УДК 004.3

Г.Г. Киричек, канд. техн. наук, доц.,
В.І. Курай, магістр
Запорізький національний технічний університет, м. Запоріжжя, Україна
kirichek@zntu.edu.ua

Визначення об'єктів з використанням дерева квадрантів

В роботі приводиться аналіз методів сегментації та порівняння зображень і пропонується використання методу дерева квадрантів для рекурсивної розбивки і порівняння зображень, як можливе рішення, коли компактне представлення зображення та високоефективний доступ до елементів є ключовими вимогами до системи.

Ключові слова: сегментація, структури даних, зображення, квадро-дерево.

DOI: 10.31474/1996-1588-2018-1-26-19-24

Вступ

Область обробки цифрових зображень розвивається за двома напрямками: вдосконалення механізмів обробки і маніпуляції над графічною інформацією, з метою її подальшої інтерпретації людиною та аналіз даних для подальшого сприйняття їх машиною, наприклад при обробці відбитків пальців або інших біометричних даних [1]. Невід'ємною частиною обробки зображень є ефективний спосіб представлення зображень, що зберігаються, у вигляді структури даних. Однією з найбільш вивчених і добре зарекомендувавших себе структур даних, які використовуються для представлення двовимірних зображень, є квадро-дерево (quadtree) [2].

Існує багато інструментів, які дозволяють вести дослідження у цьому напрямку. Найбільш поширеними є: перцептивні хеш-алгоритми [3] та штучна нейронна мережа [4]. За допомогою процесу навчання такі мережі можна навчати роботі із зображеннями. Найвідомішою бібліотекою для роботи з зображеннями наразі є OpenCV, яка надає засоби для обробки і аналізу вмісту зображень, у тому числі розпізнавання об'єктів на фотографіях [5]. Перевагами бібліотек типу OpenCV є наявність документації та прикладів їх використання. Недоліки - відсутність реалізації для конкретної мови програмування та необхідність підключення цілої бібліотеки для використання конкретних функцій, що не завжди є виправданим.

Постановка задачі

Хоча наведені вище рішення є добре відточеними алгоритмами і бібліотеками для роботи з зображеннями, все ж таки інколи розгортання подібної системи є невиправданим з точки зору швидкодії, задіяння ресурсів обладнання та складності роботи з ними. Тому в даній роботі планується дослідити та запропонувати свій підхід до вирішення цієї проблеми, шляхом впровадження системи визначення об'єктів з використанням методу дерева квадрантів, для порівняння зображень, як можливе рішення, коли компактне представлення зображення та високоефективний доступ до елементів, являються ключовими вимогами до системи.

Мета роботи – проведення досліджень та розробка методів, моделей і засобів обробки цифрових зображень для визначення об'єктів, з реалізацією методу дерева квадрантів для рекурсивного розбиття і порівняння зображень.

Об'єкт дослідження – система визначення об'єктів.

Предмет дослідження – моделі, методи та інструментальні засоби обробки цифрових зображень.

При розробці програмного забезпечення, необхідно вирішити наступні задачі: дослідити метод дерева квадрантів для рекурсивного розбиття двомірного простору; провести порівняння результатів, отриманих при розбитті зображень, на наявність відмінностей; на підставі отриманих відмінностей створити третє - результуюче зображення, що зберігає різницю перших двох вхідних. Програмне забезпечення розробляється як консольна утиліта, тому параметри для його роботи, такі як вхідні зображення та шлях до міста збереження результуючого зображення, потрібно вказувати як аргументи командного рядка.

Технології проектування

Розглянемо популярні мови програмування, що підходять для реалізації даної задачі.

