УДК 632.381

Р.А. Мельник, О.А. Алексєєв
Національний університет "Львівська політехніка",
кафедра ПЗ

ДОСЛІДЖЕННЯ АЛГОРИТМІВ ДЕКОМПОЗИЦІЇ ВІЗУАЛЬНИХ ОБРАЗІВ ЗА ХАРАКТЕРИСТИКАМИ КОМПРЕСІЇ

© Мельник Р.А., Алексєєв О.А., 2004

Запропоновано підхід декомпозиції візуальних образів на основі процедур глобального та локального характеру. Перша об'єднує алгоритми згортання складових мікрооб'єктів візуальних образів, а друга – обхід ієрархічного дерева для повного перебору варіантів покриття візуальних образів прямокутниками.

The approach for visual image decomposition based on the global and local optimization procedures is considered. The first one combines the rolling up algorithms for image micro components and the second one uses the hierarchical tree for the exhaustive search of coverage variants by rectangles.

Вступ. Для створення оптимальних алгоритмів опрацювання, зберігання та розпізнавання візуальних образів останні повинні бути оптимальними та формально описані. Одним з можливих способів формалізації візуальних образів є розбиття їх на правильні геометричні фігури, наприклад, прямокутники. Їх можна прийняти як початкові дані для процедур формування, спрощення та кодування образів. Наступний матеріал присвячено розробленню та дослідженню деяких алгоритмів кодування візуальних образів прямокутниками за різними стратегіями.

Покриття кластерів зображення прямокутниками. Для задач розпізнання, зокрема для компресії опису образу, корисним є відтворення його за допомогою фігур правильної форми та побудови ієрархічного дерева вкладеності складових мікрообразів у зазначених фігурах. Необхідні також ознаки розпізнання образу загалом та його складових частин на різних рівнях дерева кластеризації мікрообразів.

Одним з підходів до розкладу та компресії образів є кластеризація мікрообразів [1,3]. Результатом згортання мікрооб'єктів образів, на які образ розбивається сіткою з заданим кроком, є підобрази складної форми. Їх опис є громіздкий, а це не приносить виграшу в об'ємі пам'яті для зберігання самого образу. Для розкладу образу на правильні прямокутники можливі два способи: згортання до рівня підкластерів, форма яких відповідає прямокутній, та пошук мінімального покриття прямокутниками на основі повного перебору комбінацій покриття образу прямокутниками [2]. Перший спосіб характеризується неможливістю побудови оптимального покриття за одне згортання мікрооб'єктів, а другий значними обчислювальними затратами для складних форм образів.

Накладемо на образ, що розбивається, сітку з кроком, який можна зобразити як *1x2, 2x1, 2x2* тощо. Клітини сітки містять різну кількість чорних пікселів і їх можливе об'єднання визначають критерії алгоритмів кластеризації. Наприклад, для випадку поділу площини горизонтальними лініями з кроком в один піксель (*1x2)* яскравість клітинки може приймати одне з набору значень: *0, 1, 2, 3.*

До мікроклітинок E образу O застосуємо алгоритм згортання. Він має багато ступенів керування: згортання тільки повністю заповнених клітинок, згортання їх разом з половинками, згортання половинок тощо. Результатами згортання є повністю або частково заповнені кластери-прямокутники або кластери-фігури неправильної форми. Часткова наповненість кластерів коливається від 50% (об'єднані тільки половинки) до 100 % (об'єднані тільки заповнені клітинки). На рис. 1 показано приклад побудови дерева згортання T мікроклітинок гіпотетичного образу. На дереві зображені рівні P^i , P^{i-1} розкладу образу на складові частини O_4, O_1.

