально в цьому випадку має місце обмін інформацією. У площину зображення потрапляє інформація виходячи з наявності інформації в площині об'єкту, а також в залежності від того, який "кодований носій" інформації використовується в даному методі візуалізації. Так фотографія формується за рахунок перенесення фотонів.

Розподіл світлової енергії може бути представлено у вигляді функції $C(x,y,t,\lambda)$ - що вказує її залежність від просторових координат x, y, часу t і довжини хвилі λ.

Оскільки енергія випромінювання пропорційна квадрату амплітуди електричного поля, то є дійсною позитивною величиною. Так як в реальних відеосистемах максимальне значення яскравості обмежено деяким значенням випромінюючих (люмінофор) або пропускних (фотоплівка) пристроїв, то величина $C(x,y,t,\lambda)$ обмежена:

$$0 \le C(x,y,t,\lambda) \le A, \qquad (3.1)$$

де А - максимальна яскравість зображення.

З метою спрощення, будемо вважати, що зображення відмінні від "0" в деякій прямокутній області:

$$-Lx \le x \le Lx \qquad (3.2)$$
$$-Ly \le y \le Ly \qquad (3.3)$$

Зображення спостерігається протягом певного проміжку часу:

$$-T \le t \le T \qquad (3.4)$$

Таким чином, величина $C(x,y,t,\lambda)$ є обмеженою безперервною функцією чотирьох обмежених змінних.

Відчуття світлини, що виникає в зоровій системі, визначається миттєвої яскравістю світлового потоку:

$$Y(x,y,t)=\int_0^\infty C(x,y,t,\lambda)\,V_s(\lambda)\,d\lambda \qquad (3.5)$$

де $V_s(\lambda)$ - спектральна чутливість людського зору.

Колірні відчуття можна описати набором координат кольору, пропорційних інтенсивностям *R, G, B*, суміш яких дає заданий колір. Для такої системи координати визначають:

$$R(x,y,t)=\int_0^\infty C(x,y,t,\lambda)\,R_s(\lambda)\,d\lambda$$
$$G(x,y,t)=\int_0^\infty C(x,y,t,\lambda)\,G_s(\lambda)\,d\lambda$$
$$B(x,y,t)=\int_0^\infty C(x,y,t,\lambda)\,B_s(\lambda)\,d\lambda \qquad (3.6)$$

де $R_s(\lambda)$, $G_s(\lambda)$, $B_s(\lambda)$ - питомі координатам для основних кольорів, що дорівнюють координатами кольору монохроматичного випромінювання одиничної інтенсивності.

Для спектрального (монохромного) зображення справедливо співвідношення:

$$F_i(x, y, t) = \int_0^{\infty} C(x, y, t, \lambda) \square S_i(\lambda) d\lambda \qquad (3.7)$$

де $S_i(\lambda)$ - спектральна чутливість і-го датчика.

Для системи відтворення $F_i(x, y, t)$ є однією з координат кольору. У багатьох системах параметр t може бути опущений.

Набір вихідних двовимірних функцій $F_1(x,y)$, $F_2(x,y)$ ... $F_N(x,y)$ відображається в набір двовимірних функцій $G_1(x,y)$, $G_2(x,y)$ ... $G_M(x,y)$. Це відображення може бути задано набором операторів, який визначемо як $\vartheta_m\{\cdot\}$, при m= 1,2 ... M:

$$\begin{aligned} G_1(x, y) &= \vartheta_1\{F_1(x, y), F_2(x,y) \dots F_N(x,y)\} \\ G2(x, y) &= \vartheta 2\{F1(x,y), F2(x,y) \dots FN(x,y)\} \\ GM(x,y) &= \vartheta M\{F1(x,y), F2(x,y) \dots FN(x,y)\} \end{aligned} \qquad (3.8)$$

Число вхідних функцій N може бути більше, менше або дорівне числу вхідних функцій при N = M= 1:

$$G(x,y) = \vartheta\{F(x,y)\} \qquad (3.9)$$

Існують спеціальні оператори відображення, що застосовуються при аналізі двовимірних систем. У комп'ютерних системах обробки зображень це *дельта-функція Дірака*.

Основні властивості, які використовують у практичних цілях, наведені у наступних рівняннях:

$$\int \int_{-\varepsilon}^{\varepsilon} \delta(x,y)\,\mathrm{dxdy} = 1, \text{ при } \varepsilon > 0 \tag{3.10}$$

$$\int \int_{-\infty}^{\infty} F(\xi,\eta) \square \delta(x-\xi, y-\eta)\,d\xi d\eta = F(x,y) \tag{3.11}$$

.

Використовуючи властивість дельта-функції, функцію *F(x,y)* на виході системи обробки можна представити у вигляді:

$$F(x,y) = \int \int_{-\infty}^{\infty} F(\xi,\eta) \square \delta(x-\xi, y-\eta)\,d\xi d\eta \tag{3.12}$$

де $F(\xi,\eta)$ - ваговий множник дельта-імпульсу (рис. 3.2), що має координати ξ,η на площині *(x, y)*.

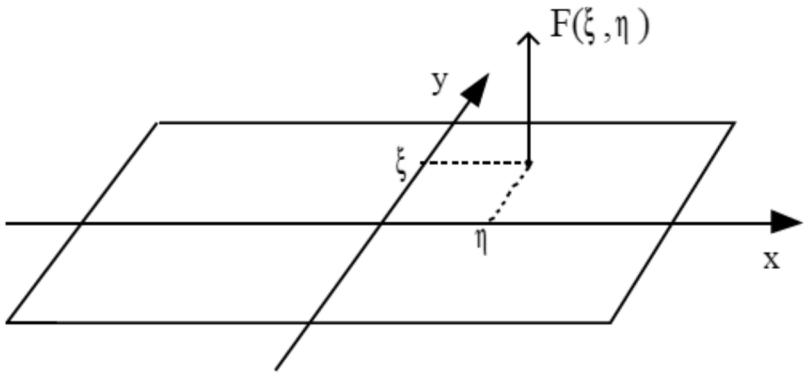


Рис. 3.2. Геометричне представлення обробки дельта-функцією Дірака

Тобто, обробка якогось розподілу оператором, який формує «відгук» за дельта-функцією Дірака, повертає одиночний імпульс у точці із заданими координатами. У технічній системі це можна трактувати як сигнал від фотоелементу, розташованого у відповідній точці фотосенсору.

Оскільки $G(x,y) = \vartheta\{F(x,y)\}$, отримаємо:

$$G(x,y)=\int_{-\infty}^{\infty} F(\xi,\eta)\,\vartheta\{\delta(x-\xi,y-\eta)\}d\xi d\eta \qquad (3.13)$$

Запишемо множник як $\vartheta\{\delta(x-\xi,y-\eta)\} = H(x,y,\xi,\eta)$ - лінійний оператор, який діє на той підінтегральний множник, який залежить від просторових змінних х, у. Функцію *H(x,y,ξ,η)* називають імпульсним відгуком оптичної системи.

Тоді, попередній вираз може бути представлено у вигляді:

$$G(x,y)=\int_{-\infty}^{\infty} F(\xi,\eta)\,H(x,y,\xi,\eta)d\xi d\eta \qquad (3.14)$$

Лінійна система називається просторово-інваріантною (незалежною, ізопланарною), якщо її імпульсний відгук залежить тільки від різниці координат *x- ξ, y- η*.

Для просторово-інваріантної системи справедливо:

$$H(x, y, \xi, \eta) = H(x-\xi, y-\eta), \qquad (3.15)$$

а інтеграл суперпозиції прийме форму:

$$G(x,y)=\int_{-\infty}^{\infty} F(\xi,\eta)\,H(x-\xi,y-\eta)d\xi d\eta \qquad (3.16)$$

,

звану інтегралом згортки. Операція згортки символічно записується у вигляді:

$$G(x, y) = F(x, y) * H(x, y) \qquad (3.17)$$

Таким чином, функція *G(x,y)* може бути описана як сканування вхідних функцій «ковзним вікном» - зверненим імпульсним відгуком і інтеграцією по області, в якій ці функції перекриваються.

Даний приклад показує, яким чином здійснюється дискретизація безперервного зображення.

3.1.3. Квантування зображень

Будь-яка аналогова величина, що підлягає обробці в ЕОМ або цифровий системі, повинна бути представлена у вигляді цілого числа, пропорційного значенням цієї величини. Процес такого перетворення називається квантуванням.

Позначимо через f і $\hat{f}$ відповідно відлік дійсного скалярного сигналу до і після квантування. Передбачається, що f – випадкова величина з щільністю ймовірності $p(f)$, яка лежить в межах:

$$a_L \le f \le a_U \qquad (3.18)$$

де a_L та a_U –відповідно нижня і верхня границі інтервалу.

В процесі квантування, вихідна величина порівнюється з набором порогових рівнів, і якщо відлік потрапляє в якийсь діапазон між двома граничними рівнями, то йому відповідно ставиться фіксований рівень квантування:

$$\text{якщо } d_j \le f \le d_{j+1}\text{ , тоді } \hat{f} = r_j \qquad (3.19)$$

Рівні квантування і порогові рівні вибирають так, щоб зменшити до мінімуму деяку завдану величину, що характеризує помилку квантування (тобто ступінь відмінності між f і $\hat{f}$).

Зазвичай в якості такого заходу вибирають середньоквадратичну помилку:

$$\varepsilon = E\left\{\left(f-\hat{f}\right)^2\right\} = \int_{a_L}^{a_U}\left(f-\hat{f}\right)^2 p(f)df = \sum_{j=0}^{J-1}\int_{d_j}^{d_{j+1}}\left(f-r_j\right)^2 p(f)df \qquad (3.20)$$

Якщо J велике, то щільність ймовірностей значень сигналу, що квантуємо, на кожному інтервалі $(dj, dj + 1)$ можна вважати постійною і рівною $p(r_j)$, звідки:

$$\varepsilon = \sum_{j=0}^{J-1} p(r_j) \int_{d_j}^{d_{j+1}} \left(f - r_j\right)^2 df \,. \qquad (3.21)$$

Після обчислення інтегралів отримаємо:

$$\varepsilon = \frac{1}{3} \sum_{j=0}^{J-1} p(r_j) \left[(d_{j+1} - r_j)^3 - (d_j - r_j)^3 \right] \qquad (3.22)$$

Оптимальне рішення можна знайти, вирішивши задачу про мінімум похибки ε як функції від r_j:

$$\frac{d\varepsilon}{dr} = 0 \,, => r_j = \frac{d_{j+1} + d_j}{2} / \qquad (3.23)$$

Таким чином, при вказаних припущеннях оптимальним положенням рівня квантування є середина інтервалу між сусідніми граничними рівнями (рис. 3.3).

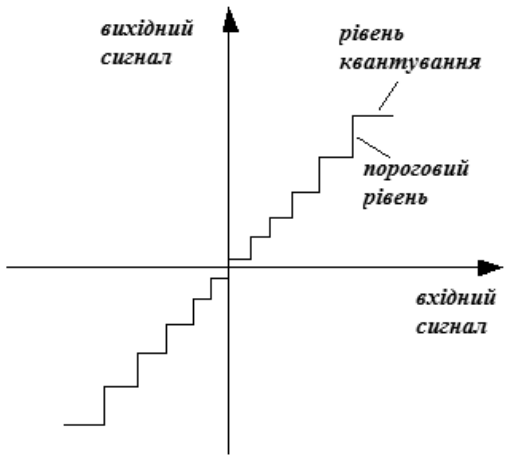


Рис. 3.3. Відповідність вхідних та квантованих сигналів у загальному виді

3.1.4. Вибір порогових рівнів та рівнів квантування

Оптимальне положення порогових рівнів можна визначити, знаходячи мінімум помилки ε методом Лагранжа. За допомогою цього методу Пантер і Дайте показали, що положення порогових рівнів точно визначається за формулою:

$$d_j = \frac{(a_U - a_L)\int_{a_L}^{a_j}[p(f)]^{-\frac{1}{3}}df}{\int_{a_L}^{a_U}[p(f)]^{-\frac{1}{3}}df} + a_L, \qquad (3.24)$$

Де $$a_j = \frac{j\cdot(a_U - a_L)}{J} + a_L, \quad j = 0,1,2,..J$$

$$a_j = \frac{j\cdot(a_U - a_L)}{J} + a_L, \quad j = 0,1,2,..J$$

Якщо щільність ймовірностей значень відліків рівномірна, то порогові рівні будуть розставлені рівномірно. При нерівномірній щільності порогові рівні частіше на тих ділянках, де щільність ймовірності вище і рідше там, де вона мала.

Вчений Макс вирішив задачу визначення рівнів квантування для Гаусової щільності і склав таблиці оптимального розміщення порогових рівнів в залежності від числа рівнів квантування.

Для окремого випадку рівномірного розподілу ймовірностей СКО дорівнює:

$$\varepsilon_{min} = \frac{1}{12\cdot J^2}. \qquad (3.25)$$

Нерівномірне квантування можна привести до рівномірного за допомогою нелінійного перетворення:

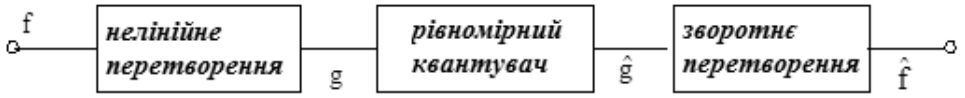


Рис. 3.4. Квантування зі стисненням

У таких системах прагнуть зробити щільність ймовірності перетворених відліків g на вході квантувача.

Перетворений відлік є:

$$g = T\{f\}, \tag{3.26}$$

Де нелінійне перетворення *T{f}* вибрано таким, що щільність ймовірності *g* виявляється рівномірною, тобто:

$$p(g)=1. \tag{3.27}$$

В інтервалі $-1/2 \le g \le 1\backslash 2$. Таким чином, якщо *f* – випадкова величина с нульовим середнім, то шукана характеристика нелінійного елементу має наступний вид:

$$T\{f\} = \int_{\infty}^{f} p(z)dz - \frac{1}{2} \tag{3.28}$$

У наступній таблиці 3.5 наведено характеристики прямого і зворотного перетворень для щільності розподілу Гаусса, Релея, Лапласа.

З геометричної точки зору трактовка цього процесу виглядає досить просто – на графіку функції ймовірності вхідного сигналу потрібно розділити горизонтальними рівномірно розташованими лініями. З точок перетину з графіком опустити вертикалі до вісі вхідних аргументів – таким чином отримаємо порогові значення (рис. 3.5).

При квантуванні вектору $\vec{f}$ N-мірний векторний простір поділяють на *J* осередків квантування D_j, кожному з яких відповідає один з *J* квантованих векторів. Векторний сигнал

замінюється на квантований сигнал, якщо потрапляє в осередок D_j.

Така постановка завдання означає, що векторний сигнал $\vec{f}$ перетвориться в вектор r_j, але компоненти вектора f при цьому не обов'язково будуть квантованими окремо по набору дискретних порогових рівнів.

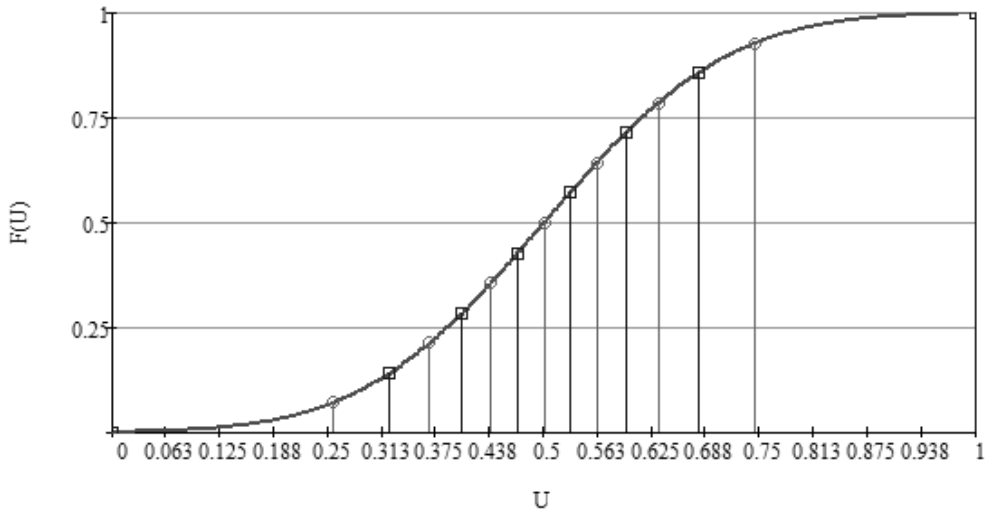


Рис. 3.5. Геометричне рішення задачі вибору порогів та рівнів квантування

Таблиця 3.1

Співвідношення перетворень

Щільність	Пряме перетворення	Зворотне перетворення
Гауса $p(f)=\sqrt{2\pi\sigma^2}*\exp\left\{\frac{-f^2}{2\sigma^2}\right\}$	$g=\frac{1}{2}*\mathrm{erf}\left\{\frac{f}{\sqrt{2\sigma}}\right\}$	$\hat{f}=\sqrt{2\sigma}*\mathrm{erf}^{-1}\{2\hat{g}\}$
Релея $p(f)=\frac{f}{\sigma^2}*\exp\left\{\frac{-f^2}{2\sigma^2}\right\}$	$g=\frac{1}{2}-\exp\left\{-\frac{f^2}{2\sigma^2}\right\}$	$\hat{f}=\left[\sqrt{2\sigma^2}*\ln\left(\frac{1}{1/2-\hat{g}}\right)\right]^{\frac{1}{2}}$
Лапласа $p(f)=\frac{\alpha}{2}*\exp\left\{\frac{-\alpha}{\lvert f\rvert}\right\},\ \alpha=\frac{\sqrt{2}}{\sigma}$	$g=\begin{cases}\frac{1}{2}(1-\exp\{-\alpha f\}), & f\ge 0\\ -\frac{1}{2}(1-\exp\{\alpha f\}), & f<0\end{cases}$	$\hat{f}=\begin{cases}-\frac{1}{\alpha}*\ln(1-2\hat{g}), & g\ge 0\\ \frac{1}{\alpha}*\ln(1+2\hat{g}), & g<0\end{cases}$

Оптимальне положення квантованих векторів при фіксованих осередках квантування *Dj* неможливо визначити, не знаючи спільної щільності ймовірності $p(\vec{f})$. Однак, часто така інформація відсутня. Тому процедуру векторного квантування спрощують, квантуючи всі компоненти вектору окремо.

Тоді, наприклад, в тривимірному просторі осередки *Dj* набувають форму паралелепіпедів. При формуванні набору осередків квантування *Dj* мінімізують значення середньоквадратичної помилки квантування.

Був запропонований метод, коли для кожної компоненти вектору задається фіксоване число рівнів квантування *J(i)*, де *i = 1, 2, ... N*, і всі компоненти квантуються незалежно, при цьому:

$$\prod_{j=1}^{N} J(i) = J \qquad (3.29)$$

де J – фіксоване число рівнів квантування для кожного вектору.

У цифрових системах зазвичай вибирають число рівнів квантування кратним ступеню двійки:

$$J(i)=2b(i) \qquad (3.30)$$

де *b(i)* - ціле число кодових розрядів для і-ої компоненти вектору. Загальна кількість кодових розрядів для вектору визначається як:

$$B = \sum_{i=1}^{N} b(i) \qquad (3.31)$$

Для квантування величини з Гаусовим розподілом, Реді та Уімц (1968 г.) запропонували наступний алгоритм:

1) обчислити *b(i)* за формулою:

$b(i) = \frac{B}{N} + 2\lg\left[\sigma^2(i)\right] - \frac{2}{N}\sum_{j=1}^{N}\lg\left[\sigma^2(j)\right]$, де $\sigma^2(i)$ – дисперсія і-го відліку;

2) округлити *b(i)* до найближчого цілого;

3) змінювати отриманий розподіл, доки не буде виконано умову (3.31).

Прикладом такого розподілення є один з форматів представлення кольору з 16 бітною сіткою. У цьому форматі виділяють по 5 біт на червону та синю компоненти, та 6 біт на зелену. Такий розподіл пов'язаний з тим, що зелена компонента сприймається більш ретельно. Про це буде інформація далі.

Як відомо, частота дискретизації за принципом імпульсно-кодової модуляції (ІКМ) вибирається виходячи з теореми Котельникова:

$$f_{дискр.} \le 2F_{max}, \qquad (3.32)$$

де F_{max} - максимальна частота відеосигналу при заданих параметрах сканування.

При неправильному виборі $f_{діскр}$. можуть виникнути спотворення дискретизації через неідеальні частотні і фазові характеристики обмежуючого фільтру. Ці спотворення проявляються як хитання зображення, дроблення дрібних деталей і контурів малюнка, зниження чіткості і різкості зображення. Найбільш ефективний спосіб боротьби з цими спотвореннями – збільшення частоти дискретизації. Однак, збільшення частоти дискретизації призводить до зниження швидкості передачі інформації.

В результаті квантування копія сигналу набуває скачки, внаслідок чого спектр цього сигналу розширюється і в зображення будуть проникати складові всього спектра квантованого сигналу.

Потужність шумів квантування визначається виразом:

$$P_{кв} = \sum_i p_i \frac{\Delta_i^2}{12} \qquad (3.33)$$

де p_i – ймовірність потрапляння сигналу в і-ю зону квантування,

Δ_i – шаг квантування.

Для лінійної шкали квантування потужність шумів обчислюється як:

$$P_{кв} = \frac{\sigma^2}{12} \qquad (3.34)$$

а відношення потужності корисного сигналу і шумів квантування в Дб визначається як:

$$\frac{P_c}{P_{кв}} = 10 \cdot \lg 12 \frac{U^2_{огр}}{\sigma^2} = 10 \cdot \lg \frac{12 \cdot U^2_{огр}}{\left(\frac{U_{огр}}{2^{n_p}}\right)^2} \approx 6 \cdot n_p + 10{,}8 \qquad (3.35)$$

де $U_{огр}$ – амплітуда динамічного діапазону квантованого сигналу;

n_p – розрядність коду для кодування сигналу.

Експериментально було показано, що співвідношення сигнал / шум має бути не менше 40дБ, а $n_p \geq 5 \div 6$.

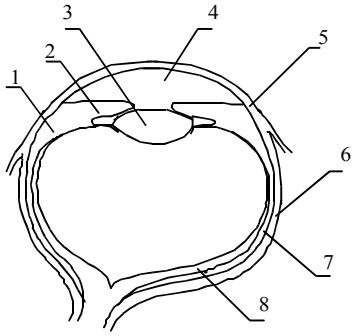
Збільшення перешкодозахищенності сигналів ІКМ може бути досягнуто при трансформаційних змінах нелінійної шкали квантування, що зменшує також обсяг переданої інформації.

