МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
імені ВОЛОДИМИРА ДАЛЯ

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять з дисципліни
"ПРОГРАМНІ ЗАСОБИ ОБРОБКИ ЗОБРАЖЕНЬ"
(для здобувачів вищої освіти за спеціальністю 122 "Комп'ютерні науки")
(Електронне видання)

ЗАТВЕРДЖЕНО
на засіданні кафедри
комп'ютерних наук та інженерії
Протокол № 2 від 04.10.2021 р.

Сєвєродонецьк 2021

УДК 004.02

Методичні вказівки до практичних занять з дисципліни "Програмні засоби обробки зображень" (для здобувачів вищої освіти спеціальності 122 "Комп'ютерні науки") / Уклад. : Л.О. Шумова, В.М. Барбарук – Сєвєродонецьк: вид-во СНУ ім. В. Даля, 2021. – 71 с.

Методичний матеріал підготовлено відповідно до робочої програми з дисципліни "Програмні засоби обробки зображень" та Освітньо-професійної програми підготовки магістрів за спеціальністю 122 "Комп'ютерні науки".

Методичні вказівки містять необхідні теоретичні й практичні матеріали для набуття практичних знань та навичок роботи з програмними засобами обробки зображень та підготовки здобувачів вищої освіти з питань розробки програмних засобів обробки зображень.

Укладачі: Л.О. Шумова, к.т.н., доц.,
В.М. Барбарук, к.т.н., доц.

Рецензент Є.В. Щербаков, к.т.н., доц.

ЗМІСТ

ВСТУП

"Програмні засоби обробки зображень" – це дисципліна, у якій вивчаються принципи обробки графічної інформації в комп'ютерних системах у процесі її покращення або розпізнавання. Розглядаються методи попередньої підготовки зображень, сегментації і фільтрації зображень, способи пошуку меж на зображенні та виявлення об'єктів на зображенні.

Мета дисципліни "Програмні засоби обробки зображень" – розвиток інженерного мислення на засадах вивчення базових положень, алгоритмів та засобів обробки графічної інформації, забезпечення майбутнім спеціалістам достатнього рівня знань методів та засобів обробки графічних даних, необхідних при проектуванні сучасних систем з аналізу візуальної інформації.

Завдання дисципліни "Програмні засоби обробки зображень" – вивчення принципів та набуття навичок обробки графічної інформації в комп'ютерних системах у процесі її покращення або розпізнавання.

Метою практичних занять за дисципліною є набуття знань та навичок роботи в середовищі готових програмних продуктів, закріплення теоретичних знань, практична підготовка здобувачів вищої освіти з питань розробки програмних засобів обробки зображень.

Для самостійного вивчення матеріалів курсу студенти використовують методичні матеріали, що розміщені на сайті Центру електронних навчальних матеріалів інституту, доступ до якого провадиться через Інтернет (сайт: http://moodle.snu.edu.ua/).

1 ПРАКТИЧНЕ ЗАНЯТТЯ №1
«Сучасні програмні засоби обробки зображень»

1.1 Мета і задачі заняття

Практичне заняття проводиться у формі семінару дослідницького типу з тематикою з окремих проблем курсу для поглибленого їх опрацювання. Студенти готують невеликі доповіді, які обговорюють учасники семінару.

Семінар призначений для поглибленого вивчення дисципліни, опанування методології наукового пізнання, розвитку у студентів культури наукового мислення.

Основною метою семінару є систематизація та узагальнення знань з теми «Сучасні програмні засоби обробки зображень», формування умінь працювати з додатковими джерелами інформації, зіставляти і порівнювати точки зору, висловлювати свою точку зору.

1.2 Постановка завдання

Провести огляд та порівняльний аналіз сучасних програмних засобів обробки зображень.

Обрати окрему проблему завдань програмної обробки зображень, дослідити її та підготувати доповідь.

1.3 Рекомендовані теми доповідей

Сучасні графічні програмні засоби для досягнення оптимальної гнучкості і функціональності в предметної області. Приклади завдань і засоби їх вирішення.

Сучасні архітектурні рішення для графічних систем.

Сучасні методи графічного представлення даних з предметної області і алгоритми що їх реалізують для розширення функціональності та гнучкості графічних систем.

2 ПРАКТИЧНЕ ЗАНЯТТЯ №2
Тема: Перетворення на площині

2.1 Мета і задачі

Отримання практичних навичок в реалізації двомірних перетворень за допомогою однорідних координат і матриці перетворення розмірністю 3 * 3. Програмна реалізація перетворень в площині.

2.2 Короткі теоретичні відомості

2.2.1 Перетворення на площині.

Для початку зауважимо, що точка на площині задається за допомогою двох її координат. Таким чином, геометрично кожна точка задається значеннями координат вектору щодо обраної системи координат. Координати точок можна розглядати як елементи матриці *[x y]*, тобто у вигляді вектор-рядка або вектор-стовпця. Положенням цих точок керують шляхом перетворення матриці.

Точки на площині *xy* можна перенести в нові позиції шляхом додавання до координат цих точок констант переносу (2.1). Таким чином, для переміщення точки на площині треба до матриці її координат додати матрицю коефіцієнтів перетворення.

$$[x' y'] = [x\ y] + [a\ b] = [x+a \quad y+b] \tag{2.1}$$

Розглянемо результати матричного множення матриці *[x, y]*, яка визначає точку *P*, і матриці перетворень *2*2* загального вигляду (2.2).

$$\begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} a & b \\ c & d \end{bmatrix} = \begin{bmatrix} (a*x + c*y) & (b*x + d*y)) \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} \tag{2.2}$$

Проведемо аналіз отриманих результатів, розглядаючи x' і y' як перетворені координати. Для цього досліджуємо кілька окремих випадків.

Розглянемо випадок, коли $a = d = 1$ і $c = b = 0$. Матриця перетворень призводить до матриці, ідентичної вихідної. При цьому змін координат точки *P* не відбувається.

Якщо $d = 1,\ b = c = 0,\ a = const$, то це призводить до зміни масштабу в напрямку *x*, так як $x' = a*x$. Отже, дане матричне

перетворення еквівалентно переміщенню вихідної точки в напрямку *x*. (2.3).

$$\begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} a & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (a*x+0*y) & (0*x+1*y)) \end{bmatrix} = \begin{bmatrix} a*x & y \end{bmatrix} = \begin{bmatrix} x' & y' \end{bmatrix} \qquad (2.3)$$

Тепер покладемо $b = c = 0$ (2.4).

$$\begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} a & 0 \\ 0 & d \end{bmatrix} = \begin{bmatrix} (a*x+0*y) & (0*x+d*y)) \end{bmatrix} = \begin{bmatrix} a*x & d*y \end{bmatrix} = \begin{bmatrix} x' & \end{bmatrix} \qquad (2.4)$$

В результаті отримуємо зміну масштабів в напрямках *x* і *y*.

Якщо $a \neq d$, то переміщення уздовж осей неоднакові. Якщо $a = d > 1$, то має місце збільшення масштабу координат точки *P*.

Якщо $0 < a = d < 1$, то буде мати місце зменшення масштабу координат точки *P*.

Якщо *a* або (і) *d* негативні, то відбувається відображення координат точок.

Розглянемо це, поклавши $b = c = 0$, $d = 1$ і $a = -1$ (2.5).

$$\begin{bmatrix} x & y \end{bmatrix} * \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} (-1*x+0*y) & (0*x+1*y)) \end{bmatrix} = \begin{bmatrix} -*x & d*y \end{bmatrix} = \begin{bmatrix} x' & \end{bmatrix} \qquad (2.5)$$

Сталося відображення точки відносно осі *y*.

У разі $b=c=0$, $a=1$, $d=-1$, відображення відбувається відносно осі *x*. Якщо $b=c=0$, $a=d<0$, то відображення відбуватиметься щодо початку координат.

Зауважимо, що відображення і зміна масштабу викликають тільки діагональні елементи матриці перетворення.

Перетворення загального вигляду, застосоване до початку координат не приведе до зміни координат точки $(0, 0)$. Отже, початок координат інваріантний при загальному перетворенні. Це обмеження долається за рахунок використання однорідних координат.

2.2.2 Перетворення повороту і відображення

Загальну матрицю *2*2*, яка здійснює поворот фігури щодо початку координат, можна отримати з розгляду обертання одиничного квадрата навколо початку координат.

На рисунку 2.1 видно, що точка B з координатами *(1, 0)* перетворюється в точку B', для якої $x' = (1)*\cos\theta$ та $y' = (1)*\sin\theta$, а точка D, з координатами *(0, 1)* переходить в точку D' з координатами $x' = (-1)\sin\theta$ и $y' = (1)*\cos\theta$.

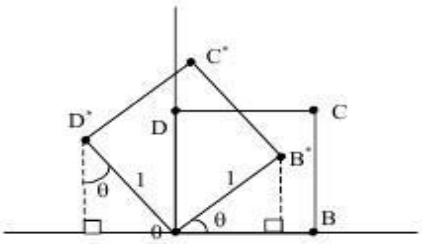


Рисунок 2.1 - Обертання фігури щодо початку координат

Матриця перетворення загального вигляду записується так:

$$\begin{bmatrix} \cos\theta & \sin\theta \\ -\sin\theta & \cos\theta \end{bmatrix}$$

Для окремих випадків. Поворот на 90° можна здійснити за допомогою матриці перетворення:

$$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}.$$

Якщо використовувати матрицю координат вершин, то отримаємо, наприклад:

$$\begin{bmatrix} 3 & -1 \\ 4 & 1 \\ 2 & 1 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 3 \\ -1 & 4 \\ -1 & 2 \end{bmatrix}.$$

Поворот на 180° виходить за допомогою матриці:

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

У той час як чисте двовимірне обертання в площині xy здійснюється навколо осі, перпендикулярної до цієї площини, відображення визначається поворотом на 180° навколо осі, що лежить в площині xy.

Таке обертання навколо лінії $y=x$ відбувається при використанні матриці

$$\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}.$$

Перетворені нові вирази визначаються співвідношенням:

$$\begin{bmatrix} 8 & -1 \\ 7 & 3 \\ 6 & 2 \end{bmatrix} * \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 8 \\ 3 & 7 \\ 2 & 6 \end{bmatrix}.$$

Обертання навколо *y=0* виходить при використанні матриці:

$$\begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix}.$$

2.2.3 Однорідні координати

Перетворення перенесення, масштабування і повороту записуються в матричної формі у вигляді:

$$P' = P + T,$$
$$P' = P * S,$$
$$P' = P * R.$$

Очевидно, що перенесення, на відміну від масштабування і обертання, реалізується за допомогою складання. Це обумовлено тим, що вводити константи переносу всередину структури загальної матриці розміру *2*2* не представляється можливим. Бажаним є уявлення перетворень в єдиній формі - за допомогою множення матриць. Цю проблему можна вирішити за рахунок введення третьої компоненти в вектори точок *[x y]* и *[x′ y′]*, тобто представляючи їх у вигляді *[x y 1]* та *[x′ y′ 1]*. Матриця перетворення після цього стає матрицею розміру *3*3*, наприклад:

$$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & 1 \end{bmatrix}$$

Третій елемент тут можна розглядати як додаткову координату вектора положення. Отже, вектор положення *[x y 1]* при впливі на нього

матриці *3*3* стає вектором положення в загальному випадку виду *[X Y H]*. Представлене перетворення було виконано так, що

$$[X\ Y\ H] = [x'\ y'\ 1].$$

Перетворення, що має місце в тривимірному просторі, в нашому випадку обмежена площиною, оскільки $H=1$. Якщо третій стовпець матриці перетворення T розміру *3*3* відмінний від *0*, то в результаті матричного перетворення отримаємо $[x\ y\ 1] * T = [X\ Y\ H]$, де $H \neq 1$.

Площина, в якій тепер лежить перетворений вектор положення, знаходиться в тривимірному просторі.

Перетворені звичайні координати виходять за рахунок нормалізації однорідних координат, тобто:

$$x' = \frac{X}{H}, y' = \frac{Y}{H}$$

Геометрично все перетворення x і y відбуваються в площині $H = 1$ після нормалізації перетворених однорідних координат. Перевага введення однорідних координат виявляється при використанні матриці перетворень загального вигляду *3*3:*

$$\begin{bmatrix} a & b & p \\ c & d & q \\ m & n & s \end{bmatrix}$$

за допомогою якої можна виконувати і інші перетворення, такі як зміщення, операції зміни масштабу і зсуву, обумовлені матричними елементами *a, b, c* і *d*. Зазначені операції розглянуто раніше.

Основна матриця перетворення розміру *3*3* для двовимірних однорідних координат може бути підрозділена на чотири частини:

$$\left[\begin{array}{cc|c} a & b & p \\ c & d & q \\ \hline m & n & s \end{array}\right].$$

Як ми бачимо, *a, b, c* і *d* здійснюють зміну масштабу, зрушення і обертання; *m* і *n* виконують зсув, а *p* і *q* - отримання проекцій. Елемент *s*, виробляє повну зміну масштабу.

Щоб показати це, розглянемо перетворення:

$$[X \quad Y \quad H] = [x \quad y \quad 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & s \end{bmatrix} = [x \quad y \quad s]$$

Де $X=x$, $Y=y$, $H=s$. Це дає $x' = x/s$ и $y' = y/s$. В результаті перетворення *[x y 1] –> [x/s y/s 1]* має місце однорідна зміна масштабу вектора положення. При $s<1$ відбувається збільшення, а при $s>1$ зменшення масштабу.

2.2.4 Комбіновані перетворення

Розглянемо комбіновані перетворення на прикладі повороту навколо довільної точки.

