УДК 683.519

ГАНЖА І. М.,
ШЕМСЕДИНОВ Т. Г.,
АЛЄНІН О. І.,
СТІРЕНКО С. Г.

МОДИФІКАЦІЯ ТА ЗАСТОСУВАННЯ АЛГОРИТМУ МАРКУВАННЯ ЗВ'ЯЗНИХ КОМПОНЕНТ НА БІНАРНОМУ ЗОБРАЖЕННІ

В статті розглянуто двухпрохідний порядковий алгоритм маркування зв’язних компонент, приведено спосіб реалізації алгоритму та запропоновано використання динамічного масиву для збереження класів еквівалентності. Алгоритм маркування зв’язних компонент використовується для виділення зв’язної компоненти, що використовується для подальшого морфологічного аналізу компоненти.

In the article considered the two-pass consistent connected components labeling algorithm, a way to implement the algorithm and the use of a dynamic array to store the equivalence classes. Connected components labeling algorithm used to select a connected component, which used for further morphological component analysis.

Ключові слова: зв'язані компоненти, морфологічний аналіз, цифрова обробка зображень, розпізнавання зображень.

1. Вступ

Двухпрохідний порядковий алгоритм маркування зв'язних компонент або класичний алгоритм маркування зв'язних компонент на бінарному зображенні [2] виявляє зв'язні компоненти (області) переднього плану та надає виявленій компоненті чисельне значення або маркер. В результаті роботи алгоритму, з початкового зображення, в якому всі пікселі компонент мають значення 1, формується маркіроване зображення, в якому пікселі компоненти мають значення маркера компоненти. На рис. 1а наведено початкове бінарне зображення, в якому всі чотири компоненти мають однакові значення пікселів. На рис. 1б зображено результат алгоритму маркування, кожна компонента отримала свій маркер, всі пікселі компоненти мають значення маркера, внаслідок чого компоненти зображені різними значеннями інтенсивності.

Маркування зв'язних компонент необхідне для подальшого морфологічного аналізу виявлених компонент. Морфологічний аналіз формує множину ознак компоненти, дані ознаки використовуються в класифікації або розпізнаванні об’єкта. Також алгоритм може використовуватись для маркування зв’язних компонент фону, маючи марковані компоненти переднього плану та фону можливо скласти граф зв’язності компонент[1].

а. Початкове зображення *б. Маркіроване зображення*

Рис. 1. Приклад роботи алгоритму маркування зв’язних компонент

2. Опис алгоритму

Класичний алгоритм виконується за два проходи. На першому проході формуються класи еквівалентності та надаються тимчасові маркери зв'язним компонентам або їх частинам. На другому проході тимчасові маркери заміняються маркерами відповідно до кореня структури об'єднання-пошуку класу еквівалентності.

Клас еквівалентності являє собою множину тимчасових маркерів окремої зв'язної компоненти. Клас еквівалентності в класичному алгоритмі маркування представлено в структурі об'єднання-пошуку, яка зберігає тимчасові мітки в деревоподібній структурі, з'єднує дерева між собою та дозволяє швидко знаходити корінь дерева [1]. На рис. 2 наведено приклад структури об'єднання-пошуку у вигляді таблиці та графів дерев.