Наприклад, в фототелеграфній апаратурі з поліграфічним (субтрактивним) синтезом компресія здійснюється за допомогою логарифмічних підсилювачів, що вирішує ще й завдання переходу до зональних оптичних щільностей, необхідних для проведення електронної корекції.

Такий метод дає виграш в завадостійкості 20-30Дб.

Тема 3.2. Фізіологічні та алгоритмічні основи представлення інформації про колір

Зоровий аналізатор (ЗА) призначений для прийому і аналізу інформації в світловому діапазоні 400-700нм. Око є периферичною частиною органу зору (рис. 3.5). За допомогою нервових шляхів воно пов'язане з мозковим центром.



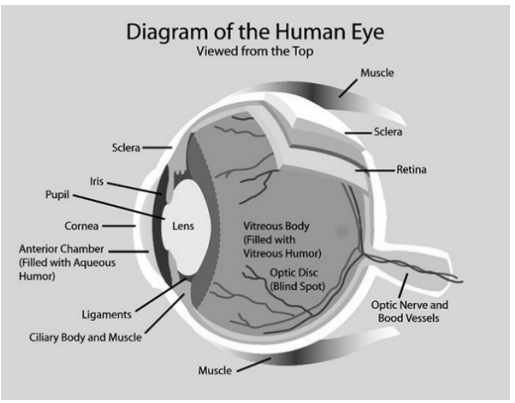


Рис. 3.6. Будова ока

Око – це кулясте тіло, масою 7,5 г. Стінки ока утворені трьома оболонками. Зовнішня фіброобразна оболонка (склера) (6), найміцніша. Вона не прозора, в її передній частині є маленьке віконце з діаметром 12мм – рогівка (5).

Зсередини до склери прилягає судинна оболонка (7), позаду рогівки знаходиться райдужка (2), вона містить клітини з пігментом-барвником, кількість якого визначає колір очей. У центрі райдужної оболонки є зіниця (4), яка звужуючись або розширюючись змінює інтенсивність проходження світлового потоку (енергії).

Позаду райдужки на тонких нитках війкового тіла підвішений кришталик (3) – двояко опукла лінза діаметром 10 мм. Фокусування зору досягається зміною форми кришталика за допомогою війкового м'яза (1).

На внутрішній оболонці очного яблука знаходиться шар фоторецептора (8). Він містить колбочки і палички. Палички відповідають за сприйняття світла, а колбочки за сприйняття кольору. Паличок на порядок більше, ніж колб. Палички і колбочки розподілені нерівномірно (рис. 3.7).

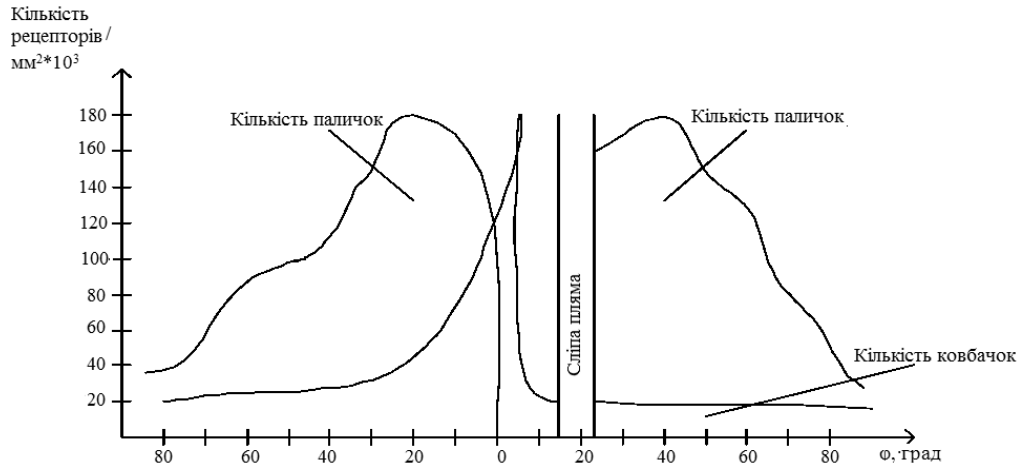


Рис. 3.7. Графіки просторової залежності розподілу фоторецепторів ока

У місці виходу з ока зорового нерву фоторецептори відсутні – цю пляму називають «сліпою». ЇЇ розміри 5,5 градусів по горизонталі і 7,5 по вертикалі.

Форма і колір сприймаються тільки при яскравості не менше 10 кд / м2. При яскравості 0,003кд / м2 функціонують тільки палички, які забезпечують «сутінковий» зір.

Виділяють наступні групи характеристик ока та зору:

- енергетичні характеристики – діапазон сприймання яскравості, контраст, сліпуча яскравість, відносне кольоро-відчуття;
- інформаційні характеристики – пропускна здатність;
- просторові характеристики – гострота зору, поле зору, обсяг сприйняття;
- тимчасові характеристики – латентний період реакції, час адаптації, тривалість інерції відчуття, критична частота миготіння, тривалість інформаційного пошуку.

Основна характеристика зорового аналізатора – чутливість. Її діапазон лежить в межах від 10^{-7} до 10^{5}кд / м2. Нижня межа визначається мінімальною інтенсивністю світлового потоку, що викликає відчуття. Цю величину прийнято називати порогом світловий чутливості.

Абсолютний поріг світлової чутливості зорового аналізатора характеризує найбільш високу чутливість, що досягається в ході темпової адаптації протягом декількох годин (до 3-4 годин).

Абсолютна чутливість досить висока і відповідає світловому потоку в кілька квантів. При практичних розрахунках рекомендується брати величину 5,2 * 10^{-6} кд / м2.

Для оцінки яскравості об'єктів використовують поняття адаптивної яскравості. Вона визначається як середньозважене значення яскравості, що потрапляють в поле зору. Найбільш сприятливі умови роботи оператора від декількох десятків до декількох сотень кд/м^2. Суб'єктивна оцінка яскравості залежить від яскравості об'єкту.

Розрізняють прямий контраст (темний об'єкт на світлому фоні) і зворотний:

$$Кпр = (Вф\text{-}ВОБ)/Вф;$$
$$Коб = (В_{ОБ}\text{-}В_{Ф})/В_{ОБ}, \qquad (3.36)$$

де $В_Ф$ – яскравість фону;
$В_{об}$ – яскравість об'єкту.

Для умов нормальної роботи значення контрасту повинні бути в діапазоні від 6,5 до 0,95.

Абсолютно сліпуча яскравість відповідає 225000 кд/м^2. Ефект засліплення може бути досягнутий, якщо в полі зору знаходяться сигнали різної інтенсивності.

Розглянуті характеристики відносяться до роботи палочкового апарату, що забезпечує сприйняття ахроматичного світла.

Поріг кольорової чутливості дуже малий. У жовтому і блакитному кольорах довжина хвилі, необхідна для відмінності кольоровості, може досягати 1 нм.

Мінімально помітна різниця довжин хвиль залежить від яскравості і кутових розмірів об'єктів. Розрізнення погіршується при зменшенні розмірів. При розмірах об'єктів менше 10 просторових хвилин колір випромінювання перестає помічатися оком.

При середніх розмірах об'єктів і яскравості вище 10 кд/м^2 число помітних об'єктів досягає декількох сотень. При великих розмірах об'єктів, розташованих поруч, око здатне розрізняти до 10^7 світлових відтінків.

Збільшення або зменшення яскравості знижує чутливість до колірних тонів.

Важливою характеристикою ока є чутливість до різних ділянок світлового спектру:

$$K_\lambda = S_\lambda S_{550} \quad . \tag{3.37}$$

де S_{550} – відчуття від джерела випромінювання хвилі довжиною *550нм*;

S_λ – відчуття, від джерела тієї ж потужності при довжині хвилі λ.

Крива відносної чутливості до різних ділянок світлового спектру має вид, наведений на рис. 3.8.

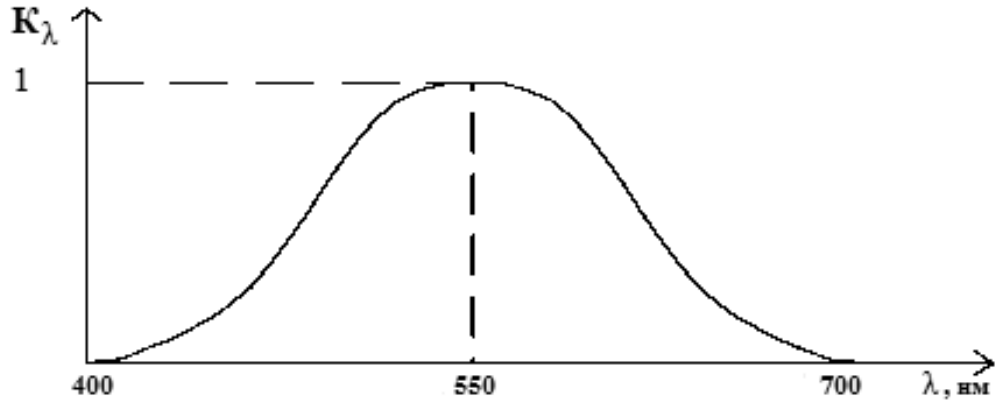


Рис. 3.8. Крива відносної чутливості

Існують три основні характеристики відчуття світла – світлість, колірний тон і насиченість.

Ознака, за якою відрізняється, наприклад, червоний колір від зеленого, називається колірним тоном. Однак, довжина світлової хвилі не є адекватною характеристикою кольору, оскільки в природі зустрічаються такі кольори, які відсутні в веселці, наприклад, пурпурний (вузько смуговий червоний + вузько смуговий синій).

Якщо два джерела світла з однаковими спектральними розподілами спостерігати в однакових умовах, їх колірний тон буде однаковим. Однак, можна взяти два таких джерела світла з різними спектральними розподілами, які будуть сприйматися як такі, що з однаковим колірним тоном. Вони називаються метамеричною парою.

Насиченість, по суті, описує «білизну» кольору і дозволяє відрізняти спектральний колір від пастельного бляклого кольору такого ж кольорового тону.

Для класифікації кольорів зручно їх розглядати як точки деякого кольорового простору (рис. 3.9).

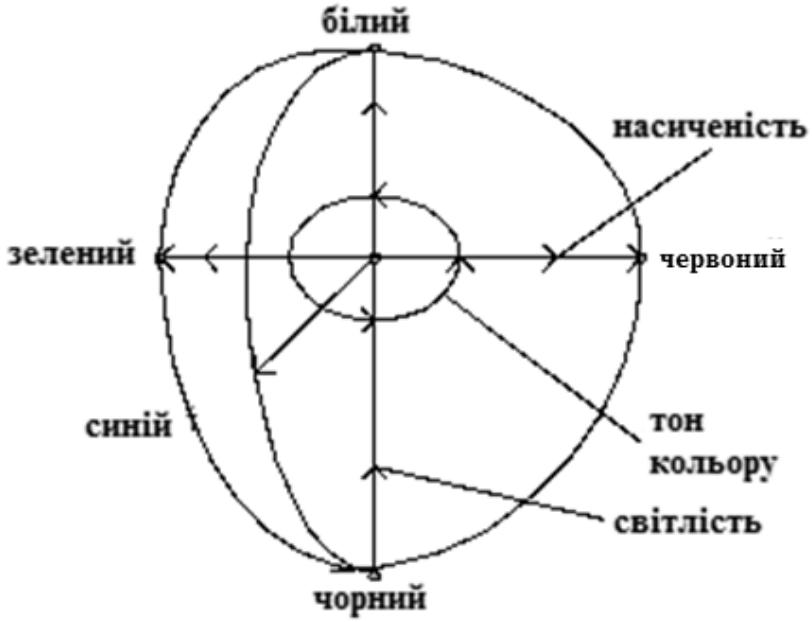


Рис. 3.9. Модель системи координат кольорів людини

3.2.1. Моделі кольорового зору

Теорії, що пояснюють кольоровий зір, розроблялися ще з часів Ньютона. Класична модель трикольорового зору, розроблена Юнгом в 1802р., передбачає, що око має три види елементів чутливих в різних областях спектру. Підтвердження цієї теорії було отримано тільки в 1960 р.

Одна з моделей, запропонована Фреєм, має наступну структуру – рецептори зі спектральної чутливості $S_1(\lambda)$, $S_2(\lambda)$,$S_3(\lambda)$ створюють сигнали:

$$\begin{aligned} e_1 &= \int C(\lambda) \cdot S_1(\lambda)\, d\lambda; \\ e_2 &= \int C(\lambda) \cdot S_2(\lambda)\, d\lambda; \\ e_3 &= \int C(\lambda) \cdot S_3(\lambda)\, d\lambda; \end{aligned} \qquad (3.38)$$

де $C(\lambda)$ - спектральна щільність енергії світла.

Сигнали e_1, e_2, e_3 (рис. 3.9) піддаються потім логарифмуванню і об'єднуються в такий спосіб:

$$\begin{aligned} d_1 &= \log e_1; \\ d_2 &= \log e_2 - \log e_1 = \log(e_2 e_1); \\ d_3 &= \log e_3 - \log e_1 = \log(e_3 e_1). \end{aligned} \qquad (3.39)$$

Ці сигнали проходять через лінійні фільтри $H1 \div H3$ і отримані на виході сигнали g_1, g_2 , g_3 визначають сприйняття кольорів мозком.

У такій моделі d2 і d3 характеризують колір, а d1 пропорційний яскравості (рис. 3.10).

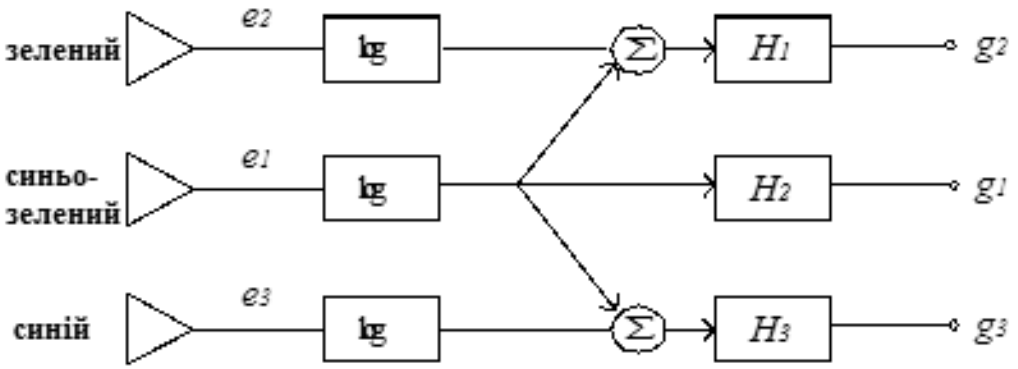


Рис. 3.10. Структурна схема перетворення кольорової інформації

Виявилося, що ця модель дозволяє дуже точно передбачити багато явищ кольорового зору.

3.2.2. Рівняння кольорів

В основі триколірної теорії кольорового зору лежить можливість подання довільного кольору складанням в потрібній пропорції трьох основних кольорів.

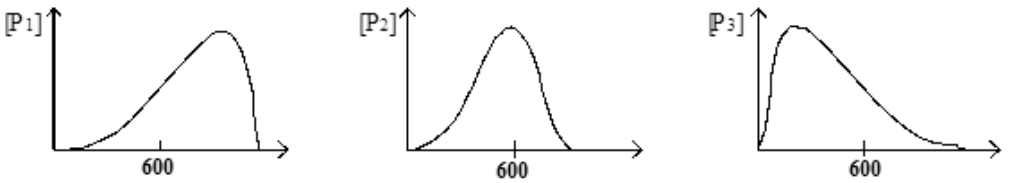
Виділяють адитивне і субтрактивне відтворення кольорів.

Адитивне рівняння кольорів проводиться за наступною схемою.

На поверхню ідеального дифузного відбивача проектується пляма світла [C] з довільною спектральною щільністю:

Проектується пляма опорного білого кольору [W] зі спектральної щільністю:

а також, перекриваються плями [P1], [P2], [P3] трьох основних кольорів, з їх спектральними щільністями:



Спочатку інтенсивності [P1], [P2], [P3] встановлюються таким чином, щоб загальна частина плям відповідала опорному білому кольору [W]. Інтенсивності A1[W], A2[W], A3[W] вимірюють в будь-яких одиницях, наприклад, BAT. Вони є величинами, що урівноважують опорний білий.

Потім інтенсивності встановлюються рівними у кольорі [C]. Якщо рівняння досягається, записують інтенсивності *A1(C), A2(C), A3(C)* і обчислюють нормовані величини:

$$T_i = A_i(C) A_i(W) \text{ , i = 1,2,3} \qquad (3.40)$$

які називаються координатами кольору. Такий метод є «прямим методом кількісного опису кольорів».

Явні недоліки цього методу:

- суб'єктивність оцінки результатів;
- громіздкість.

Встановлено експериментальним шляхом, що іноді повного зрівняння досягти не вдається при дуже сильному або дуже слабкому освітленні. Так само результати залежать від кольорового оточення. Проте виявилося, що прості правила рівняння кольорів залишаються справедливими в широкому діапазоні умов експерименту.

Грассман ввів 8 аксіом, що визначають триколірні рівняння кольорів, які є основою колориметрії:

1. Будь який колір може бути представлений сумішшю не більше ніж трьох кольорів.

2. Рівняння, досягнуте за даної інтенсивності світла, зберігається в широкому діапазоні інтенсивності.

3. Суміш кольорів не може бути розділена людським оком на окремі компоненти.

4. Яскравість суміші кольорів дорівнює сумі яскравості її компонент.

5. Закон складання: якщо колір [M] еквівалентний кольору [N], а колір [P] - кольору [Q], то суміш кольорів [M] і [P] еквівалентна суміші [N] і [Q]:

$$[M] \triangleq [N];\ [P] \triangleq [Q] \Rightarrow [M] \overset{+}{\triangle} [P] \triangleq [N] \overset{+}{\triangle} [Q], \qquad (3.41)$$

де $\triangleq$ - оператор рівняння (еквівалентності);

$\overset{+}{\triangle}$ - адитивна суміш;

$\overset{\bullet}{\triangle}$ - одиниця суміші.

6. Закон віднімання: якщо суміш кольорів [M] і [P] еквівалентна суміші [N] і [Q], і якщо колір [P] еквівалентний кольору [Q], то колір [M] еквівалентний кольору [N]:

$$[M] \triangleq [P] = [N] \triangleq [Q], [P] \triangleq [Q] \Rightarrow [M] \triangleq [N]. \quad (3.42)$$

7. Закон транзитивності: якщо колір [M] еквівалентний кольору [N], а [N] = [P], то колір [M] еквівалентний кольору [P]:

$$[M] \triangleq [N], [N] \triangleq [P] \Rightarrow [M] \triangleq [P]. \quad (3.43)$$

8. Зрівнювання кольорів. Справедливо одне з трьох співвідношень:

а) З одиниці кольору [C] зрівнюють суміш M одиниць [M], N одиниць [N] і P одиниць [P]:

$$З \triangleq [C] \triangleq M \triangleq [M] \triangleq N \triangleq [N] \triangleq P \triangleq [P]; \quad (3.44)$$

б) суміш C одиниць [C] і M одиниць [M] зрівнюють суміш N одиниць [N] і P одиниць кольору [P]:

$$C \triangleq [C] \triangleq M \triangleq [M] \triangleq N \triangleq [N] \triangleq P \triangleq [P]; \quad (3.45)$$

в) суміш C одиниць [C], M одиниць [M] і N одиниць [N] зрівнюють P одиниць [P]:

$$C \triangleq [C] \triangleq M \triangleq [M] \triangleq N \triangleq [N] \triangleq P \triangleq [P]. \quad (3.46)$$

З восьмої аксіоми випливає, що колір [C] може бути зрівняний сумішшю трьох основних кольорів [P1], [P2], [P3], тобто

$$[C] \triangleq A1\ (C) \triangleq [P1] \triangleq A2\ (C) \triangleq [P2] \triangleq A3\ (C) \triangleq [P3] \quad (3.47)$$

де *A1(C), A2(C), A3(C)* - зрівнюють величини кольору [C].

Згідно аксіомі 4 і цьому відношенню, спектральна щільність $C(\lambda)$ може бути замінена еквівалентною спектральною щільністю суміші основних кольорів:

$$C(\lambda) \triangleq A_1(C)P_1(\lambda) + A_2(C)P_2(\lambda) + A_3(C)P_3(\lambda) = \sum_{j=1}^{3} A_j(C) \cdot P_j(\lambda) \quad (3.48)$$

Дане співвідношення має простий сенс: спектральні щільності, пов'язані оператором еквівалентності $\triangleq$, викликають однакові колірні відчуття. Оскільки координати кольоровості визначаються як:

$$T_j = A_j(C) A_j(W) \ , \quad (3.49)$$

то підставивши (3.49) в (3.48) і на основі 4 аксіоми визначимо яскравість кольору:

$$\mathrm{C}(\lambda) \triangleq \sum_{j=1}^{3} T_j(C) \cdot A_j(W) \cdot P_j(\lambda) \, . \quad (3.50)$$

Яскравість тоді визначається як:

$$L(C) = \int C(\lambda) \cdot V(\lambda) d\lambda = \sum_{j=1}^{3} \int A_j(C) \cdot P_j(\lambda) \cdot V_j(\lambda) d\lambda = \sum_{j=1}^{3} \int T_j(C) \cdot A_j(W) \cdot P_j(\lambda) \cdot V_j(\lambda) d\lambda \quad (3.51)$$

де $V(\lambda)$ - відносна світлова ефективність.

Співвідношення (3.50) і (3.51) є кількісною основою колориметрії. Колориметрія – це наука про вимірювання кольорів.