Вище було розглянуто обертання зображення близько початку координат. Однорідні координати забезпечують поворот зображення навколо точок, відмінних від початку координат. У загальному випадку обертання близько довільної точки може бути виконано шляхом перенесення центру обертання в початок координат, поворотом щодо початку координат, а потім перенесенням точки обертання в початкове положення. Таким чином, поворот вектора положення *[x y 1]* близько точки *(m, n)* на довільний кут може бути виконаний за допомогою перетворення:

$$[x \quad y \quad 1] * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -m & -n & s \end{bmatrix} * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ m & n & s \end{bmatrix} = [X \quad Y \quad H].$$

Виконавши дві операції множення матриць, можна записати:

$$[X \quad Y \quad H] = [x \quad y \quad 1] * \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ -m*(\cos\theta - 1) + n*(\sin\theta) & -m*(\sin\theta) - n*(\cos\theta - 1) & 1 \end{bmatrix}.$$

2.3 Програмна реалізація

На першому етапі спроектуйте віконний інтерфейс. Додайте на формі кнопки для перенесення, масштабування і повороту. При необхідності розмістить елементи для введення значень: наприклад, для введення значень кутів повороту. Приклад вікна приведений на рис. 2.2.

Опишіть і задайте початкові значення для матриці, де будуть зберігатися координати двомірної фігури. Назвемо цю матрицю матрицею тіла і опишемо як двомірний масив *Sq* типу *double*. Опишіть і задайте як одиничну матрицю перетворення *3*3* (двомірний масив *T* типу *double*). Оскільки масиви *Sq* і *T* будуть використовуватися повсюдно, то доцільно поставити їх в якості атрибутів класу доступних у всіх методах і не витрачати час на передачу в якості параметрів.

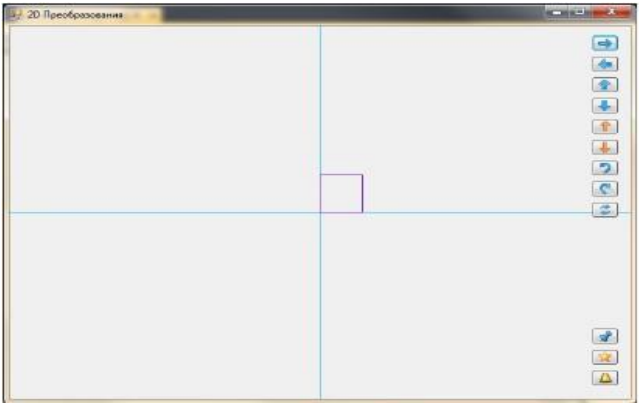


Рисунок 2.2 – Інтерфейс для перетворень на площині

При натисканні на кнопку повинен запускатися наступний процес:

1) завдання матриці перетворення
2) множення матриці тіла на матрицю перетворення
3) нормалізація

4) перерисовка фігур.

Етап завдання коефіцієнтів матриці перетворення зводиться до завдання елементів двомірного масиву. На другому етапі обрана стратегія збереження змін при перетвореннях в матриці тіла, а не накопичення змін в матриці перетворень. Виходячи з цього, програмний код алгоритму методу, що реалізує даний етап, може виглядати наступним чином (рис. 2.3).

```
b = new double[3];//массив для хранения промежуточных данных
  for (int j=0; j<4; j++)//Цикл по вершинам фигуры (4 вершины для квадрата)
      {
          for (int i = 0; i < 3; i++)
          {
              b[i] = 0;
              for (int k = 0; k < 3; k++)
                  b[i] = b[i] + Sq[j, k]*T[k, i];
          }
          for (int k=0; k<3; k++)
          Sq[j, k]= b[k];
      }
```

Рисунок 2.3 - Програмний код алгоритму множення матриці тіла на матрицю перетворення

Третій етап (нормалізація) призводить до одиниці останню координату шляхом ділення:

```
for (int j = 0; j < 4; j++)
    {
        Sq[j, 0] = Sq[j, 0] / Sq[j, 2];
        Sq[j, 1] = Sq[j, 1] / Sq[j, 2];
        Sq[j, 2] = 1;
    }
```

Перерисовка повинна знаходитися в обробнику події Paint (рис. 2.4))

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
    Graphics gfx = e.Graphics;
    Pen skyBluePen = new Pen(Brushes.DeepSkyBlue);
    int cx = ClientSize.Width/2;
    int cy = ClientSize.Height/2;
    gfx.DrawLine(skyBluePen, cx, 0, cx, ClientSize.Height);
    gfx.DrawLine(skyBluePen, 0, cy, ClientSize.Width, cy);

    Pen SqPen = new Pen(Brushes.BlueViolet);

