УДК 004.932.2

О.В.Самощенко, С.В.Ходус
Донецький національний технічний університет
samoshchenko@cs.donntu.edu.ua

Передискретизація зображень у гексагональній системі координат hex2

Розглядається перетворення зображень, які присутні у гексагональній системі координат, у псевдогексагональному виді, що дозволяє представити результати візуально без використання гексагональної решітки або апаратних гексагонально-орієнтованих пристроїв. Досліджено та перевірено властивості зображень, які показують перевагу гексагональної структури у порівнянні зі стандартною прямокутною структурою пікселів

Гексагонал, координатна система, решітка пікселів, псевдопіксель, фільтр визначення краю

Вступ

Численні дослідження підтверджують привілеї використання гексагональної решітки при перетворенні зображень. Багато досліджень також присвячено перетворенням між гексагональною та стандартною прямокутною решіткою, тобто, проведенням звичайних операцій у декартових координатах, наприклад перетворенням Фур'є, градієнту, дифузії, фільтрації відповідно до гексагональної мозаїки і визначенню системи координат в гексагональній області [1-3]. Подання системи координат повинно мати щонайменше наступні властивості:

- повнота – достатнє представлення будь-якої точки на площині (у двомірному просторі);
- унікальність – має бути тільки одна пара чисел, яка описує положення будь-якої точки в просторі;
- конвертованість – просте перетворення до декартових координат та зворотно;
- ефективність – зручне подання за критеріями технічної оцінки.

Представлення координатної системи hex2

Система координат, яку обрано і аналізуємо, показана на рисунку 1. Деякі аспекти цій системі координат були досліджені раніше. Наприклад, Лучак та Розенфельд [4] визначили відстані гексагональної метрики за допомогою такої системи, а Белл та ін. [5] використовували складні формалізми для вирішення геометричних перетворень.

На рисунку 1 показано, що координати кожного пікселя представлено горизонтальним відхиленням вгору і вправо та пов'язано з парою одиничних векторів *u* та *v*. Для подальшого використання цю систему координат названо "hex2".

Точка R^2 позначається, як $p = u\boldsymbol{u} + v\boldsymbol{v}$, де невиділені жирним шрифтом символи вказують на величину в напрямку одиничного вектора, який позначається жирним шрифтом. У випадку двох або більше точок позначатиме різні вектори використанням індексів, тобто $p_i = u_i\boldsymbol{u} + v_i\boldsymbol{v}$.

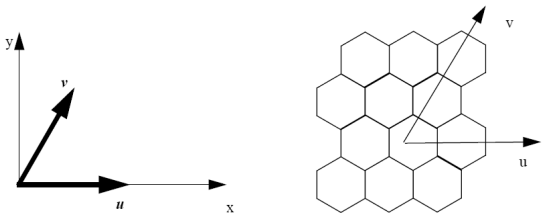


Рисунок 1 – Гексагональна система координат

Також буде використовуватися позначення вектор-стовпець для таких точок:

$$p_i = [u_i, v_i]^T .$$

Для звичайного декартового представлення справедливо

$$p_i = [u_i, v_i]^T = [x_i, y_i]^T$$

з відповідними значеннями *u, v, x, y*.

Аналіз системи hex2 показує наступне.

Лема 1: Будь-яка впорядкована пара [*u,v*] відповідає рівно одній парі [*x,y*].

Доказ: Використовуючи просту тригонометрію і відзначаючи, що косинус 60 градусів дорівнює 1/2, можна вивести:

$$x = u + \frac{v}{2}, \quad y = \frac{v\sqrt{3}}{2} .$$

Лема 2: Будь-якій впорядкованій парі декартових координат [*x,y*] відповідає рівно одна пара [*u,v*] в гексагональній системі.