Для трикольорової моделі зору, як ми вказували раніше:

$$e_i(C) = \int C(\lambda) \cdot S_i(\lambda) d\lambda, \quad \text{где } i = 1 \ldots 3 \quad (3.52)$$

Згідно аксіомі Грассмана, якщо спостерігач бачить еквівалентну суміш первинних кольорів, а не вихідний колір [С], то заміна спектральної щільності С*(λ)* еквівалентною спектральною щільністю не повинна приводити до зміни колірного відчуття. Отже,

$$e_i(C) = \sum_{j=1}^{3} T_j(C) \cdot A_j(W) \cdot \int P_j(\lambda) \cdot S_i(\lambda) d\lambda, \quad \text{где } i = 1 \ldots 3 \qquad (3.53)$$

Визначивши коефіцієнти

$$k_{ij} = \int P_j(\lambda) \cdot S_i(\lambda) d\lambda, \qquad (3.54)$$

Співвідношення (3.53) може бути переписано у вигляді:

$$\begin{bmatrix} e_1(C) \\ e_2(C) \\ e_3(C) \end{bmatrix} = \begin{bmatrix} k_{11} & k_{12} & k_{13} \\ k_{21} & k_{22} & k_{23} \\ k_{31} & k_{32} & k_{33} \end{bmatrix} \cdot \begin{bmatrix} A_1(W) & 0 & 0 \\ 0 & A_2(W) & 0 \\ 0 & 0 & A_3(W) \end{bmatrix} \cdot \begin{bmatrix} T_1(C) \\ T_2(C) \\ T_3(C) \end{bmatrix} \qquad (3.55)$$

$$e(C) = K \cdot A \cdot T(C) \qquad (3.56)$$

$$T(C) = (K \cdot A)^{-1} \cdot e(C) \qquad (3.57)$$

Для заданого набору основних кольорів і базового білого кольору коефіцієнти матриці К і матриці А залишаються незмінними.

Монохроматичний колір $[C_\varphi]$з довжиною хвилі φ і одиничною енергією має спектральну щільність: $C_\phi(\lambda) = \delta(\lambda - \phi)$. При цьому, справедливо співвідношення:

$$e_i(C_\varphi) = \int \delta(\lambda - \varphi) \cdot S_i(\lambda) d\lambda = \sum_{j=1}^{3} A_j(W) \cdot P_j(\lambda) \cdot T_{S_i}(\varphi) \cdot S_i(\lambda) d\lambda \qquad (3.58)$$

де $T_{S_i}(\phi)$ – координати кольору спектральних (вузькосмугових) випромінювань одиничної енергії, які прийнято називати функціями складання.

Якщо розглядати довільний колір [C], то $C(\varphi)$ цього світла при довжині хвилі φ мають координати $T_{S_i}(\phi)$, а значить:

$$\int C(\varphi)\delta(\lambda-\varphi)\cdot S_i(\lambda)d\lambda = \sum_{j=1}^{3}\int A_j(W)\cdot P_j(\lambda)\cdot T_{S_i}(\varphi)\cdot S_i(\lambda)d\lambda \quad (3.59)$$

Звідки для функції, яка описує сигнали колб, отримаємо:

$$e_i(C) = \sum_{i=1}^{3}\int\int A_j(W)\cdot P_j(\lambda)\cdot C(\varphi)\cdot T_{S_i}(\varphi)\cdot S_i(\lambda)d\varphi d\lambda \qquad (3.60)$$

Порівнюючи дане рівняння з (3.53), можемо стверджувати, що координати кольору [C]:

$$T_j(C) = \int C(\varphi)\cdot T_{S_j}(\varphi)d\varphi \qquad (3.61)$$

Координати T1, T2, T3 можна розглядати як координати точок в деякому тривимірному просторі (рис. 3.11), а колір - як вектор в цьому просторі.

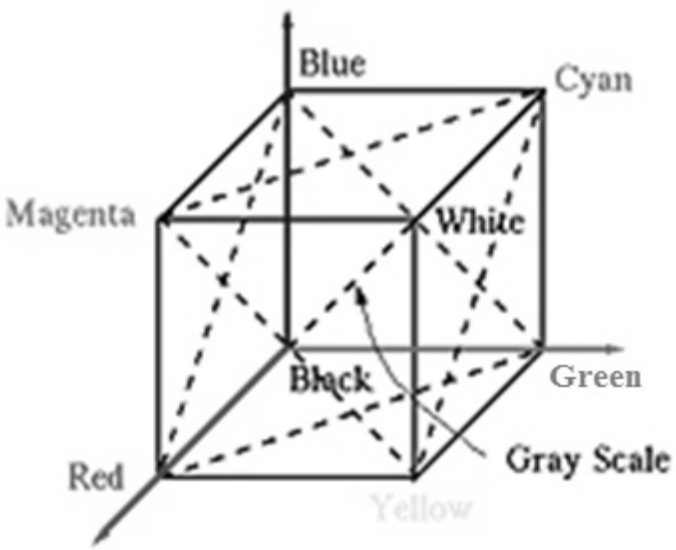


Рис. 3.11. Представлення кольору як координати

На ньому зображений трикутник Максвелла (формується точками R, G, D). Відстань від вершин цього трикутника до точки перетину кольорового вектору з площиною трикутника характеризує колірний тон і насиченість кольорів.

Якщо яскравість кольору не враховується, то в такому випадку тон і насиченість кольору можна виразити через координати кольоровості, які є нормованими координатами кольору:

$$t_1 = \frac{T_1}{T_1 + T_2 + T_3}, \quad t_2 = \frac{T_2}{T_1 + T_2 + T_3}, \quad t_3 = \frac{T_3}{T_1 + T_2 + T_3} \qquad (3.62)$$

$t_3 = 1 - t_2 - t_1$, і отже, для опису кольору необхідні тільки дві координати. На рис. 3.12 представлений графік кольоровості для набору типових основних кольорів. За допомогою реальних джерел світла можуть бути відтворені тільки ті кольори, які лежать всередині трикутника, визначеного основними кольорами.

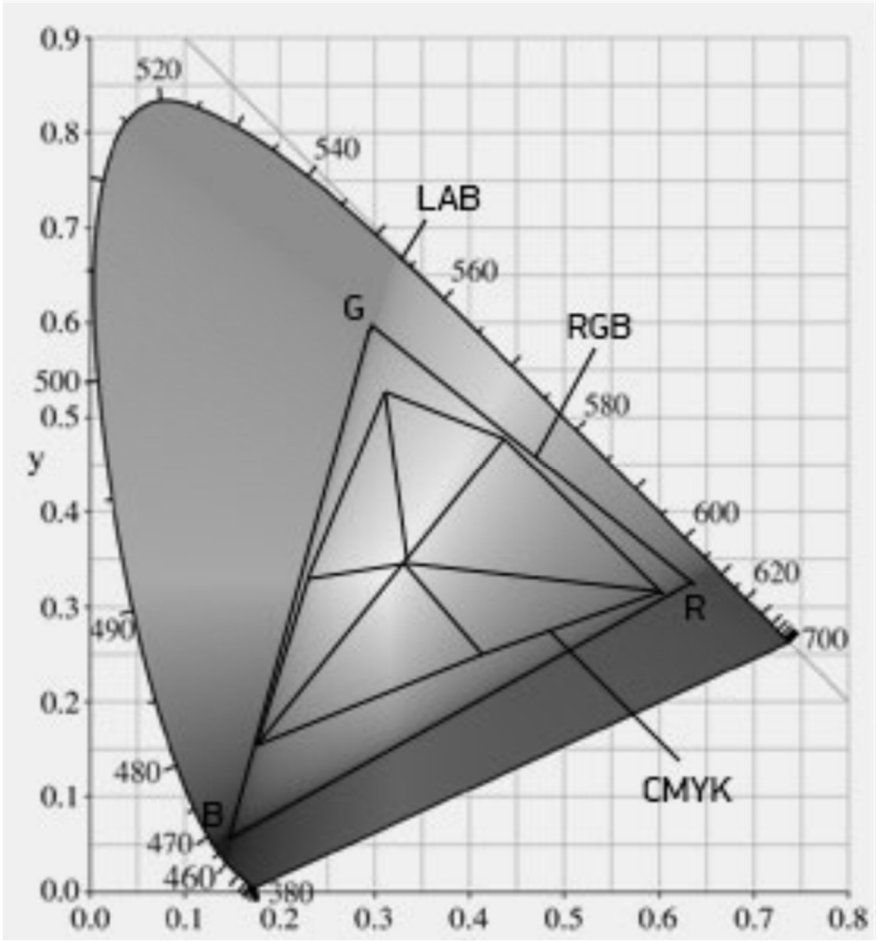


Рис. 3.12. Графік кольоровості для типових основних кольорів

З раніше розглянутого виразу (3.52) слідує, що набір основних кольорів можна задавати безліччю способів.

Припустимо, [P1], [P2], [P3] – вихідний набір основних кольорів із спектральними щільностями *P1(λ), P2(λ), P3(λ)*, і їх інтенсивностями, що урівноважують опорний білий *A1(W), A2(W), A3(W)*.

Припустимо також, що є інший набір основних кольорів $[\widetilde{P_1}],[\widetilde{P_2}],[\widetilde{P_3}]$, із спектральними щільностями $\widetilde{P_1}(\lambda),\widetilde{P_2}(\lambda),\widetilde{P_3}(\lambda)$. Опорний білий колір $[\widetilde{W}]$, який може відрізнятися від *[W]*, зрівнюється при інтенсивностях $\widetilde{A_1}(W),\widetilde{A_2}(W),\widetilde{A_3}(W)$ нових кольорів.

Довільний колір [C] має координати *T1(C), T2(C), T3(C)* в вихідній системі координат і для нового набору.

Для цих наборів справедливе співвідношення:

$$e(C) = K \cdot A(W) \cdot T(C) = \widetilde{K} \cdot \widetilde{A}(W) \cdot \widetilde{T}\,(C) \qquad (3.63)$$

Встановимо тепер одиниці виміру нових координат кольору, викликавши замість кольору [C] опорний білий $[\widetilde{W}]$:

$$e(\widetilde{W}) = K \cdot A(W) \cdot T(\widetilde{W}) = \widetilde{K} \cdot \widetilde{A}(W) \cdot \widetilde{T}\,(\widetilde{W}), \qquad (3.64)$$

оскільки $\widetilde{T_1}(\widetilde{W})= \widetilde{T_2}(\widetilde{W})= \widetilde{T_3}(\widetilde{W})= 1$, то підставивши в (3.64) нові основні кольори $[\widetilde{P_1}], [\widetilde{P_2}], [\widetilde{P_3}]$ з їх координатами кольоровості в вихідну систему координат кольорів, отримаємо:

$$\begin{aligned} e(\widetilde{P}_1) &= K \cdot A(W) \cdot T(\widetilde{P}_1) = \widetilde{K} \cdot \widetilde{A}(\widetilde{W}) \cdot \widetilde{T}\,(\widetilde{P}_1); \\ e(\widetilde{P}_2) &= K \cdot A(W) \cdot T(\widetilde{P}_2) = \widetilde{K} \cdot \widetilde{A}(\widetilde{W}) \cdot \widetilde{T}\,(\widetilde{P}_2); \\ e(\widetilde{P}_3) &= K \cdot A(W) \cdot T(\widetilde{P}_3) = \widetilde{K} \cdot \widetilde{A}(\widetilde{W}) \cdot \widetilde{T}\,(\widetilde{P}_3); \end{aligned} \qquad (3.65)$$

Де $\widetilde{T}(\widetilde{P}_1) = \begin{vmatrix} 1 \\ \overline{\widetilde{A}_1(\widetilde{W})} \\ 0 \\ 0 \end{vmatrix}$, $\widetilde{T}(\widetilde{P}_2) = \begin{vmatrix} 0 \\ 1 \\ \overline{\widetilde{A}_2(\widetilde{W})} \\ 0 \end{vmatrix}$, $\widetilde{T}(\widetilde{P}_3) = \begin{vmatrix} 0 \\ 0 \\ 1 \\ \overline{\widetilde{A}_3(\widetilde{W})} \end{vmatrix}$

Спільне рішення системи рівнянь (3.63 – 3.65) може бути записано у вигляді:

$$\begin{vmatrix} \widetilde{T}_1(C) \\ \widetilde{T}_2(C) \\ \widetilde{T}_3(C) \end{vmatrix} = \begin{vmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{vmatrix} \cdot \begin{vmatrix} T_1(C) \\ T_2(C) \\ T_3(C) \end{vmatrix} \qquad (3.66)$$

Де $m_{ij} = \dfrac{\Delta_{ij}}{\Delta_i}$.

$$\Delta_1 = T_1(\widetilde{W}) \cdot \Delta_{11} + T_2(\widetilde{W}) \cdot \Delta_{12} + T_3(\widetilde{W}) \cdot \Delta_{13};$$
$$\Delta_2 = T_1(\widetilde{W}) \cdot \Delta_{21} + T_2(\widetilde{W}) \cdot \Delta_{22} + T_3(\widetilde{W}) \cdot \Delta_{23};$$
$$\Delta_3 = T_1(\widetilde{W}) \cdot \Delta_{31} + T_2(\widetilde{W}) \cdot \Delta_{32} + T_3(\widetilde{W}) \cdot \Delta_{33};$$

$$\Delta_{11} = t_2(\widetilde{P}_2) \cdot t_3(\widetilde{P}_3) - t_3(\widetilde{P}_2) \cdot t_2(\widetilde{P}_3);$$
$$\Delta_{12} = t_3(\widetilde{P}_2) \cdot t_1(\widetilde{P}_3) - t_1(\widetilde{P}_2) \cdot t_3(\widetilde{P}_3);$$
$$\Delta_{13} = t_1(\widetilde{P}_2) \cdot t_2(\widetilde{P}_3) - t_2(\widetilde{P}_2) \cdot t_1(\widetilde{P}_3);$$
$$\Delta_{21} = t_3(\widetilde{P}_1) \cdot t_2(\widetilde{P}_3) - t_2(\widetilde{P}_1) \cdot t_3(\widetilde{P}_3);$$
$$\Delta_{22} = t_1(\widetilde{P}_1) \cdot t_3(\widetilde{P}_3) - t_3(\widetilde{P}_1) \cdot t_1(\widetilde{P}_3);$$
$$\Delta_{23} = t_2(\widetilde{P}_1) \cdot t_1(\widetilde{P}_3) - t_1(\widetilde{P}_1) \cdot t_2(\widetilde{P}_3);$$
$$\Delta_{31} = t_2(\widetilde{P}_1) \cdot t_3(\widetilde{P}_2) - t_3(\widetilde{P}_1) \cdot t_2(\widetilde{P}_2);$$
$$\Delta_{32} = t_3(\widetilde{P}_1) \cdot t_1(\widetilde{P}_2) - t_1(\widetilde{P}_1) \cdot t_3(\widetilde{P}_2);$$
$$\Delta_{11} = t_1(\widetilde{P}_1) \cdot t_2(\widetilde{P}_2) - t_2(\widetilde{P}_1) \cdot t_1(\widetilde{P}_2);$$

Для кількісного опису кольорів запропоновано багато різних систем координат. Розглянемо деякі з них.

3.2.3. Система координат спектральних основних кольорів МКО

Система розроблена в 1931р. Її утворюють монохроматичні кольори – червоний (700нм), зелений (516,1нм), синій (435,8нм). Для неї функції складання (рис. 3.13) обрані таким чином, щоб опорний білий мав рівномірну спектральну щільність у видимій частині спектру. Графік кольоровості наведений на рис. 3.14.

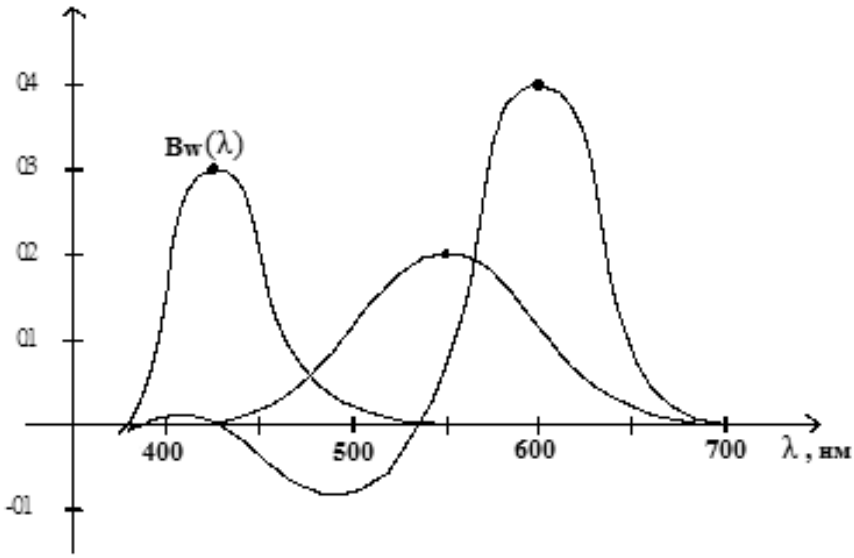


Рис. 3.13. Зовнішній вигляд функції для опорного білого кольору в даній системі координат

Ці криві отримані в експериментах щодо вирівнювання кольорів з великим числом спостерігачів. За результатами експериментів був визначений так само «стандартний наглядач».

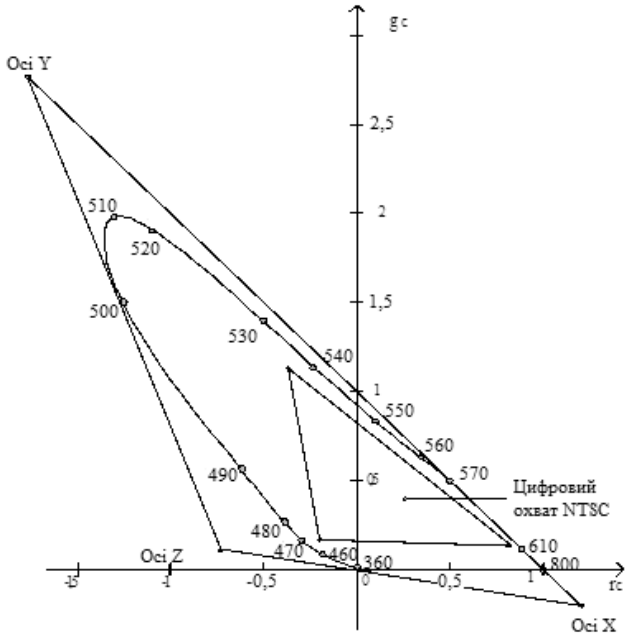


Рис. 3.14. Кольоровість спектральних основних кольорів МКО

3.2.4. Система координат XYZ МКО

Для того, щоб не отримати негативні значення координат (рис 3.15), МКО розробила штучну систему , в якій для графіка кольоровості опорного білого з рівномірною щільністю виходять тільки позитивні значення. Координата Y еквівалентна яскравості *L(C)* кольору C.

3.2.5. Система координат приймача NTSC

Система координат триколірних люмінофорів приймачів NTSC може бути пов'язана з системою координат спектральних основних кольорів простим лінійним перетворенням. У цій системі одиниці виміру координат кольору нормовані так, що значення координат, при яких зрівнюється опорний білий, однакові. Люмінофори приймача не є джерелами монохроматичного кольору.

Координати кодуються трьома складовими: Y, I, Q. Координата Y збігається з координатою Y в системі XYZ МКО і відповідає яскравості. Інші дві координати описують колірний тон і насиченість.

Переваги системи наступні:

- сигнал Y може бути використаний телеприймачем однокольорового зображення;
- вдається передавати повний аналоговий сигнал кольорового зображення в тій же смузі частот, що і при одноколірному зображенні.

Графіки кольоровості для даних систем координат мають такий вигляд, представлений на рис. 3.16.

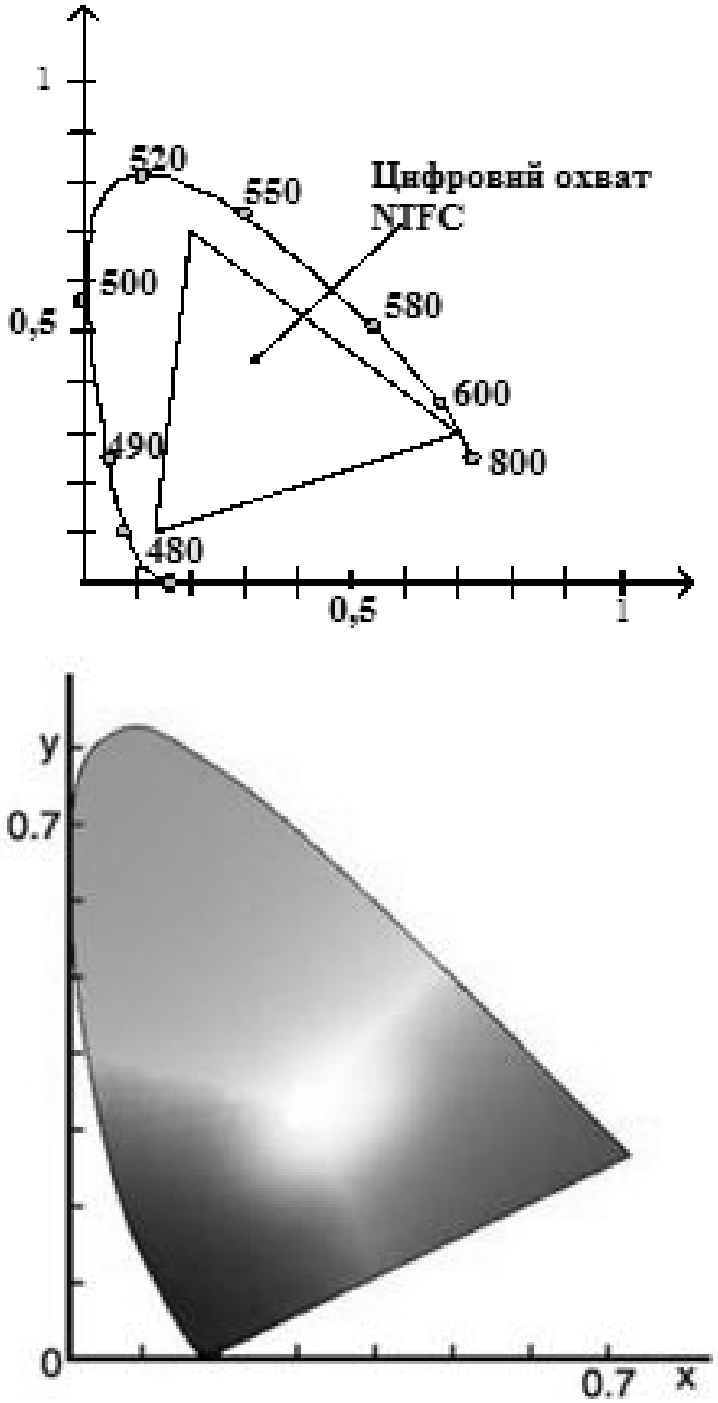


Рис. 3.15. Графік кольоровості в системі XYZ МКО

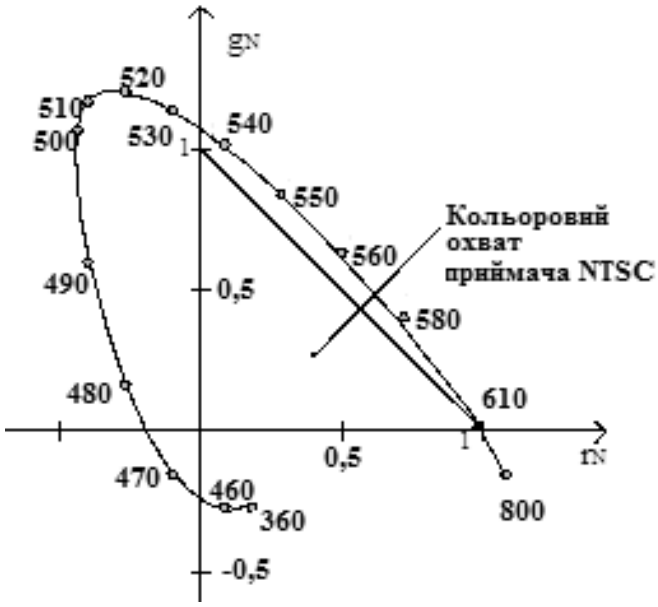


Рис. 3.16. Графік кольоровості в системі координат NTSC

Система координат трибарвних люмінофорів приймачів NTSC може бути пов'язана з системою координат спектральних основних кольорів простим лінійним перетворенням:

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.522 & 0.311 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$
$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 & 0.956 & 0.623 \\ 1 & -0.272 & -0.648 \\ 1 & -1.105 & 1.705 \end{bmatrix} * \begin{bmatrix} Y \\ I \\ Q \end{bmatrix} \qquad (3.67)$$

У цій системі одиниці вимірювання координат кольору нормовані так, що значення координат, при яких зрівнюється опорний білий, однакові. Люмінофори приймача не є джерелами монохроматичного кольору. Тому, кольорове охоплення вужче, ніж при використанні спектральних основних кольорів МКО.