    gfx.DrawLine(SqPen, (int) (cx + Sq[0, 0]), (int) (cy - Sq[0, 1]),
                        (int) (cx + Sq[1, 0]), (int) (cy - Sq[1, 1]));
    gfx.DrawLine(SqPen, (int) (cx + Sq[1, 0]), (int) (cy - Sq[1, 1]),
                        (int) (cx + Sq[2, 0]), (int) (cy - Sq[2, 1]));
    gfx.DrawLine(SqPen, (int) (cx + Sq[2, 0]), (int) (cy - Sq[2, 1]),
                        (int) (cx + Sq[3, 0]), (int) (cy - Sq[3, 1]));
    gfx.DrawLine(SqPen, (int) (cx + Sq[3, 0]), (int) (cy - Sq[3, 1]),
                        (int) (cx + Sq[0, 0]), (int) (cy - Sq[0, 1]));
}
```

Рисунок 2.4 - Програмний код перерисовки

Змінні *cx* і *cy* це координати центру вікна, використовуються для прорисовки осей і самої фігури. Приведення до цілого типу, а отже округлення, проводиться безпосередньо перед промальовуванням і не зберігається в матриці *Sq*. Тому не буде відбуватися накопичення помилок округлення.

Запустіть і налагодьте програму. За результатами підготуйте звіт.

2.4 Завдання

Оберіть варіант сукупності фігур (див. додаток А). Реалізуйте із заданою сукупністю фігур всі види афінних перетворень: перенесення по осі *OX* і осі *OY*, відображення щодо координатних осей і прямий *Y=X*, масштабування, поворот на задані кути щодо центру координат і щодо довільної точки, про яку йдеться в ході виконання програми. Передбачити відновлення вихідної позиції фігур. Управління організувати як через елементи інтерфейсу (меню, кнопки, рядки редагування і ін.), так і через «гарячі» клавіші.

2.5 Питання для самоперевірки

Що розуміється під афінним перетворенням над точками?

Охарактеризуйте операції зсуву, масштабування, обертання.

Запишіть афінні перетворення: перенесення, масштабування, обертання і відображення у матричній формі.

Чому відображення і зміна масштабу викликають тільки діагональні елементи матриці перетворення?

Чим обумовлено використання однорідних координат?

2.6 Рекомендована література

При підготовці до заняття необхідно вивчити тему афінні перетворення над точками, використовуючи конспект лекцій, а також [5, с. 281-292; 7, с. 65-68].

3 ПРАКТИЧНЕ ЗАНЯТТЯ №3
Тема: Основи обробки зображень в графічному редакторі GIMP

3.1 Мета заняття

Обробка зображень засобами графічного редактора GIMP. Ввід і редагування зображень. Отримання практичних навичок фотомонтажу в редакторі GIMP.

3.2 Короткий опис роботи в GIMP
3.2.1 Основні поняття

GIMP (https://www.gimp.org/downloads/) – багатоплатформне програмне забезпечення для редагування растрових зображень (GIMP - GNU Image Manipulation Program). Редактор GIMP придатний для вирішення безлічі завдань зі зміни зображень, включаючи ретуш фотографій, об'єднання і створення зображень.

Однією з сильних сторін GIMP є його доступність з багатьох джерел для багатьох операційних систем. GIMP входить до складу більшості дистрибутивів GNU / Linux. GIMP також доступний і для інших операційних систем на кшталт Microsoft Windows ™ або Mac OS X ™ від Apple (Darwin). GIMP - безкоштовне програмне забезпечення, яке випускається під ліцензією GPL (General Public License). GPL надає користувачам право доступу до вихідного коду програм і право змінювати його. Будучи досить потужним продуктом, GIMP здатний стати незамінним помічником в таких областях, як підготовка графіки для Web-сторінок і поліграфічної продукції, оформлення програм (малювання піктограм, заставок і т. П.), Створення анімаційних роликів, обробка кадрів для відеофрагментів і побудова текстур для тривимірної анімації. Дуже корисна функція створення і обробки анімаційних роликів, що дозволяє накладати анімацію на об'єкт як текстуру і виконувати певні фінішні операції після рендеринга.

Зображення - основний об'єкт, з яким працює GIMP, це один файл з розширенням TIFF або JPEG. Зображення в GIMP може бути достатньо складним. Найбільш правильною аналогією буде не аркуш паперу, а, скоріше, книга, сторінки якої називаються шарами.

Шари. Якщо зображення подібно книзі, то шар можна порівняти зі сторінкою всередині книги. Найпростіше зображення містить тільки один шар і, продовжуючи аналогію, є аркушем паперу. Шари можуть бути прозорими і можуть покривати не весь простір зображення.

Канали в GIMP є найменшою одиницею підрозділу стека шарів, з яких створюється зображення. Кожен канал має той же розмір, що і шар, і складається з тих же пікселів. Сенс цього значення залежить від типу каналу, наприклад, в колірній моделі RGB значення каналу R означає кількість червоного кольору, який додається до інших кольорів пікселів.

Для зміни частини зображення існує **механізм виділення областей**. У кожному зображенні можна створити виділену область, яка, як правило, відображається в вигляді рухомої пунктирною лінії.

3. 2.2 Основні прийоми використання GIMP

На рис. 3.1 показано стандартне розташування вікон GIMP. Елементами вікон є:

1 - панель інструментів, яка містить кнопки для вибору інструментів виділення, малювання, трансформації зображення і т.і.;

2 - параметри обраного інструменту (в даному випадку це інструмент «Кадрування»);

3 - вікно зображення (в GIMP можна відкрити одночасно настільки велика кількість зображень, наскільки дозволяють системні ресурси);

4 – вікно діалогів *Слои/Каналы/Контуры/Отменить* відображає структуру шарів активного зображення і дозволяє управляти ними;

5 – вікно діалогів *Кисти/Текстуры/Градиенты* дозволяє управляти кистями, текстурами і градієнтами.

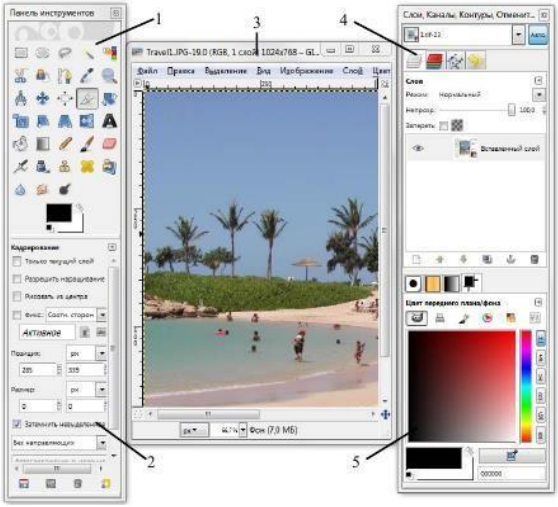


Рисунок 3.1 - Загальний вигляд редактора GIMP

Наведений набір - це мінімальний набір вікон. Більш детальну інформацію про діалоги і панелі, що використовуються в GIMP, можна отримати на сайті Gimp [8].

Робота з файлами в GIMP (створення нового зображення, відкриття і збереження зображення) здійснюється стандартними способами.

У діалоговому вікні «Створити нове зображення» (рис. 3.2), можна встановити початкові ширину і висоту файлу. При виборі додаткових параметрів встановлюється роздільна здатність, колірна модель і колір фону.

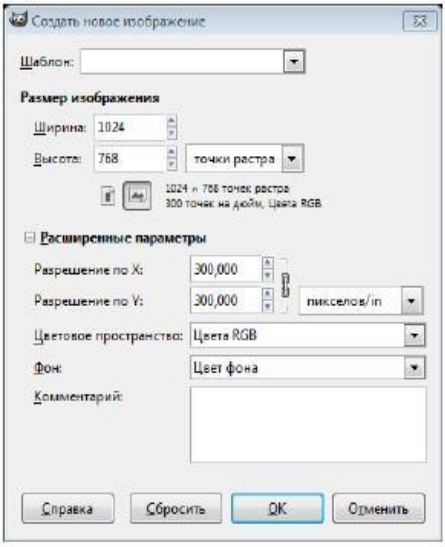


Рисунок 3.2 - Діалог «Створити нове зображення»

При вставки рисунка з буфера обміну будуть встановлені розміри зображення, яке знаходиться в буфері обміну, також можливе захоплення зображенні з екрану, сканера або фотокамери.

При збереженні зображення в деякі формати, можуть з'являтися додаткові вікна для завдання параметрів зображення. Відзначимо формат JPG, при збереженні в якому можна встановити якість зображення. Чим вище буде задано якість, тим більший розмір буде у файлу, що зберігає зображення.

У ряді випадків, наприклад, при обробці деяких відносно невеликих областей, виникає необхідність зміни масштабу відображення

зображення на екрані. Це можна здійснити або через інтерфейс програми, або через клавіатуру і мишу.

Майже все, що робиться із зображенням, може бути скасовано. Ви можете скасувати останню дію, вибравши в меню зображення *Правка→ Отменить*, або поєднанням клавіш ***Ctrl + Z***.

Саме скасування також може бути скасована через меню зображення пункт *Правка→Повторить* або з використанням клавіші швидкого доступу ***Ctrl + Y***.

Якщо ви часто використовуєте скасування та повернення на безліч кроків за раз, зручніше буде працювати з діалогом *Истории действий* - панель, яка показує невеликі ескізи кожної точки в історії скасування, дозволяючи вам переміщатися назад або вперед до точки, за якою ви натискаєте (рис. 3.3).

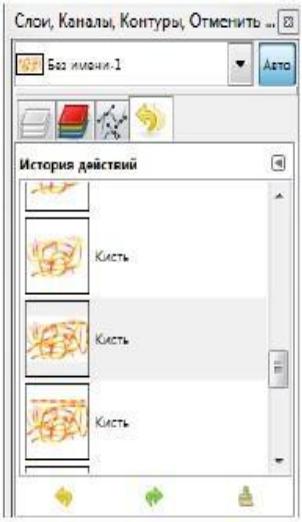


Рисунок 3.3 - Панель «*Історія дій*»

3.3 Завдання

Запустіть GIMP.

Створіть зображення розміром 640 * 480 пікселів з роздільною здатністю 72 dpi. При цьому використовуйте колірну модель RGB.

Використовуючи різні інструменти малювання, створіть зображення. Обов'язково використовуйте кисті різних форм і розмірів,

різні режими накладення квітів, спеціальні ефекти. Також використовуйте ластик, заливку і градієнтну заливку.

Отримане зображення збережіть в різних форматах: xcf, bmp, tif (використовуючи LZW компресію), png, gif, gif з градаціями сірого кольору, jpg (з різним ступенем стиснення: 90, 60, 40).

Створіть звіт в тестовому редакторі MS Word вставте в звіт зображення з отриманих файлів і запишіть після кожного розмір отриманого файлу.

Створіть зображення 20*20 пікселів.

Збільште масштаб відображення і створіть рисунок з емблемою будь-якої компанії, використовуючи обмежену кількість кольорів.

Збережіть отримане зображення в різних форматах: bmp, tif (використовуючи LZW компресію), gif, jpg (з різним ступенем стиснення: 90, 60, 40).

Додайте до звіту отримане зображення з збережених файлів.

Запишіть після кожного зображення розмір отриманого файлу.

Проаналізуйте отримані результати в розгорнутому висновку. При цьому оцінюйте якість отриманих зображень і розміри файлів.

3.4 Питання для самоперевірки

Назвіть три основні вікна GIMP.
Навіщо потрібні шари?
Навіщо потрібно виділення?
Перелічіть основні категорії інструментів GIMP.
Навіщо призначені інструменти кольору?
Навіщо потрібна маска шару?
Який власний формат файлу GIMP?

3.5 Рекомендована література

При підготовці до заняття необхідно вивчити основи роботи в GIMP [9].

4 ПРАКТИЧНЕ ЗАНЯТТЯ №4
Тема: Обробка точкових зображень

4.1 Мета і задачі

Отримання практичних навичок в обробки точкових зображень.

4.2 Короткі теоретичні відомості

4.2.1 Корекція кольору

Для корекції кольору в GIMP існує ряд інструментів в меню «*Цвет*» (рис. 4.1).

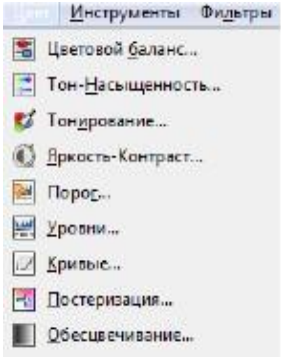


Рисунок 4.1 - Інструменти для корекції кольору

Перший інструмент «*Цветовой баланс*» дозволяє регулювати співвідношення основних кольорів з RGB і CMY моделей. Інструмент найбільш корисний для виправлення кольорів, що переважають на цифрових фотографіях. Регулювання відбувається або для світлих ділянок зображення, або для півтонів, або для тіней за допомогою спеціального діалогу (рис. 4.2).

4.2.2 Корекція тону, освітленості, насиченості

Наступний інструмент дозволяє змінювати значення тону, насиченості і яскравості обраного колірного діапазону в активному шарі або виділення з допомогою спеціального діалогу (рис. 4.3).

Тон дозволяє відрізняти між собою основні кольори в моделі HSV: червоний, зелений, синій, блакитний, малиновий (пурпурний), жовтий.

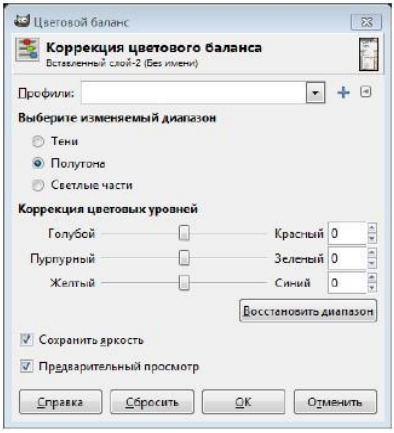


Рисунок 4.2 - Діалог для коректування колірного балансу

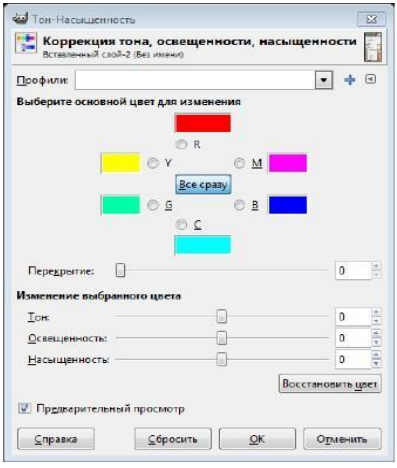


Рисунок 4.3 - Діалог корекції тону, освітленості, насиченості

У теорії кольору насиченість - це інтенсивність певного тону. Можна сказати, що насиченість це характеристика кольору, що визначає його чистоту. Насичений колір можна назвати соковитим, глибоким, менш насичений - приглушеним, наближеним до сірого. Також насиченість дозволяє відрізняти червоний колір від рожевого, зелений від світло-зеленого і т.і.

Повністю ненасичений колір буде відтінком сірого. Насиченість (saturation) - одна з трьох координат в колірних просторах моделей HSL і HSV.

Світлота (від англ. Lightness) - одна з основних характеристик кольору поряд з насиченістю і тоном. Світлота це суб'єктивна яскравість ділянки зображення, що дозволяє відрізняти, наприклад, сірий колір від чорного, а білий від сірого.

Інструмент «*Тонирование*», так само можна використовувати для зміни тону, освітленості і насиченості через спеціальний діалог (рис. 4.4). Але на відміну від попереднього інструмента, тонування застосовується для зображень в градаціях сірого кольору для отримання кольоровості. Використання інструментів виділення окремих областей і тонування дозволяють чорно-білу фотографію (зображення в градаціях сірого кольору) перевести в колір. Також для розфарбовування чорно білих зображень можна використовувати команду *Цвет→Окрашивание.*

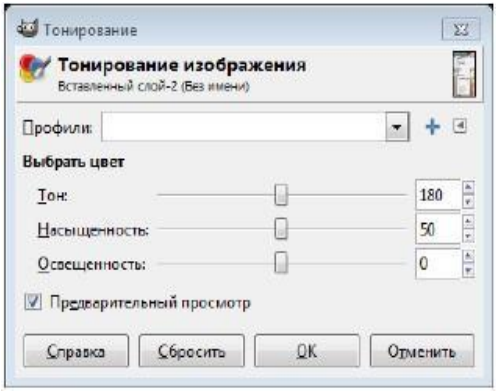


Рисунок 4.4 - Діалог тонування

4.2.3 Яскравість і контраст

Досить просто змінюється яскравість і контрастність зображення або виділення при використанні інструменту. «*Яркость-контраст*». Яскравість і контраст є суб'єктивними характеристиками зображення, які сприймалися людиною.

Яскравість є характеристикою, що визначає те, на скільки сильно кольору пікселів відрізняються від чорного кольору. Наприклад, якщо оцифрована фотографія зроблена в сонячну погоду, то її яскравість буде

значною. З іншого боку, якщо фотографія зроблена ввечері або вночі, то її яскравість буде невелика.

Контраст є характеристикою того, наскільки великий розкид мають кольори пікселів зображення. Чим більший розкид мають значення кольорів пікселів, тим більший контраст має зображення.

4.2.4 Гістограма зображення

Кілька наступних команд в меню «*Цвет*» використовують гістограму зображення (рівні) як основний або додатковий інструмент. **Гістограма** (в фотографії) - це графік розподілу півтонів зображення, в якому по горизонтальній осі представлена Яскравість, а по вертикалі - відносна кількість пікселів з даними значенням яскравості (рис. 4.5). **Гістограма зображення дозволяє оцінити кількість і різноманітність відтінків зображення, а також загальний рівень яскравості зображення**.

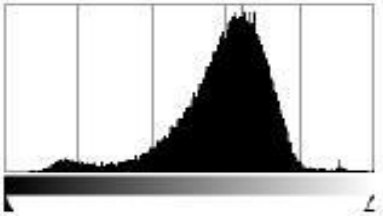


Рисунок 4.5 – Приклад гістограми

Найбільш просто побудову гістограми можна пояснити для зображення в градація сірого кольору. В цьому випадку, гістограма являє собою діаграму, де по горизонтальній шкалі відкладаються градації сірого від 0 (чорний) до 255 (білий), а по вертикальній - кількість точок відповідної градації в цьому зображенні. Чим вище стовпець, тим більше точок відповідного відтінку сірого міститься в фотографії. Так гістограма, приведена на рис. 4.5., відображає кількість пікселів певного тону для рис. 4.6.

Рисунок 4.6 – Приклад зображення для побудови гістограм

При виборі інструмента «*Уровни*» і завданні параметрів, як показано на рис. 4.7. (бігунки справа зліва зрушені до середини), розподіляємо значення інтенсивностей рівномірно по всій області градацій сірого кольору. Цим самим підвищуємо контрастність зображення (рис. 4.8).

У деяких випадках, описаний метод дозволяє поліпшити зображення, а в деяких навпаки зменшити художню цінність зображення.

Гістограма для кольорового зображення будується по яскравості, або по кожному окремому каналу для основних кольорів колірної моделі. Наприклад, для RGB моделі гістограма може бути побудована для трьох каналів R, G і B відповідно.

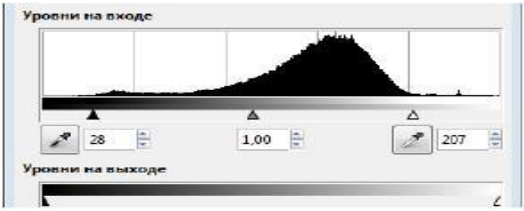


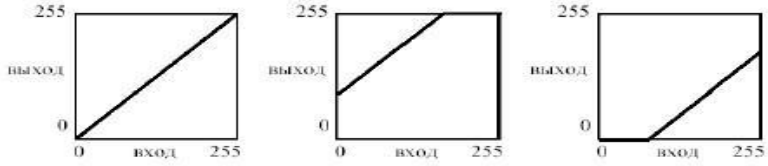
Рисунок 4.7 - Зміна гістограми

Рисунок 4.8 - Зображення з підвищеною контрастністю

4.2.5 Корекція колірних кривих

Другим значимим елементом після гістограми є «*Кривые*», які дозволяють управляти функцією яскравості і контрастності. Розглянемо функцію для управління яскравістю і контрастністю, областю визначення і значень якої є значення колірних компонент в моделі RGB. Аргументом функції є колір пікселя вихідного зображення. Значення функції являє собою колір пікселя обробленого зображення. Для зміни яскравості/ контрасту функція застосовується для кожного пікселя зображення.

Якщо яскравість і контраст зображення не змінюються в процесі перетворення, то функція має графік, представлений на рис. 4.9, а. З рисунка видно, що функція в цьому випадку просто передає на вихід значення свого аргументу.

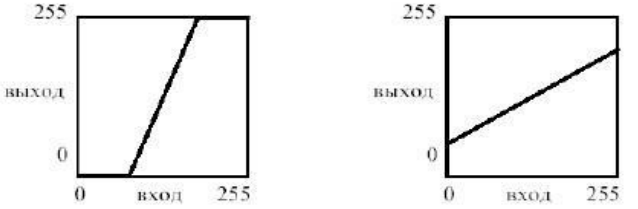


а) б) в)

Рисунок 4.9 Графіки яскравості

Яскравість для даної функції являє собою зсув прямої лінії в вертикальному напрямку. Яскравість зображення збільшується пропорційне зсуву прямої. Якщо пряма зсувається вгору (рис. 4.9, б), яскравість зображення збільшується, а якщо пряма зсувається вниз (рис. 4.9, в) - зменшується.