Доказ: За рішенням попередніх рівнянь для [*u,v*] слідує:

$$u = x - \frac{y}{\sqrt{3}}, \quad v = 2\frac{y}{\sqrt{3}} .$$

Безліч векторів $\lambda_1,\lambda_2, \ldots \lambda_d$ приймається основою для векторного простору R^d якщо будь-який вектор в R^d може бути записаний у вигляді лінійної комбінації $\lambda_1,\lambda_2, \ldots \lambda_d$. Якщо λ_i є ортонормованим, це є достатнім для твердження, що вектори утворюють базис. Однак, u та v нормалізуються, але вони не ортогональні. Щоб переконатися в цьому, представляють u та v в декартових координатах і виражають їх внутрішню величину:

$$u = x, \quad v = \frac{1}{2}x + \frac{\sqrt{3}}{2}y, \quad u^T v = [1,0]\begin{bmatrix}\frac{1}{2}\\ \frac{\sqrt{3}}{2}\end{bmatrix} = \frac{1}{2}$$

Теорема: вектори u та v формують базис для R^2.

Доказ: Оскільки x та y є основою для R^2, можна представити будь-яку точку p в R^2 як впорядковану пару $p = [x,y]^T = x\mathbf{x}+y\mathbf{y}$. Але з леми 2 маємо:

$$p = xu + y\frac{2v-u}{\sqrt{3}} = \left(x - \frac{y}{\sqrt{3}}\right)u + \frac{2y}{\sqrt{3}}v$$

Таким чином, будь-яка точка в R^2 може бути вказана як зважена сума u та v, що й треба було довести.

Зберігання даних для системи hex2

Множина даних в системі hex2 не має природного вирівнювання на прямокутні масиві в пам'яті, як у стандартній системі координат. Щоб дослідити це, необхідно розглянути рисунок 2, де наведено координати u,v.

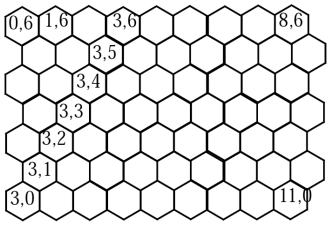


Рисунок 2 – Система нумерації пікселів через u,v, яка не містить негативних індексів

Система нумерації пікселів була обрана таким чином, щоб на гексагональній сітці не було негативних координат. На цьому масиві пікселів розмірністю *7x9*, нумерація йде від $u = 0$ до $u = 11$, а нумерація на v коливається від 0 до 7. Необхідно відзначити, що не існує пікселів у диапазоні [11,2] ... [11,6]. З огляду на розподіл пам'яті таким марнотратним способом, можна було б задекларувати масив пам'яті з горизонтальною віссю u з номерами від 0 до 11, а по вертикальній осі v - від 0 до 6, тобто маючи в цілому 84 точки, у яких використовуються лише 63 пікселі. Загалом, для зображення з nr рядків і nc стовпців, така схема розподілу пам'яті матиме коефіцієнт використання:

$$kus = \frac{nr \times nc}{nr \times \left(nc + \frac{nr}{2}\right)}$$

Порівнюючи з квадратним масивом, можна спрощено виразити відношення пам'яті, що використовується, до пам'яті, що виділяється, як 2/3. Тим не менш, прості та ефективні схеми розподілу пам'яті можуть бути розраховані на використання всієї виділеної пам'яті, і робити це навіть швидше, ніж двійково індексовані масиви. Щоб виділити для системи hex2 масив рядків і стовпців, перш за все, треба виділити вектор для зберігання зображення: pixel image[nr*nc].

Потім виділити вектор вказівників на пікселі та провести адресу на початку кожного рядка: pixptr fistpix[nr].

Простий цикл ініціалізує кожен вхід fistpix, що містить адресу першого з пікселів на кожному рядку. Потім, щоб отримати доступ до будь-якого з пікселів, припустимо, на r-рядку і c-стовпці, треба зробити вказівник і додати: ptr = fistpix[r] + c, а доступ зображення за допомогою цього покажчика отримати так: value = *ptr. Це більш ефективно, ніж value = image[r][c], тому що останнє включає в себе множину інструкцій, відсутніх в попередній схемі.