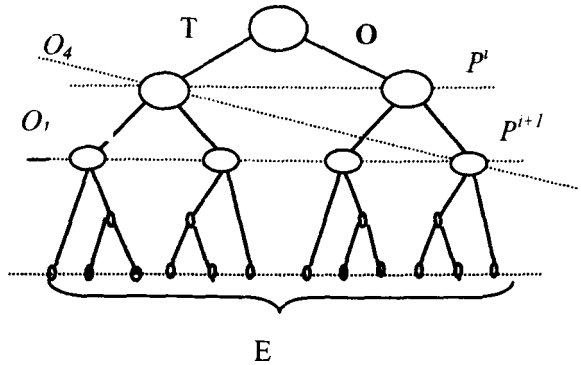

Рис. 1. Розбиття образу та ієрархічне дерево

Основні кроки алгоритму згортання:

П1. Розбиття простору.

П2. Поки не будуть виділені усі кластери, об'єднання суміжних підкластерів за умови дозволу.

П3. Виділення кластера, якщо він не має суміжних підкластерів.

В основі другого способу покриття образу прямокутниками – метод штучного інтелект-методу повернень, для якого пошук прямокутників покриття кластера можна зобразити ієрархічним деревом (рис. 2). Клітинки вузлів дерева містять розмір знайденого прямокутника на певному кроці алгоритму. Кореневий вузол неявно містить повну вагу образу (наприклад, в пікселях, але не обов'язково). Розгалуження у вузлах дерева пошуку відповідають всім можливим способам формування покриваючих прямокутників, пошук яких здійснюється на площині. Накладання прямокутників здійснюється методом повного перебору: сканування прямокутниками всіх можливих розмірів початкового прямокутника, який охоплює сканований образ.

Основні кроки алгоритму сканування:

П1. Розбиття простору.

П2. Визначення координат простору сканування (прямокутника охоплення).

П3. Поки не виділені усі кластери, накладання скануючого прямокутника до збігу його площі із сумарною площею образу. Видалення його з простору сканування.

П4. Зменшення розмірів прямокутника сканування до мінімального, або відсутність непокритих образів.

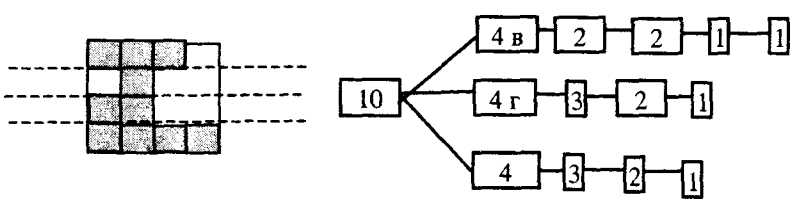

Рис. 2. Образ та дерево пошуку оптимального покриття

На рис 2, а зображено образ, що складається з 10 пікселів. Фрагмент дерева пошуку оптимального розбиття образу на прямокутники показано на рис. 2, б. Показані три шляхи: 1) першим знайдено максимальний вертикальний прямокутник (4 пікселі), 2) першим є горизонтальний прямокутник (4 пікселі), 3) першим є квадрат (4 пікселі). Два останні з наведених шляхів у графі пошуку приводить до оптимальної кількості прямокутників (4). Перший шлях приводить до п'яти прямокутників

На рис. 3 зображені приклади фрагментації образу букви „а" (розмір файла 1196 байт) за допомогою алгоритму кластеризації множини E – заповнених 1-клітинок (рис. 3, б) та розбиття букви алгоритмом повного перебору (рис. 1, в). У першому випадку отримано 59 (розмір файла 295 байт) прямокутників за 1 с, у другому – скануванням протягом 12 с отримано 46 (розмір файла 210 байт) прямокутників.

Рис. 3. Прямокутники, отримані кластеризацією та скануванням

Для образів тоншої структури різниця у часових характеристиках збільшується, що пояснюється ростом дерева пошуку як в глибину, так і в ширину. На рис. 4 зображені приклади фрагментації образу букви „к" (розмір файла 2436 байт) за допомогою алгоритму кластеризації (рис. 4, б) та розбиття букви алгоритмом повного перебору (рис. 4, в). У першому випадку отримано 67 (розмір файла 335 байт) прямокутників за 0.1 с, у другому – скануванням протягом 121 с отримано 43 (розмір файла 265 байт) прямокутників.