Координати кодуються трьома складовими: Y, I, Q. Координата Y співпадає з координатою Y в системі XYZ МКО і

відповідає яскравості. Решта двох координат описує кольоровий тон і насиченість.

Переваги системи наступні:

- сигнал Y може бути використаний телеприймачами одноколірного зображення;
- вдається передавати повний аналоговий сигнал кольорового зображення у тій же смузі частот, що і при одноколірному зображенні.

3.2.6. Система координат Lab

Більшість колориметрів працює в цій системі. Система не має обмеження в охваті кольорів та є апаратно незалежною. Координати кольору визначаються наступним чином:

$$\begin{cases} L = 25 \cdot \left(\frac{100Y}{Y_0} \right)^{\frac{1}{3}} - 16 \\ a = 500 \cdot \left[\left(\frac{X}{X_0} \right)^{\frac{1}{3}} - \left(\frac{Y}{Y_0} \right)^{\frac{1}{3}} \right], \\ b = 200 \cdot \left[\left(\frac{Y}{Y_0} \right)^{\frac{1}{3}} - \left(\frac{Z}{Z_0} \right)^{\frac{1}{3}} \right] \end{cases} \quad (3.68)$$

де X0, Y0, Z0 – координати опорного білого кольору в системі XYZ;

L – визначає яскравість;

a – співвідношення червоного і зеленого;

b – співвідношення синього і жовтого.

3.2.7. Система координат RGB

Вимірювання колірних відчуттів людини є кінцевим результатом експерименту CIE (CIE — Communication Internationale de l'Eclairage - Міжнародна Комісія з Освітлення -

МКО) і лежить в основі всієї сучасної колориметрії - науки про вимірювання кольору. Фізіологічна колірна координатна система, одержана в результаті експериментів CIE, носить назву "CIE RGB".

В 1931 році була прийнята кольорова система координат RGB, де як осі координат була прийнята тріада кольорів - червоний $\lambda=700$ нМ, $P=243$ Вт; зелений $\lambda=546$,1 нМ, $P=4,66$ Вт; синій $\lambda=435,8$ нМ, $P=3,38$ Вт.

У експерименті CIE істотну частину чистих спектральних кольорів зрівняти не вдалося, внаслідок чого в кольоровій координатній системі CIE RGB деякі кольори мають негативні координати. Це створює великі незручності при математичних розрахунках і незабаром після виникнення CIE RGB, була запропонована інша колірна координатна система, одержана примусовим математичним перерахунком із початкової CIE RGB. Ця система одержала назву CIE XYZ (по трьох координатних осях — XYZ). ККС CIE XYZ не має негативних значень і має ряд позитивних властивостей, що спрощують обчислення. Зв'язок цієї системи з системою RGB описується співвідношеннями:

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.166 & 0.125 & 0.093 \\ 0.060 & 0.327 & 0.005 \\ 0.000 & 0.004 & 0.460 \end{bmatrix} * \begin{bmatrix} R \\ G \\ B \end{bmatrix} \tag{3.69}$$

Існують також приведені координати, які дозволяють обмежити колірний простір з боку максимуму:

$$\begin{cases} x = \dfrac{X}{X+Y+Z} \\ y = \dfrac{Y}{X+Y+Z} \\ z = \dfrac{Z}{X+Y+Z} \end{cases} \tag{3.70}$$

Оскільки, основні кольори XYZ не є реальними, то в кубічному колірному просторі лише частину об'єму займають кольори реальні.

Коли виникає необхідність продемонструвати кольорове охоплення (кількість відтворних кольорів) того або іншого пристрою, що показується завжди порівняно з кольоровим охопленням людського зору (рис. 3.12), вдаються до ще однієї координатної системи — xyY. Ця система одержана з XYZ шляхом простого математичного перерахунку:

$$x = X/(X+Y+Z);\ y = Y/(X+Y+Z);\ Y = Y \qquad (3.71)$$

Осі "x" і "y" — це осі кольоровості, а вісь "Y" — вісь світлості. На діаграмах прийнято зображати не саме охоплення, а його *проекцію* на площину "*xy*".

Добре збалансована структура ККС L*a*b* заснована на тій теорії, що колір не може бути одночасно зеленим і червоним або жовтим і синім. Отже, для опису атрибутів "червоний/зелений" і "жовтий/синій" можна скористатися одними і тими ж осями.

У ККС CIE (рис.3.17) Lab, величина L позначає світлість (Luminance, Light), a - величину червоної/зеленої складової, b - величину жовтої/синьої складової.

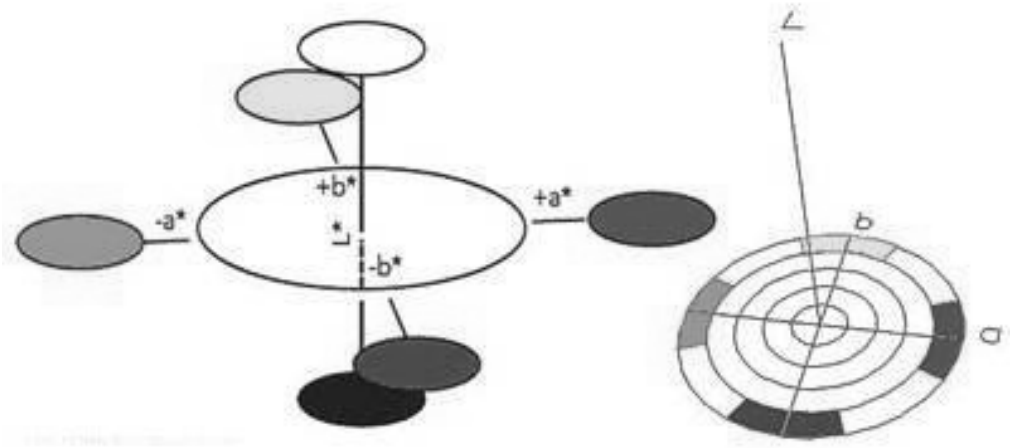


Рис. 3.17. Lab у двох різних графічних поданнях

ЦКС CIE Lab найширше застосовується для всіх математичних розрахунків, вироблюваних комп'ютерами при роботі з кольором. Крім того, при кольорокорекції цифрових зображень криві Lab дають користувачу комплект можливостей, що доповнюють традиційний інструментарій растрових редакторів.

Тема 3.3. Практичне використання рівняння кольорів

3.3.1. Індивідуальні завдання до визначення

Визначити координати кольору і кольоровості для заданих значень інтенсивності базових кольорів опорного білого і шуканого кольору (табл. 3.2). Вказати за якою схемою (8 аксіома Грассмана) визначаються шукані координати і чому.

Таблиця 3.2

Похідні данні для визначення координат кольорів

Варіант	Опорний білий			Шуканий колір			
	A1	A2	A3	C	A1 (C)	A2 (C)	A3 (C)
0	120	100	80	100	30	30	40
1	120	100	80	120	140	90	170
2	120	100	80	140	125	90	105
3	120	100	80	160	120	173	133
4	120	100	80	10	23	43	76
5	120	100	80	100	100	75	75
6	120	100	80	220	60	80	80
7	120	100	80	95	10	104	1
8	120	100	80	260	120	140	0
9	120	100	80	28	118	10	80

3.3.2. Порядок виконання роботи

Для вирішення поставленого завдання необхідно ознайомитися з матеріалами, викладеними в п.п. 3.2.1-3.2.6.

Відповідно зазначеними матеріалами, методика рішення задачі полягає у виконанні 3-х етапів:

- визначенні схеми по 8-ій аксіомі Грассмана, відповідно до якої виконується зрівнювання шуканого кольору і трьох опорних кольорів;
- визначенні координат кольору;
- визначенні координат кольоровості.

Для виконання першого етапу необхідно згадати, що властивості зору не дозволяють розділити суміш кольорів на окремі компоненти (3-я аксіома Грасмана) і яскравість суміші кольорів дорівнює сумі яскравостей її компонент (4-я аксіома Грасмана). Восьма аксіома визначає, по суті, що з чим зрівнюється:

- в першій схемі яскравість шуканого кольору зрівнюється сумою трьох опорних кольорів;
- у другій схемі зрівнюється яскравість кольору, отриманого змішуванням шуканого кольору і одного з опорних, з яскравістю кольору отриманого змішуванням двох інших опорних кольорів;
- в третій схемі яскравість одного з опорних кольорів зрівнюється з яскравістю суміші шуканого і двох інших опорних кольорів.

Однак, зрівняння проводиться тільки при однакових спектральних щільності - тобто, колірний тон сумішей (або окремих кольорів) в правій і лівій частинах повинен бути однаковим.

Так, наприклад, у варіанті 4 очевидно, що зрівнювання кольорів буде проводитися за 3-ою схемою, оскільки сума яскравостей шуканого кольору I(C), A1(C) і A2(C) буде дорівнює яскравості залишився кольору A3(C) і щодо аксіом Грассмана буде справедливим вираз:

10 △· [C] △+ 23 △· [A1] △+ 46 △· [A2] △= 76 △· [A3]

На другому етапі обчислюються координати кольору за співвідношенням:

$$T_j = {A_j(C)}\Big/{A_j(W)}.$$

При цьому слід пам'ятати, що координати опорних кольорів, які переносяться в ліву частину рівняння в схемі порівнювання, беруться зі знаком «мінус». Так, для нашого прикладу за варіантом 4, координати кольору будуть мати наступні значення:

T1= -23 / 120 □ -0.192; T2= -46 / 100 = -0.460; T3= 76 / 80 = 0.95

На третьому етапі знаходяться координати кольоровості. За базовим співвідношенням ці координати визначаються як:

$$t_i = \frac{T_i}{T_1 + T_2 + T_3}.$$

В відповідності до цього, отримуємо:

T1+T2+T3 ≈ -0.192 + (-0.46) + 0.95 ≈ 0.298;

t1 = T1 / (T1+T2+T3) ≈ -0.192 / 0.298 ≈ -0,346;
t2 = T2 / (T1+T2+T3) ≈ -0.46 / 0.298 ≈ -1.8904
t3 = T3 / (T1+T2+T3) ≈ 0.76 / 0.298 ≈ 2,550/

Таким чином, отримуємо усі необхідні результати.

Тема 3.4. Формат зберігання зображень DIB BMP

Для зберігання даних використовують різні файлові структури. Одним з перших форматів став формат BMP, який створювався з орієнтиром на використовувані тоді апаратні засоби відеоконтролерів, відтворення кольорів у яких було обмежено досить малою кількістю кольорів і тому використовувався принцип «палітри». Палітра визначає набір кольорів, доступний до використання та представляємий у вигляді таблиці.

Опис зображення здійснюється співставленням індексу кольору у палітрі, який розташований у координатах точки, що повинна бути відображена цим кольором. За аналогією, використовується «непряма» адресація кольорів.

Такий принцип є основою device-independent bitmap - DIB формату BMP, структура якого розглядається нижче. Поняття "незалежне" має на увазі, що інформація про колір представляється у формі, не залежній від типу пристрою виведення і методів реалізації кольорового зображення.

3.4.1. Загальна структура файлу формату bmp

Кожен файл у форматі BMP містить в своєму складі заголовок файлу (табл. 3.3), заголовок бітової карти (образу), палітру і безпосередньо бітову карту зображення. Заголовок файлу містить інформацію про тип, розмір і властивості файлу. Заголовок бітового образу (табл. 3.4) визначає розподільчу здатність зображення, використовувані методи стиснення графічної інформації і форматах представлення колірної інформації.

3.4.2. Структура заголовка файлу

Таблиця 3.3

Структура даних в заголовку файлу

Розмір	Призначення
word (2 байти)	тип файлу (для бітового образу - BM)
dword (4 байти)	розмір файлу в dword
word	не використовується
word	не використовується
dword	зсув даних бітового образу від заголовка в байтах

3.4.3. Структура заголовку бітового образу

Таблиця 3.4

Структура даних в заголовку бітового образу

№ з/п	Розмір	Призначення
1	dword	число байт для структури BITMAPINFOHEADER
2	dword	ширина бітового образу в пікселах
3	dword	висота бітового образу в пікселах
4	word	число бітових площин пристрою
5	word	число бітів на піксель
6	dword	тип стиснення, може приймати значення: 0 - стиснення відсутнє; 1 - стиснення для формату 8 біт/піксель 2 - стиснення для формату 4 біт/піксель
7	dword	розмір картинки в байтах
8	dword	горизонтальна розподільна здатність пристрою, пікселів/м
9	dword	вертикальна розподільна здатність пристрою, пікселів/м
10	dword	число використовуваних кольорів, 0 - якщо використовуються всі кольори палітри

№ з/п	Розмір	Призначення
11	dword	число «важливих» кольорів

Четверте поле завжди рівне 1, оскільки колір кодується послідовними бітами.

П'яте поле визначає число кольорів, використовуваних бітовим чином. Залежно від способу кодування, може приймати значення:

- 1 - бітовий образ монохромний, і таблиця кольорів повинна містити два елементи. Кожен біт в масиві даних кодує один піксел. Якщо значення біта - 0, то піксель стає першим кольором з таблиці кольорів, якщо 1 - піксель стає другим кольором таблиці.
- 4 - бітовий образ має максимум 16 кольорів і таблицю кольорів має до 16 елементів. Колір кожного піксела визначається по таблиці кольорів за допомогою чотирьохбітового індексу, наприклад, якщо перший байт даних має значення 3Ah, то при відображенні бітового образу колір першого піксела визначає четвертий елемент таблиці кольорів, а колір другого - одинадцятий.
- 8 - бітовий образ має максимум 256 кольорів, таблиця кольорів має до 256 елементів. Кожен байт масиву даних визначає колір одного піксела.
- 24 - бітовий образ має максимум 2 в 24-у ступеню кольорів. Таблиця кольорів порожня, а колір пікселів визначається пакетами з трьох байтів, що описують колірні інтенсивності червоного, зеленого і блакитного кольорів.

3.4.4. Методи стиснення бітового образу

Windows versions 3.0 і подальші версії підтримують метод кодування довгими серіями (run-length encoded - RLE) для стиснення бітового образу, що використовує 4 і 8 біт на піксель.

Стиснення для бітової карти в режимі 8 біт на піксель може реалізовуватися наступними методами – кодованим і абсолютним. Обидва методи можуть використовуватися одночасно в одному файлі.

Кодований режим полягає в нижченаведеному.

Одиниця інформації складається з 2 байт. Перший байт визначає число пікселів послідовності з індексом кольору, що міститься в наступному байті. Перед першим байтом послідовності повинні стояти нулі, що визначають наявність послідовності, що управляє. Наступний байт може визначати 3 варіанти:

0 – кінець рядка;

1 – кінець бітового образу;

2 – приріст, що містить беззнакові значення зсуву по горизонталі і вертикалі до наступної крапки з поточної позиції.

Абсолютний режим визначається послідовністю, в якій перший байт – нулі, а значення другого байту лежить у діапазоні від 0x03 ÷ 0xFF.

Другий байт визначає довжину наступної послідовності, в якій кожен байт містить індекс кольору для одного пікселя. Кожна серія повинна бути вирівняна по 2-байтовій межі. Приклади використання кодових послідовностей представлені в табл. 3.5.

Таблиця 3.5

Приклади кодування даних в RLE-8

Стиснені дані	Розгорнені дані
03 04	04 04 04
05 06	06 06 06 06 06
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	Move 5 right and 1 down
02 78	78 78
00 00	End of line
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	End of RLE bitmap

При стисненні 4-бітових послідовностей використовуються ті ж принципи, що і при стисненні 8-бітових послідовностей, що ілюстровано даними табл. 3.6.

Таблиця 3.6

Приклади кодування даних в RLE-4

Стиснені дані	Розгорнені дані
03 04	0 4 0
05 06	0 6 0 6 0
00 06 45 56 67 00	4 5 5 6 6 7
04 78	7 8 7 8
00 02 05 01	Move 5 right and 1 down
04 78	7 8 7 8
00 00	End of line
09 1E	1 E 1 E 1 E 1 E 1
00 01	End of RLE bitmap

3.4.5. Представлення палітри

Інформація про колір представляється у файлі четвірками байт, в яких байти відповідають інтенсивності червоної, зеленої і синьої складових, а четвертий байт зарезервований.

Тема 3.5. Практичне застосування квантування відеосигналу

3.5.1. Індивідуальні завдання до виконання

Визначити співвідношення сигнал/шум (Дб) для нормованого в межах від 0 до 1 сигналу відеозображення, при заданому законі розподілу одержуваних при його обробці значень, із заданою кількістю рівнів квантування.

При необхідності визначити мінімально необхідну кількість рівнів квантування для умови, що співвідношення сигнал/шум має бути не гірше зазначеного (табл. 3.7).

Таблиця 3.7

Похідні дані для розрахунків

Варіант	Кількість рівнів квантування	Співвідношення Сигнал/шум, Дб	Закон розподілення
0	4	20	Лінійний
1	5	25	Гауса
2	6	30	Лінійний
3	7	35	Гауса
4	8	40	Лінійний
5	10	20	Гауса

3.5.2. Порядок та приклад виконання роботи

Для виконання завдання необхідно вивчити матеріали, викладені вище. Розглянемо рішення поставленого завдання для варіанта 4.

Виходячи з наведеного матеріалу, в загальному випадку, методика розрахунку полягає у виконанні наступних дій:

- будуємо графік функції ймовірності;
- виходячи з умови забезпечення рівної ймовірності попадання відліків в будь-який з діапазонів квантування, розбиваємо шкалу ймовірності на рівні діапазони (в нашому випадку - на 8 діапазонів);
- провівши в межах отриманих діапазонів горизонтальні лінії до перетину з графіком функції ймовірності і опустивши з точок перетину вертикальні лінії на вісь абсцис, отримаємо шукані значення порогів квантування d_i;
- знайдемо ширину діапазонів квантування Δ_i, як різницю між верхнім значенням і нижнім значеннями сусідніх порогів квантування –
  $\Delta_i = d_{i+1} - d_i$;
- провівши серединні лінії в діапазонах квантування за шкалою ймовірності до перетину з графіком функції ймовірності і опустивши вертикальні лінії на вісь

абсцис, отримаємо шукані значення рівнів квантування r_i;

- підставивши в (3.64) значення ширини діапазонів квантування і їх кількості, отримаємо шукане значення співвідношення сигнал/шум квантування;
- порівнюємо отримане значення з заданим за спрощеним варіантом і, якщо воно вище заданого, то умова виконана, а якщо немає, то необхідно збільшити кількість діапазонів квантування і повторити розрахунок спочатку.

Однак, для лінійного закону розподілу є спрощена формула розрахунку шуканого співвідношення сигнал/шум квантування:

$$\frac{P_C}{P_{кв}} \approx 6 \cdot n_p + 10{,}8 \qquad (3.72)$$

Тобто, в нашому випадку при 8 діапазонах квантування, $n_p = 3$, так як $2^3 = 8$.

Тоді, співвідношення сигнал/шум квантування при заданих значеннях дорівнюватиме:

$$\frac{P_C}{P_{кв}} \approx 6 \cdot n_p + 10{,}8 \approx 6 \cdot 3 + 10.8 \approx 28.8 Дб \qquad (3.73)$$

Дане значення набагато менше, ніж необхідно мати за умовою завдання - 40 Дб. На підставі наведеної формули, можна вирішити зворотне завдання – знайти шукане значення. Оскільки, кількість діапазонів квантування має бути $2^5 = 32$ для задоволення поставлених умов.

Пороги квантування будуть визначатися для нашої задачі як:

$$di = i/32,\ ri = (i/32) + 1/64. \qquad (3.74)$$

Для повного вирішення завдання, необхідно привести числові значення di і ri.

Тема 3.6. Практичні завдання з обробки графічних даних у форматі BMP

3.6.1. Завдання на індивідуальну роботу

Метою роботи є ознайомлення з принципами опису і обробки графічної інформації на прикладі зображень, представлених у вигляді бітових карт у форматі BMP. У роботі необхідно реалізувати інтерфейс прикладної програми, що передбачає функції завантаження зображення у форматі bmp та їх відтворення на екрані без використання стандартних функцій API операційної системи чи мов програмування високого рівня.

3.6.2. Порядок виконання роботи

1) Розробити прототипи відповідних класів та структур даних для роботи з графічним файлом та супровідними даними.

2) Запустити комп'ютер і завантажити ОС.

3) Завантажити інструментальні засоби програмування.

4) Реалізувати інтерфейс прикладної програми, що передбачає функції завантаження зображення у форматі bmp.

5) Реалізувати прототипи класів, відповідно до п.1.

6) Відкомпілювати програму та протестувати її роботу на різних файлах.

3.6.3. Контрольні запитання

1) Моделі кольорового зору. Правила рівняння кольорів.