Перетворення зображень із прямокутної до гексагональної решітки

В галузі гексагональної обробки зображень, перетворення, тобто передискретизація, з регулярного квадратного зображення у гексагональне зображення, як правило, є необхідним. Основні методи для моделювання гексагональної сітки на основі оригінальної квадратної сітки, вказані у [6-10]. В свою чергу, Вютрих запропонував метод створення псевдогексагональних пікселів, відомий як гіперпел (hyperpel), з кластеру квадратних пікселів [11]. Хоча це не відтворює ідеальне уявлення шестикутника, згідно методу створюється шестикутникоподібна форма, що відповідає основним гексагональним властивостям. По відношенню к гексагональному зображенню можливе виконання необхідних перетворень перед його відображенням на прямокутній сітці.

В роботі запропоновано передискретизацію, яку основано на методі Вютріха. Метод модифіковано представленням кожного пікселя *mxn* піксельним блоком з метою створення псевдопіксельного ефекту, який дозволяє псевдопіксельну кластеризацію, яка показана на рисунку 3. Це обмежує втрату роздільної здатності зображення. Кожен піксель представлений блоком 13x16 рівної інтенсивності в новому зображенні. Кожен піксель оригінального зображення представлений блоком 16x16 пікселів. Це створює збільшене

зображення тієї ж роздільної здатності в якості початкового зображення з можливістю відображення кожного пікселя в якості групи *mxn* псевдопікселів.

Рисунок 3 – Регулярне прямокутне зображення

При розробці прямокутного псевдопікселя враховується параметр scale, який передається у функцію image_expand, що безпосередньо виконує формування вихідної матриці. Параметри sourceRowsNum, sourceColumnsNum визначають розмір вхідної матриці, а параметри destinationRowsNum та destinationColumnsNum - розмір вихідної матриці. Останні два параметри визначаються множенням перших двох на параметр factor, який у даній функції є аналогом scale. Заповнення кінцевої матриці прямокутних псевдопікселів відбувається за допомогою двох циклів, перший з яких діє по стовпцях, а другий по строках матриці. Кожному пікселю вхідної матриці ставиться у співвідношення псевдопіксель вихідної псевдопіксельної матриці, який має такі ж самі характеристики, з єдиною відмінністю в тому, що він у factor*factor разів більший. Приклад регулярного прямокутного зображення, побудованого за допомогою псевдопікселів наведений на рисунку 3. Функцію, яка формує матрицю псвдопікселів, наведено на рисунку 4.

```
function [ destination ] = image_expand(source,  factor)
    [sourceRowsNum, sourceColumnsNum] = size (source);
    destinationRowsNum = sourceRowsNum * factor;
    destinationColumnsNum = sourceColumnsNum * factor;
    destination = zeros (destinationRowsNum, destinationColumnsNum);
    desinationRow = 1;
    destinationColumn = 1;
    for row = 1 : sourceRowsNum
        for column = 1 : sourceColumnsNum
            destination (desinationRow : desinationRow + factor - 1,
destinationColumn : destinationColumn + factor - 1) = source (row, column);
            destinationColumn = column * factor + 1;
        end

        destinationColumn = 1;
        desinationRow = row * factor + 1;
    end
```

Рисунок 4 – Функція, яка формує матрицю прямокутних псевдопікселів

На рисунку 5 наведений приклад матриці прямокутних псевдопікселів який отримано за допомогою MatLab.

При розробці гексагонального псевдопікселя було враховано розмір прямокутного псевдопікселя і, виходячи з цього, розмір був обраний таким, що дорівнює 13x16 пікселів. Для формування псевдопікселя розроблена функція `hex_pixel_spread`, а для формування нового зображення використовується функція `hex_image_expand`. огляд яких наведено далі. Для формування пікселя, яке відбувається по стовпцях, виконується пять послідовних кроків, на кожному з яких відбувається залиття поточного стовпця визначеним кольором, причому формування верхнього та нижнього кута відбувається одночасно, тому що вони мають однакову (але горизонтально симетричну) форму.