0	1	2	3	4	5	6	7	8
0	2	6	6	8	6	0	8	0

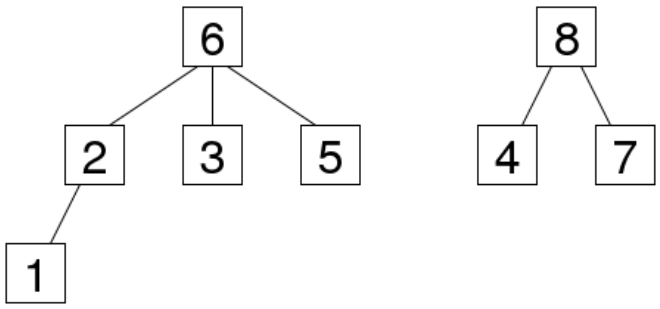

Рис. 2. Структура об'єднання-пошуку у вигляді таблиці та графу

Таблиця на рис. 2 в першому рядку містить значення маркерів від 1 до кількості маркерів, значення 0 не враховується адже відповідає значенню фону зображення. Другий рядок містить значення маркера-батька для маркера з першого рядка, значення 0 означає, що маркер являється коренем дерева. На рис. 2 зображено два графи дерев, що відповідають двом структурам об'єднання-пошуку або класам еквівалентності. Два дерева об'єднують дві множини маркерів – {6, 5, 3, 2, 1} та {8, 7, 4}. Корені дерев – маркери 6 та 8, тому вони мають значення 0 в другому рядку таблиці. Маркер 1 має маркер-батька 2, тому в другому рядку має значення 2, маркери 2, 3, 5 мають маркера-батька 6, тому значення в другому рядку в усіх 6, маркери 4, 7 мають маркера-батька 8, тому в другому рядку міститься 8. Оскільки в першому рядку значення являють нумерацію, то дану структуру можливо представити у вигляді одномірного масиву, в якому індекси – це дані першого рядка, а значення масиву – це дані другого рядка.

При необхідності об'єднати два маркери в структуру, або об'єднати дві структури, якщо їхні корені не співпадають, виконується операція об'єднання. Для виконання операції об'єднання двох дерев досить в таблиці в другий рядок коре-ня дерева, що додається замінити 0 на значення кореня дерева, в яке виконується додавання. Якщо необхідно знайти батька для маркера, щоб визначити до якого класу еквівалентності він належить, виконується операція пошуку. Для пошуку кореня в даній структурі виконується перебирання значень батьків маркерів до того моменту доки не буде знайдено 0.

На першому проході алгоритм надає тимчасовий маркер ще не поміченій області та намагається розповсюдити маркер правим та нижнім сусідам компоненти. Якщо виникає ситуація, що два не однакових маркера намагаються розповсюдитись на один і той же піксель то обирається маркер з меншим значенням. Кожен маркер заноситься в структуру об'єднання-пошуку та при наявності двох різних маркерів в одній компоненті маркери об'єднуються за допомогою описаної вище операції об'єднання. Після першого проходу класи еквівалентності сформовані у вигляді структур об'єднання-пошуку та мають унікальний маркер або ідентифікатор, що відповідає кореню дерева структури. На другому проході тимчасові маркери замінюються ідентифікаторами класів еквівалентності, застосовуючи операцію пошуку кореня дерева, яка описана вище.

На рис. 3 показано результат роботи першого та другого проходів алгоритму та наведено структуру об'єднання-пошуку, яка має класи еквівалентності {2, 1} та {4, 3}.

1	1	0	0	0	0	0	0	0	0
1	1	0	0	2	2	2	2	2	0
1	1	0	0	2	2	2	2	2	0
1	1	1	1	1	1	0	2	2	2
1	1	1	1	1	1	0	2	2	2
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	3	3	0
0	0	0	0	0	0	0	3	3	0
0	4	4	4	4	4	4	3	3	0
0	4	4	4	4	4	4	3	3	0

а. Результат першого проходу

1	1	0	0	0	0	0	0	0	0
1	1	0	0	1	1	1	1	1	0
1	1	0	0	1	1	1	1	1	0
1	1	1	1	1	1	0	1	1	1
1	1	1	1	1	1	0	1	1	1
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	2	2	0
0	0	0	0	0	0	0	2	2	0
0	2	2	2	2	2	2	2	2	0
0	2	2	2	2	2	2	2	2	0

б. Результат другого проходу

0	1	2	3	4
0	2	0	4	0

в. Структура об'єднання-пошуку після першого проходу

Рис. 3. Результати роботи двох проходів та структура об'єднання-пошуку

3. Реалізація алгоритму

Для збереження класів еквівалентності або структур об'єднання-пошуку використано динамічний масив, що дозволяє створювати масив відповідно до кількості тимчасових маркерів.