При використанні перетворення контрасту пряма лінія змінює свій нахил. При збільшенні контрасту зображення (рис. 4.10, а) нахил прямої збільшується, при зменшенні контрасту - зменшується (рис. 4.10, б). При цьому зсув прямої в горизонтальному напрямку означає, що крім контрасту змінюється і яскравість зображення. Комбінації нахилу і зсуву прямої дозволяють одночасно змінювати і яскравість, і контраст зображення. Наприклад, на рис. 4.11. представлений графік функції, що підсилює контраст і збільшує яскравість зображення.



а) б)

Рисунок 4.10 Графіки контрастності

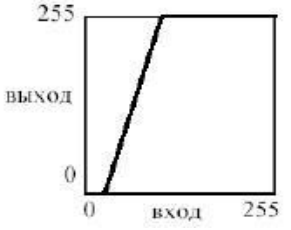


Рисунок 4.11 Збільшення яскравості і контрастності

Перетворення яскравості/контрасту може бути застосовано і до окремих компонентів моделі RGB, наприклад до компоненту червоного кольору. Тоді яскравість / контраст будуть змінюватися тільки для червоного компонента, а для інших компонент вони залишаться незмінними. Більш того, можна задавати різні перетворення яскравості / контрасту одночасно для кожного компонента моделі RGB.

4.2.6 Фільтри

Фільтр - спеціальний вид інструменту, який бере вхідний шар або зображення, застосовує до нього математичний алгоритм і повертає змінений шар або зображення в новому форматі. Фільтри дозволяють накладати на зображення різні ефекти, наприклад: розмиття, різкість, деформацію, шум і т.і.

Для роботи з фільтрами в GIMP виділено спеціальне меню «*Фильтры*». При роботі з фільтрами активно використовуються діалогові вікна для завдання параметрів фільтрів.

В GIMP можна використовувати такі фільтри:

- фільтри розмиття;
- фільтри поліпшення;
- фільтри спотворення;
- фільтри світло і тінь;
- фільтри виділення краю;
- фільтри імітації;
- фільтри візуалізації.

Більш повну інформацію по роботі з фільтрами в GIMP можна отримати в онлайн довіднику [9].

4.3 Завдання

Підібрати вихідне зображення в градаціях сірого кольору. Помістити зображення в звіт.

Перекласти зображення в режим RGB моделі, вибравши *Изображение → Режим → RGB*.

Використовуючи інструменти виділення, тонування і фарбування зробити зображення кольоровим.

Відрегулювати колірний баланс, яскравість і контрастність.

Зберегти отримане зображення. Описати хід роботи і отримані результати.

Підібрати кілька різних кольорових зображень і досліджувати зображення за допомогою гістограми.

При необхідності збільшити контрастність зображень. Відобразити результати дослідження в звіті, в якому привести вихідні та отримані зображення і їх гістограми.

Створити нове зображення і залити його градієнтом від чорного до білого кольору, зліва направо.

Використовую корекцію колірних кривих як показано на рис. 4.13.

Перетворити зображення. Пояснити результат в звіті.

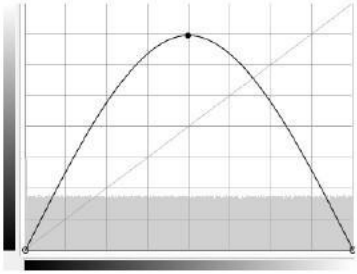


Рисунок 4.13 - Крива корекції кольору

Підібрати кольорову фотографію. Поліпшити зображення, використовуючи гістограму (рівні).

Створити виділення по краях зображення. Для цього використовуйте прямокутне виділення із закругленими краями. Для симетричності виділення в параметрах інструменту «*Прямоугольное выделение*» позицію і розмір. Потім інвертуйте виділення командою *Выделение → Инвертировать* (Ctrl + I).

Застосуйте до виділення кілька різних фільтрів для отримання оригінальних рамок. Можна спробувати вибрати *Фильтры → Карта →*

Фрактальный след. Хороші результати збережіть у звіті. Використовуйте історію для скасування невдалих дій і нових спроб.

Підберіть кольорові зображення для подальших експериментів.

Досліджуйте групи фільтрів: *«Искажение», «Выделение края», «Имитация»* застосовуючи їх в цілому до всього зображення. Найбільш цікаві результати занесіть в звіт.

Створіть нове зображення 640 * 480.

Дослідіть на цьому зображенні можливості групи фільтрів «*Визуализация*».

Досліджуйте можливості побудови геометричних і алгебраїчних фракталів.

Досліджуйте можливості побудови векторних примітивів на зображенні за допомогою фільтра Gfig.

Напишіть розгорнутий висновок, де проаналізуйте основні можливості GIMP в порівнянні з іншими графічними редакторами. Висловіть і обґрунтуйте свою думку.

4.4 Питання для самоперевірки

Які є інструменти в GIMP для корекції кольору?

Що у теорії кольору розуміється під насиченістю та світлотою?

Який інструмент можна використовувати для зміни тону, освітленості і насиченості?

Що оцінюють по гістограмі зображення?

Навіщо служать фільтри?

Які фільтри можна використовувати в GIMP?

4.5 Рекомендована література

При підготовці до заняття необхідно вивчити засоби перетворення яскравості зображень, а також методи просторової фільтрації, використовуючи конспект лекцій та [1, с.3-5; 2, с. 6-8; 3, с. 121-139; 9].

5 ПРАКТИЧНЕ ЗАНЯТТЯ №5
Тема: Основи обробки зображень в графічному редакторі Inkscape

5.1 Мета і задачі

Отримання практичних навичок роботи в редакторі Inkscape. Створення векторного логотипу.

5.2 Короткі теоретичні відомості

Inkscape (Інкскейп) - вільно розповсюджуваний векторний графічний редактор, що підтримує відкритий формат SVG (Scalable Vector Graphics - Масштабируемая векторна графіка). Inkscape використовує формат SVG для своїх файлів. Відкритий стандарт SVG широко використовується в графічних пакетах. Формат SVG використовує мову розмітки XML, тому файли в цьому форматі можуть редагуватися будь-яким текстовим або XML-редактором (окремо від Inkscape).

Крім SVG, в Inkscape можна працювати і з іншими форматами (наприклад, EPS і PNG).

Детальну інформацію по роботі в редакторі Inkscape можна отримати в онлайн курсі [11].

5.2.1 Розміщення тексту вздовж контуру

У Incscape існує можливість розміщення тексту уздовж будь-яких ліній (в тому числі і кривих) і вздовж будь-якої складної фігури, що створена шляхом перетворення фігури в криву. Складна фігура, створена шляхом злиття, автоматично стає кривої. Розташовувати текст уздовж фігур (прямокутників, еліпсів і т.і.) не можна, тому попередньо такі фігури потрібно обов'язково перетворювати в криві. Алгоритм розміщення тексту уздовж контуру виглядає наступним чином:

1) натисніть на іконку  і створіть текстовий об'єкт;
2) натисніть на кнопку для створення потрібної фігури, намалюйте фігуру;
3) натисніть на кнопку  і виберіть обидва об'єкти. Вибрати обидва об'єкти можна послідовно клацнувши по ним, утримуючи *Shift*;
4) у меню, виберіть *Текст→ Разместить по контуру*.

5.2.2 Виконання логічних операцій над фігурами

Над кількома виділеними об'єктами можливе виконання логічних операцій додавання, віднімання, перетину, що виключає АБО і т.і.

Команда *Сумма* (додавання) об’єднує два об'єкти і робить з них один. Може застосовуватися до будь-якої кількості об'єктів. Одержуваний в результаті виконання операції об'єкт завжди використовує настройки стилю (заливки і штриха) нижнього об'єкта. Алгоритм складання двох об'єктів наведено нижче.

1) створіть дві фігури за допомогою інструментів;

2) натисніть на кнопку  і виберіть обидва об'єкти (вибрати обидва об'єкти можна послідовно клацнувши по ним, утримуючи ***Shift***);

3) у меню виберіть *Контур → Сумма*.

Команда *Разность* (віднімання) видаляє у об'єкта області, що перекриваються вищерозміщеним виділеним об'єктом (або об'єктами). Може застосовуватися тільки до двох об'єктах.

Алгоритм виконання операції для двох об'єктів наведено нижче.

1) створіть дві фігури за допомогою інструментів;

2) натисніть на кнопку  і виберіть обидва об'єкти;

3) у меню виберіть *Контур → Разность*.

Команда *Пересечение* (перетин) дозволяє створити новий об'єкт, який включає в себе область перетину двох або більше об'єктів, виділених перед злиттям. Якщо виділено більше двох об'єктів, то необхідно, щоб у всіх виділених об'єктах був сегмент перетину. Якщо такого сегмента немає, то команда ігнорується. Фрагмент перетину може бути тільки один. Одержуваний в результаті виконання операції об'єкт завжди використовує настройки стилю (заливки і штриха) нижнього об'єкта. Алгоритм виконання операції для двох об'єктів аналогічний розглянутим вище.

Команда *Исключающее ИЛИ* (що виключає або) робить пересічні області прозорими. Може застосовуватися тільки до двох об'єктах.

Команда *Разделить* (розділити) поєднує в себе команди *Разность* і *Пересечение*. Число виділених об'єктів не може бути більше двох. Алгоритм виконання операції для двох об'єктів аналогічний розглянутим вище.

5.2.3 Робота з вузлами. Інструменти для керування вузлами

Інструмент керування вузлами  призначений для редагування і вибору вузлів. Що б активізувати інструмент для управління вузлами, можна використовувати бокове вікно панелі інструментів, цей інструмент розташований в ньому другим зверху або натиснути клавішу ***F2***. При цьому зміниться склад кнопок контекстної панелі інструментів. Вона стане виглядати так, як показано на рис. 5.1.

Рисунок 5.1 - Панель інструментів управління вузлами

Для того щоб на активній фігурі виділити вузли, необхідно вибрати кнопку  «Перетворити вибраний об'єкт в контур» або натиснути комбінацію клавіш ***Shift*** + ***Ctrl*** + ***C***.

Якщо інструмент управління вузлами активний, то по контуру активної фігури можуть відображатися вузли у вигляді квадратиків. Для того щоб вибрати вузол, просто клацніть по ньому. Клацання по контуру між вузлами вибирає обидва цих вузла. Якщо ви хочете додати або видалити вузол утримуйте при натисканні миші клавішу ***Shift***. Так само додати вузол можна за допомогою подвійного клацання лівою кнопкою миші по контуру активної фігури.

Якщо необхідно вибрати всі вузли фігури, то можна скористатися комбінацією клавіш ***Ctrl*** + ***A*** в цьому випадку будуть вибрані всі вузли, крім вкладених. Для того, щоб вибрати всі вузли, включаючи вкладені, слід використовувати комбінацію клавіш ***Ctrl*** + ***Alt*** + ***A***. Для зміни форми можна використовувати наступні кнопки:  . Клавіша  показує або приховує окреслення контуру.

Переміщати вузли можна за допомогою миші звичайним чином. Якщо утримувати при переміщенні вузла клавішу ***Ctrl***, то вузол зможе переміщатися тільки по вертикалі або по горизонталі. Утримуючи комбінацію клавіш ***Ctrl*** + ***Alt***, можна переміщати вузол строго уздовж його направляє. Переміщати вузли можна також за допомогою стрілок на клавіатурі. У цьому випадку об'єкт буде переміщатися з кроком 2 пікселя (за замовчуванням, але цю настройку можна змінити).

Після того як вузол обраний, якщо це можливо для даного виду вузла, то буде відображатися його направляюча. Розташування

направляючої також впливає на вигляд кривої цього вузла. За допомогою маркерів на кінцях напрямної можна змінювати її довжину і обертати її. Утримуючи при обертанні направляючої клавішу ***Ctrl***, можна обертати її з інтервалом 15 градусів. Утримання клавіші ***Alt*** блокує зміна довжини направляючої. Клавіша Shift дозволяє переміщати обидві напрямних.

При роботі в редакторі зручно використовувати Швидкий доступ -Гарячі клавіші.

Shift утримуйте для вибору декількох вузлів.

TAB вибирає наступний вузол.

Shift + ***Tab*** вибирає попередній вузол.

Ctrl + ***Alt*** видаляє вузол.

Ctrl + клацання покажчиком миші на маркер направляючої обнуляє її довжину. Щоб витягнути направляючу назад з вузла використовуйте клавішу ***Shift***.

5.3 Завдання

Використовуючи прийоми, що описані в розділі, створіть векторне зображення за зразком (див. додаток Б).

Проаналізуйте виконану роботу у висновку. Також, опишіть у висновку переваги векторного формату зображення перед растровим і переваги формату SVG.

5.4 Питання для самоперевірки

Назвіть векторні графічні редактори, виділіть їх особливості у обробці зображень.

У чому особливість графічного редактора Inkscape?

Охарактеризуйте команди Inkscape для виконання логічних операцій над фігурами.

Для чого призначені інструменти керування вузлами?

5.5 Рекомендована література

При підготовці до заняття необхідно ознайомитись із програмним забезпеченням векторної графіки, вивчити основи обробки зображень в Inkscape, використовуючи конспект лекцій та [10].

6 ПРАКТИЧНЕ ЗАНЯТТЯ №6
Тема: Програмування графіки

6.1 Мета і задачі

Вивчення можливостей Visual Studio зі створення найпростіших графічних зображень. Програмна реалізація побудови на екрані різних графічних примітивів.

6.2 Короткі теоретичні відомості

6.2.1 Повідомлення WM_PAINT

ОС Microsoft Windows стежить за переміщенням і зміною розміру вікон і при необхідності сповіщає додатки, про те, що їм слід перемалювати вміст вікна. Для сповіщення в чергу програми записується повідомлення з ідентифікатором WM_PAINT. Отримавши таке повідомлення, функція вікна повинна виконати перемальовування всього вікна або його частини, в залежності від додаткових даних, отриманих разом з повідомленням WM_PAINT.

Для полегшення роботи по відображенню вмісту вікна весь вивід у вікно зазвичай виконують в одному місці додатка - при обробці повідомлення WM_PAINT в функції вікна. Додаток має бути зроблено таким чином, щоб в будь-який момент часу при надходженні повідомлення WM_PAINT функція вікна могла перемальовувати все вікно або будь-яку його частину, задану своїми координатами. Останнє неважко зробити, якщо додаток буде зберігати в пам'яті свій поточний стан, користуючись яким функція вікна зможе перемалювати вікно в будь-який момент часу. Тут не мається на увазі, що програма має зберігати образ вікна у вигляді графічного зображення і відновлювати його при необхідності, хоча це і можна зробити. Додаток має зберігати інформацію, на підставі якої зможе в будь-який момент часу перемалювати вікно.

Повідомлення WM_PAINT передається функції вікна, якщо стала видна область вікна, прихована раніше іншими вікнами, якщо користувач змінив розмір вікна або виконав операцію прокрутки зображення у вікні. Додаток може передати функції вікна повідомлення WM_PAINT явно, викликаючи функції програмного інтерфейсу Win32 API, такі як UpdateWindow, InvalidateRect або InvalidateRgn.

Іноді ОС Microsoft Windows може сама відновити вміст вікна, не посилаючи повідомлення WM_PAINT. Наприклад, при переміщенні курсору миші або значка згорнутого додатки ОС відновлює вміст вікна.

Якщо ж ОС не може відновити вікно, функція вікна отримує від ОС повідомлення WM_PAINT і перерисовує вікно самостійно.

6.2.2 Подія Paint

Для форм класу *System.Windows.Forms* передбачений зручний об'єктно-орієнтований спосіб, що дозволяє додатку при необхідності перерисовувать вікно форми в будь-який момент часу. Коли вся клієнтська область вікна форми або частина цієї області вимагає перерисування, формі передається подія Paint. Все, що потрібно від програміста, це створити обробник цієї події, наповнивши його необхідною функціональністю (рис. 6.1).

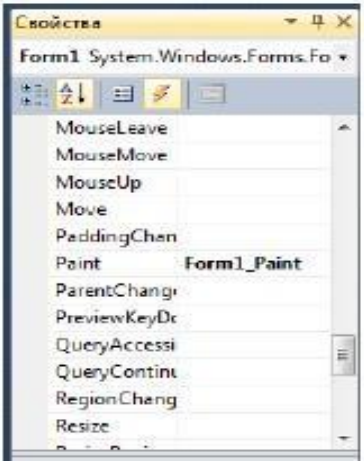


Рисунок 6.1 - Створення обробника події Paint

6.2.3 Объект Graphics

Перед тим як малювати лінії і фігури, відображати текст, виводити зображення і керувати ними в GDI необхідно створити об'єкт *Graphics*. Об'єкт *Graphics* представляє поверхню малювання GDI і використовується для створення графічних зображень. Нижче представені два етапи роботи з графікою:

1) створення об'єкта *Graphics*;

2) використання об'єкта Graphics для малювання ліній і фігур, відображення тексту або зображення і управління ними.

Існує кілька способів створення об'єктів Graphics. Одним з найбільш використовуваних є отримання посилання на об'єкт Graphics через об'єкт *PaintEventArgs* при обробці події Paint форми або елемента управління:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
Graphics g = e.Graphics; // Объявляется объект Graphics
// Далее вставляется код рисования  }
```