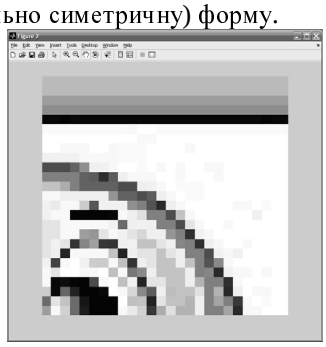


Рисунок 5 – Приклад прямокутних псевдопікселів

Приклад регулярного гексагонального зображення, побудованого за допомогою псевдопікселів, наведений на рисунку 6. Функцію, яка формує матрицю псевдопікселів, наведено на рисунку 7. На рисунку 8 наведений збільшений приклад матриці гексагональних псевдопікселів, який отримано за допомогою MatLab.

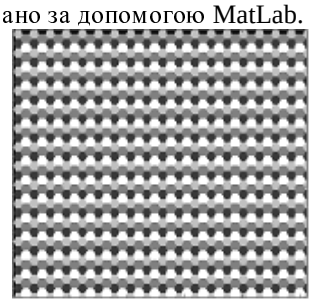


Рисунок 6 – Регулярне гексагональне зображення

```
function hex_pixel_spread (x, y, pixel)
        templateWidth = 16;
        templateHeight = 13;
        columnShift = 0;
        out (x + columnShift, y + 6 : y + 7) = pixel;
        out (x + templateWidth - columnShift -1, y + 6 : y + 7) = pixel;
columnShift = columnShift + 1;
        out (x + columnShift, y + 4 : y + 9) = pixel;
        out (x + templateWidth - columnShift -1, y + 4 : y + 9) = pixel;
columnShift = columnShift + 1;
        out (x + columnShift, y + 3 : y + 10) = pixel;
        out (x + templateWidth - columnShift -1, y + 3 : y + 10) = pixel;
columnShift = columnShift + 1;
        out (x + columnShift, y + 1 : y + 12) = pixel;
        out (x + templateWidth - columnShift -1, y + 1 : y + 12) = pixel;
columnShift = columnShift + 1;
        out (x + columnShift : x + templateWidth - columnShift -1, y : y +
12) = pixel;
    end
```

Рисунок 7 – Функція, яка формує матрицю гексагональних псевдопікселів

Для формування вихідного гексагонального зображення в першу чергу враховується параметр `scale`, який показує у скільки разів повинна бути збільшена вихідна матриця пікселів у порівнянні зі вхідною. У `main` функції відбувається завантаження вхідного зображення, а також вручну користувачем задається параметр `scale`. Ці два параметри і передаються у функцію обробки `hex_image_expand`, яка виконує перетворення у гексагональні псевдопікселі.

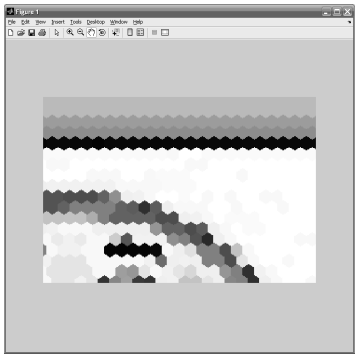


Рисунок 8 – Приклад гексагональних псевдопікселів

На початку роботи функції визначається розмір вхідної матриці і формується нова (вихідна) матриця пікселів, розмір якої по строках та стовпцях збільшується у `scale` разів. Нова матриця створюється за допомогою функції `Zeros`. Обробка здійснюється по стовпцях, причому спочатку формуються всі непарні стовпці, а потім, зсунувши початкову точку на півпікселя, обробляються всі парні стовпці. У функцію `hex_pixel_spread`, яка формує кожен новий псевдопіксель окрім поточних координат передається параметр `pixel`, який формується у функції `get_pixel (x, y)`. У змінну `pixel = in (x, y)` записується значення пікселя на поточних координатах. Програмний код функції реалізації та пересемплування на віртуальній решітці показаний на рисунку 9.

```
function [ out ] = hex_image_expand (in, scale)
    [inWidth, inHeight] = size (in);
    out = zeros (inWidth * scale, inHeight * scale);
    out (:, :) = 255;

    function pixel = get_pixel (x, y)
        pixel = in (x, y);
    end

function hex_pixel_spread (x, y, pixel)
      .
      .
      .
end
    i = 1;
    for x = 1 : 2 : inHeight
        j = 1;
        for y = 1 : 1 : inWidth
            pixel = get_pixel (x, y);
            hex_pixel_spread (i, j, pixel);
            j = j + 13;
        end
        i = i + 24;
    end
     i = 13;
    for x = 2 : 2 : inHeight
        j = 8;
        for y = 1 : 1 : inWidth
            pixel = get_pixel (x, y);
            hex_pixel_spread (i, j, pixel);
            j = j + 13;
        end
        i = i + 24;
    end
end
```