2) Аксіоми рівняння кольорів.

3) Поняття колірних координат і координат кольоровості.

4) Загальні положення перетворення координат кольору.

5) Моделі представлення кольорових зображень RGB, CMY, CIE та їх взаємозалежність.

6) Яка структура форматів машинного представлення зображень BMP, PCX?

7) Методи обробки квантованих величин.

Тема 3.7. Практичне застосування перетворення систем координат кольорів

Мета роботи полягає в тім, щоб визначити ефективні методи роботи з інформацією про колір елементів зображення та методів її обробки.

Ця робота є проміжною до роботи з покращення зображень засобами ковзанкової фільтрації зображень. На перед слід зазначити, що особливістю фільтрації є необхідність обробки яскравості у точці або кожної з компонент RGB-зображення, або еквівалентної яскравості. Хоча у другому варіанті ніби то потрібно виконувати додаткові операції з перетворення системи координат, насправді такий підхід є більш економним з огляду на те, що при обробці RGB компонент здійснюється спотворення кольору за рахунок операцій округлення цілочисельних байтових значень. Це призводить до додаткових витрат на відновлення якості. При обробці еквівалентної яскравості, кольорова компонента не використовується взагалі, за рахунок чого кольори є сталими.

3.7.1. Порядок виконання роботи

За основу для виконання цієї роботи потрібно взяти попередню роботу (п.3.6), з візуалізації забраження з файлу у форматі BMP.

Оскільки при виконанні попередньої роботи створюються структури, які несуть інформацію про колір пікселів у форматі RGB, то вже є похідні дані для виконання поточної роботи. Далі потрібно наступне:

1) розробити структури даних для представлення їх у форматі YIQ стандарту NTSC, де компонента I відповідає за еквівалентну яскравість (монохромне представлення зображення), а компоненти YQ – колірні координати;
2) реалізувати процедури перетворення кольорів між форматами RGB та YIQ згідно з відношеннями, наведеними у п.3.2.5;
3) реалізувати процедуру візуалізації монохромного зображення на основі компоненти яскравості;
4) дослідити працездатність та коректність реалізованих алгоритмів, їх вплив на якість зображення після виконання подвійного перетворення;
5) проаналізувати та пояснити отримані результати.

Запитання для самоперевірки до розділу 3

1) Моделі кольорового зору. Правила рівняння кольорів.
2) Аксіоми рівняння кольорів Грассмана.
3) Поняття колірних координат і координат кольоровості.
4) Перетворення координат кольору.
5) Моделі представлення кольорових зображень RGB, CMY, CIE і їх особливості і взаємозалежності.
6) Методи обробки квантованих величин.
7) Основні положення теорії стиснення зображень.
8) Класи зображень і прикладень для їх обробки.
9) Алгоритми стиснення без втрат – типові представники і їх характеристики.

РОЗДІЛ 4. ПОКРАЩЕННЯ ЗОБРАЖЕНЬ ТА ПОПЕРЕДНЯ ОБРОБКА ЗОБРАЖЕНЬ

Тема 4.1. Загальні питання організації систем обробки зображень та розпізнавання образів

Розробники комп'ютерних засобів завжди намагалися створити системи, які були б подібні до людини та мали східні з людиною властивості – слух, зір, інтелект та почуття гумору. Але ці задачі й досі не є вирішеними, оскільки не з'ясовано саме як людина реалізує ці можливості.

У сьогоденні засоби штучного інтелекту технічного зору розвинулися у самостійні фундаментальні дисципліни та реалізуються своїми, вже не дуже схожими на людські, алгоритмами. Ці алгоритми вимагають таких механізмів представлення та обробки даних, які спрощують саме ці процеси для машинної обробки.

Тому, сьогодні можна виділити два класи систем обробки зображень орієнтованих на людину та орієнтованих на машинну обробку. Деякі принципи однаково часто застосовуються як у одному так і в іншому класі.

Предметом поточної роботи є засвоєння базових принципів та підходів, які можна використовувати у системах обробки зображень та розпізнавання образів.

Тема 4.2. Особливості практичної реалізації систем

Прикладами широко використовуваних систем технічного зору в промисловому виробництві є завдання контролю:

- виробництва друкованих плат і монтажу радіоелементів;
- якості нанесення лакофарбових виробів;
- сортування виробів за видами і якості на конвеєрах;
- контроль стану промислових об'єктів і їх охорони.

Більшість із завдань, що вирішуються в промисловому виробництві для підвищення ефективності, наводяться до

завдань розпізнавання «плоских» зображень. Однак, є завдання, які вимагають просторового аналізу в процесі ідентифікації об'єктів і їх розпізнавання. В першу чергу, до них відносяться завдання управління промисловими роботами різного типу і призначення. Роботи, що використовують системи технічного зору можуть виконувати завдання складання виробів, їх фарбування, враховувати наявність людей в області своєї роботи (для підвищення безпеки людей), займатися транспортуванням виробів.

При вирішенні таких завдань недостатньо визначити наявність або відсутність того чи іншого об'єкта. Необхідно враховувати їх взаємне розташування і рух, що в значній мірі підвищує складність вирішення завдань.

При цьому, можна умовно виділити два класи роботизованих систем - роботи, що не міняють своє розташування з плином часу (наприклад, складальні роботи-маніпулятори на конвеєрі), і роботи, що змінюють своє розташування (мобільні логістичні роботи).

Розвиток рішень для мобільних роботів знайшло на сучасному етапі новий напрямок для впровадження - створення безпілотних транспортних систем. Тобто систем, які можуть пересуватися і обирати напрямок руху без участі людини.

Більш точно, дана область отримала розвиток знову ж з військової сфери - створення автономних систем управління ракетами, що переміщаються на малих висотах і враховують особливості рельєфу місцевості. Зараз ці завдання у військовій сфері перенесені на різні системи дронів і безпілотних літальних апаратів.

В області цивільного використання за допомогою систем технічного зору вирішуються завдання забезпечення безпеки пасажирських і вантажних перевезень, забезпечення безпеки і захисту інформації.

Сучасні технології не виділяють окремо термін «Технічний зір». На відміну від цього, існує два пов'язаних базових поняття, відмінність між якими вкрай відносно - це поняття «Машинний зір» [1,2] і «Комп'ютерний зір» [3,4]. Найбільш загальне

визначення, дане все-таки в англійському варіанті Computer vision - це «термін, що визначає технології та методи, які використовуються для отримання інформації з зображення». Поняття «Комп'ютерне зір» в цьому відношенні більш вузьке і в Вікіпедії визначається як «... міждисциплінарна область, спрямована на ефективне вирішення завдань розуміння цифрових зображень і відео ...» і «... автоматизації завдань, що вирішуються зорової системою людини ...».

Виходячи з цих визначень, можна зробити висновок, що «Комп'ютерне зір» є складовою частиною систем «Машинного зору» і механізмом для його реалізації. Крім цього, завдання машинного і комп'ютерного «розуміння» тісно пов'язані з системами штучного інтелекту.

Більш чітке і суворе визначення дано в [5] для Систем Машинного Зору (СМЗ, MVS - Machine Vision Systems): *«A machine vision system (MVS) is a type of technology that enables a computing device to inspect, evaluate and identify still or moving images»*, тобто, це технологія, що дозволяє комп'ютерним пристроям виявляти, аналізувати і ідентифікувати статичні і рухомі зображення.

Оскільки мова йде про розуміння саме зображень, представлених в статичної або динамічної (відео) формі, то в структурі технічних систем прийнято виділяти три основні компоненти - відео-сенсори, апаратну і програмну частини [6].

Сенсорна частина вирішує завдання «сприйняття» навколишнього середовища або окремої її «сцени» шляхом створення цифрового еквівалента розподілу електромагнітних випромінювань в просторі.

Слід відразу зазначити, що для СМЗ зовсім не обов'язково проводити аналіз саме тієї частини спектра випромінювань, які сприймаються людиною. У цьому сенсі, вони більш наближені до природи, дозволяючи обробляти дані в більш широкому діапазоні частот, що характерно для тваринного світу. Саме така особливість, в тому числі, використовується в спеціалізованих людино-машинних системах, які допомагають людям,

наприклад, в тепловізорах або системах, які суміщають дані від сенсорів декількох видів [7].

Якщо необхідне рішення задачі саме в діапазоні видимого випромінювання, то деякі джерела [8,9] виділяють в окрему компоненту оптичні елементи і компоненти для «підсвічування» об'єктів. Вказується також, що перераховані компоненти можуть мати як окремі конструктивні рішення, так і входити до складу єдиного вироби. Конкретне конструктивне рішення, природно, буде залежати від розв'язуваної задачі і області застосування.

Обчислювальна платформа, на базі якої реалізується СМЗ може бути самого різного роду. Тут для апаратної частини можуть використовуватися як пристрої загального призначення, так і спеціалізовані рішення.

В якості першого варіанту можуть бути обрані як звичайні комп'ютери і контролери з достатньою продуктивністю (типу Raspberry Pi) так і мобільні пристрої типу смартфонів.

У другому випадку, використовуються спеціалізовані архітектури, в тому числі і на базі процесорів машинного зору [9, 10] або спеціалізованих мікроконтролерних систем на основі універсальних процесорних елементів.

Для кожної такої системи використовується відповідне програмне забезпечення - власне для спеціалізованих платформ і рішення, в тому числі, на основі стандартних бібліотек або відкритого коду, наприклад, OpenCV.

Найбільш типовими завданнями, які вирішують СМЗ, в області промисловості, є [11]:

- контроль нанесення і вмісту маркування;
- зчитування і розпізнавання штрих-коду і текстової інформації;
- перевірка цілісності упаковки і тари;
- контроль нанесення етикетки;
- сортування товарів, контроль відповідності зовнішнього вигляду продукту;
- виявлення, вимір і рахунок об'єктів;
- контроль наявності вкраплень, сторонніх включень.

Такі завдання вирішуються системами з 1 камерою на кожен контрольований процес.

Однак, крім позитивних факторів, використання СМЗ висуває і додаткові вимоги як до організації виробничих процесів, так і по створенню певних умов для їх експлуатації. До основних факторів і вимогам по експлуатації можна віднести:

- забезпечення заданої відстані до об'єкта;
- облік величини об'єкта;
- узгодження швидкості руху;
- забезпечення відповідного середовища експлуатації камери;
- забезпечення відповідного освітлення;
- необхідність синхронізації та інтеграції в загальну систему.

Початкова вартість інтегрованої системи для вирішення подібних завдань становить (за даними на 2016-й рік) близько 1500 €.

Ефективність використання, яка полягає в прийнятті конкретного рішення по кожному аналізованого об'єкта, доходить до 100% [10].

Як правило, професійні цифрові камери сьогодні йдуть з повною підтримкою 3D і вирішують більш складні завдання з аналізом просторового розташування об'єктів або орієнтування в просторі. Фактично, такі камери вирішують завдання стереоскопічного (бінокулярного, 3D) зору. Такі системи використовуються з роботами-маніпуляторами і в якості компонентів різних рухливі систем.

Оскільки точність визначення відстані на основі зображень досить обмежена і залежить від безлічі факторів, серед яких не тільки просторова роздільна здатність камери, але і відстань між оптичними елементами, то часто виникає необхідність використання додаткових сенсорних систем [22].

Так, для підвищення точності в сучасних системах можуть використовуватися спеціалізовані лазерні далекоміри, оптичні й ультразвукові датчики.

Слід зазначити, що сьогодні в деяких СМЗ для побудови моделі навколишнього простору в якості альтернативи у вигляді полігональної сітки використовуються виключно спеціалізовані лазерні системи - лідари - Light Identification Detection and Ranging, що дозволяють згенерувати і зняти «хмара» міток з навколишнього простору в короткий проміжок часу з їх точними координатами щодо скануючого пристрою. Це дозволяє виключити складні процеси обробки зображень для виділення координатних даних.

На сьогодні існує досить велика кількість різноманітних програмно-апаратних платформ для вирішення задачі комп'ютерного зору. Компанія Omron (Японія) [15] є одним з основних постачальників компонентів для автоматизації виробничих процесів широкого спектра. У тому числі це стосується і систем технічного зору, представлених широким набором пристроїв. Загальна лінійка характеризується параметрами, представленими на рис. 4.1.

Основні характеристики датчиків наступні:

- функції автонастройки;
- вбудований сенсорний екран;
- швидкість зчитування до 96 кадрів / секунду;
- до 20 інструментів обробки зображення;
- до 128 параметрів, що настроюються;
- мережевий інтерфейс Ethernet 100BASE-TX / 10BASE-T;
- послідовні інтерфейси USB2.0, RS-232C, RS-422;
- 33 дискретних входу / виходу;
- карта пам'яті SD.

Хоча в даній класифікації модель F-160 віднесена до датчиків, її контролер дозволяє вирішувати і завдання розпізнавання і управління так само.

У загальному випадку, серія F являє собою повнофункціональні системи технічного зору. До складу серії входять контролер з можливістю підключення декількох камер (2-4 в залежності від серії). Високошвидкісні камери мають функції обмеження площі сканування, що дає затримку

спрацьовування до 2 мс. Вони можуть мати інтелектуальну підсвітку, що забезпечує належне освітлення.

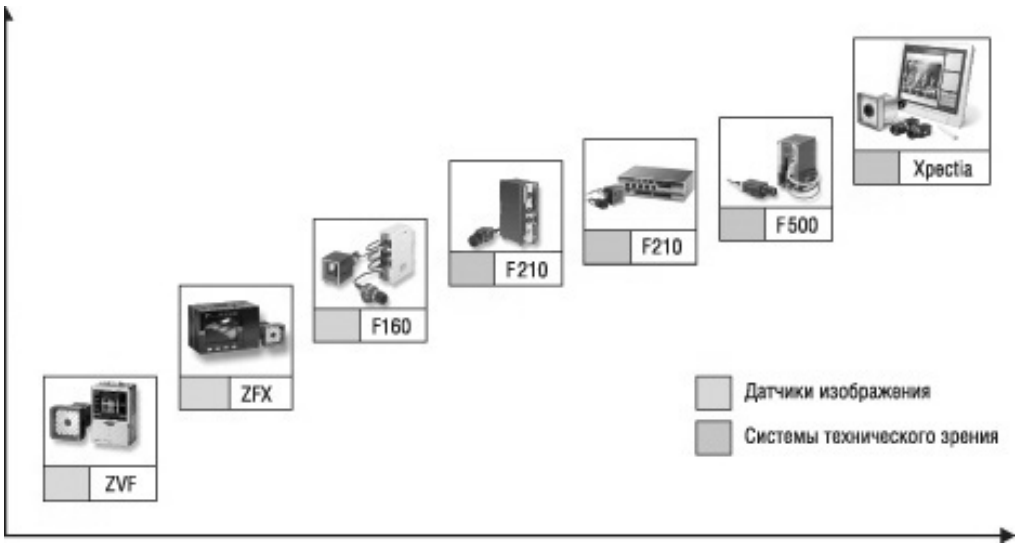


Рис. 4.1. Лінійка пристроїв технічного зору Omron

Всі серії містять слоти карт пам'яті Compact Flash для зберігання зображень, даних вимірювання і даних розширення функціоналу.

Контролери не мають вбудованого монітору, але дозволяють підключати зовнішні дисплеї та підтримують графічний інтерфейс користувача з системою меню. Користувач може налаштовувати інтерфейс для простого і наочного аналізу даних, що відображаються. Як інтерфейс управління застосовується дистанційний пульт.

4.2.1. Проект OpenMV.io

Проект «персональної» системи технічного зору [31, 32] реалізований в Китаї на відносно простий апаратній платформі і має наступні характеристики і параметри:

- процесор STM32F765VI ARM Cortex M7 з частотою 216 МГц;
- 512 Кбайт оперативної пам'яті;

- 2 МБ флеш-пам'яті;
- напруга логічного рівня 3,3;
- роз'єм μSD Card, зі швидкістю до 100 Мбіт / с;
- Матриця OV7725 для виведення 8-бітних зображень або 16-бітних RGB565 роздільною здатністю 640x480;
- FPS 60 (320x240) або 120 (320x240) кадрів в секунду;
- Камера з 2.8-міліметровим об'єктивом на стандартному кріпленні M12;
- мова програмування Python.

Система реалізує велику кількість функцій розпізнавання - Виявлення лінії, кола, особи, відстеження трекінгу очей, відстеження маркерів і квітів, виявлення і декодування штрих і QR-кодів, надання шаблонів зображень, захоплення зображень і відео.

Система має власну середу для програмування і підтримує, в тому числі, російську мову.

4.2.2. Відео-процесорні пристрої від Intel

Провідна Американська компанія в області виробництва компонентів обчислювальної техніки вже майже десятиліття [10] займається розробкою апаратної підтримки систем технічного та бінокулярного зору.

Сьогодні існує два неконкурирующих напрямки, що розробляються цією компанією - EyeQ і Movidius ™ [33].

У першому випадку, це інтегральне рішення для створення безпілотних автономних систем в автомобілебудуванні, що підтримує взаємодію з безліччю сенсорних пристроїв автомобіля, кількома камерами високої роздільної здатності та системи LiDAR. Структура апаратної платформи, реалізованої у вигляді SoC, представлена на рис. 4.2. дана платформа орієнтована на створення повністю автономних мобільних пристроїв відповідно до нових вимог безпеки.

Технологія Movidius є універсальним рішенням для безлічі практичних задач в різних областях і втілюється у вигляді окремого відео-процесорного пристрою, побудованого на основі

багаторівневих нейромереж, що реалізують до 16 незалежних потоків обробки відеоінформації, які надходить з 8 камер високого розподілення (4К) та обробляє до 700 млн. пікселів в секунду.

Тема 4.3. Програмне забезпечення систем обробки та розпізнавання зображень

Робота з програмним забезпеченням можна розділити на кілька груп. В першу чергу, це продукти від виробників систем технічного зору, які не доступні для модифікації і дозволяють використовувати його у вузьких рамках допустимих параметрів.

Другу групу утворюють пакети і бібліотеки щодо універсального призначення, які можуть бути як ліцензійними, так і вільно поширюваними.

Як ліцензійного програмного забезпечення слід відзначити систему LabView від National Instruments [17, 18], яка містить в своєму складі блоки і елементи, орієнтовані на вирішення задач технічного зору.

Відкрите програмне забезпечення, яке можна використовувати в своїх проектах, представляється підключається бібліотекою OpenCV [19], який має реалізації для призначених для користувача проектів на мовах C, Python, JS.

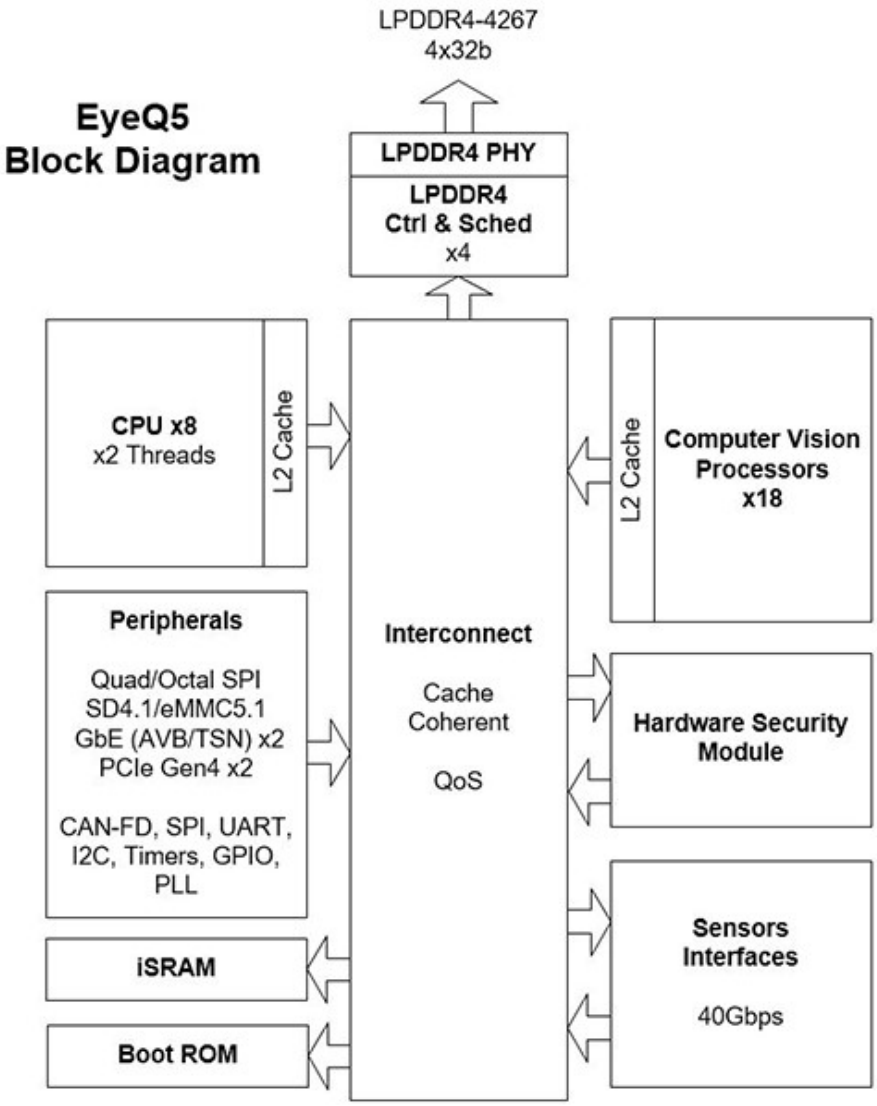


Рис. 4.2. Структура чипу по технології EyeQ5

Модулі для роботи з тривимірним введенням зображень представлені кількома бібліотеками:

- Calib3d - Camera Calibration and 3D Reconstruction;
- cnn_3dobj - 3D object recognition and pose estimation API;
- ccalib. Custom Calibration Pattern for 3D reconstruction;
- rapid - silhouette based 3D object tracking;
- face - Face Analysis;
- stereo - Stereo Correspondance Algorithms.

Прикладом використання даного програмного забезпечення є робота, представлена в [20] і реалізована на платформі OrangePi 3+ і двох ширококутних камер з USB інтерфейсом.

Тема 4.4. Математичне та алгоритмічне підґрунтя

4.4.1. Склад конвеєра з обробки зображень

У загальному випадку, механізм і алгоритми обробки інформації в системі відповідають загальноприйнятій конвеєру обробки зображень в системах розпізнавання з додаванням операцій, пов'язаних з ідентифікацією (рис. 4.3).