C # - об'єктно-орієнтована мова програмування з безпечною системою типізації для платформи .NET [6]. Java - об'єктно-орієнтована мова програмування, має менше низькорівневих можливостей для роботи з апаратним забезпеченням, що в порівнянні, наприклад, з C++ зменшує швидкість роботи програм [7]. Python - інтерпретована об'єктно-орієнтована мова програмування високого рівня з строгою динамічною типізацією. Вона підтримує декілька парадигм програмування, зокрема: об'єктно-орієнтовану, процедурну, функціональну та аспектно-орієнтовану [8]. Основні архітектурні риси Python: динамічна типізація; авто-

матичне керування пам'яттю; повна інтроспекція; механізм обробки виключень; підтримка багатопоточних обчислень та зручні високорівневі структури даних. На вибір Python, у якості мови програмування для системи, яка розробляється, вплинули: підтримка Python об'єктно-орієнтованого підходу, простота синтаксису, а також наявність великої кількості вбудованих функцій і структур даних, які прискорили швидкість розробки програмного забезпечення.

Сегментація - процес розділення цифрового зображення на декілька сегментів (множина пікселів). Мета сегментації полягає у спрощенні і/або зміні представлення зображення для полегшення його аналізу [9]. До популярних методів сегментації зображення можна віднести: метод заснований на кластеризації - ітеративний метод, що використовують для розділити зображення на K кластерів [9]; метод виділення країв - добре вивчена область обробки зображень [9]; методи з використанням гістограми - підхід використовує сегментацію, засновану на рухомих об'єктах і нерухомому оточенні, що дає інший вид сегментації, що є корисним у відео трекінгу [9] та метод дерева квадрантів (Q-дерево) використовується для рекурсивного розбиття двомірного простору по 4 квадранта (області), що є прямокутниками або мають довільну форму [2].

Переваги такої структури в тому, що регулярний поділ забезпечує просте і ефективне накопичення, відновлення і обробку даних. Простота виникає з геометричною регулярністю розбиття, а ефективність - за рахунок зберігання лише вузлів з даними, які представляють інтерес [9]. Основна ідея квадро-дерева - комбінування однакових або схожих елементів даних і кодування великих однорідних сукупностей даних малою кількістю бітів. Ділянки одного кольору заповнюються великими "вузлами" дерева, натомість у місцях "стику" кольорів можна побачити скупчення маленьких квадратів. Тому дерево квадрантів для зображень це дерево зі збільшенням глибини якого збільшується кількість елементів дерева. Цей алгоритм має наступну залежність часу виконання O (u + p + q). Де u – кількість полігонів, p – периметр полігонів, роздільна здатність зображення [10]. Причини вибору цього методу розбиття зображень наступні: компактне представлення зображення та високоефективний доступ до елементів, який досягається за рахунок деревовидної структури даних, якою і є дерево-квадрантів.

Реалізація системи

Виходячи з основних завдань роботи наведемо метод реалізації дерева квадрантів. Наступний код демонструє опис дерева квадрантів на мові програмування Python (лістинг 1).

Лістинг 1 – Опис дерева квадрантів на мові програмування Python.

```
class QuadTreeRect(Rect):
def __init__(self, pt1, pt2):
    super().__init__(pt1, pt2)
@property
  def color_distance(self):
    return self._color_distance
@color_distance.setter
  def color_distance(self, color_distance):
    self._color_distance = color_distance
@property
  def average_color(self):
    return self._average_color
@average_color.setter
def average_color(self, average_color):
self._average_color = average_color
def __eq__(self, other):
return self.average_color == other.average_color and (self.top_left() == other.top_left() and self.bottom_right() == other.bottom_right())
```

Клас унаслідується від класу Rect, який є реалізацією прямокутника та містить у собі метод для розбиття прямокутника на 4 рівні частини, що необхідно для створення дочірніх вузлів дерева квадрантів [11].

Щодо класу QuadTreeRect, то він містить гетери та сетери для параметрів average_color (середнє значення кольору), яке міститься у відповідному прямокутнику дерева (вузлі), а також color_distance. Color_distance - значення отримане в результаті обчислення різниці між кольором кожного пікселя в середині прямокутника та середнім значенням кольору цього прямокутника.

Початок розбиття зображення - отримання зображення у вигляді двомірного масиву пікселів, шляхом розробки методу get_pixels_from_image() (лістинг 2).