Рисунок 9 – Програмний код функції реалізації та пересемплування на віртуальній решітці

Далі наведено приклад для візуального порівняння оригінального зображення та двох зображень, перетворених у прямокутні та гексагональні псевдопікселі відповідно. Отримані зображення наведено на рисунках 10,11,12 (оригінал, прямокутні псевдопікселі та гексагональні псевдопікселі відповідно).

Для оцінки продуктивності та дослідження властивостей перетвореного зображення в рамках розробленої програми, було проведено випробування виявлення краю. Зображення, розміром 256 на 256 пікселів (рисунок 13), було обрано, оскільки воно містить суміш кривих і прямих ліній, а також зміну контрастності. Були розглянуті два різних детектори визначення краю. Перший оператор – детектор визначення краю Прюітта, а другий - лапласіан гауссовий детектор (Laplacian of Gaussian).

Рисунок 10 – Оригінальне зображення

Рисунок 11 – Зображення, перетворене у прямокутні псевдопікселі

Рисунок 12 – Зображення, перетворене у гексагональні псевдопікселі

Виявлення краю складається з двох кроків: розрахунок величини країв і порогова классифікація. Для порівняння прямокутного зображення із гексагональним зображенням, порогова класифікація була спрямована на досягнення найкращих результатів у якості, а потім було обчислено співвідношення крайових пікселів до розміру зображення.

Фільтр визначення краю Прюітта заснований на використанні горизонтального і вертикального фільтрів послідовно. Обидва

фільтри застосовуються до зображення і підводять до форми кінцевого результату.

Лапласіан Гауссівський фільтр спочатку застосовує Gaussian Blur, потім - фільтр Лапласа і, нарешті, перевіряється перетин нуля (тобто коли в результаті значення зростає з негативного на позитивний або навпаки). Кінцевим результатом цього фільтра є визначення краю.

Результати на детекторі краю Прюітта показано на рисунку 14. Маска розміром 9 пікселів була використана для квадратного зображення і розміром 7 пікселів - для гексагонального зображення. Огляд двох вихідних зображень показав, що:

- форма монети виглядає більш круговою в гексагональному зображенні;
- ківі, папороть, а також число «20» являються більш чіткими в гексагональному зображенні.

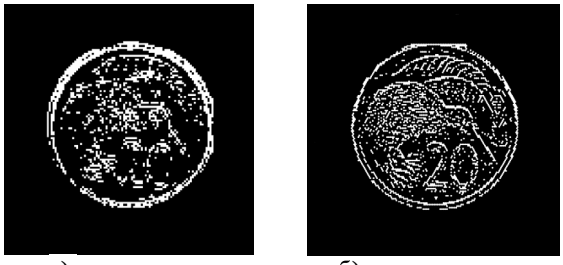
Рисунок 13 – Тестове зображення

а) прямокутне зображення б) гексагональне зображення

Рисунок 14 – Результати використання детектору краю Прюітта

В цілому, гексагональне зображення є менш зашумленим. Це пов'язано з тим, що виявлені краї у прямокутному зображенні є товщими. Відношення краю пікселів до розміру зображення у двох випадках є 12,6% для гексагонального зображення і 15,9% - для прямокутного зображення.

Результати для лапласіан гаусівського визначення краю, використовуючи той самий образ випробувань, наведено на рисунку 15. Обидві маски орієнтовані на лапласіан гауссові функції і складаються з 49 точок. Для гексагонального зображення, криві показані з хорошою чіткістю. Тут можна також розгледіти листя папороті. Низька продуктивність виявлення краю в прямокутному зображенні пов'язана з тим, що маска є не дуже придатною для лапсасіан-гауссівської функції. Щоб підвищити ефективність квадратного зображення потребується більша маска, яка також призведе до збільшення обчислювальних витрат. Відношення краю пікселів до розміру зображення для гексагонального зображення є 16,3% і 17,9% - для прямокутного зображення.