6.2.4 Методи та властивості класу Graphics

Імена великої кількості методів, визначених в класі Graphics, починається з префікса *Draw** і *Fill**. Перші з них призначені для малювання тексту, ліній і не зафарбованих фігур (наприклад, прямокутні рамки), а другі - для малювання зафарбованих геометричних фігур. Розглянемо застосування тільки найважливіших з цих методів, а повну інформацію можна знайти в документації.

Метод *DrawLine* малює лінію, що сполучає дві точки з заданими координатами. Нижче наведені прототипи різних версій цього методу:

```
public void DrawLine(Pen, Point, Point);
public void DrawLine(Pen, PointF PointF);
public void DrawLine(Pen, int, int, int, int);
public void DrawLine(Pen, float, float, float);
```

Перший параметр задає інструмент для малювання лінії - перо.

Пір'я створюються як об'єкти класу *Pen*, наприклад:

```
Pen p = new Pen(Brushes.Black,2);
```

Тут створюється чорне перо товщиною 2 пікселя. Створюючи перо, можна вибрати його колір, товщину і тип лінії, а також інші атрибути. Інші параметри перевантажених методів *DrawLine* задають координати точок, що з'єднуються. Ці координати можуть бути задані як об'єкти класу *Point* і *PointF*, а також у вигляді цілих чисел і чисел з плаваючою десятковою крапкою.

У класах *Point* і *PointF* визначені властивості *X* і *Y*, що задають, відповідно, координати точки по горизонтальній і вертикальної осі. При цьому в класі *Point* ці властивості мають цілочисельні значення, а в класі *PointF* - значення з плаваючою десятковою крапкою. Третій і четвертий варіант методу *DrawLine* дозволяє задавати координати точок, що з'єднуються у вигляді двох пар чисел. Перша пара визначає координати першої точки по горизонтальній і вертикальної осі, а друга - координати другої точки по цим же осях. Різниця між третім і

четвертим методом полягає в використанні координат різних типів (цілочисельних *int* і з плаваючою десятковою крапкою *float*). Щоб випробувати метод *DrawLine* в роботі, створіть додаток *DrawLineApp* (аналогічно тому, як створювали попередній додаток). У цьому додатку створіть наступний оброблювач події *Paint*:

```
private void Form1_Paint(object sender, PaintEventArgs e)
{
Graphics g = e.Graphics;
g.Clear(Color.White);  for (int i = 0; i < 50; i++)
  {
  g.DrawLine(new Pen(Brushes.Black, 2), 10, 4 * i + 20, 200, 4 *  i +
20);
  }
}
```

Тут викликається метод DrawLine в циклі, малюючи 50 горизонтальних ліній (рис. 6.2).

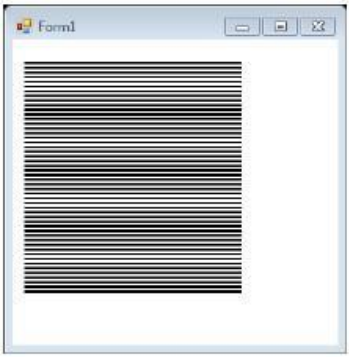


Рисунок 6.2 - Приклад використання DrawLine

Викликавши один раз метод *DrawLines*, можна намалювати відразу кілька прямих ліній, з'єднаних між собою. Іншими словами, метод *DrawLines* дозволяє з'єднати між собою кілька точок. Ко-ординати цих точок по горизонтальній і вертикальної осі передаються методу через масив класу *Point* або *PointF*:

```
public void DrawLines(Pen, Point[]);  public void DrawLines(Pen, PointF[];
```

Для демонстрації можливостей методу *DrawLines* створіть додаток. Створіть кисть pen для малювання ліній:

Pen pen = new Pen(Color.Black, 2);
а також масив точок *points*, які потрібно з'єднати лініями:
Point[] points = new Point[50];

```
for(int i = 0; i < 20; i++)
{  int xPos;
if(i%2 == 0)
{  xPos = 10;
}
 else  {  xPos = 400;
}
points[i] = new Point(xPos, 10 * i);
}
```

Код буде виглядати наступним чином (6.3).

```
public partial class Form1 : Form
{
Point[] points = new Point[50];
Pen pen = new Pen(Color.Black, 2);

public Form1()
{
InitializeComponent();
}
private void Form1_Paint(object sender, PaintEventArgs e)
{
Graphics g = e.Graphics;
g.DrawLines(pen, points);
}
private void Form1_Load(object sender, EventArgs e)
{
for (int i = 0; i < 20; i++)
{    int xPos;    if (i % 2 == 0)
{    xPos = 10;    }    else    {    xPos = 400;
}
points[i] = new Point(xPos, 10 * i);
}
}
}
```

Рисунок 6.3 – Код малювання 50 прямих ліній, з'єднаних між собою

Результат наведено на рис. 6.4.

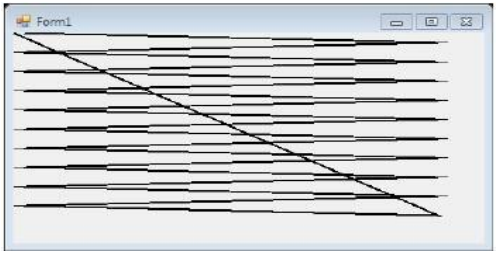


Рисунок 6.4 – Приклад використання масиву точок

Для промальовування прямокутників можна використовувати метод *DrawRectangle (Pen, int, int, int, int)*; В якості першого параметра передається перо класу *Pen*. Інші параметри задають розташування і розміри прямокутника.

Для промальовування багатокутників можна використовувати метод

DrawPolygon (Pen, Point []);

Метод D*rawEllipse* малює еліпс, вписаний в прямокутну область, розташування і розміри якої передаються йому в якості параметрів. За допомогою методу *DrawArc* програма може намалювати сегмент еліпса. Сегмент задається за допомогою координат прямокутної області, в яку вписаний еліпс, а також двох кутів, що обчислюються в напрямку проти годинникової стрілки. Перший кут *Angle1* задає розташування одного кінця сегмента, а другий *Angle2* - розташування іншого кінця сегмента (рис. 6.5).

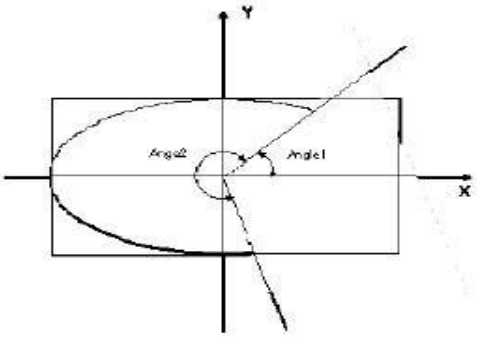


Рисунок 6.5 – Кути і прямокутник, що задають сегмент еліпса

У класі *Graphics* визначено ряд методів, призначених для малювання зафарбованих фігур. Імена деяких з цих методів, що мають префікс *Fill*:

FillRectangle (малювання закрашеного прямокутника),

FillRectangles (малювання безлічі зафарбованих багатокутників),

FillPolygon (малювання закрашеного багатокутника), FillEllipse (малювання закрашеного еліпса) FillPie (малювання закрашеного сегмента еліпса)

FillClosedCurve (малювання закрашеного сплайна)

FillRegion (малювання зафарбованою області типу Region).

Є дві відмінності методів з префіксом *Fill* від однойменних методів з префіксом *Draw*. Перш за все, методи з префіксом *Fill* малюють зафарбовані фігури, а методи з префіксом *Draw* - не зафарбовані. Крім цього, в якості першого параметра методам з префіксом *Fill* передається не перо класу *Pen*, а кисть класу *SolidBrush*. Нижче наведено приклад виводу зафарбованого прямокутника:

```
SolidBrush B=new SolidBrush(Color.DeepPink);
g.FillRectangle(B,0,0,100,100);
```

Таким чином платформа .Net містить велику кількість класів з багатьма методами і властивостями. Немає сенсу описувати всі класи, методи, оскільки з будь-якого методу або класу можна отримати MSDN довідку набравши найменування методу в середовищі *Visual Studio* і натиснувши на ньому клавішу *F1*. Також, при наборі методу в редакторі коду Середовища показує коротку довідку про переданих параметрах.

6.3 Завдання

Вивчіть за допомогою довідки MSDN методи і властивості класів Graphics, Color, Pen і SolidBrush. Створіть власний додаток виводить на форму малюнок, що складається з різних об'єктів (ліній, багатокутників, еліпсів, прямокутників і ін.), Які не зафарбованих і зафарбованих повністю. Використовуйте різні кольори і стилі ліній (суцільні, штрихові, штрих-пунктирні).

6.4 Питання для самоперевірки

Охарактеризуйте графічні можливості Visual Studio.
Опишіть можливості виводу графічних зображень засобами .NET Framework.

Назвіть методи класу Graphics.
Опишіть класи Pen і Brush.
Опишіть можливості виведення тексту.
У чому полягає використання бітових карт?
Назвіть основні властивості і методи класу Bitmap.

6.5 Рекомендована література

При підготовці до заняття необхідно вивчити графічні можливості Visual Studio, використовуючи конспект лекцій та [10].

7 ПРАКТИЧНЕ ЗАНЯТТЯ №7
Тема: Колірні моделі в комп'ютерній графіці

7.1 Мета і задачі

Вивчення основ колориметрії. Аналіз колірних моделей.

7.2 Короткі теоретичні відомості

7.2.1 Принципи побудови кольору.

Колір - одна з найважливіших характеристик графічних зображень. Для характеристики кольору використовуються наступні атрибути: кольоровий тон, яскравість, насиченість.

Кольоровий тон дозволяє відрізняти один колір від іншого - наприклад, зелений від червоного, жовтого й інших.

Яскравість визначається енергією, інтенсивністю кольору.

Насиченість (чистота тону) виражається часткою присутності білого кольору. В ідеально чистому кольорі домішки білого відсутні. Якщо, наприклад, до чистого червоного кольору додати у визначеній пропорції білий колір, то вийде світлий блідо-червоний колір.

Зазначені три атрибути дозволяють описати всі кольори й відтінки. Наука, що вивчає колір і його властивості, називається колориметрією. Вона описує загальні закономірності сприйняття кольору людиною. Одними з основних законів колориметрії є закони змішування кольорів. Ці закони в найбільш повному вигляді були сформульовані в 1853 році німецьким математиком Германом Грассманом.

1. Колір трьохмірний - для його опису необхідні три компоненти. Будь-які чотири кольори знаходяться в лінійній залежності, хоча існує необмежене число лінійно незалежних сукупностей із трьох кольорів.

Іншими словами, для будь-якого заданого кольору (К) можна записати таке кольорове рівняння, що виражає лінійну залежність кольорів:

$$К=aК1+bК2+cК3,$$

де К1, К2, К3 – деякі базисні, лінійно незалежні кольори,

a, b, c – коефіцієнти, що вказують на кількість відповідного кольору, що змішується.

Лінійна незалежність кольорів К1, К2, К3 означає, що жоден з них не може бути виражений зваженою сумою (лінійною комбінацією) двох інших.

Цей закон можна трактувати й у більш широкому сенсі, а саме, у сенсі тривимірності кольору. Необов'язково для опису кольору застосовувати суміш інших кольорів, можна використовувати й інші величини - але їх обов'язково повинно бути три.

2. Якщо в суміші трьох кольорових компонентів одна міняється безупинно, у той час, як дві інші залишаються постійними, колір суміші також змінюється безупинно.

3. Колір суміші залежить тільки від кольорів компонентів, що змішуються, і не залежить від їх складів.

Зміст третього закону стає більш зрозумілим, якщо врахувати, що той самий колір (у тому числі і колір компонентів, що змішуються) може бути отриманий різними способами. Наприклад, певний компонент може бути отриманий, у свою чергу, змішуванням інших компонентів.

Кольори, які змішуються для отримання інших кольорів називаються основними. Кольори, отримані в результаті змішування основних кольорів, називаються складеними.

7.2.2 Стандартна колориметрична система RGB

Міжнародні колориметрические стандарти розробляє Міжнародна комісія з освітлення (МКО). В системі RGB-МКО основними кольорами є три спектральних (монохроматичних) кольору з наступними довжинами хвиль: для червоного (R) - 700 нм, для зеленого (G) - 546,1 нм і для синього (B) - 435,8 нм.