Лістинг 2 – Отримання пікселів з зображення у вигляди двомірного масиву.

```
def get_pixels_from_image(self):
im = Image.open(self.image_name)
colors = list(im.getdata())
width, height = im.size
return [colors[i * width:(i + 1) * width] for i in range(height)]
```

Метод повертає двомірний масив довжиною 3 (трьома елементами кодується колір пікселя). Використовуючи доступ до елементів через індекс, отримали значення кольору, у відповідній точці зображення. Змінюючи мінімальний розмір вузла дерева, можна регулювати якість вихідного зображення за рахунок обмеження подальшого створення нащадків. Так як, великі області одного

кольору представляться у вигляді одного великого прямокутника, то на їх подальше розбиття зміна мінімального розміру вузла не впливає. Для програмної реалізації визначення середнього значення кольору в заданому вузлі дерева створено метод, що приймає на вхід вузол дерева (лістинг 3).

Лістинг 3 – Середнє значення кольору

```
def calc_average_color(self, rect):
    red_sum = 0
    green_sum = 0
    blue_sum = 0
    for row_i in range(rect.top, rect.bottom):
        for column_j in range(rect.left, rect.right):
red_sum += self.pixels[row_i][column_j][0]
green_sum += self.pixels[row_i][column_j][1]
blue_sum += self.pixels[row_i][column_j][2]
rect_area = rect.get_rect_area()
return int(red_sum / rect_area), int(green_sum / rect_area), int(blue_sum / rect_area)
```

Далі у вкладеному циклі, через індекси проводиться доступ до пікселів зображення, що знаходяться в області вузла дерева. Проводиться підрахунок трьох складових кольору та кожна складова кольору ділиться на кількість пікселів у заданому вузлі, для знаходження середнього значення. Так метод повертає три середніх значення: червоної, зеленої та синьої складових кольору. Для отримання значення різниці кольорів, знайдемо середнє значення кольору в цьому прямокутнику.

Наступним кроком є знаходження різниці за модулем між середнім значенням кольору в квадранті та значенням кольору пікселя. Для цього у вкладеному циклі знайдемо різницю між складовою кольору кожного пікселя та середнім значенням кольору. Розрахунок проводиться три рази, для кожної складової кольору. Результати розрахунків підсумовуються.

Після завершення підрахунків метод повертає значення різниці кольору поділене на площу прямокутника та помножену на три. Множення площі прямокутника на три, відбувається через те, що для тієї ж самої площі розрахунок проводився три рази, для кожної складової кольору. Представимо етапи роботи програми та її окремих функцій у вигляді блок схеми, яка відображає процес розбиття зображення на квадранти (рис. 1).

Результатом розбиття зображення виступає масив квадрантів вузлів дерева. Кожний такий вузол містить в собі об'єкт типу прямокутника та значення середнього кольору і відстані кольорів. Маючи такий набір характеристик для кожного вузла дерева, можна провести порівняння двох зображень через порівняння їх дерев квадрантів [11].

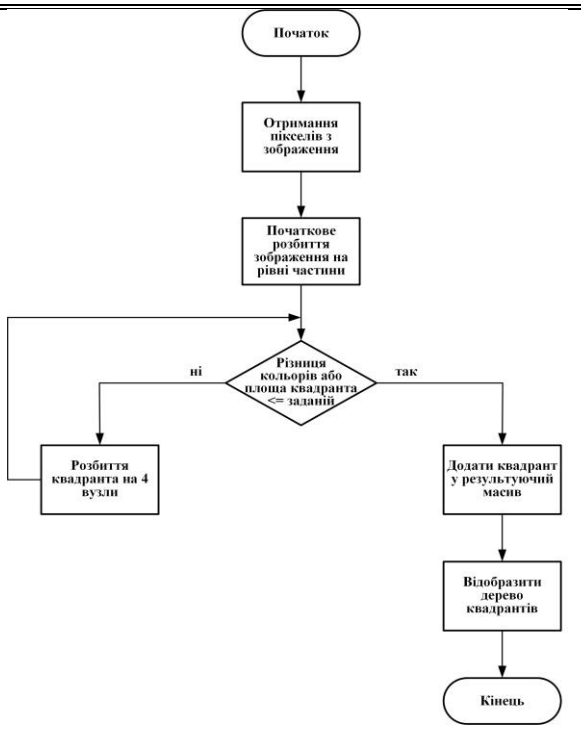

Рисунок 1 – Блок схема розбиття зображення

Для коректного порівняння елементів дерева квадрантів треба для класу QuadTreeRect змінити метод, що відповідає за порівняння об'єктів (лістинг 4).