По відношенню до обробки 2D-зображень, крім збільшення обсягів обчислень в 2 рази вводиться операція, пов'язана з ідентифікацією однієї і тієї ж точки на двох зображеннях стереопари. Для систем технічного зору, крім того, необхідно визначати в режимі реального часу відстань до об'єктів, розташованих в сцені.

Введення зображень здійснюється за допомогою стандартних засобів і методів - за допомогою двох цифрових камер. В результаті виходить пара зображень. Однак, такі зображення як правило, не готові для використання в СМЗ, оскільки є викривленими за рахунок фізичних ефектів і явищ, властивих оптичним системам.

Для усунення цих ефектів, а також для вирішення завдань приведення зображення до форми, зручної для використання, наприклад, поворот і масштабування зображення, виконують операції геометричній корекції.

Після цього, виконуються операції по «синхронізації» вмісту зображень, які полягають у виділенні так званих опорних точок і визначенні їх розташування на обох зображеннях. Така операція виконується до поліпшення зображення (попередня обробка), оскільки як правило на даному етапі втрачається частина інформації внаслідок використання низькочастотної

фільтрації до вмісту обох кадрів, яка прибирає дрібні деталі зображення.

Рис. 4.3. Конвеєр перетворень в 3D-системі розпізнавання образів

Подальші завдання в процесі розпізнавання полягають у тому, щоб отримати формальний опис контурів об'єктів, представлених в зображеннях, за якими в подальшому і буде виконуватися процес розпізнавання та ідентифікації. В ході попередньої обробки намагаються виділити ділянки, на яких представлені різні об'єкти, а потім отримати інформацію про їх межах і заповненні.

Для виділення контурів знову-таки використовують методи фільтрації, але вже високочастотної для посилення перепадів яскравості, бінарізація зображень, операції стоншення і стиснення. Основне завдання даного етапу - уявити контури і скелети лініями товщиною в 1 піксель.

На наступному етапі, отримані лінії описують в деякому формальному вигляді, придатному для порівняння з шаблонами або «лінгвістичного» аналізу з метою виділення унікальних характерних ознак, за якими можна визначити, що це за об'єкти представлені на зображенні.

Особливістю рішення задач розпізнавання є обробка яскравості або колірних компонентів зображення, які є первинними джерелами інформації. Інформація про координати об'єктів може бути отримана тільки після виконання більшої частини з перерахованих етапів.

Кожен з перерахованих етапів є складним завданням, що залежить від безлічі специфічних факторів та умов зйомки. Далі розглянемо базові характеристики і особливості обробки даних в СМЗ бінокулярного типу.

4.4.2. Математичні принципи подання бінокулярного зображення

В основі технічних систем бінокулярного зору лежить принцип, що описує роботу зорових аналізаторів людини. Інша назва властивості «бінокулярне» є «стереоскопічне», тобто таке, яке дозволяє аналізувати взаємне просторове розташування предметів і людини в навколишньому світі. Його особливістю є обробка двох (бі-) зображень, які сприймаються як одне.

Це властивість пояснюється наявністю «фузіоного рефлексу» - двоїння предметів, що потрапляють на функціонально різні області сітківки ока [21] (рис. 4.4).

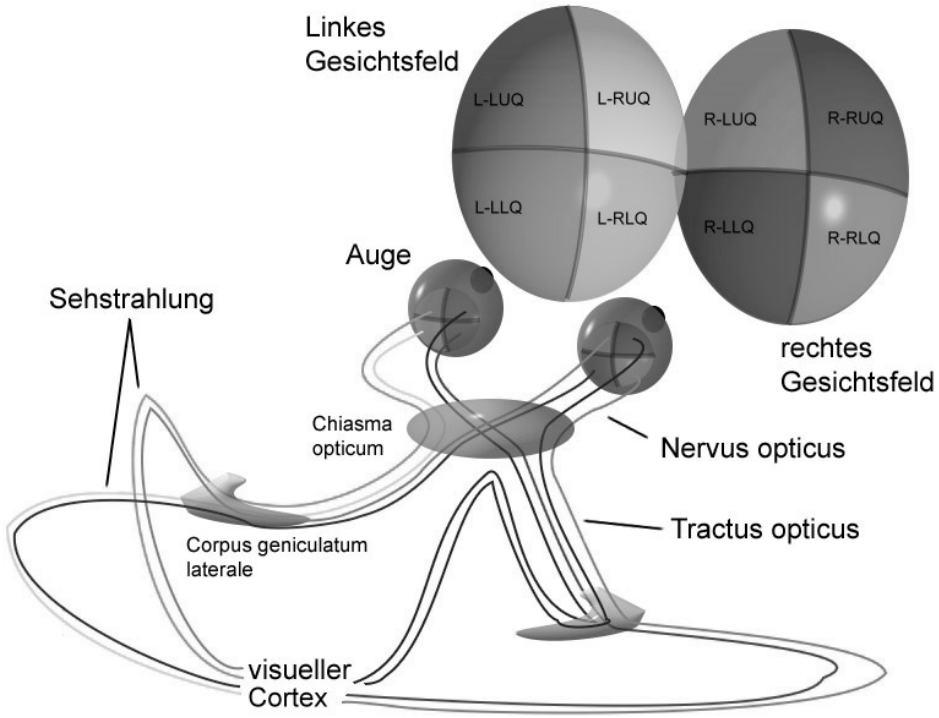


Рис. 4.4. Функціональні зв'язку системи зорового аналізатора

Інша властивість людського зору полягає в «бінокулярній діспарантності» або різного сприйняття зображення правим і лівим оком (рис. 4.5). Це властивість визначає можливість ідентифікації відстані між різними об'єктами.

Слід зазначити, що велика кількість різних систем, пов'язаних з передачею візуальних даних, використовують в більшій чи меншій мірі особливості і властивості людського зору і, тим більше, це відноситься до 3D-систем візуалізації.

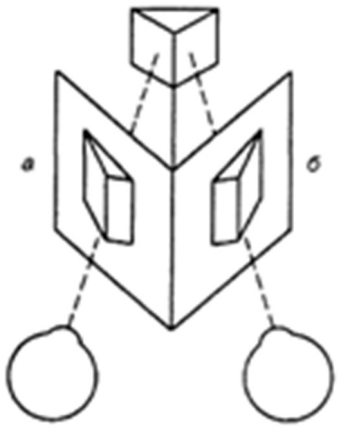


Рис. 4.5. Принцип різності сприйняття очами

З фізичної точки зору, система зорового аналізу на периферичному рівні представляється двома оптичними системами (очима), аналогами яких в техніці є оптичні датчики - цифрові камери.

Особливістю людського зору є наявність «конвергенції» - зближення точки перетину оптичних осей очей при фіксації на об'єктах, які розташовуються ближче до спостерігача. При фіксації на віддалених об'єктах оптичні осі очей практично паралельні один одному (рис. 4.6).

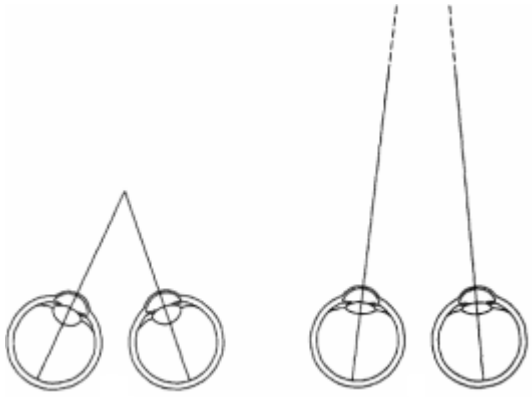


Рис. 4.6. Принцип конвергенції зору

У загальному випадку, при введенні стереоскопічних зображень в технічних системах виділяють кілька варіантів

взаємного розташування оптичних датчиків, наведених на рис. 4.7 і визначають особливості обчислення координат.

Серед представлених варіантів тільки в випадок 4.7.г відповідає «людської» конвергентної моделі зору, але в СМЗ практичного використання майже не має. Всі інші – не характерні для людського зору і, більш того, визначають деякі різновиди його дефектів, однак, для реалізації в технічних системах використовуються. Найбільш часто зустрічається використання варіанту 4.7.а.

Розглянутий випадок відповідає розміщенню двох фотодатчиків в одній площині. Лінію, на якій вони знаходяться прийнято називати «базисом» зйомки В.

Для визначення відстані до заданої точки та її координат в просторі, вся система повинна використовувати «елементи орієнтування», що визначають розташування проекційних систем оптичних елементів в просторі. Прийнято виділяти елементи внутрішнього і зовнішнього орієнтування.

Внутрішніх елементів три - фокусна відстань *f* і координати головної точки *x0, z0*. Практично, вони задають положення центру проекції знімка.

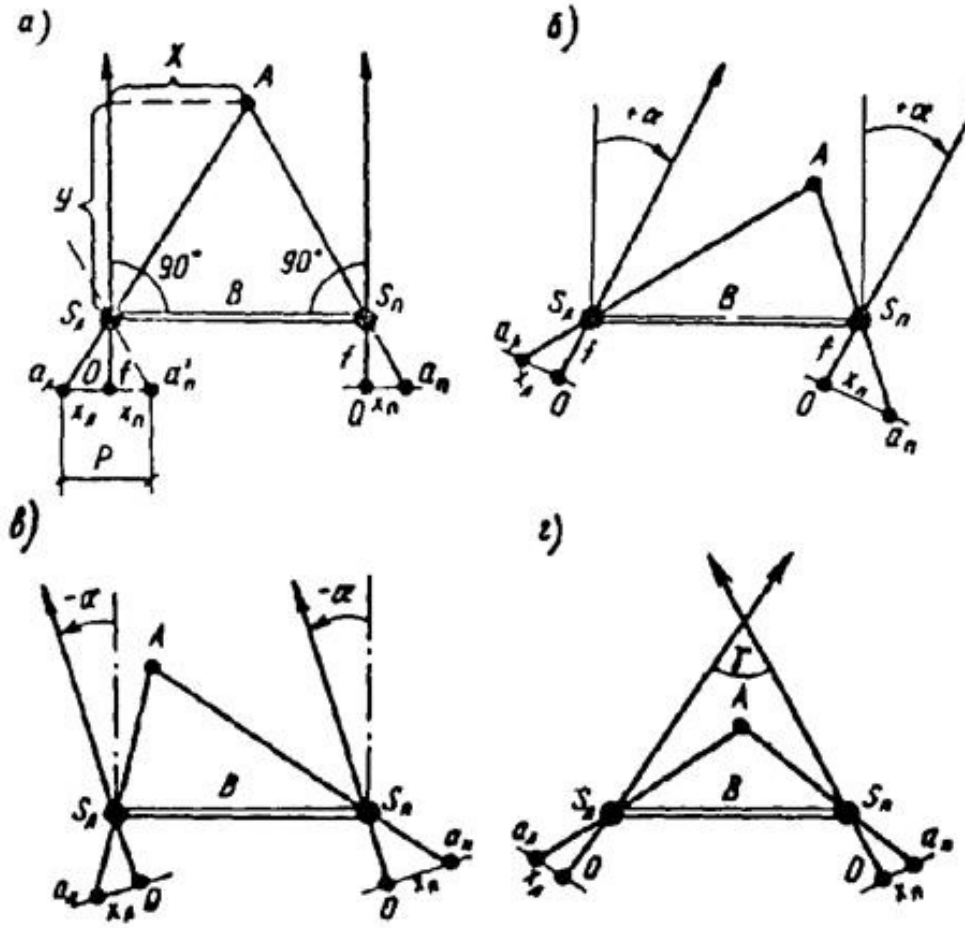


Рис. 4.7. Основні варіанти орієнтування оптичних елементів:
а) нормальний; б) і в) рівномірно відхилені; г) конвергентний.

Зовнішніх елементів кілька і їх завдання вказати на положення камер в просторовій системі координат. У практичних розрахунках може використовуватися 12 або 15 зовнішніх елементів орієнтування. Це лінійні і кутові характеристики розташування проекційної системи.

У разі нормальної зйомки, координати об'єкта (точки) визначаються по наступних співвідношеннях:

$$Y = B\frac{f}{p};\ X = B\frac{x}{p};\ Z = B\frac{z}{p}, \tag{4.1}$$

де B – базис зйомки,

f – фокусна відстань,

$p=x_л - x_п$ – поздовжній паралакс шуканої точки,

x, z – координати шуканої точки на лівому знімку.

Для випадку з рівномірно відхиленої зйомкою, базові розрахунки відповідають такими виразами:

$$Y = B\frac{f}{p}\left(\cos\alpha + \frac{x_п}{f}\sin\alpha\right);$$
$$X = B\frac{x_л}{p}\left(\cos\alpha + \frac{x_п}{f}\sin\alpha\right); \qquad (4.2)$$
$$Z = B\frac{z_л}{p}\left(\cos\alpha + \frac{x_п}{f}\sin\alpha\right),$$

де α – кут між оптичною віссю і базисом.

В даних розрахунках приймається, що центр просторової системи координат пов'язаний з лівим фотоелементом. Для переходу до реальних систем координат необхідно вводити відповідні коригування, що враховують всі елементи орієнтування [22].

Дані методи розрахунку координат використовуються для отримання точних значень. У тому випадку, коли необхідно ідентифікувати об'єкти, їх характеристики або взаємне розташування об'єктів, використовують спрощені методи для визначення глибини сканованою точки.

Є деякі інші обмеження, які слід враховувати при формуванні оптичної системи для бінокулярного зору, які детально розглядаються, наприклад в [23].

4.4.3. Геометрична корекція зображень

Геометрична корекція зображень пов'язана з необхідністю усунення ряду спотворень, що вносяться в процесі отримання зображення цифрового еквівалента сцени. В ході геометричних перетворень, для нового зображення на основі отриманого зображення необхідно отримати прямокутну матрицю з деякими

значеннями яскравості або кольору. Таким чином, вирішуються дві основні задачі - визначення нової координати для точки, представленої на оригінальному документі і визначення її яскравості (або кольору). Для тих точок нового зображення, які не мають прямого відповідності на оригінальному документі, значення яскравості (кольору) визначаються різними інтерполюючими методами.

До основних видів спотворень, що вносяться оптичною системою, відносять бочкоподібне спотворення (рис. 4.8) і бочкоподібне, які називаються спотвореннями дисторсії і відносять до ефектів аберації.

Нижче (рис. 4.9) наведено приклад [24], що представляє такий тип спотворень при зйомках камерою типу «риб'яче око», яка характеризується широким кутом захоплення зображення.

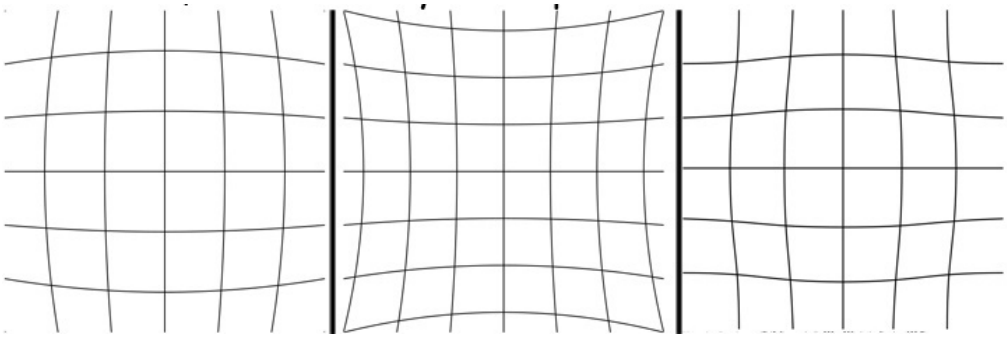


Рис. 4.8. Спотворення дисторсії – бочкоподібне, подушкоподібне та комплексне (зліва на право)

Крім цього, є ще специфічні спотворення, пов'язані з оптичними елементами - це віньетируванням і хроматичні аберації.

Віньетирування проявляється в тому, що ближче до країв зображення більш темне, ніж в центрі. Хроматичні аберації проявляються внаслідок того, що оптична система по-різному впливає на випромінювання різної довжини.

Рис. 4.9. Приклад бочкоподібного спотворення

Для усунення першого типу спотворень використовується еквалізація (кероване регулювання в різних діапазонах) гістограм зображення, суть якої полягає в тому, що по заданому закону змінюється динамічний діапазон шляхом заміни пікселів однієї яскравості на пікселі інший яскравості.

У другому випадку, процес обробки більш складний і вимагає попередніх операцій кольороподілу, виділення колірних контурів і геометричної корекції в окремих колірних шарах.

У загальному випадку, механізм геометричній корекції полягає у виконанні ряду геометричних перетворень, які описуються в комп'ютерних системах в однорідних координатах (описаний у другому розділі), властивих проективної геометрії.

4.4.4. Основи проективної геометрії

Будь-яке перетворення в проективної системі представляється операцією множення вектора вихідних координат на квадратну матрицю розмірністю 3x3 елемента для

2 мірного евклідова простору, 4x4 для 3-х мірного евклідова простору і (N + 1) x (N + 1) для N-мірного простору.

У загальному випадку, операція проектування задається співвідношенням [25] x = PX, де x - вектор однорідних координат точки в проекційної площини, X - вектор точки в однорідних координатах простору, P - матриця камери.

Матриця P знаходиться як:

$$P = KR[I|-c] = K[R|t], \qquad (4.3)$$

де K - верхня трикутна матриця внутрішніх параметрів камери розміру 3 × 3 (конкретний вид наведено нижче);
R - ортогональна матриця розміру 3 × 3, яка визначає поворот камери щодо глобальної системи координат;
I - одинична матриця розміру 3 × 3;
c - вектор координати центру камери;
t = -Rc.

У загальному випадку, матриця параметрів камери має вигляд:

$$\mathbf{K} = \begin{bmatrix} f_u & 0 & u_0 \\ 0 & f_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \qquad (4.4)$$

де fu і fv - фокусні відстані об'єктива, виражені в одиницях ширини і висоти пікселя відповідно;
(u_0,v_0) - 2D-координати головної точки.

4.4.5. Колірні перетворення у попередній обробці зображень

Попередня обробка служить для підвищення якості зображення перед подальшим аналізом. Зазвичай в неї включаються такі операції:

– корекція яскравості і контрастності по всьому полю зображення;

– колірні перетворення;

– виключення шумів;

– перетворення зображення (підкреслення перепадів).

Основним змістом попередньої обробки зображень є покращення зображення. Процедури покращення зображень зводяться до виконання комплексу операцій з метою або поліпшення візуального сприйняття зображення, або перетворення його в форму, більш зручну для візуального або машинного аналізу. У системах покращення зображення не робиться спроби наблизити відтворене зображення до деякого ідеалізованому оригіналу. У деяких випадках спотворене зображення суб'єктивно сприймається краще, ніж неспотворений оригінал. Прикладом може служити зображення з підкресленими кордонами, або видаленими шумами.

Основу операцій покращення зображень становлять колірні і контрастні перетворення.

Слід зазначити, що колірні перетворення використовуються не тільки для поліпшення зображень, але і для оптимізації обробки зображень в технічних засобах. Так, наприклад, для перегляду зображення і його виведення на жорсткий носій можуть використовуватися дві різні системи представлення кольорів - адитивна (система монітора) і субтрактивна (система кольорового принтера). При оцифруванні телевізійного сигналу (наприклад, в стандарті NTSC) для зберігання і обробки в комп'ютері (формат RGB) використовуються різні колірні моделі.

У деяких випадках колірні перетворення застосовуються з метою підвищення примітності об'єктів оператором. Зазвичай [27] розрізняють два основних види колірних перетворень - "фальшивих" і "неправдивих" кольорів.

Неприродні кольори утворюються при поелементному лінійному або нелінійному перетворенні координат кольору вихідного кольорового зображення або набору компонент спектрозонального зображення, в результаті якого виходять

координати відтвореного кольору. Це перетворення застосовують для того, щоб отримати зображення, об'єкти якого мають змінені (помилкові) кольори, що відрізняються від очікуваних. Одна з можливих цілей такого залучення уваги спостерігача. Наприклад, актуальна на сьогодні задача температурного моніторингу відвідувачів на основі поєднання інфрачервого (перетворюваного у видимий діапазон) та звічайного зображень.

Неприродні кольори можуть також застосовуватися для кращого використання можливостей зорової системи людини. Відомо, що яркісна чутливість паличок і колбочок сітківки максимальна в зеленій області видимого спектру. Таким чином, якщо об'єкт червоного кольору перефарбувати в помилковий зелений колір, його буде легше виявити.

При створенні помилкових кольорів червона (R_D), зелена (G_D) і синя (B_D) координати відтворюваних кольорів пов'язані з координатами вихідних кольорів (R_S, G_S, B_S) наступним чином:

$$\begin{vmatrix} R_D \\ G_D \\ B_D \end{vmatrix} = M \cdot \begin{vmatrix} R_S \\ G_S \\ B_S \end{vmatrix}. \qquad (4.5)$$

Природний наслідок таких перетворень - необхідність використання напівтонової палітри (true color) зображень в зв'язку з тим, що знову одержувані кольори не будуть належати похідній палітрі і будуть її збільшувати.

Перехід від адитивної системи RGB до субтрактивної системи CYM здійснюється відповідно до формули:

$$\begin{vmatrix} Y \\ I \\ Q \end{vmatrix} = \begin{vmatrix} 1 \\ 1 \\ 1 \end{vmatrix} - \begin{vmatrix} R \\ G \\ B \end{vmatrix} \cdot \qquad (4.6)$$

4.4.6. Контрастні перетворення зображень

Слабкий контраст найбільш поширений дефект фотографічних телевізійних зображень, обумовлений обмеженим діапазоном відтворюваних яркостей або нелинейностью передачі рівнів камерою. У багатьох випадках контраст можна підвищити, змінюючи яскравість кожного елемента зображення. Різні варіанти таких перетворень представлені на рис. 4.10.

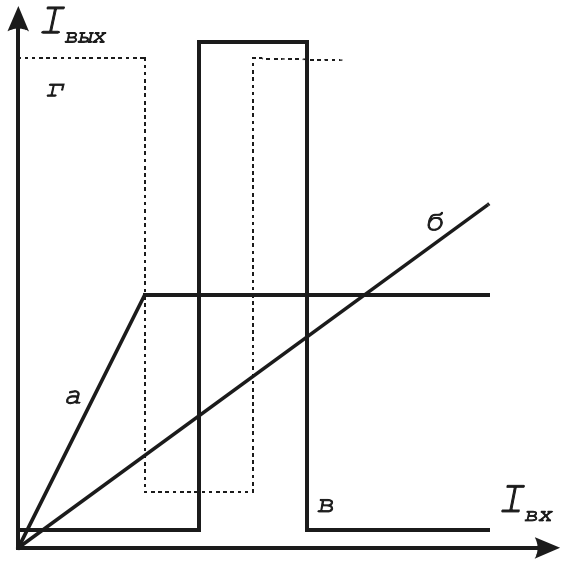


Рис. 4.10. Стандартні форми контрастних перетворень:
а) лінійне масштабування з обмеженням;
б) лінійне масштабування;
в) «прямий» зріз яскравості;
г) «зворотний» зріз яскравості.