Рис.4. Прямокутники, отримані кластеризацією та скануванням

Всі розглянуті приклади опрацьовувались алгоритмами без втрат, тобто точними в сенсі побудованих прямокутників.

Покриття зображення прямокутниками з втратами. Розглянемо застосування розроблених алгоритмів за допущення певної втрати інформації. Під час побудови початкових клітинок вони залежно від розмірів та заповнення можуть мати різну характеристику яскравості. Керування параметром яскравості кластера є ключем до відсотка компресії та можливих втрат, якщо для кластера зберігати тільки характеристику яскравості без координат незаповнених клітинок. Втрати зображення виникають через довільне (наприклад, випадкове) доповнення кластера при відновленні зображення.

При покритті образів прямокутниками згортанням необхідне обчислення функцій, що відповідають за дозвіл на утворення прямокутника. Введемо позначення: m_i – кількість всіх пікселів в кластері, z_i, p_i – кількість заповнених і пустих пікселів у кластері. Тоді характеристика $v_i = p_i /(p_i + z_i)$ відношення кількості пустих до загальної кількості заповнених пікселів, b_i – коефіцієнт подібності фігури кластера до прямокутника тощо. Наявність таких характеристик дозволяє під час сканування не тільки виділяти повністю заповнені прямокутники, але і апроксимувати прямокутниками певні області, близькі до них згідно з заданими обмеженнями на коефіцієнти заповнення тощо. Нестрогим покриттям можна компенсувати строгість кластеризації зображення і навпаки.

Розглянемо приклади застосування алгоритмів покриття образів з втратами. На рис. 5 показано приклад покриття букви „а" кластеризацією початкових клітинок *2x1*. При коефіцієнті втрат 50% (рис. 5, а) отримано 33 прямокутники, для коефіцієнта втрат 25% кластеризацією отримано 40 прямокутників (рис. 5, а). Для початкових клітинок *2x1* і тих же коефіцієнтів втрат маємо відповідно 29 і 32 прямокутників. Часові характеристики алгоритму практично не залежать від коефіцієнта втрат.

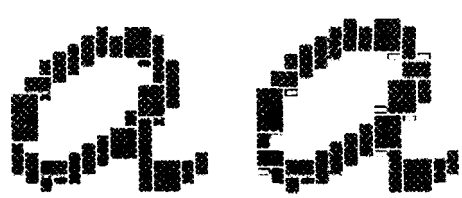

Рис. 5. Розбиття кластеризацією з втратами

Застосування коефіцієнта втрат для методу сканування полягає у введенні похибки під час визначення сумарної площі підкластерів, які увійшли до складу прямокутника-сканування. Для методу кластеризації коефіцієнт втрат враховується під час перевірки на правильність форми двох суміжних підкластерів. Ілюстрацією застосування алгоритму сканування до букви „а" є рис. 6, де відповідно отримано 14 прямокутників (коефіцієнт втрат 50%) та 33 прямокутників (коефіцієнт втрат 25%).