Одиниці виміру координат кольору, які в цій системі позначають через r ', g', b ', були обрані таким чином, щоб для опорного білого кольору типу E, що характеризується постійною спектральною інтенсивністю у видимій частині спектру, вони були б однакові,

7.3 Завдання

7.3.1 Завдання 1

Провести аналіз розглянутих колірних моделей:

- модель, параметри моделі;
- застосування;
- можливості перетворення (конвертації) в інші колірні координатні системи.

Результати аналізу оформити у вигляді таблиці (табл. 7.1).

Таблиця 7.1 – Макет таблиці аналізу колірних моделей

Модель (абреві-атура)	Параметри моделі	Де застосовується?	В які колірні координатні системи конвертується?	Правило конвертації
Колориметрична система Манселла	…	…	…	…
L*a*b*	…	…	…	…
XYZ	…	…	…	…
CMY(K)	…	…	…	…
RGB	…	…	…	…
YIQ	…	…	…	…

7.3.2 Завдання 2

Визначити координати кольору і кольоровості для заданих значень інтенсивностей базових кольорів опорного білого і шуканого кольору (табл. 8.2). Вказати за якою схемою (8 теорема Грассмана) визначаються шукані координати і чому.

Таблиця 8.2 – Варіанти завдання

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

7.3.3 Завдання 3

Реалізувати інтерфейс прикладної програми, яка передбачає функції:

- завантаження зображення в форматі bmp;
- перетворення формату з RGB в YIQ і назад для завантаженого зображення.

7.4 Приклад розв'язання завдання 2

7.4.1 Постановка завдання

Визначити координати кольору і кольоровості для заданих значень інтенсивностей базових кольорів опорного білого (A1 = 120, A2 = 100, A3 = 80) і шуканого кольору (C = 100, A1 = 30, A2 = 30, A3 = 40). Вказати за якою схемою визначаються шукані координати і чому.

7.4 Рішення завдання

Восьма аксіома Грассмана визначає схеми зрівнювання кольорів при різних умовах, не даючи відразу прямого рішення. Для того, щоб визначити відповідно з якою схемою зрівнювання кольорів треба визначати координати кольору і кольоровості, необхідно скористатися четвертої аксіомою Грассмана. В цієї аксіомі стверджується, що яскравість суміші кольорів (або інтенсивність шуканого кольору, в нашому випадку) дорівнює сумі інтенсивностей кольорів її складових. Перебір можливих поєднань чотирьох компонент - C, A1, A2 і A3 дає наступний результат:

$$I[C] = I[A1(C)] + I[A2(C)] + I[A3(C)].$$

На підставі цього, можна стверджувати, що зрівнювання квітів при заданих умовах відповідно до 8-ої аксіомою Грассмана виконується за першою схемою. На підставі цього, координати шуканого кольору З визначатимуться по наступних співвідношеннях:

$$T_1 = \frac{A_1(C)}{A_1(W)} = \frac{30}{120} \approx 0.25; \quad T_2 = \frac{A_2(C)}{A_2(W)} = \frac{30}{100} = 0.3;$$

$$T_3 = \frac{A_3(C)}{A_3(W)} = \frac{40}{80} = 0.5$$

Координати кольоровості визначаються як:

$$t_1 = \frac{T_1}{T_1 + T_2 + T_3} \approx \frac{0{,}25}{0{,}25 + 0{,}3 + 0{,}5} = \frac{0{,}25}{1{,}05} \approx 0{,}238$$

$$t_2 = \frac{T_2}{T_1 + T_2 + T_3} \approx \frac{0{,}3}{0{,}25 + 0{,}3 + 0{,}5} = \frac{0{,}3}{1{,}05} \approx 0{,}286$$

$$t_3 = \frac{T_3}{T_1 + T_2 + T_3} \approx \frac{0{,}5}{0{,}25 + 0{,}3 + 0{,}5} = \frac{0{,}5}{1{,}05} \approx 0{,}476.$$

7.5 Представлення результатів роботи

Результати виконання завдань надати викладачеві в формі звіту.

7.6 Контрольні питання

Назвіть способи вираження результуючого кольору.

Яке призначення міжнародних колориметричних стандартів. Хто їх розробляє?

Сформулюйте аксіоми Грассмана і закони змішування кольорів.

Охарактеризуйте відомі Вам колориметричні системи.

Охарактеризуйте координати кольору в системі - R, G, B.

Якими співвідношеннями пов'язані координати кольору і їх яскравості?

Охарактеризуйте колориметричну систему XYZ. Виділіть переваги системи XYZ.

Чим визначається зв'язок між системами координат RGB і XYZ?

6.5 Рекомендована література

При підготовці до заняття необхідно вивчити тему «Відтворення кольору на зображеннях із керуванням кольорів», використовуючи конспект лекцій, а також [5, с. 49-84]

8 ПРАКТИЧНЕ ЗАНЯТТЯ №8
Тема: Квантування сигналів яскравості

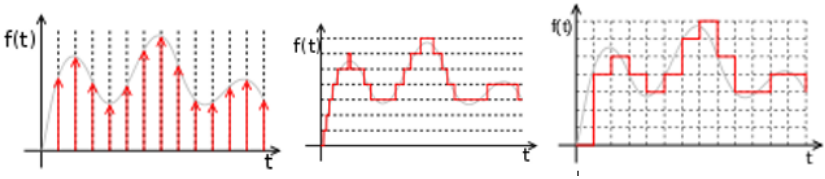
8.1 Мета і задачі

Визначення співвідношення сигнал/шум для сигналу відеозображення і числа рівнів квантування для забезпечення необхідної якості квантування сигналу.

8.2 Короткі теоретичні відомості

Процес перетворення аналогового сигналу на цифрову форму складається з трьох стадій: дискретизації, квантування та кодування (рис. 8.1).

Квантування - заміна величини відліку найближчим значенням з набору фіксованих величин. Щодо зображень це означає зменшення кількості значень атрибутів для кожного пікселя (зменшення кількості кольорів у зображенні). При цьому потрібно, щоб якість зображення погіршилася якнайменше.



а) б) в)

а) сигнал із дискретним часом;
б) квантований сигнал;
в) цифровий сигнал.

Рисунок 8.1 - Процес перетворення аналогового сигналу

Квантування потрібне для економії пам'яті; покращення властивостей послідовностей для стиснення; підготовки для подальшої обробки; додавання ефектів.

Пояснимо принцип квантування з прикладу квантування скалярної величини. Позначимо через f та $\hat{f}$ відповідно відлік дійсного скалярного сигналу до і після квантування. Передбачається, що f – випадкова величина із щільністю ймовірності $p(f)$, яка лежить у межах:

$$a_L \le f \le a_U, \qquad (8.1)$$

де a_L та a_U – відповідно нижня та верхня межі інтервалу.

У процесі квантування, вихідна величина порівнюється з набором порогових рівнів, і якщо відлік потрапляє у якийсь діапазон між двома пороговими рівнями, йому відповідно ставиться фіксований рівень квантування, тобто, якщо $d_j \le f \le d_{j+1}$, то $\hat{f} = r_j$.

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

Зазвичай як такий захід вибирають середньоквадратичну помилку:

$$\varepsilon = \mathrm{E}\{(f - \hat{f})^2\} = \int_{a_L}^{a_U} (f - \hat{f})^2 p(f) df = \sum_{j=0}^{J-1} \int_{d_j}^{d_{j+1}} (f - \hat{f})^2 p(\tag{8.2}$$

Якщо j велике, то щільність ймовірностей значень квантованого сигналу кожному інтервалі (d_j, d_{j+1}) можна вважати постійною та рівною $p(r_j)$, тоді:

$$\varepsilon = \sum_{j=0}^{J-1} p(r_j) \int_{d_j}^{d_{j+1}} (f - r_j)^2 \, df \tag{8.3}$$

і після обчислення інтервалів отримаємо:

$$\varepsilon = \frac{1}{3} \sum_{j=0}^{J-1} p(r_j) \left[(d_{j+1} - r_j)^3 - (d_j - r_j)^3 \right] \tag{8.4}$$

Оптимальне рішення можна знайти, вирішуючи задачу про мінімум помилки ε як функції r_j:

$$\frac{d\varepsilon}{dr} = 0 \tag{8.5}$$

$$r_j = \frac{d_{j+1} + d_j}{2} \tag{8.6}$$

Таким чином, при допущеннях (8.1-8.6) оптимальним положенням рівня квантування є середина інтервалу між сусідніми пороговими рівнями (рис. 8.2).

Підставивши (8.6) до (8.4), отримаємо:

$$\varepsilon = \frac{1}{12}\sum_{j=0}^{J-1} p(r_j)\left[d_{j+1} - d_j\right]^3 \qquad (8.7)$$

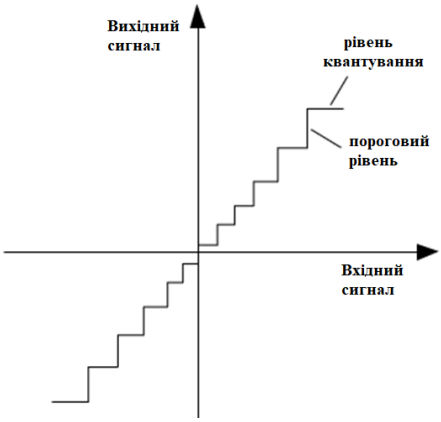


Рисунок 8.2 - Графік функції перетворення для операції квантування у загальному вигляді

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

$$d_j = \frac{(a_U - a_L)\int_{a_L}^{a_j}[p(f)]^{-\frac{1}{3}}df}{\int_{a_L}^{a_U}[p(f)]^{-\frac{1}{3}}df} + a_L \qquad (8.8)$$

, де

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

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

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

Якщо кількість рівнів квантування ***j*** невелика, то слід використовувати точний вираз для помилки (8.1).

Диференціюючи його за змінними ***dj*** і ***rj*** і прирівнюючи похідні до нуля, отримуємо:

$$\frac{\partial \varepsilon}{\partial d_j} = (d_j - r_j)^2 \varphi(d_j) - (d_j - r_{j-1})^2 p(d_j) = 0$$

$$\frac{\partial \varepsilon}{\partial r_j} = 2 \int_{d_j}^{d_{j+1}} (f - r_j) p(f) df = 0$$

Після перетворень:

$$r_j = 2d_j - r_{j-1}$$

$$r_j = \frac{\int_{d_j}^{d_{j+1}} f_p(f) df}{\int_{d_j}^{d_{j+1}} p(f) df}$$

8.3 Завдання

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. 8.1)

Таблиця 8.1 - Вихідні дані до завдання

Варіант	Кількість рівнів квантування	Сигнал/шум, Дб	Закон розподілу
1	4	20	лінійний
2	5	25	Гаусса

3	6	36	лінійний
4	7	35	Гаусса
5	8	40	лінійний
6	10	20	Гаусса

8.4 Приклад розв'язання завдання

Визначити співвідношення сигнал/шум (Дб) для нормованого в межах від 0 до 1 сигналу відеозображення, при лінійному (рівномірному) законі розподілу одержуваних при його обробці значень, з кількістю рівнів квантування 6. При необхідності визначити мінімально необхідну кількість рівнів квантування для умови, що співвідношення сигнал/шум має бути не гірше 30дБ.

Рішення завдання було виконано в системі Mathcad (рис. 8.3).

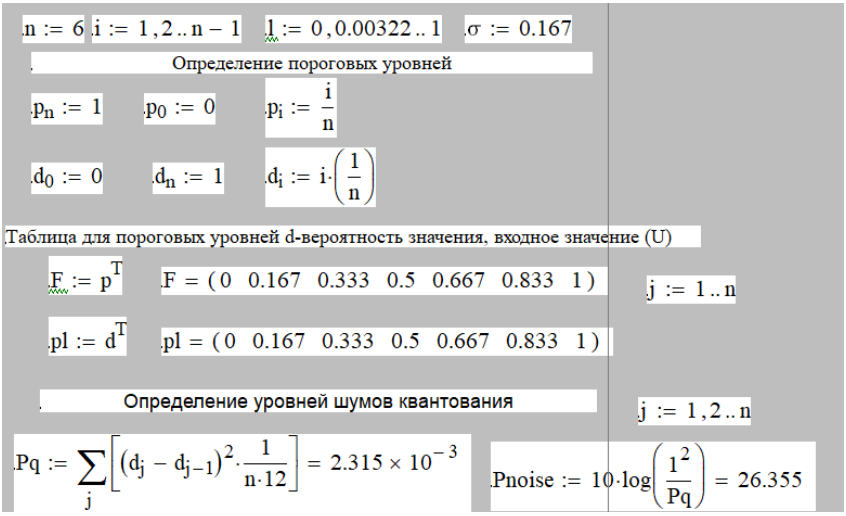


Рисунок 8.3 - Фрагмент робочого вікна Mathcad з рішенням завдання

8.5 Представлення результатів роботи

Результати виконання завдання надати викладачеві в формі звіту.

8.6 Контрольні питання

Охарактеризуйте цифрові методи обробки і передачі зображень.
Для чого здійснюється квантування?
У чому полягає квантування?
Назвіть основні характеристики квантування сигналів зображень.
Як можна визначити оптимальне положення порогових рівнів?

8.7 Рекомендована література

При підготовці до заняття необхідно вивчити тему «Квантування цифрових зображень», використовуючи конспект лекцій, а також [2, с. 11-21; 5, с. 150-154; 12].

9 ПРАКТИЧНЕ ЗАНЯТТЯ №10
Тема: Методи кодування зображень.

9.1 Мета і задачі

Вивчення характеристик та особливостей алгоритмів стиснення. Програмна реалізація алгоритмів стиснення і декомпресії послідовності яскравостей зображення.