Лістинг 4 – Перевизначення __eq__().

```
def __eq__(self, other):
return self.average_color == other.average_color and (self.top_left() == other.top_left() and self.bottom_right() == other.bottom_right())
```

Відповідно до нової реалізації метод __eq__() поверне значення True, у тому разі якщо співпадає середній колір квадрантів, а також координати їхньої верхньої лівої та нижньої правої точок. Наступним кроком є знаходження елементів дерева квадрантів зображення 1, яких не містить зображення 2. Тобто необхідно знайти різницю множин дерев квадрантів. Для цього розроблено метод compare_images_rects(self, list1, list2) (лістинг 5).

Лістинг 5 – Знаходження різниці множин.

```
def compare_images_rects(self, list1, list2):
    outputlist = []
    list3 = list1 + list2
    for item in list3:
        if item not in list2:
            outputlist.append(item)
    return outputlist
```

Алгоритм порівняння зображень такий:

– провести початкове розбиття двох зображень, задавши значення мінімального розміру вузла;

– зробити порівняння отриманих вузлів дерева першого і другого зображень;

– для знайдених квадрантів, повторювати порівняння, поки досягнемо заданого значення мінімального розміру квадранта;

– після отримання результатів, створити новий графічний файл, та на основі отриманого дерева квадрантів отримати зображення.

Приклад роботи системи

В результаті проведення досліджень та визначення основних етапів роботи системи для реалізації порівняння зображень розроблено консольну утиліту, яка використовує для передачі аргументів з консолі клас ArgumentParser.

Користувач (або система), при цьому, має можливість передати в програму наступні аргументи: шлях до зображень, які порівнюються; мінімальний розмір вузла дерева; мінімальну різницю кольорів та шлях до міста збереження результуючого зображення.

У лістингу 6 наведено код, який відповідає за передачу аргументів з консолі та запуск роботи програми.

Лістинг 6 – Передача аргументів з консолі та запуск роботи програми.

```
def get_args():
parser = ArgumentParser()
parser.add_argument('--
image1',required=True,type=str)
parser.add_argument('--
image2',required=True, type=str)
parser.add_argument('--
save_to',required=True, type=str)
parser.add_argument('--min_color_diff',
required=False,type=int, default=5)
parser.add_argument('--
min_quad_size',required=False,type=int, default=4)
args = parser.parse_args()
return args
if __name__ == '__main__':
args = get_args()
image_comparator = ImageComparator(
  args.image1,
  args.image2,
  args.save_to,
  args.min_color_diff,
  args.min_quad_size)
image_comparator.compare_images()
```

Розглянемо приклад роботи програми при порівняні зображень супутникових знімків аеродрому. Вхідні зображення наведено на рисунку 2.

Рисунок 2 –Знімки аеродрому

На зображені 2 бачимо супутникові знімки аеродрому із різною кількістю літаків. Проведемо порівняння цих зображень за допомогою розробленого програмного забезпечення.

На рисунку 3 зображена послідовність кроків розбивки зображень та порівнянь їх дерев квадрантів.

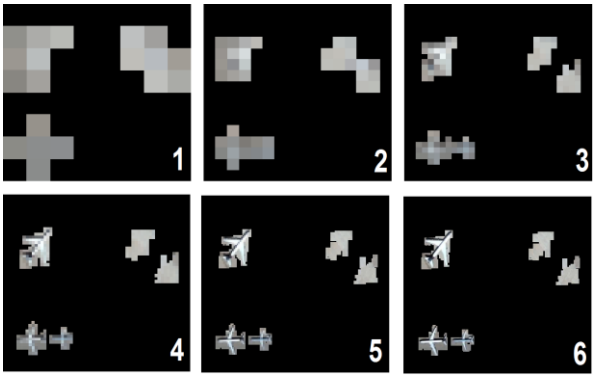

Рисунок 3 – Розбивка зображень та порівняння їх дерев квадрантів

Як видно на рисунку 3, вже на першій ітерації можна побачити області зображення, які відрізняється.