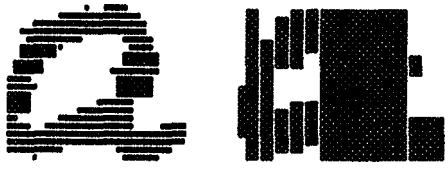

Рис. 6. Розбиття сканування з втратами

Програмна реалізація. На основі запропонованих алгоритмів та моделей розроблено програму кластеризації образів, яка містить три основні кроки: підготовку мікрооб'єктів кластеризації (клітинок), побудову дерева згортання та перетворення кластерів непрямокутної форму у набори прямокутників. Основними об'єктами в програмі є : 1) *TTree* – для позначення дерева згортання (однонапрямлений список вказівників на об'єкти типу *TCluster*). Об'єкт володіє такими властивостями та методами : *Begin* – вказівник на початок дерева, *Cell* – розмір мінімального кластера у просторі *1x1, 1x2, 2x1, 2x2, InitMicroCluter()* – метод розбиває простір на початкові підкластери, *CreateCluster()*– метод створює новий кластер, *BuildTree()*– метод побудови дерева згортання; 2) *TCluster :: TMicroCluster* – нащадок об'єкта *TMicroCluster*. Призначення об'єкта – опис кластерів, які утворюються під час аналізу суміжності підкластерів дерева згортання. Властивості:

Part1, Part2 – вказівники на суміжні між собою підкластери, *Rectangle* – ознака форми кластера (прямокутник, квадрат). Методи: *isRectangle()* – метод для визначення ознаки правильності форми кластера; **3) *TMicroCluster*** – для опису початкових мікрооб'ктів, на які розбивається простір. Властивості: *Brightness* – ознака кольору кластера, *Coordinate* – координати мікрокластера у просторі, *pRight* - вказівник на суміжний справа мікрокластер, *pDown* – вказівник на суміжний вниз мікрокластер, *Part* – ознака того, чи є кластер складовою іншого кластера. Методи: *GetBrightness()*– метод повертає ознаку яскравості. Використовується для алгоритмів покриття із втратами, *GetNeighborRight()* – повертає вказівник на суміжний справа підкластер, *GetNeighborDown()* – повертає вказівник на суміжний вниз підкластер, *isCluster()* – визначає входження кластера у склад кластерів вищих рівнів.

У таблиці наведені результати компресії чорно-білих зображень у форматі BMP методами кластеризації та сканування відповідно. Наведені також результати компресії цих же зображень сучасними методами стиску інформації. Видно, що розроблені методи дозволяють отримати кращу компресію.

Таблиця

Результати декомпозиції візуальних образів та характеристики компресії

ім'я файла	Розмір файла у байтах				
	Bmp	**Згортання**	**Сканування**	**Rar**	**Zip**
1.BMP	382	132	104	204	228
11.bmp	382	84	76	177	213
13.bmp	382	108	100	184	219
A2.BMP	318	236	176	226	257
A3-10.bmp	374	240	168	231	267
A3.BMP	270	236	168	212	244
A4.BMP	310	220	160	221	248

Продовження таблиці

1	2	3	4	5	6
b.bmp	382	92	80	177	205
b1.bmp	382	124	116	204	226
b3.bmp	382	116	104	195	223
b4.bmp	382	112	96	189	217
b5.bmp	382	100	84	180	212
c.bmp	162	52	48	148	173
C1.BMP	170	104	76	179	210
d.bmp	382	152	140	207	241
d1.bmp	382	136	128	203	235
d2.bmp	382	84	72	174	203
d3.bmp	382	172	148	214	246
d4.bmp	382	96	96	196	224
DDD.bmp	414	160	148	225	257
e.bmp	382	128	104	196	222
e1.bmp	382	148	116	196	227
e2.bmp	382	92	76	180	208
e3.bmp	382	132	112	200	229
g.bmp	382	108	96	200	221
g1.bmp	382	124	92	191	220
g2.bmp	382	76	68	177	204
g3.bmp	382	108	84	194	218
I3.BMP	198	108	88	179	210
myS.bmp	162	84	68	163	197
o.bmp	162	52	48	152	178
v.bmp	382	212	180	217	255
v1.bmp	382	140	104	203	241
v2.bmp	382	176	144	213	245
v3.bmp	382	132	120	197	230
v33.bmp	510	208	176	244	276
Y.bmp	382	188	172	217	248
Y1.bmp	382	240	184	236	266
Y2.bmp	382	300	240	239	271
z.bmp	382	148	132	208	242
z1.bmp	382	120	100	196	223
z2.bmp	382	184	14	215	252
z3.bmp	382	124	108	200	233