Перший прохід реалізовано двома послідовними циклами, в яких оброблюються пікселі зі значенням 1 вхідного зображення. Відносно поточного пікселя перевіряться наявність сусідів зі значенням 1 зліва та зверху, з задачею ініціювати маркер MC. Якщо є сусід зліва з маркером ML, то MC := ML, якщо сусід зверху також існує і його маркер MT менше MC, то MC:=MT. Якщо сусідів не виявлено то в MC записується значення наступного маркера по порядку. Маркер MC записується в піксель результуючої матриці.

Якщо маркер MC не дорівнює значенню зліва або зверху то виконується об'єднання маркерів в структурі об'єднання-пошуку.

На другому проході також виконується прохід двома циклами, в яких оброблюються пікселі зі значенням більше 0. Піксель зберігає значення тимчасового маркера, по якому за допомогою операції пошуку знаходиться корінь структури об'єднання-пошуку. Для уникнення збою в порядку нумерації маркерів створюється окремий масив, в якому значенню кореня структури присвоюється порядковий номер. Порядковий номер, що відповідає кореню структури записується в пікселі зв'язної компоненти та є її остаточним маркером.

4. Порівняння класичного та рекурсивного алгоритмів

Порівняння класичного та рекурсивного алгоритмів проводиться на основі сформованого вручну зображення в форматі PGM розміром 800x800 пікселів. Щоб отримати множину зображень різного розміру, початкове зображення розміром 800x800 зменшується на 50 пікселів по ширині та висоті. В результаті маємо 16 однакових по змісту тестових зображень розміром від 50x50 до 800x800 пікселів з кроком 50 пікселів.

Для кожного типу алгоритму проводиться по 100 експериментів та визначається середній час роботи алгоритму. Час роботи алгоритмів показано на рис. 4.

На рис. 4 видно, що рекурсивний алгоритм працює швидше класичного, адже класичний алгоритм робить два проходи з обробко пікселів, рекурсивний алгоритм робить всього один прохід. При розмірі зображення 500x500 пікселів рекурсивний алгоритм видає помилку StackOverflow , що означає що програма вичерпала кількість викликів рекурсивних методів, тому на графіках значення часу 0. Оскільки кількість пікселів в компонентах збільшилась, а на кожен піксель при обробці необхідно викликати рекурсивну функцію, значить збільшується кількість рекурсивних викликів функцій. Налаштування JVM при експерименті стандартне – heap size 700 Мб.

Класичний алгоритм працює при розмірі зображення 500x500 і більше, адже відсутній рекурсивний виклик функцій, маркери назначаються пікселям на основі значень сусідів та структури об’єднання-пошуку. При збільшенні розміру зображення збільшується час роботи алгоритму.

Класичний алгоритм можливо застосовувати для великих зображень на відміну від рекурсивного. Зі збільшенням якості зображень, які формують сучасні фотокамери, потреба в обробці зображень великого розміру стає актуальною.