9.2 Короткі теоретичні відомості
9.2.1 Актуальність завдань архівації графіки

Актуальність алгоритмів архівації графіки обумовлена особливостями зображення як типу даних:

- зображення займають набагато більше місця в пам'яті, ніж текст;
- особливість людського зору дозволила створити спеціальні алгоритми стиснення, орієнтовані тільки на зображення;
- при створенні алгоритму компресії графіки ми використовуємо особливості структури зображення (зображення має надмірність у 2-х вимірах, тобто. як правило, сусідні точки, як по горизонталі, так і по вертикалі, у зображенні близькі за кольором. Крім того, ми можемо використовувати подібність між колірними площинами R, G і B у наших алгоритмах, що дає можливість створити ще ефективніші алгоритми)..

Щоб говорити про алгоритми стиснення зображень, необхідно визначити наступні питання:

1. Які критерії ми можемо запропонувати для порівняння різних алгоритмів?
2. Які класи зображень існують?
3. Які класи додатків, що використовують алгоритми компресії графіки, існують, і які вимоги вони висувають до алгоритмів?

Основною величиною, що характеризує метод стиснення є **к**оефіцієнт стиснення (***Кс***), визначає у скільки разів файл, який зберігає стислий зображення, менше файлу, що зберігає вихідне зображення. величини ***V1*** та ***V2*** виражаються в байтах. ***Кс*** – величина безрозмірна.

$$K_C = \frac{V_1}{V_2}$$

9.2.2 Алгоритми стиснення зображень способом кодування серій (*RLE - Run Length Encoding*)

Всі алгоритми *RLE* засновані на дуже простій ідеї: групи елементів, що повторюються, замінюються на пару (кількість повторів, повторюваний елемент). Розглянемо цей алгоритм з прикладу послідовності біт. У цій послідовності чергуватимуть групи нулів та одиниць. Причому в групах найчастіше буде більше одного елемента. Тоді послідовності 11111 000000 11111111 00 буде відповідати наступний набір чисел 5 6 8 2. Ці числа позначають кількість повторень (відлік починається з одиниць), але ці числа також необхідно кодувати. Вважатимемо, що кількість повторень лежить у межах від 0 до 7 (тобто нам вистачить 3 біт для кодування числа повторів). Тоді розглянута вище послідовність кодується наступною послідовністю чисел 567012. Легко підрахувати, що для кодування вихідної послідовності потрібно 21 біт, а в стислому за методом *RLE* вигляді ця послідовність займає 18 біт.

Хоч цей алгоритм і дуже простий, але його ефективність порівняно низька. Більше того, у деяких випадках застосування цього алгоритму призводить не до зменшення, а до збільшення довжини послідовності. Наприклад розглянемо наступну послідовність 111 0000 11111111 00. Відповідна їй RL-послідовність виглядає так: 3 4 7 0 1 2. Довжина вихідної послідовності – 17 біт, довжина стиснутої послідовності – 18 біт.

Цей алгоритм найефективніший для чорно-білих зображень. Також він часто використовується як один із проміжних етапів стиснення більш складних алгоритмів.

Кращий, середній і найгірший коефіцієнти стиснення - 1/32, 1/2, 2/1.

Кодований режим стиснення для бітової картки в режимі 8 біт піксель ***RLE-8*** полягає в наступному.

Одиниця інформації складається із 2-х байт. Перший байт визначає число пікселів послідовності з індексом кольору, який полягає у 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 беззнакове значення зсуву по горизонталі та вертикалі до наступної точки з поточної позиції.

Абсолютний режим визначається керуючою послідовністю, в якій перший байт – нулі, а значення другого байта лежить у діапазоні від 0x03 ÷ 0xFF. Другий байт визначає довжину наступної послідовності, у якій кожен байт містить індекс кольору одного пікселя. Кожна серія має бути вирівняна по 2-байтному кордоні. Приклади використання кодових послідовностей представлені у таблиці 9.1.

Таблиця 9.1 - Приклади використання кодових послідовностей з *RLE-8*

Стислі дані	Розгорнуті 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

9.2.3 Алгоритм Лемпеля-Зіва (LZ-compression)

Стиснення, на відміну RLE, здійснюється за рахунок однакових послідовностей байт. Суть полягає у наступному. Пакувальник постійно зберігає деяку кількість останніх оброблених символів у буфері. В процесі обробки вхідного потоку символи, що знову надійшли, потрапляють в кінець буфера, зсуваючи попередні символи і витісняючи найстаріші. Розміри цього буфера (ковзний словник) варіюються в різних реалізаціях.

Після побудови хеш-таблиць, виділяють (шляхом пошуку в словнику) найдовший початковий підрядок вхідного потоку, що збігається з одним з підрядків у словнику, і видають на вихід пару (***length*** -довжина знайденого у словнику підрядка, ***distance***- відстань від нього до вхідного підрядка). Якщо такий підрядок не знайдено, в вихідний потік просто копіюється черговий символ вхідного потоку.

Існує велике сімейство ***LZ***-подібних алгоритмів, що розрізняються, наприклад, методом пошуку повторюваних ланцюжків. Один з досить простих варіантів цього алгоритму передбачає, що у вхідному потоці йде або пара ***<лічильник, зміщення щодо поточної позиції>***, або просто ***<лічильник>*** байт, що пропускаються і ***самі значення байтів***.

При розархівації для пари ***<лічильник, зміщення>*** копіюються ***<лічильник>*** байт зі вихідного масиву, отриманого в результаті

розархівації, на ***<зміщення>*** байт раніше, а ***<лічильник>*** (число дорівнює лічильнику) значень байт, що пропускаються, просто копіюються у вихідний масив з вхідного потоку.

Алгоритм Лемпеля-Зіва-Велч (Lempel-Ziv-Welch, LZW замінює рядки символів на деякі коди. Це робиться без аналізу вхідного тексту. Натомість при додаванні кожного нового рядка символів переглядається таблиця рядків. Стиснення відбувається, коли код замінює рядок символів. Коди, що генеруються ***LZW***-алгоритмом, можуть бути будь-якої довжини, але вони повинні містити більше біт, ніж одиничний символ. Перші 256 кодів (коли використовуються 8-бітові символи) за промовчанням відповідають стандартному набору символів. Інші коди відповідають оброблюваним алгоритмом рядків.

LZW-алгоритм відрізняють висока швидкість роботи як при упаковці, так і при розпакуванні, низькі вимоги до пам'яті та проста апаратна реалізація.

Недоліком є низька ступінь стиснення у порівнянні зі схемою двоступеневого кодування.

Існує велике сімейство LZW - подібних алгоритмів.

Коефіцієнти стиснення: 1/1000, 1/4, 7/5. *Коефіцієнт 1/1000 досягається тільки на одноколірних зображеннях розміром більше 4 Мб.* Ситуація, коли алгоритм збільшує зображення, зустрічається вкрай рідко. Стиснення забезпечується за рахунок однакових підланцюжків в потоці.

LZW універсальний - саме його варіанти використовуються в звичайних архіваторах, реалізований в форматах GIF, TIFF и TGA.

9.3 Завдання

Виконайте процес стиснення і декомпресії для зазначеної послідовності яскравостей зображення, відповідно до заданого типу алгоритму. Вкажіть характеристики і особливості використаного алгоритму.

Таблиця 10.1 - Вихідні дані до завдання

Варіант	Алгоритм	Послідовність
1	RLE-8	44 44 44 44 4c 4c 44 44 44 44 44 4c c4 44 4c 44 44 44 44 44 44 c4 4c 44 44 44 44 44 4c 44 c4 44
2	RLE	44 44 44 44 4c 4c 44 44 44 44 44 4c c4 44 4c 44 44 44 44 44 44 c4 4c 44 44 44 44 44 4c 44 c4 44

3	LZ	44 44 44 44 4c 4c 44 44 44 44 44 4c c4 44 4c 44 44 44 44 44 44 c4 4c 44 44 44 44 44 4c 44 c4 44
4	LZW	44 44 44 44 4c 4c 44 44 44 44 44 4c c4 44 4c 44 44 44 44 44 44 c4 4c 44 44 44 44 44 4c 44 c4 44

9.4 Приклад розв'язання завдання

Процес стиснення і декомпресії виконаний для наступній послідовності яскравостей зображення: 44 44 44 44 4c 4c 44 44 44 44 44 4c c4 44 4c 44 44 44 44 44 44 c4 4c 44 44 44 44 44 4c 44 c4 44, відповідно до алгоритму RLE-4.

Відповідно до алгоритму, вхідна послідовність перетворюється в вихідну з використанням спеціальних керуючих кодів, що позначають наступні дії:

00 … … … 00 – послідовність даних що не повторюються;

00 01 – кінець бітового образу;

00 02 – зміщення в бітовому образі.

Тоді, вихідна послідовність перетворюється до наступного вигляду (з урахуванням того, що один байт містить інформацію про 2 точки):
09 44 04 C4 0A 44 00 4C C4 44 4C 00 0C 44 00 C4 4C 00 0A 44 00 4C 44 C4 44 00 00 01.

Вихідна послідовність має довжину 32 байту, а вихідна - 28 байт, тобто, коефіцієнт стиснення складає 32/28≈1,14.

9.5 Представлення результатів роботи

Результати виконання завдання надати викладачеві в формі звіту.

9.6 Контрольні питання

Чим зумовлена поява алгоритмів архівації графіки?

Для чого і як можна класифікувати всі зображення?

Для чого і як можна класифікувати додатки, що використовують алгоритми компресії?

Поясніть вимоги додатків до алгоритмів компресії.

Поясніть критерії порівняння алгоритмів.

Охарактеризуйте алгоритми стиснення без втрат.

9.7 Рекомендована література

При підготовці до заняття необхідно вивчити тему «Алгоритми стиснення зображень», використовуючи конспект лекцій, а також [2, с. 21-26; 5, с. 473-546; 12].

10 ПРАКТИЧНЕ ЗАНЯТТЯ №10
Тема: Покращення зображень методами ковзної фільтрації.

10.1 Мета і задачі

Вивчення методів просторово фільтрації зображень. Визначення довжини медіанного лінійного фільтра для усунення імпульсних перешкод для зазначених сигналів.

10.2 Короткі теоретичні відомості

Процес виділення певних компонентів просторового складу зображення з їх подальшим посиленням, ослабленням або видаленням називають **просторовою фільтрацією**. Фільтрація зображення може здійснюватися лінійними або нелінійними законами, мати локальний або глобальний характер, виконуватися безпосередньо в просторової області або в області просторових частот по рекурсивної або нерекурсивною схемою або адаптивно. Фільтрація може застосовуватися до безперервних, дискретних або цифрових зображень, які можуть бути багатозональними, кольоровими, напівтоновими, бінарними.

Традиційно просторова фільтрація зображень застосовується для вирішення завдань поліпшення зображень (*Image Enhancement*) - це процес представлення зображення в формі, більш придатної для його подальшого використання: завдання видалення з зображення просторової перешкоди і шуму; завдання підкреслення меж перепадів яскравості; завдання зміни контрасту і т.і. Також просторова фільтрація застосовується для вирішення завдань відновлення зображень (*Image Restoration*) - це процес отримання такого зображення, яке близьке до ідеального зображення настільки, наскільки це можливо.

Будь-який процес просторової фільтрації напівтонового зображення можна розглядати як певне перетворення. У загальному вигляді можна описати двома рівняннями: операційним рівнянням (10.1) в просторової області, де ***A*** - певний оператор, за допомогою якого виконується просторове перетворення зображення ***f (x, y)*** в "відфільтроване" зображення ***g(x, y)***; та рівнянням (10.2) в просторово-частотній області, де ***B***-певний оператор, за допомогою якого виконується перетворення просторового спектра $F(w_x, w_y)$ зображення ***f(x,y)*** в просторовий спектр $G(w_x, w_y)$ "відфільтрованого" зображення ***g(x, y)***.

$$g(x,y)=\mathbf{A}[f(x,y)] \qquad (10.1)$$

$$G(\omega_x, \omega_y) = \mathbf{B}[F(\omega_x, \omega_y)] \qquad (10.2)$$

$\mathbf{A}=\mathbf{F}^{-1}\mathbf{B}\mathbf{F}^{1}$, $\mathbf{B}=\mathbf{F}^{1}\mathbf{A}\mathbf{F}^{-1}$.

10.3 Завдання

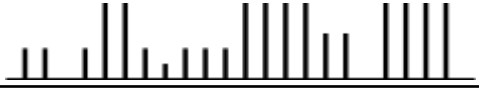
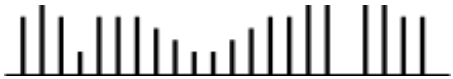
10.3.1 Завдання 1

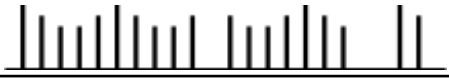
Вивчити методи просторової фільтрації. Провести аналіз завдань просторової фільтрації і способів вирішення цих завдань.

10.3.2 Завдання 2

Вкажіть, де на Вашу думку знаходиться в зазначеній послідовності імпульсні перешкоди, визначте довжину медіанного лінійного фільтра для усунення імпульсних перешкод для зазначених сигналів (табл. 10.1). Пояснити, на підставі чого отримані такі результати. Вкажіть, якими альтернативними способами можна домогтися вирішення поставленого завдання.

Таблиця 10.1 - Вихідні дані до завдання

Варіант	Послідовність
1	
2	
3	
4	

5	

10.4 Приклад розв'язання завдання

Для наведеного варіанту 1 цифровий послідовності можна припустити, що перешкоди імпульсного виду знаходяться в позиціях 5, 8 і 18.

Оскільки дана помилка виникає нерегулярно, то вона може бути усунена за допомогою лінійного медіанного фільтра довжиною 3 одиниці, так як при роботі медіанного фільтра усуваються імпульсні перешкоди довжиною не перевищує половини довжини фільтра. Обробка медіанного фільтром, в даному випадку, найбільш ефективний спосіб, так як при обробці ковзаючими фільтрами, що усереднюють перешкода повністю не усувається з зображення, а тільки послаблюється.