При лінійному масштабування може використовуватися обмеження екстремальних значень яскравості обробленого зображення максимальним і (або) мінімальним порогом, що забезпечує суб'єктивно більш високу якість зображення, особливо якщо оброблене зображення містить відносно мало елементів з перевищенням рівня обмеження.

З метою отримання зображення з широким динамічним діапазоном, використовується пилкоподібне контрастне масштабування. Перетворення типу "зріз яскравості" дозволяє

виділити певний інтервал діапазону яскравості вхідного зображення. З'являється можливість відображення в якомусь яскравому кольорі (наприклад, в червоному) ділянок одноколірного зображення, яскравість яких знаходиться в довільному вузькому інтервалі, інші ділянки цього зображення можуть відображатися звичайним способом.

Реалізація таких процедур перетворення здійснюється або підпрограмами, або за допомогою функціональних таблиць перетворення, що розміщуються між кадровою пам'яттю зображення і цифро-аналоговим перетворювачем.

4.4.7. Перетворення ковзанковою фільтрацією

У великій кількості випадків можна говорити про те, що основна інформація про об'єкт міститься в його контурі, остові (кістяку), фактурі поверхні, зв'язності областей, контурів та ін. Завданням етапу препарації і є виділення цих основних ознак. Відзначимо наступні основні операції цього етапу:

– виділення перепадів;
– виділення контурів;
– бінарізація зображення;
– операції стиснення і розширення;
– виділення зв'язкових областей;
– виділення скелета (кістяка) зображення;
– сегментація зображення.

масив вхідного зображення *f(n,m)* відображається оператором *L* в масив вихідного зображення *g(n,m)* відповідно до співвідношення:

$$g(n,m) = L\{f(n,m)\} \qquad (4.7)$$

де *n, m* - параметри точок зображення в просторі сигналів.

Фільтром такої системи є співвідношення:

$$h(k,l) = L\{f_o(n,m)\} \tag{4.8}$$

де $f_o(n,m)$ - одиничний відклик

$$f_o(n,m) = \begin{vmatrix} 1, при & (n=0, m=0) \\ 0, при & інших \quad значеннях \end{vmatrix} \tag{4.9}$$

k,l - координати зсуву одиничного імпульсу.

Відповідно до цього, співвідношення (4.7) можна розглядати як згортку масиву f(n,m) початкового зображення з фільтром h(к,l). Це є ковзанковий фільтр, що являє собою вікно (маску) *h*, що рухається по вхідному зображенню і охоплює *(2c + 1) (2d + 1)* елементів цього зображення. Маска розташовується симетрично щодо фільтруємого елементу зображення, тобто *f(x,y)* знаходиться в центрі маски. Нове значення *g(x,y)* виходить в результаті перетворення з урахуванням оточення елемента. Закон перетворення задається ваговими коефіцієнтами маски за наступною схемою:

$$g(x,y) = \sum_i \sum_j f(x-i, y-j) h(i,j). \tag{4.10}$$

Підвищити рівень різкості зображення, можна посиливши скачки яскравості на контурах об'єктів. Зміна яскравості в зоні контуру, що розділяє два об'єкти можна уявити згладженої ступінчастою кривої (рис. 4.11). Яскравість змінюється не різко, а протягом кількох елементів зображення. Завдання підвищення різкості полягає в тому, щоб світлі елементи зображення зробити ще більш світлими, а темні - ще темнішими. Для цього спочатку слід відшукати зони, в яких відбуваються найбільш сильні зміни яскравості.

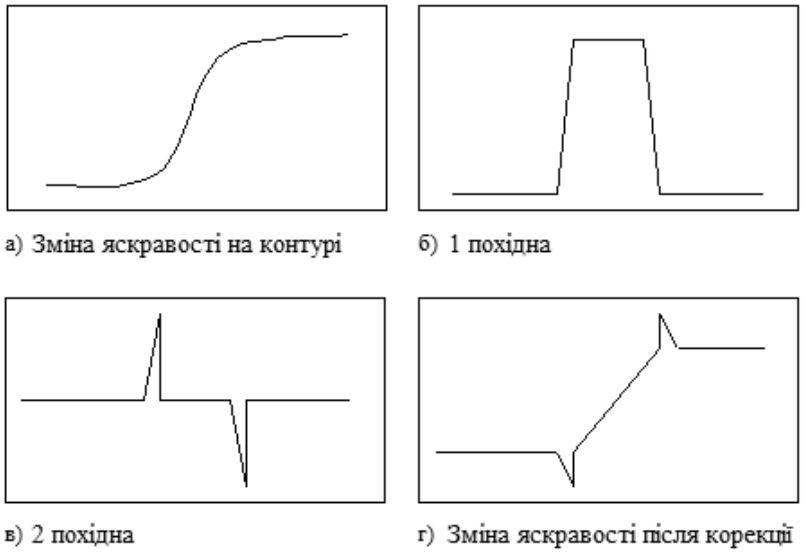


Рис. 4.11. Принцип зміни контрасту фільтрацією

Інтенсивність зміни значення функції можна виразити за допомогою її першої похідної, оскільки вона вказує крутизну графіка. Хід першої похідної показаний на графіку (рис. 4.11, б). Зміна яскравості посилюється в зоні контуру, досягає рівної ділянки та повертається до нуля. Щоб відшукати зони, в яких відбуваються найбільш сильні зміни яскравості, слід знайти другу похідну, яка має крутизну першої похідної. Як показує графік (рис. 4.11, в), найсильніші скачки яскравості припадають на початок і кінець контурного переходу. Їх потрібно посилити, для чого друга похідна інвертується і підсумовується з вихідною функцією. Результуючий графік зміни яскравості показаний на рис. 4.11,г. При цьому контур виявляється сильно підкресленим завдяки посиленій зміни яскравості на початку і кінці контуру.

Найбільшого поширення набула маска у вигляді матриці 3x3 елемента. Особливість обробки полягає в тому, що для розрахунку завжди використовуються тільки незмінні вхідні значення. Обчислені вихідні значення заносяться в нову бітову матрицю. У наведеній нижче масці, коефіцієнт X визначає вплив на відносну величину корекції контурів - який відсоток другої похідної додається до початкового зображення. Чим вище його значення, тим слабкіше ефект загострення.

$$h = \begin{vmatrix} -1 & -1 & -1 \\ -1 & X & -1 \\ -1 & -1 & -1 \end{vmatrix} \qquad (4.11)$$

Коефіцієнт X обчислюється за формулою:

$$X = int(100/S - 1 + 8), \qquad (4.12)$$

де S - це ступінь загострення, що збігається зі значенням процентної добавки другої похідної;

Int - це округлення до наступного цілочисельного значення.

Однак, застосування фільтрів надає на якість зображення і негативний вплив - збільшення шуму в зображенні. Такий фільтр відноситься до фільтрів верхніх частот і тому посилює деталі і шумів.

Особливі труднощі виникають при обробці кольорових зображень, коли в зображеннях міститься різний фоновий шум. В результаті, при виділенні контурів можуть виникати кольорові окантовки і плями.

Інший вид фільтрації - це «згладжування». Такий фільтр, зменшує різкість зображення і відноситься до класу фільтрів нижніх частот. Він пригнічує деталі зображення і завдяки цьому послаблює шум.

Ще їх називають усередненням, змазуванням або згладжуванням. Типові згорткові маски для такого фільтра представлені нижче:

$$M_1 = \begin{vmatrix} 1 & 4 & 1 \\ 4 & 16 & 4 \\ 1 & 4 & 1 \end{vmatrix} \quad (4.13)$$

$$M_2 = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 8 & 1 \\ 1 & 1 & 1 \end{vmatrix} \quad (4.14)$$

При обробці фільтром обчислюється зважене середнє сусідніх елементів зображення. Оскільки вплив елемента зображення на середнє значення зменшується зі збільшенням відстані від фокуса маски, перша маска виявляється більш дієвою завдяки тому, що елементи в діагональних напрямках враховуються з меншою вагою. Ефективне згладжування забезпечує також матриця виду:

$$M_3 = \begin{vmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{vmatrix}. \quad (4.15)$$

Однак оскільки в ці матриці всі елементи входять з рівним вагою, вони призводять до дуже значної втрати деталей зображення.

4.4.8. Алгоритми бінарізації зображення

Одним з найбільш часто зустрічаються методів бінарізації зображень є метод порогового обмеження по яскравості [27]. Метод заснований на тому, що об'єкти, що цікавлять, мають відносно однакову яскравість по відношенню до фону з іншою яскравістю. Наприклад, для друкованих провідників яскравість є відмітною ознакою, щоб визначити положення об'єкту. При цьому, якщо об'єкт має білий колір на чорному тлі або навпаки, то визначення точок об'єкта представляє собою задачу встановлення порога середньої яскравості.

Для виконання обмеження по яскравості, використовують кілька методів, в основі яких лежить статистична обробка даних.

Один з методів орієнтований на класи зображень, в яких об'єкти явно виділяються на тлі, наприклад, штрихкод. Цей метод заснований на усередненні «поперекових перерізів» профілю яскравості (рис. 4.12). Горизонтальні і вертикальні поперечні перерізи визначаються за формулами:

$$H(k)=\frac{1}{N}\sum_{j=0}^{N-1}F(j,k) \qquad (4.16)$$

$$V(j)=\frac{1}{M}\sum_{k=0}^{M-1}F(j,k) \qquad (4.17)$$

де *N, M* - розміри зображення відповідно по горизонталі і вертикалі.

Аналіз наведених алгоритмів вказує на необхідність виконання додаткових операцій статистичної обробки яскравості компонент одержуваних зображень.

Рис. 4.12. Приклад будови поперекових перерізів по зображенню.

4.4.9. Операції побудови кістяка зображення

Операції побудови кістяка зображення ставлять перед собою завдання такого перетворення зображення, в результаті якого виділені об'єкти будуть представлені об'єктами з одиничною товщиною. За допомогою стиснення область зводиться до єдиного елементу. Робота зазначених алгоритмів ґрунтується на поняттях зв'язності області.

Деякий елемент А (рис. 4.13), оточений вісьмома сусідніми елементами 0, 1, 2, 3, 4, 5, 6, 7. Елемент А має властивість S (яскравість, текстура, колір). За визначенням чотири- зв’язності (зв'язність вгору - вниз - вліво - вправо) елемент А і елементи пов'язані, якщо обидва мають властивість S. Восьми-зв'язність дозволяє пов'язувати елемент А ще і з сусідами по діагоналі.

3	2	1
4	A	0
5	6	7

Рис. 4.13. Визначення восьми-зв'язної області пікселя А

Елемент *I(i,j)* двійкового зображення 8-пов'язаний з елементом I(m,n), якщо має місце відношення:

$$\boldsymbol{max}\left(|i-m|,|j-n|\right)\le 1. \qquad (4.18)$$

8-зв'язкове оточення елемента є безліч, що складається з самого елемента і всіх його 8 найближчих сусідів.

Елемент зображення *I(i,j)* 4-пов'язаний з елементом *I(m,n)* якщо:

$$\left(\left|i-m\right|+\left|j-n\right|\right)\le 1. \qquad (4.19)$$

Загалом, два ненульових елемента називаються зв'язковими, якщо між цими двома елементами існує безперервна послідовність ненульових 8-зв'язкових елементів.

Стиснення є операцію перетворення зображення, яка полягає в тому, то кожному ненульовому елементу присвоюється значення 0, якщо:

а) при 4-зв'язковому стисненні хоча б один з 4-зв'язкових сусіда мають значення 0:

$$A=\overline{\left(0\vee 2\vee 4\vee 6\right)} \qquad (4.20)$$

б) при 8-зв'язковому стисненні хоча б один з 8-зв'язкових сусіда мають значення 0:

$$A=\overline{\left(0\vee 1\vee 2\vee 3\vee 4\vee 5\vee 6\vee 7\right)} \qquad (4.21)$$

Операція стиснення проводиться за правилами ковзанкової фільтрації: знову набуті значення елементів не використовуються при поточному проході по зображенню.

В результаті стиснення розміри об'єкта зменшуються, можуть відбутися розриви топології, а при багаторазовому повторенні об'єкт взагалі зникне. Тому, робота алгоритму здійснюється в інтерактивному режимі з суб'єктивною оцінкою отриманого результату оператором.

За допомогою операції стоншення [30] область наводиться до мінімальної ширини поперечного перерізу з виконанням умов:

- після перетворення все лінії мають товщину в один елемент;

– після перетворення не порушується топологія символу, тобто лініях і вузлів вихідного зображення відповідають лінії і вузли перетвореного зображення;
– перетворення чи не порушує основних розмірів символу.

Особливістю є те, що неоднаковість ширини вихідних ліній відноситься до систематичних відхилень, а до числа відхилень, що роблять негативний вплив, відносять виступи і западини по довжині ліній - «бахрома».

Іншим видом відхилень є порожнечі (елементи всередині ліній), які можуть призводити до розривів в топології, тому вони повинні бути усунуті.

Загальний алгоритм стоншення складається в аналізі восьми- зв'язних сусідів, знаходженні крайніх правих, лівих, верхніх і нижніх елементів і винесенні рішення про можливість їх видалення.

Елемент А вважається крайнім зверху, якщо він і елемент 6 зафарбовані, а елемент 2 не зафарбований, що в булевом описі задається функцією:

$$g_в = \bar{2} \wedge 6 \wedge A = 1. \qquad (4.22)$$

Розглянутий елемент може бути видалений, якщо не порушується зв'язність між його сусідами, що визначається за функцією:

$$f_в = \left(\bar{1} \wedge 4\right) \vee \left(\bar{3} \wedge 0\right) \vee \left(0 \wedge 4\right) \qquad (4.23)$$

Таким чином, елементу А можна надати значення «0», якщо дорівнює одиниці добуток $g_в f_в$.

За аналогією, виноситься висновок про крайні лівій, нижньої і правої точках:

$$g_л = \overline{4} \wedge 0 \wedge A = 1 \quad (4.24)$$

$$f_л = \left(\overline{3} \wedge 6\right) \vee \left(\overline{5} \wedge 2\right) \vee \left(2 \wedge 6\right) = 1 \quad (4.25)$$

$$g_н = \overline{6} \wedge 2 \wedge A = 1 \quad (4.26)$$

$$f_н = \left(\overline{5} \wedge 0\right) \vee \left(\overline{7} \wedge 4\right) \vee \left(4 \wedge 0\right) = 1 \quad (4.27)$$

$$g_п = \overline{0} \wedge 4 \wedge A = 1 \quad (4.28)$$

$$f_п = \left(\overline{7} \wedge 2\right) \vee \left(\overline{1} \wedge 6\right) \vee \left(6 \wedge 2\right) = 1 \quad (4.29)$$

Результуючі значення отримують з використанням функцій:

$$A_в = A \oplus g_в \quad (4.30)$$

$$A_л = A \oplus g_л \quad (4.31)$$

$$A_н = A \oplus g_н \quad (4.32)$$

$$A_п = A \oplus g_п \quad (4.33)$$

Робота алгоритму полягає в послідовному стиранні на кожному циклі всіх крайніх елементів до отримання ліній одиничної товщини.

Шуми в вигляді «бахроми» можуть з'являтися як на оригінальному документі, так і в результаті виконання операції стоншення. Для підвищення надійності розпізнавання, рекомендується перед циклом стоншення виконувати операцію корекції випадкових відхилень. Послідовність дій при цьому наступна:

- заливка порожнин;
- стирання кінцевих точок (видалення «бахроми»);
- стоншення.

При заповненні пустот використовують мажоритарний алгоритм - НЕ зафарбований елемент фарбується, якщо не менше п'яти з восьми його сусідів зафарбовані.

Для стирання кінцевих точок необхідно виконати перетворення відповідно до формулами (верхніх, лівих, нижніх та правих):

$$A_в = A \oplus \left(A \wedge \bar{0} \wedge \bar{1} \wedge \bar{2} \wedge \bar{3} \wedge \bar{4}\right)\left(\bar{5} \vee 6\right) \quad (4.34)$$

$$A_л = A \oplus \left(A \wedge \bar{2} \wedge \bar{3} \wedge \bar{4} \wedge \bar{5} \wedge \bar{6}\right)\left(\bar{7} \vee 0\right) \quad (4.35)$$

$$A_н = A \oplus \left(A \wedge \bar{4} \wedge \bar{5} \wedge \bar{6} \wedge \bar{7} \wedge \bar{0}\right)\left(\bar{1} \vee 2\right) \quad (4.36)$$

$$A_п = A \oplus \left(A \wedge \bar{6} \wedge \bar{7} \wedge \bar{0} \wedge \bar{1} \wedge \bar{2}\right)\left(\bar{3} \vee 4\right) \quad (4.37)$$

При видаленні «бахроми» стирання всіх точок можна здійснювати за один прохід по узагальненій формулі:

$$A_{в,л,н,п} = A\left(\begin{array}{l}\left(0 \vee 1 \vee 2 \vee 3 \vee 4 \vee 5 \vee \bar{6}\right) \wedge \left(2 \vee 3 \vee 4 \vee 5 \vee 6 \vee 7 \vee \bar{0}\right) \wedge \\ \left(4 \vee 5 \vee 6 \vee 7 \vee 0 \vee 1 \vee \bar{2}\right) \wedge \left(6 \vee 7 \vee 0 \vee 1 \vee 2 \vee 3 \vee \bar{4}\right)\end{array}\right) \quad (4.38)$$

4.4.10. Кольорові контури

Існує кілька визначень колірних контурів.

1. Колірний контур визначається тільки як область з перепадом яскравості, ко-торая визначається як зважена сума:

$$Y=\alpha_1 T_1+\alpha_2 T_2+\alpha_3 T_3, \quad (4.39)$$

де αi - вагові коефіцієнти; Ti - координати кольору.

2. Досліджується кожна з 3х компонент зображення Ti, і колірної контур існує, якщо перепад яскравості виявлений у всіх трьох компонентах.

3. Контур існує, якщо відстань між колірними векторами двох осередків перевищує певний поріг h.

В останніх двох випадках результат в значній мірі залежить від обраної системи колірних координат.

4.4.11. Ефективність алгоритмів виявлення перепадів

При виборі критеріїв ефективності для детекторів роблять відмінність між обов'язкової і допоміжною інформацією.

Обов'язкова - положення перепаду, висота і крутизна.

Корисна - достовірність рішення про перепаді.

Існують наступні основні види помилок (рис. 4.14):

- Пропуск істинного перепаду.
- Помилка у визначенні положення перепаду.
- Прийняття шумів за перепад.

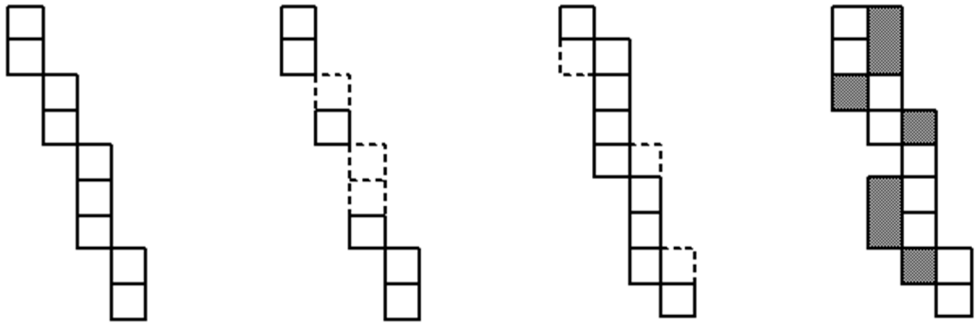


Рис. 4.14. Варіанти отриманих контурів (зліва на право): ідеальний, дроблений, зі зміщенням, змазаний.

Точність положення перепаду можна оцінити величиною:

$$R = \frac{1}{T_N} \sum_{i=1}^{I_A} \frac{1}{1 + \alpha \cdot d^2} \qquad (4.40)$$

де $I_N = \max(I_I, I_A)$;

I_I - число точок в ідеальному перепаді;

I_A - число точок в реальному перепаді;

α - масштабний коефіцієнт;

d - відстань між точкою дійсного перепаду і лінією, що складається з точок ідеального перепаду, вимірюється по нормалі до цієї лінії.

R нормалізовано так, що R = 1 для точок точно виділеного перепаду.

Масштабний множник можна підібрати таким чином, щоб встановлювати штрафи для перепадів, справжній стан яких відрізняється.

Множник 1 / TN забезпечує величину штрафу за змащене або розбите зображення-ня. Наприклад, якщо α = 1/9, то при

виявленні вертикального перепаду, що відстоїть на один елемент щодо істинного, R = 0.90. Зрушення на два елементи призводить до значення R = 0.69. Зміщений контур шириною в 3 елементи зображення, центр якого збігається з центром істинного вертикального перепаду, дає значення R = 0,93, при ширині в 5 елементів - R = 0.84.

4.4.12. Узагальнення з математичних основ

Аналіз наведених і далеко не всіх математичних співвідношень, що описують процеси обробки даних, одержуваних з фотокамер при вирішенні задач розпізнавання образів в системах машинного зору вказує на їх високу складність і трудомісткість.

Це пояснює, зокрема, виробництво спеціалізованих контролерів, орієнтованих на рішення вузькоспеціалізованих завдань в конкретних виробничих умовах, що призводить до деякого зниження обчислювальної навантаження.

Така висока складність пояснює і необхідність використання в системах реального часу елементів систем штучного інтелекту, що знайшли втілення в розробках провідних світових виробників.

Наведена інформація свідчить про велику кількість труднощів, які необхідно вирішити при реалізації високоефективної системи технічного зору і особливо з підтримкою стерео-зору. Труднощі виникають різного роду:

- технічні, пов'язані з необхідністю наявності компактної оптичної системи хорошої якості і мінімальними спотвореннями, що вносяться в одержувані зображення, високою продуктивністю обчислювального ядра і пропускної здатності інтерфейсних каналів;
- алгоритмічні, пов'язані з необхідністю реалізації ефективних алгоритмів обробки даних і високою складністю конвеєра обробки інформації;
- механічні, пов'язані з тим, що платформа оптичної системи повинна бути, як правило, рухомого

(мобільного), мати досить велику базу для підвищення точності розпізнавання, досить жорсткою для збереження сталості параметрів.