На подальших кроках, відбувається розбивка дерева квадрантів, отриманого на попередньому кроці. І так до тих пір поки не буде досягнута задана точність. Для прикладу, в таблиці 1 наведена залежність часу виконання роботи програми, в залежності від площі мінімального вузла.

Таблиця 1 – Залежність часу роботи програми від площі вузла

Мінімальна площа вузла (пікселі)	Час виконання програми (секунди)
2500	2,81
625	2,95
144	3,29
36	5,7
9	33,6

На основі таблиці 1 побудуємо графік залежності часу роботи програми від площі вузла (рис. 4).

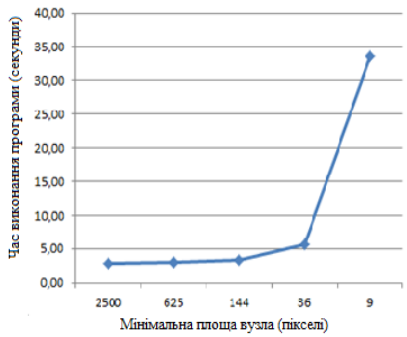

Рисунок 4 – Залежність часу роботи програми від площі вузла

Як видно на рисунку 4 із зменшенням мінімальної площі вузла дерева, збільшується час виконання програми, але разом з цим збільшується і якість вихідного зображення. Збільшення часу виконання програми пов'язано із зростанням числа елементів дерева квадрантів, а також їх подальшим порівнянням один з одним.

Даний метод дозволяє порівнюючи карти місцевості із зображеннями, отриманими в результаті періодичної зйомки, виявляти об'єкти які з'являються або зникають у місцях проведення контролю місцевості, без участі людини.

Висновки

В роботі проведено дослідження, отримано та наведено варіант програмної реалізації метода дерева квадрантів для порівняння двомірних зображень. Встановлена залежність між якістю порівняння зображень та такими характеристиками дерева квадрантів як мінімальний розмір вузла і відстань кольорів в середині вузла.

Автори виконали програмну реалізацію системи визначення об'єктів, з реалізацією методу дерева квадрантів для рекурсивного розбиття і порівняння зображень, за допомогою якої порівняно супутникові знімки. В подальшому планується дослідити роботу даного методу при порівнянні зображень, з урахуванням нічної зйомки та зйомки в умовах різних перешкод.

Список літератури

1. Потапов А.А. Новейшие методы обработки изображений // А.А. Потапов, Ю.В. Гуляев, С.А.Никитов, А.А. Пахомов, В.А. Герман. – М.: ФИЗМАТЛИТ, 2008. – 496 с.
2. Spann M. A quad-tree approach to image segmentation which combines statistical and spatial information // M. Spann, R.Wilson / Pattern Recognition, 1985. – vol. 18 (issue 3-4) . – pp. 257-269.
3. «Выглядит похоже». Как работает перцептивный хеш. – [Електронний ресурс]. – Режим доступу: https://habrahabr.ru/post/120562/.
4. Лукін В.Є. Аналіз використання технології штучних нейронних мереж в якості нового підходу до обробки сигналів // В.С. Лукін / Телекомунікаційні та інформаційні технології. – №3. – Київ, 2014. – С. 81-88.
5. Петин В. Микрокомпьютеры Raspberry Pi: Практическое руководство // В. Петин. – СПб: Питер, 2015. – 240 с.
6. Виссер Дж. Разработка обслуживаемых программ на языке C# // Дж.Виссер. – СПб: ДМК Пресс, 2017. – 192 с.
7. Лафоре Р. Структуры данных и алгоритмы Java // Р. Лафоре. – СПб: Питер, 2013. – 704 с.
8. Dierbach Ch. Python as a first programming language // Ch. Dierbach / Journal of Computing Sciences in Colleges, 2014. – vol. 29 (issue 6). – pp. 153-154.
9. Zaitoun N.M. Survey on Image Segmentation Techniques // N.M. Zaitoun, M.J. Aqel / Procedia Computer Science, 2015. – vol. 65. – pp. 797-806.
10. Hunter G.M. Operations on Images Using Quad Trees // G.M. Hunter, K. Steiglitz / IEEE Transactions on Pattern Analysis and Machine Intelligence, 1979. – vol. PAMI-1 (issue 2). – pp. 145 – 153.
11. Kirichek G. Implementation quadtree method for comparison of images // G. Kirichek, V. Kurai / Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, February 20 – 24, 2018. – p. 49.