Після обробки вихідна послідовність матиме вигляд, представлений на рисунку 10.1.

Рисунок 10.1 - Вихідна послідовність після обробки

10.5 Представлення результатів роботи

Результати виконання завдання надати викладачеві в формі звіту.

10.6 Контрольні питання

Яке призначення фільтрації?
Пояснити алгоритм ковзної фільтрації.
Пояснити алгоритм придушення шуму (усереднення).
Що являє собою медіанний фільтр?

10.6 Рекомендована література

При підготовці до заняття необхідно вивчити тему «Методи фільтрації зображень», використовуючи конспект лекцій, а також [1, с. 34-53; 2, с. 26-27; 5, с.215-277].

11 ПРАКТИЧНЕ ЗАНЯТТЯ №11
Тема: Простеження контурів. Класичний алгоритм «жука».

11.1 Мета і задачі

Вивчення алгоритмів простеження контурів. Програмна реалізація алгоритму «жука».

11.2 Короткі теоретичні відомості

11.2.1 Огляд основних методів контурного аналізу для виділення контурів об'єктів

Аналіз зображення - це процес виділення потрібної інформації з зображення за допомогою автоматичних систем.

Ознака зображення - це його найпростіша характеристика або властивість. Деякі ознаки є природними в тому розумінні, що вони встановлюються візуальним аналізом зображення, тоді як інші, так звані штучні ознаки, виходять в результаті його спеціальної обробки і вимірювань. Природні ознаки: світлота (яскравість), текстура різних областей зображення і форма контурів об'єктів

Зазвичай аналіз зображення включає в себе отримання зовнішнього контуру зображених об'єктів і запис координат точок цього контуру. Частіше за все потрібно отримати зовнішній контур у вигляді замкнутої кривої або сукупності відрізків дуг. Є три загальних підходи до подання кордонів об'єкта:

- апроксимація кривих;
- простеження контурів;
- зв'язування точок перепадів.

Контури зображень є областями з високою концентрацією інформації, яка слабо залежить від кольору і яскравості.

При розгляді будь-якого об'єкта в свідомості людини формується зоровий образ. При сприйнятті око відстежує лінію контуру, що призводить до створення образу з характерними деталями. Існує думка, що при сприйнятті в свідомості людини формуються два образи: контуру і внутрішньої частини зображення. Необхідним етапом сприйняття вважається сканування по лінії контуру.

Контурний аналіз є сукупністю методів виділення, опису та перетворення контурів зображень та розпізнавання зорових образів. Контур цілком визначає форму зображення і містить всю необхідну інформацію для розпізнавання зображень за їх формою. Такий підхід дозволяє не розглядати внутрішні точки зображення і тим самим значно

скоротити обсяг інформації, що переробляється. Наслідком цього часто стає можливість забезпечення роботи системи в режимі реального часу.

Під контуром розуміється просторово-протяжний розрив, перепад або стрибкоподібна зміна значень яскравості. Існує ряд проблем при виділенні контурів зображення:

- розриви контуру в місцях, де яскравість змінюється не дуже швидко;
- наявність помилкових контурів внаслідок шуму на зображенні;
- широкі контурні лінії через розмитість або шуму.

Контурний аналіз дозволяє описувати, зберігати, порівнювати і робити пошук об'єктів, представлених у вигляді своїх зовнішніх обрисів - контурів, а також ефективно вирішувати основні проблеми розпізнавання образів - перенесення, поворот і зміна масштабу зображення об'єкта

Методи контурного аналізу:

- метод активних контурів;
- метод активних контурів без попереднього виділення кордонів;
- детектор границь Кенні;
- простеження контурів;
- кластеризація;
- локальна обробка;
- аналіз за допомогою графів.

11.2.2 Виділення контурів на бінарному зображенні. «Класичний алгоритм жука»

Це задача двоїста виділенню зв'язкових областей. При цьому вважається, що контурна точка області - це будь-яка точка, що належить області, в околиці якої є точки, які не належать даній області.

Алгоритми, що відстежують, засновані на тому, що на зображенні відшукується об'єкт (перша точка об'єкту, що зустрілася) і контур об'єкту відстежується і векторизується. Перевагою даних алгоритмів є їх простота. До недоліків можна віднести їх послідовну реалізацію і деяку складність при пошуку і обробці внутрішніх контурів.

Приклад такого алгоритму - "алгоритм обходу контуру", або "алгоритм жука", - наведено на рис. 11.1. "Жук" починає рух з білою області у напрямку до чорної. Як тільки він потрапляє на чорний елемент, він повертає ліворуч і переходить до наступного елементу. Якщо цей елемент білий, то жук повертається направо, інакше - наліво. Процедура повторюється до тих пір, поки жук не повернеться у вихідну

точку. Координати точок переходу з чорного на біле і з білого на чорне і описують границю об'єкту.

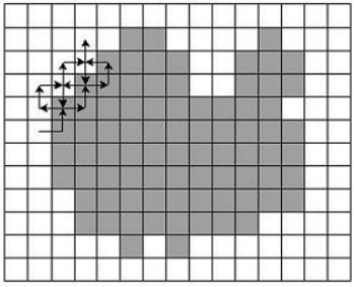


Рисунок 11.1 – Схема роботи алгоритму «жука»

За «класичним алгоритмом жука» може бути пропущений контур, що примикає (рис. 11.2).

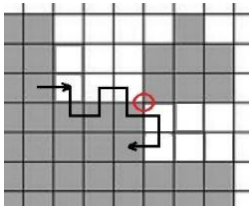


Рисунок 11.2 – Приклад пропущення контуру, що примикає у алгоритму «жука»

Щоб цього не сталося біля кожної контурної точки відбувається круговий обхід по 8 напрямках, щоб виявити точку, яка можливо примикає до контуру (рис. 11.3). Якщо така точка виявлена, то вона запам'ятовується, щоб повернутися до неї для обходу пропущеного контуру.

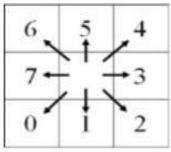
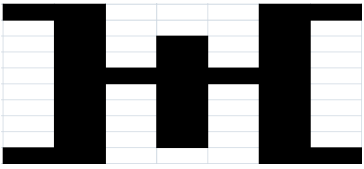


Рисунок 11.3 – Напрямки обходу

11.2.3 Постановка задачі

Нехай є зображення, яке математично описується наступним чином (рис. 11.4), де i – довжина зображення, j – ширина зображення.



$$Image_{ij} = \begin{bmatrix} 1100011 \\ 0100010 \\ 0101010 \\ 0101010 \\ 0111110 \\ 0101010 \\ 0101010 \\ 0101010 \\ 0101010 \\ 1100011 \end{bmatrix},$$

Рисунок 11.4 – Вихідне зображення

Потрібно знайти контур на зображенні, складеного з пікселів зі значенням 1.

11.2.4 Рішення задачі

На першому кроці починаємо переглядати стовпці зображень з першого пікселя по останній, якщо зустрічається один чи кілька поспіль пікселів (група пікселі) зі значенням один, то запам'ятовуємо їх координати.

На другому кроці серед груп пікселів двох сусідніх стовпців шукаємо точки дотику. Точки дотику визначаються за такими правилами (11.1), де x_1 и y_1 – координати першої групи пікселів, x_2 и y_2 – координати другої групи пікселів.

$$\begin{cases} |x_1 - x_2| < 2 \\ |y_1 - y_2| < 2 \\ x_1 > x_2 and y_1 < y_2 \\ x_1 < x_2 and y_1 > y_2 \end{cases}, \quad (11.1)$$

Приклад виконання другого кроку показано на рисунку 11.5.

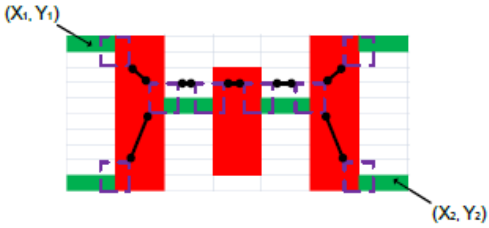


Рисунок 11.5 – Приклад виконання алгоритму

11.4 Завдання

Вивчити представлений алгоритм. Програмно реалізувати завдання, що описано в пунктах 11.2.3, 11.2.4.

11.5 Представлення результатів роботи

Результати виконання завдання надати викладачеві у формі звіту.

11.6 Контрольні питання

Які існують підходи до подання меж об'єкту?

Які методи відносять до методів контурного аналізу?

Наведіть приклад простеження контуру для об'єкта, заданого в аналоговій формі.

Охарактеризуйте властивості алгоритму простежування контурів.

Поясніть на прикладі послідовність дій за «класичним алгоритмом жука».

11.7 Рекомендована література

При підготовці до заняття необхідно вивчити тему «Методи фільтрації зображень», використовуючи конспект лекцій, а також [7, с. 499-520; 3, с. 72-76; 1, с.101-114].

12 ПРАКТИЧНЕ ЗАНЯТТЯ №12
Тема: Розпізнавання цифрових зображень

12.1 Мета і задачі заняття

Практичне заняття проводиться у формі семінару дослідницького типу. Студенти готують невеликі доповіді, які обговорюють учасники семінару.

Семінар призначений для поглибленого вивчення дисципліни, опанування методології наукового пізнання, розвитку у студентів культури наукового мислення.

Основною метою семінару є систематизація та узагальнення знань з теми «Розпізнавання цифрових зображень», формування умінь працювати з додатковими джерелами інформації, зіставляти і порівнювати точки зору, висловлювати свою точку зору.

12.2 Постановка завдання

Вивчити методи і засоби аналізу і розпізнавання цифрових зображень (образів).

Виділити одне з актуальних завдань розпізнавання образів, вивчити можливості його вирішення на основі опублікованих досліджень.

Висловити свою думку щодо найбільш доцільного підходу.

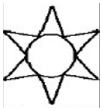
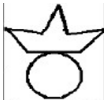
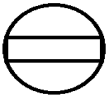
Представити роботу у формі презентації доповіді, або оформити звіт.

Представляючи роботу виділити: актуальне завдання розпізнавання образів, що досліджується; способи його вирішення з посиланнями на джерела; висновки. Обсяг звіту – 3-5 стор.

РЕКОМЕНДОВАНА ЛІТЕРАТУРА ТА ІНШІ ДЖЕРЕЛА ІНФОРМАЦІЇ

1. Вовк С.М. Методи обробки зображень та компютерний зір : навч. посіб. / С.М. Вовк, В.В. Гнатушенко, М.В. Бондаренко. – Д. : ЛІРА, 2016. – 148 с.
2. Ежова К.В. Моделирование и обработка изображений. Учебное пособие. — СПб : НИУ ИТМО, 2011. — 93 с.
3. Кобилін О. А. Методи цифрової обробки зображень: навч. посібник. / О. А. Кобилін, І. С. Творошенко – Харків : ХНУРЕ, 2021. – 124 с.
4. Кононюк А. Е. Основы фундаментальной теории искусственного интеллекта : [В 20 кн.] / А. Е. Кононюк. – Киев : Освіта України, 2017– . – ISBN 978-966-373-693-8. – Кн. 3: Зрительное восприятие изображений искусственным интеллектом. – Ч. 2 – 2017.
5. Красильников Н. Н. Цифровая обработка 2D- и 3D-изображений: учеб. пособие. - СПб.: БХВ-Петербург, 2011. - 608 с.
6. Прэтт У. Цифровая обработка изображений: пер. с англ.,-М.: Мир, 1982.- в 2-х кн.
7. Старовойтов В. В. Получение и обработка изображений на ЭВМ : учебно-методическое пособие / В.В. Старовойтов, Ю.И. Голуб. – Минск : БНТУ, 2018. – 204 с.
8. Gimp [Electronic resource] – Mode of access: http://www.gimp.org/
9. GIMP Documentation [Electronic resource] – Mode of access: https://docs.gimp.org/2.10/ru/filters.html
10. Документация по семейству продуктов Visual Studio [Electronic resource] – Mode of access: https://docs.microsoft.com/ru-ru/visualstudio/?view=vs-2022
11. Уроки INKSCAPE [Electronic resource] – Mode of access: http://uart.at.ua/publ/uroki_inkscape/16
12. Вікіпедія. Вільна енциклопедія [Electronic resource] – Mode of access: https://uk.wikipedia.org/wiki

Додаток А
Варіанти завдання до практичного заняття №2

№ варіанту	Задана сукупність фігур	№ варіанту	Задана сукупність фігур
1		8	
2		9	
3		10	
4		11	
5		12	
6		13	
7		14	

Додаток Б
Варіанти зразків до практичного заняття №5

Варіант	Зразок	Варіант	Зразок
1		8	
2		9	
3		10	
4		11	
5		12	
6		13	
7		14	

Навчальне видання

МЕТОДИЧНІ ВКАЗІВКИ
до практичних занять з дисципліни
"ПРОГРАМНІ ЗАСОБИ ОБРОБКИ ЗОБРАЖЕНЬ"
(для здобувачів вищої освіти за спеціальністю 122
"Комп'ютерні науки")
(Електронне видання)

Укладачі:
Л.О. ШУМОВА,
В.М. БАРБАРУК

Оригінал - макет

Підписано до друку _____
Формат 60*84$^{1}/_{16}$ Папір типограф. Гарнітура Times.
Друк офсетний. Умов. друк. арк. 2. Обл.-вид. арк. _____.
Тираж __ прим. Вид. № _____. Замов № _____. Ціна договірна.

Видавництво Східноукраїнського національного університету
імені Володимира Даля

Свідоцтво про реєстрацію: серія ___ № _____ від __.__._____ р.
Адреса університета: просп. Центральний 59-А
м. Сєвєродонецьк, 93400, Україна
e-mail: vidavnictvoSNU.ua@gmail.com.