а) прямокутне зображення б) гексагональне зображення

Рисунок 15 – Результати використання лапласіан гаусівського визначення краю

Список літератури

1. Carstens B. Hexagonal domain transform for shape analysis / B. Carstens, M. Quinn // Intelligent Robots and Computer Vision X: Algorithms and Techniques, SPIE. – 1991. – V.1607. – P. 197-205.

2. Watson A.B. A hexagonal orthogonal-oriented pyramid as a model of image representation in visual cortex / A.B. Watson, A.J. Ahumada, Jr. // IEEE Biomedical Engineering. – 1989. – V.36. – No.1. – P. 97-105.

3. Jan W. Conversion of Cartesian coordinates from and to generalized balanced ternary addresses / Jan W. // Photogrammetric Engineering and Remote Sensing, 54. – no. 11. – 1988. – Pp. 1565-1570.

4. Luczak E. Distance on a hexagonal grid / E. Luczak, A. Rosenfeld // IEEE Transactions on Computers.–V.C-25. – No. 5. – 1976. – P. 532-533.

5. Bell S. A digital geometry for hexagonal pixels / S. Bell, H. Fred, D. Mason // Image and Vision Computing. – V. 7. – No. 3. – 1989. – P. 194-204.

6. Allen J.D. Filter Banks for Images on Hexagonal Grid / J.D.Allen //Signal Solutions. – 2003.

7. Mollon, J.D. The Spatial Arrangement of Cones in the Primate Fovea / J.D. Mollon, J.K. Bowmaker // Nature. – 1992. – V. 360. – P. 677-679.

8. Mylopoulos J.P. On the topological properties of quantized spaces, I. the notion of dimension / J.P. Mylopoulos, T. Pavlidis // Journal of the ACM (JACM). – 1971. – V.18. – No.2. – P. 239-246.

9. Anterrieu Eric Apodization Functions for 2-D Hexagonally Sampled Synthetic Aperture Imaging Radiometers / Eric Anterrieu, Philippe Waldteufel, Andre Lannes // IEEE Transactions On Geoscience And Remote Sensing. – 2002 – V. 40. – No. 12. – P. 2531-2542.

10. Middleton L. Edge Detection in a Hexagonal-Image Processing Framework / L. Middleton, J. Sivaswamy // Image and Vision Computing 19. – 2001. – P. 1071-1081.

11. Wuthrich C.A. An Algorithmic Comparison Between Square-and Hexagonal-based Grid / C.A Wuthrich, P. Stucki // CVGIP: Graphical Models and Image Processing. – 1999. – Vol. 53 – P. 324-339.

Надійшла до редколегії 01.03.2011

А.В. САМОЩЕНКО, С.В. ХОДУС
Донецкий национальный технический университет

Передискретизация изображений в гексагональной системе координат hex2

Рассматривается преобразование изображений, присутствующих в гексагональной системе координат, в псевдогексагональном виде, что позволяет представить результаты визуально без использования гексагональной решетки или аппаратных гексагонально-ориентованных устройств. Исследованы свойства изображений, которые показывают преимущества гексагональной структуры по сравнению со стандартной прямоугольной структурой пикселей.

Гексагонал, координатная система, решетка пикселей, псевдопиксель, фильтр определения края

A.V. SAMOSHCHENKO, S.V. HODUS
Donetsk National Technical University

Image discretization in Hexagonal Coordinate System Hex2

The paper considers transformation of images in the hexagonal coordinate system and their representation in the pseudohexagonal image, which also allows presenting the results visually without using a hexagonal lattice or hardware (hexagonal-oriented output devices). The properties of images were investigated and verified, they show the superiority of hexagonal structure in comparison with the standard rectangular structure points.

Hexagonal, coordinate system, pseudopixel, determined filters of lines