References

1. Potapov A. (2008), The newest methods of image processing, Moscow: FIZMATLIT, 496 p.
2. Spann M., Wilson R. (1985), "A quad-tree approach to image segmentation which combines statistical and spatial information," Pattern Recognition, vol. 18 (issue 3-4), pp. 257-269.
3. "It looks like." How the perceptive hash works, available at: https://habrahabr.ru/post/120562/.
4. Lukin V. (2014), "Analis of the technology of technology of the neural neural units in the anchor of a new input to the signal box," Telecomunication and Informatics Technologies, vol. 3, Kiev, pp.81-88.
5. Petin V. (2015), Microcomputers Raspberry Pi: Practical guidance, Peter, 240 p.
6. Visser J. (2017), Development of Serviced Programs in C#, St. Petersburg: Press, 192 p.
7. Lafore R. (2013), Data Structures and Java Algorithms, St. Petersburg: Peter, 704 p.

8. Dierbach Ch. (2014), "Python as a first programming language," Journal of Computing Sciences in Colleges, vol. 29 (issue 6), pp. 153-154.

9. Zaitoun NM., Aqel MJ. (2015), "Survey on Image Segmentation Techniques," Procedia Computer Science, vol. 65, pp. 797-806.

10. Hunter GM., Steiglitz K. (1979), "Operations on Images Using Quad Trees," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. PAMI-1 (issue 2), pp. 145 - 153.

11. Kirichek G. (2018), "Implementation quadtree method for comparison of images", Proceedings of 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET), Lviv-Slavske, Ukraine, p. 49.

Надійшла до редакції 26.03.2018

Г.Г. КИРИЧЕК, В.И. КУРАЙ
Запорожский национальный технический университет (Украина)

ОПРЕДЕЛЕНИЕ ОБЪЕКТОВ С ИСПОЛЬЗОВАНИЕМ ДЕРЕВА КВАДРАНТОВ

В работе приводится анализ методов сегментации и сравнения изображений и предлагается, использование метода дерева квадрантов, для рекурсивной разбивки и сравнения изображений, как возможное решение, когда компактное представление изображения и высокоэффективный доступ к элементам, являются ключевыми требованиями к системе.

Ключевые слова: сегментация, структуры данных, изображения, квадро-дерево.

G.G. KIRICHEK, V.I. KURAI
Zaporizhzhia National Technical University (Ukraine)

DEFINITION OF OBJECTS USING A QUADRANT TREE

The article deals with identifying of different image areas using the quad tree method for recursive image decomposition and comparison. It may be apply when key system requirements include compact image representation and high-efficiency access to elements.

The objectives of the work are the analysis of image processing methods and the validation of recursive image partitionig and representation in form of the quadtree. Also two images partitioning results are compared in the article. Also the article is devoted to the developing of digital images processing methods, models and means to determine the objects, with quadtree method implementation for recursive partitioning and images comparison.

The proposed method for different image areas identifying provides convenient data manipulation, memory cost reducing and quick access to image elements. The method has been improved by representing of image part in the form of quadrants that contain the coordinates and color. The object detection system includes images fixation, images comparing and saving differences.

The authors proposed a graphics information processing mechanism for further human interpretation, for example, in medicine tasks (analysis of X-rays) or for map comparison (detecting of appearing/ disappearing objects). The quadrant tree data structure was investigated.

The dependence between image comparison quality and quadtree characteristics (node minimum size, color distance in the middle of the node) was established. The software implementation of quadtree method for comparison of two-dimensional images had been presented.

Keywords: segmentation, data structures, images, quad-tree.