Висновки. Проведені дослідження з алгоритмами покриття образів алгоритмами з різною ідеологією показали, що згортання є потужним швидкодіючим інструментом опрацювання образів, який має важелі для управління якістю розбиття (кількості прямокутників, їх наповнення тощо). Цими важелями є коефіцієнти втрат та розміри початкових клітинок – об'єктів згортання. Через наявність параметрів управління та непрогнозованість згортання однократним згортанням отримати найкращий результат покриття важко. Повний перебір характеризується можливістю знаход-

ження оптимального розбиття при значних часових затратах. У той же час застосування коефіцієнтів втрат для сканування зменшує часові затрати, зменшує кількість прямокутників, але, не враховуючи специфіки зображення, огрубляє наближення.

1. Мельник Р.А., Алексеев О.А., Кластеризація мікрообразів для кодування зображень // Пр. міжн. конф. Укробраз'2004. – К. – 2004, с. 81–85. 2. Мельник Р.А., Алексеев О.А. Покриття образів прямокутниками // Вісник Національного університету "Львівська політехніка" "Комп'ютерна інженерія та інформаційні технології". – 2004. 3. Мельник Р.А. Декомпозиція графів на основі нечіткого дерева згортання // Вісник Національного університету "Львівська політехніка" "Комп'ютерна інженерія та інформаційні технології". – Львів, 1998, № 351. – С. 65–69.

УДК 621.3.019 : 51.001.57

О. Ю. Лозинський, С. В. Щербовських
Національний університет "Львівська політехніка",
кафедра електроприводу та автоматизації промислових установок

СИНТЕЗ МОДЕЛЕЙ НАДІЙНОСТІ НА ОСНОВІ МЕТОДУ МОНТЕ-КАРЛО ІЗ ЗАСТОСУВАННЯМ КОНЦЕПЦІЇ ПРОСТОРУ СТАНІВ

© Лозинський О.Ю., Щербовських С.В., 2004

Запропоновано алгоритм побудови моделей надійності на основі методу Монте-Карло із застосування концепції простору станів. Подано приклад такої моделі надійності для ремонтованого об'єкта із двох елементів, які сполучені логічно послідовно.

In this paper the Monte-Carlo reliability model construction algorithm on the basis of state space conception application is offered. The example of such reliability model for two-component series repairable item is given.

Вступ. Під час проектування ремонтованих електромеханічних об'єктів одним із важливих показників, який необхідно забезпечити, є коефіцієнт готовності, що визначає імовірність перебування об'єкта в справному стані в заданий момент часу. Тривалий час єдиний спосіб розв'язання цієї задачі ґрунтувався на аналізі відповідної однорідної марківської моделі надійності такого об'єкта. Такі моделі надійності [1], в основі яких лежить концепція простору станів, формуються із умови, що всі випадкові процеси відмов та відновлення, що відбуваються в об'єкті, підпорядковуються тільки експоненціальним законам розподілу. При розрахунку коефіцієнта готовності електромеханічних об'єктів цей підхід не забезпечує високої адекватності, оскільки характеристики випадкових процесів, що відбуваються в досліджуваних об'єктах, не можуть бути з високим ступенем точності апроксимовані експоненціальним законом розподілу [2]. Із розвитком комп'ютерної техніки, для практичного розрахунку коефіцієнта готовності, набув застосування метод Монте-Карло, в основі якого лежить використання генератора випадкових чисел. На теперішній час можна констатувати, що моделі надійності на основі методу Монте-Карло багатоелементних ремонтованих об'єктів замінили собою відповідні однорідні марківські моделі надійності в більшості практичних досліджень, оскільки такі моделі взагалі не накладають обмежень на закони розподілу. Метод Монте-Карло передбачає декілька різних способів побудови моделей надійності, причому, залежно від об'єкта, різні типи таких комп'ютерних моделей можуть бути різною мірою зручними.