Однак, не дивлячись на труднощі, що розвиваються технології роблять раніше неможливі рішення дешевими і максимально наближеними до користувача - починаючи від вбудованих функцій виявлення обличчя і усмішки у фотоапаратах, до персональних систем комп'ютерного зору типу «OpenMV» і вбудованих систем автоматичного паркування автомобіля, до автономних безпілотних систем.

Особливу роль в процесі розвитку систем технічного зору грають створення систем штучного інтелекту, технології спеціалізованих і реконфігурованих логічних і процесорних елементів, створення компактних високошвидкісних фотографічних матриць високої роздільної здатності та перспективні хмарні технології і технології і датчики Інтернету речей.

Тема 4.5. Фільтрація зображень просторовими ковзанковими фільтрами

Метою роботи є ознайомлення з принципами обробки зображень, які розглянуто у п.4.4.7.

4.5.1. Похідні дані для виконання роботи

Для виконання роботи, розглянемо декілька видів фільтрів та механізми їх дії.

Рекурсивними фільтрами першого роду будуть такі фільтри, вихід Y яких формується перемножуванням вагових множників A з елементами зображення X. Для прикладу розглянемо фільтри низьких частот:

$$A_1 = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}, A_2 = \frac{1}{10}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 2 & 1 \\ 1 & 1 & 1 \end{bmatrix}, A_3 = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad (4.41)$$

Фільтром низьких частот користуються часто для того, щоб подавити шум в зображенні, зробити його менш різким. Використовуючи фільтр A_3, одержуватимемо зображення Y таким чином:

$$g_{i,j} = \frac{\left(f_{i-1,j-1} + 2f_{i-1,j} + f_{i-1,j+1} + 2f_{i,j-1} + 4f_{i,j} + 2f_{i,j+1} + f_{i+1,j-1} + 2f_{i+1,j} + f_{i+1,j+1}\right)}{16}. \quad (4.42)$$

Розглянемо і інші фільтри. Високочастотні (для підкреслення різкості зображення):

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 5 & -2 \\ 1 & -2 & 1 \end{bmatrix} \begin{bmatrix} -1 & -2 & -1 \\ -2 & 13 & -2 \\ -1 & -2 & -1 \end{bmatrix} \quad (4.43)$$

Для підкреслення орієнтації:

$$\begin{bmatrix} 1 & 1 & 1 \\ 1 & -2 & 1 \\ -1 & -1 & -1 \end{bmatrix}, \begin{bmatrix} 1 & 1 & 1 \\ -1 & -2 & 1 \\ -1 & -1 & 1 \end{bmatrix} i \quad m.д. \quad (4.44)$$

Підкреслення без урахування орієнтації (фільтри Лапласа):

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad (4.45)$$

4.5.2. Порядок виконання роботи

Для дослідження роботи фільтрів потрібно вирішити наступні задачі:

1) розробити відповідні структури даних, класи та інтерфейсні елементи для виконання операції ковзанкової фільтрації;
2) інтегрувати розроблені елементи до системи, що здійснює візуалізацію та перетворення зображення, застосовуючи відповідні механізми до складової яскравості;
3) дослідити вплив фільтрів різного типу на зображення та зафіксувати отримані результати;
4) проаналізувати та пояснити отримані результати

Тема 4.6. Медіанна фільтрація

4.6.1. Індивідуальні завдання для виконання роботи

Вкажіть, де на Вашу думку знаходиться в зазначеній послідовності імпульсні перешкоди, визначте довжину медіанного лінійного фільтру для усунення імпульсних перешкод для зазначених сигналів (табл. 4.1). Пояснити, на підставі чого отримані такі результати. Вкажіть, якими альтернативними способами можна домогтися вирішення поставленого завдання.

4.6.2. Порядок виконання роботи і приклад розрахунку

Для виконання даного завдання необхідно ознайомитися з матеріалами, викладеними в конспекті лекцій в п. 11.1.4, стор. 76-80, медіанна фільтрація.

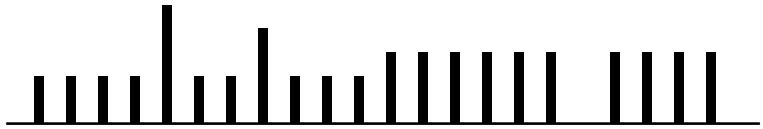
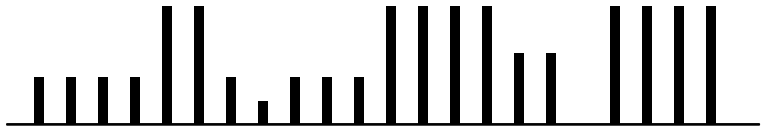
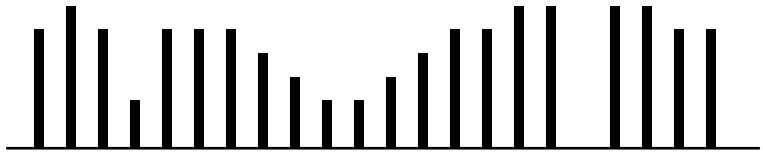
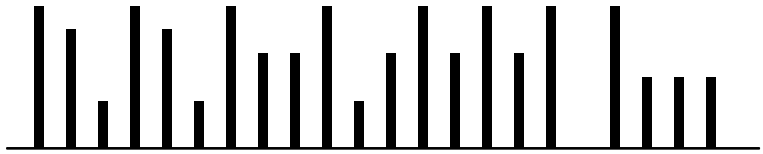
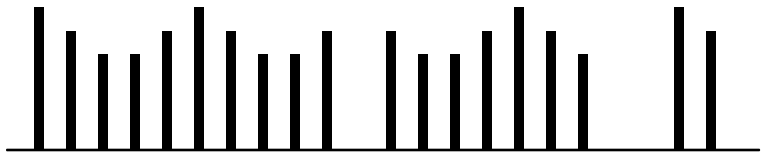
Методика розрахунку полягає в наступному:

– абстраговані дані, представлені в таблиці у вигляді графіка стовпчастих значень еквівалентних яркостей в скан-рядку, необхідно замінити наборами числових значень, наприклад, для варіанта 4 будемо використовувати наступні значення яскравості (з 0-ій

позиції): 100, 80, 60, 60, 80, 100, 80, 60, 60, 80, 0, 80, 60, 60, 80, 100, 80, 60, 0, 0, 100, 80;

Таблиця 4.1

Тестові послідовності для медіанної фільтрації

Варіант	Послідовність
0	
1	
2	
3	
4	

- наступний етап полягає у визначенні довжини імпульсної перешкоди, що впливає на сигнал, яка виходячи з регулярності наведених даних, може бути до 2-х дискретов максимум (два поспіль нуля);
- визначається довжина медіанного фільтра, необхідного для усунення впливу імпульсної перешкоди, яка в нашому випадку буде дорівнює L = 2x2 + 1 = 5;

– вихідну послідовність обробляємо медіанного фільтром з довжиною 5 елементів, в результаті чого отримуємо наступні результати
100, 80, 80 (60), 80 (60), 80, 80 (100), 80, 80 (60), 60, 60 (80), 60 (0), 60 (80), 60, 80 (60), 80, 80 (100), 80, 60, 60 (0), 60 (0), 100, 80.

Отримана після обробки послідовність матиме вигляд, представлений нижче на рис. 4.15. У розрахунках вище значення, які зазнали змін, наведені в дужках.

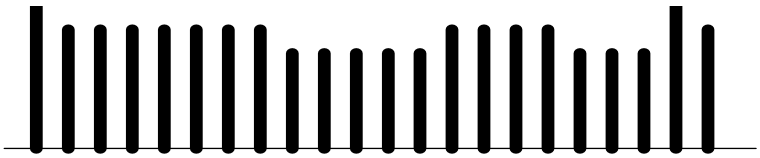


Рис. 4.15. Результат роботи медіанного фільтру

Тема 4.7. Застосування перетворення контрасту

4.7.1. Індивідуальні завдання до роботи

Покажіть структуру засобів для реалізації операції контрастування і побудуйте діаграми вмісту функціональних таблиць перетворення для заданих умов (табл. 4.2). Вихідне зображення виводиться на кольоровий монітор.

4.7.2. Порядок роботи та приклад виконання

Відповідно до завдання (див. П.1.1), необхідно вирішити варіант 0.

Відповідно до теоретичних матеріалів, методика рішення задачі полягає в тому, що б визначити структурну схему технічних засобів для реалізації заданих перетворень і визначити (в графічній формі) вміст функціональних таблиць для червоного, зеленого і синього кольорових складових.

Для виконання розрахунків і візуалізації графіків скористаємося пакетом MathCad, зміст лістингу коду в якому наведено нижче (рис. 4.16).

Таблиця 4.2

Варіанти для реалізації контрастних перетворень

Варіант	Характер зображення	Характеристики перетворення
0	Монохромне	Пікселі в діапазоні яскравості 10-25 виділити пурпуровим кольором, пропорційним вихідної інтенсивності I. У діапазоні 50-200 замінити пікселями, у яких R = 0,7 * I але не перевищує значення 100, G = 0.3 * I, B = I, інші пікселі «прибираються» з зображення.
1	Монохромне	Пікселі в діапазоні яскравості 0-25 виділити жовтим кольором максимальної інтенсивності. У діапазоні 50-75 замінити пікселями, у яких R = 0.7 * I, G = 0.9 * I, B = 0.5I, інші пікселі зберігають вихідні значення яскравості.
2	Монохромне	Динамічний діапазон для червоної складової в 2.5 рази менше від початкового із середнім значенням 120, для зеленої складової зворотна шкала яскравості в діапазоні від 100 до 200, динамічний діапазон для синьої складової збільшений в 4 рази.
3	Кольорове	Перетворити у монохромне

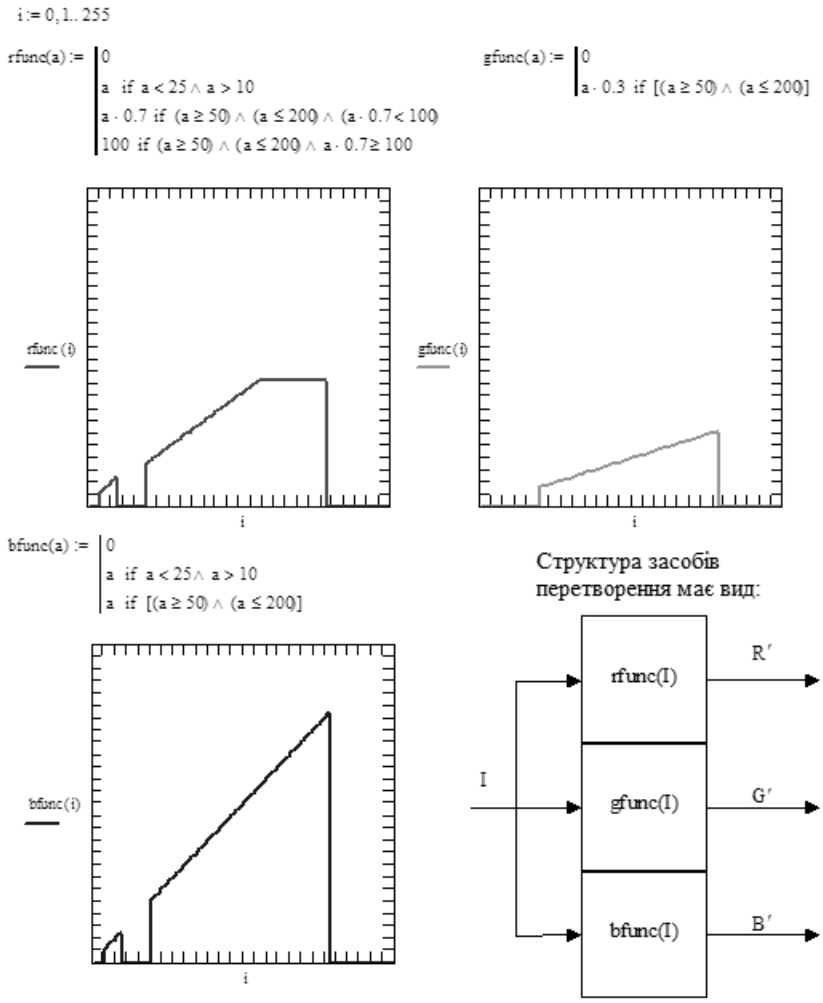


Рис. 4.16. Приклад розрахунку контрастних перетворень

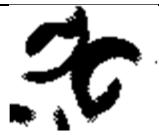
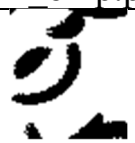
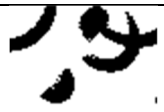
Тема 4.8. Обробка бінарізованих зображень

4.8.1. Індивідуальні завдання до виконання роботи

Виконайте 3 цикли обробки фрагменту бінарного зображення (відповідно до варіанту з табл. 4.3) операціями стиснення, стоншення і видалення «бахроми».

Таблиця 4.3

Варіанти зображень для обробки

Вариант	Изображение
0	 - 70x55 точок. Вміст в файлі 0_bmp.prn
1	 - 74x62 точки. Вміст в файлі 1_bmp.prn
2	 - 58x65 точок. Вміст в файлі 2_bmp.prn
3	 - 73x57 точок. Вміст в файлі 3_bmp.prn
4	 - 40x53 точки. Вміст в файлі 4_bmp.prn
5	 - 62x61 точку. Вміст в файлі 5_bmp.prn

4.8.2. Порядок виконання роботи та приклад розрахунку

Методика виконання роботи полягає в тому, що б реалізувати алгоритми стиснення і алгоритми стоншення, наведені у п. 4.4.9, а так само реалізувати три ітерації зазначених перетворень над заданим двійкового масивом.

Приклад виконання. Відповідно до завдання, необхідно вирішити варіант 4. Для вирішення скористаємося пакетом MathCad. Лістинг коду для проведення розрахунків наведено нижче (рис. 4.17).

$$\text{Norm}(Mt) := \left| \begin{array}{l} \text{for } j \in 0..(\text{cols}(Mt) - 1) \\ \quad \text{for } i \in 0..(\text{rows}(Mt) - 1) \\ \quad\quad R_{i,j} \leftarrow \frac{Mt_{i,j}}{255} \\ R \end{array} \right.$$

Визначемо функцію стиснення:

$$\text{szhatije}(M) := \left| \begin{array}{l} R \leftarrow M \\ \text{for } j \in 1..(\text{cols}(M) - 2) \\ \quad \text{for } i \in 1..(\text{rows}(M) - 2) \\ \quad\quad R_{i,j} \leftarrow \left(M_{i-1,j-1} \vee M_{i-1,j} \vee M_{i-1,j+1} \vee M_{i,j-1} \vee M_{i,j+1} \vee M_{i+1,j-1} \vee M_{i+1,j} \vee M_{i+1,j+1}\right) \\ R \end{array} \right.$$

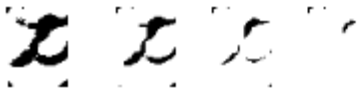
Читаємо та бінарізуємо вхідні данні:

Array := READPRN("4_bmp.prn")

Array := Norm(Array)

A1 := szhatije(Array) A2 := szhatije(A1) A3 := szhatije(A2)

Вхідні данні та результати обробки:



Array·255 A1·255 A2·255 A3·255

Рис. 4.17. Обробка бінарного зображення з метою стиснення

Література до 4 розділу

1. Machine vision. Article from Wikipedia, the free encyclopedia [Електронний ресурс]: – Режим доступу: https://en.wikipedia.org/wiki/Machine_vision.
2. Машинное зрение. Материалы Википедии, свободной энциклопедии. [Електронний ресурс]: – Режим доступу: https://ru.wikipedia.org/wiki/Машинное_зрение.
3. Компьютерное зрение. Материалы Википедии, свободной энциклопедии [Електронний ресурс]: – Режим доступу: https://ru.wikipedia.org/wiki/Компьютерное_зрение.
4. Computer vision. Article from Wikipedia, the free encyclopedia [Електронний ресурс]: – Режим доступу: https://en.wikipedia.org/wiki/Computer_vision.
5. Definition – What does Machine Vision System (MVS) mean? [Електронний ресурс]: – Режим доступу: https://www.techopedia.com/definition/30414/machine-vision-system-mvs.
6. Машины обретают зрение [Електронний ресурс]: – Режим доступу: https://www.osp.ru/lan/2015/07-08/13046648/.
7. Бондаренко М.А. Алгоритм совмещения сенсорной и синтезированной видеоинформации в авиационной системе комбинированного видения // Кибернетика и программирование. № 1. 2016. С. 236-257. DOI: 10.7256/2306-4196.2016.1.17770 (http://e-notabene.ru/kp/article_17770.html).
8. Особенности и преимущества системы технического зрения [Електронний ресурс]: – Режим доступу: http://www.techtrends.ru/techdept/techarticles/sistemy_tehnicheskogo_zreniya.php.
9. Intel Unveils Neural Compute Engine in Movidius Myriad X VPU to Unleash AI at the Edge [Електронний ресурс]: – Режим

доступу: https://newsroom.intel.com/news/intel-unveils-neural-compute-engine-movidius-myriad-x-vpu-unleash-ai-edge/#gs.o3h1bl.
10. The Evolution of EyeQ [Електронний ресурс]: – Режим доступу: https://www.mobileye.com/our-technology/evolution-eyeq-chip.
11. Система технического зрения [Електронний ресурс]: – Режим доступу: http://www.promautomatic.ru/tehnicheskoe-zrenie.html.
12. Системы машинного зрения. История, примеры, планы. [Електронний ресурс]: – Режим доступу: https://robotics.ua/shows/modernity/5844-sistemy_mashinnogo_zreniya_istoriya_primery_plany.
13. Kawasaki Vision System. Cat. No. 3L1805 Apr. ’18. [Електронний ресурс]: – Режим доступу: https://robotics.kawasaki.com/userAssets1/productPDF/Europe-Africa/Kawasaki_VisionSystem_brochure.pdf.
14. FANUC Corporation. Официальный сайт компании. [Електронний ресурс]: – Режим доступу: https://www.fanuc.co.jp/eindex.html.
15. Датчики и системы технического зрения Omron. [Електронний ресурс]: – Режим доступу: https://www.compel.ru/lib/54337.
16. Лаборатория Цифрового зрения. Решения. (Официальный сайт компании) [Електронний ресурс]: – Режим доступу: https://divisionlabs.com/resheniya/.
17. Модуль Vision Development. [Електронний ресурс]: – Режим доступу: https://www.ni.com/ru-ru/shop/data-acquisition-and-control/add-ons-for-data-acquisition-and-control/what-is-vision-development-module.html.
18. Зрение машинное, техническое или компьютерное? [Електронний ресурс]: – Режим доступу:

https://www.vitec.ru/znaniya/articles/prakticheskie-voprosi-primenenia-teh-zrenia/.
19. OpenCV modules [Електронний ресурс]: – Режим доступу: http://docs.opencv.org/trunk/.
20. Комендатенко С.Д., Когочев А.Ю. Система определения расстояния на основе двух широкоугольных видеокамер // Современные научные исследования и инновации. 2017. № 11 [Електронний ресурс]: – Режим доступу: http://web.snauka.ru/issues/2017/11/84833.
21. Зрение человека. Материалы Википедии, свободной энциклопедии [Електронний ресурс]: – Режим доступу: https://ru.wikipedia.org/wiki/Зрение_человека.
22. Руководство по применению фотограмметрических методов. [Електронний ресурс]: – Режим доступу: https://meganorm.ru/Data2/1/4294845/4294845419.htm.
23. Геометрия стереофотосъёмки [Електронний ресурс]: – Режим доступу: https://www.ixbt.com/digimage/stereogeometry.shtml.
24. Оптические аберрации (искажения) зрительной системы человека. [Електронний ресурс]: – Режим доступу: https://sovkalmykia.ru/lentochnyjj-fundament/opticheskie-aberracii-iskazheniya-zritelnoi-sistemy-cheloveka.html.
25. Основы стереозрения [Електронний ресурс]: – Режим доступу: https://habr.com/ru/post/130300/.
26. Холопов, И.С. Алгоритм коррекции проективных искажений при маловысотной съёмке / И.С. Холопов // Компьютерная оптика. – 2017. – Т. 41, № 2. – С. 284-290. – DOI: 10.18287/0134-2452-2017-41-2-284-290.
27. Прэтт У. Цифровая обработка изображений: Пер. с англ. / У. Прэтт // – М.: Мир, 1982. Кн. 2.
28. Быков Р.Е., Гуревич С.Б. Анализ и обработка цветных и объемных изображений. / Р.Е. Быков, С.Б. Гуревич // М.: Радио и Связь, 1984 г. – 248 с., ил.

29. Узилевский В.А. Передача, обработка и воспроизведение цветных изображений. / В.А. Узилевский // М.: Радио и Связь, 1981 г. – 216 с., ил.
30. Бутаков Е.А., Островский В.И., Фадеев И.Л. Обработка изображений на ЭВМ / Е.А. Бутаков, В.И. Островский, И.Л. Фадеев // – М.: Радио и связь, 1987. – 240 с.: ил.
31. OpenMV – "Ардуино" для машинного зрения [Електронний ресурс]: – Режим доступу: https://mysku.ru/blog/china-stores/72445.html.
32. Разработка нейроуправляемого автомобиля для мобилизации людей с двигательным дефицитом – нейромобиля [Електронний ресурс]: – Режим доступу: http://www.stm-journal.ru/ru/numbers/2018/4/1497/html.
33. Introducing Myriad X: Unleashing AI at the Edge. [Електронний ресурс]: – Режим доступу: https://newsroom.intel.com/editorials/introducing-myriad-x-unleashing-ai-at-the-edge/#gs.owkeej.

Смолій В.В., Савицька Я.А.,
Місюра М.Д., Шкарупило В.В.

СИСТЕМИ ВІЗУАЛІЗАЦІЇ ТА РОЗПІЗНАВАННЯ ОБРАЗІВ

Навчальний посібник

Видавець ФОП Ямчинський О.В.
03150, Київ, вул. Предславинська, 28
Свідоцтво про внесення до Державного реєстру
суб'єкта видавничої справи ДК № 6554 від 26.12.2018 р.

Формат 60×84/16. Наклад 300 пр. Ум. друк. арк. 14,8. Зам. № 143.

Виготовлювач ТОВ «ЦП «КОМПРИНТ»
03150, Київ, вул. Предславинська, 28
Свідоцтво про внесення до Державного реєстру
суб'єкта видавничої справи ДК № 4131 від 04.08.2011 р.