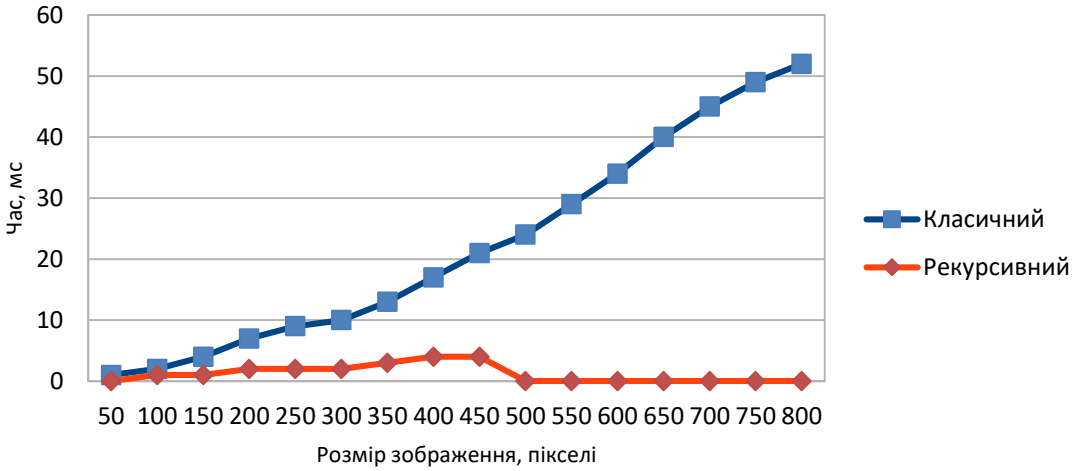

Рис. 4. Час роботи класичного та рекурсивного алгоритмів

5. Застосування алгоритму

Алгоритм маркування зв'язних компонент виконується перед розрахунком характеристик зв'язних компонент. Виділення характеристик компоненти на зображенні називається морфологічним аналізом. Алгоритм маркування зв'язних компонент виділяє компоненту, надає їй унікальний ідентифікатор, що дозволяє аналізувати певну компоненту на зображенні. Модифікація алгоритму нумерує компоненти по порядку, що спрощує реалізацію морфологічного аналізу для випадку коли більше однієї компоненти.

До морфологічних характеристик області зображення відносять:

- площа;
- центр ваги;
- центральний момент другого порядку відносно рядків, стовбців;
- змішаний центральний момент другого порядку;
- момент другого порядку відносно головної та другорядної осі.

Позначимо множину пікселів, що входять до зв'язної компоненти буквою R. Площею A компоненти на бінарному зображенні є кількість пікселів множини R[1]:

$$A = \sum_{(r,c)\in R} 1 \qquad (1.1)$$

Центр ваги є центральною точкою компоненти. Координати центра ваги розраховуються як середні значення координат пікселів області R[1]:

$$\bar{r} = \frac{1}{A} \sum_{(r,c)\in R} r \qquad (1.2)$$

$$\bar{c} = \frac{1}{A} \sum_{(r,c)\in R} c \qquad (1.3)$$

Центральний момент другого порядку показує відхилення координат пікселів множини R від координат центра ваги $\bar{r}$ (1.2), $\bar{c}$ (1.3) по рядкам або стовбцям, нормоване площею A(1.1) компоненти[1]:

$$\mu_{rr} = \frac{1}{A} \sum_{(r,c)\in R} (r - \bar{r})^2 \qquad (1.4)$$

$$\mu_{cc} = \frac{1}{A} \sum_{(r,c)\in R} (c - \bar{c})^2 \qquad (1.5)$$

Змішаний центральний момент показує відхилення по рядкам і стовбцям одночасно:

$$\mu_{rc} = \frac{1}{A} \sum_{(r,c)\in R} (r - \bar{r})(c - \bar{c}) \qquad (1.6)$$

Для компоненти можна виділити головну та другорядну осі, які проходять через центр ваги. Головна вісь об'єкту на бінарному зображенні може бути знайдена як вісь, відносно якої пікселі об'єкта мають найменший момент другого порядку. Другорядна вісь перпендикулярна головній. Головна та другорядна осі аналогічні великій та малій осям еліпса. Вісь характеризується кутом відносно вертикальної осі декартових координат. Розрахунок кута головної осі наступний[1]:

1. $\mu_{rc} = 0, \mu_{rr} > \mu_{cc}$

$$\alpha = 0^{\circ}$$

2. $\mu_{rc} = 0, \mu_{rr} \le \mu_{cc}$

$$\alpha = 90^{\circ}$$

3. $\mu_{rc} \neq 0, \mu_{rr} > \mu_{cc}$

$$\alpha = tg^{-1}\left\{\frac{\left[\mu_{cc} + \mu_{rr} + [(\mu_{cc} - \mu_{rr})^2 + 4\mu_{rc}^2]^{1/2}\right]^{1/2}}{-2\mu_{rc}}\right\}$$

4. $\mu_{rc} \neq 0, \mu_{rr} \le \mu_{cc}$

$$\alpha = tg^{-1}\left\{\frac{-2\mu_{rc}}{\mu_{rr} - \mu_{cc} + [(\mu_{rr} - \mu_{cc})^2 + 4\mu_{rc}^2]^{1/2}}\right\}$$

Кут другорядної осі $\beta = \alpha + 90^{\circ}$.

Знайшовши кут осей, розраховуємо момент другого порядку відносно головної осі за формулою[1]:

$$\mu_{\bar{r},\bar{c},a} = \frac{1}{A}\sum_{(r,c)\in R}\left((r-\bar{r})cos\alpha + (c-\bar{c})sin\alpha\right)^2 \qquad (1.7)$$

Формула для розрахунку моменту другого порядку відносно другорядної осі аналогічна.

6. Приклад застосування

Моменти другого порядку є морфологічними характеристиками та можуть застосовуватись для розпізнавання об’єктів на зображенні.

Автором статті реалізована програма розрахунку моментів другого порядку та розпізнавання об’єкта на основі готових моментів другого порядку еталонів. Об’єктом розпізнавання обрано цифри. Використано моменти другого порядку μ_{rr}(1.4), μ_{cc} (1.5), μ_{rc}(1.6), $\mu_{\bar{r},\bar{c},a}$(1.7) головної та другорядної осей. Наприклад момент μ_{cc} для цифри 1 має менше значення ніж μ_{cc} для цифри 2, моменти $\mu_{\bar{r},\bar{c},a}$ відносно головної осі матимуть різні значення для цифр 1 та 2.

На першому етапі розраховано моменти другого порядку для бінарних зображень розміром 50x50 з цифрами від 0 до 9. Ці еталонні дані збережено в файлі в структурі JSON. Наприклад для цифри 1 сформовано наступний JSON об’єкт: {"muRCaL":29.378721237182617,"muRC":24.95986 3662719727,"muRR":201.96498107910156,"muCC": 21.200931549072266,"muRCaS":193.787185668945 3,"value":1}. На другому етапі малюється цифра такого ж розміру як еталонні цифри, знаходяться моменти другого порядку цифри, знаходиться різниця моментів другого порядку з кожним еталоном, еталон з найменшою різницею є шуканим класом вхідного об’єкту.

Експерименти показали, що розпізнавання цифр 1, 2, 3, 4, 5, 7, 8 працює вірно, розпізнавання цифр 6 може дати результат 9, розпізнавання цифри 9 може дати результат 6. Таким чином даний метод не дає 100% правильний результат, адже моментів другого порядку недостатньо для повного розпізнавання цифр.

Перевагою даного методу є те, що для класифікації використовуються характеристики еталонів, самі еталони не завантажуються в пам'ять та не обробляються при розпізнаванні.

7. Висновок

В статті детально розглянуто двухпрохідний порядковий алгоритм маркування зв’язних компонент, запропоновано модифікацію алгоритму, яка полягає в упорядкуванні маркерів, та приведено опис реалізації алгоритму з використанням динамічних масивів для зберігання структур об'єднання-пошуку. Порівняння часу роботи класичного та рекурсивного алгоритму показало, що класичний алгоритм програє в швидкості рекурсивному, однак рекурсивний алгоритм при розмірі зображення 500x500 пікселів та більше призводить до переповнення стеку задач. Класичний алгоритм дозволяє обробляти зображення великих розмірів , при цьому збільшується час обробки.

Наведено приклад застосування алгоритму маркування зв’язних компонент – морфологічний аналіз маркірованих компонент та розпізнавання цифр на основі даних морфологічного аналізу. При розпізнаванні цифр за допомогою моментів другого порядку, більшість цифр розпізнаються правильно, цифри, схожі по структурі, можуть взаємо замінятись.

Список посилань

1. Л. Шапиро, Дж. Стокман, Компьютерное зрение, М.: Бином. Лаборатория знаний, 2006.
2. A. Rosenfeld, P. Pfaltz, “Sequential Operations in Digital Picture Processing”, Journal of the Association for Computing Machinery, Vol. 12., 1966.
3. Інтернет ресурс. Image moment. https://en.wikipedia.org/wiki/Image_moment