Методы и средства обработки сигналов и изображений

УДК 004.932.2

А.Ю. Варфоломєєв, О.М. Лисенко, д-р техн. наук

Порівняльний аналіз сучасних алгоритмів автоматизованої сегментації зображень

Рассмотрены алгоритмы автоматизированной сегментации на основе кластеризации по *k*-средних, максимизации-ожидания, сдвига среднего, нормализированного пересечения графов, взвешенной агрегации, статистического объединения областей, JSEG, HGS и ROI-SEG. Представлены результаты сегментации, а также проведен анализ качества и быстродействия каждого из алгоритмов на естественном, спутниковом и содержащем текстуры изображениях.

Unsupervised image segmentation algorithms based on *k*-mean clustering, expectation-maximization, mean-shift, normalized graph cut, weighted aggregation, statistical region merging, JSEG, HGS and ROI-SEG are considered. The results of segmentation obtained by mentioned algorithms on textural, satellite and natural images are presented. The analysis of quality and segmentation speed of each algorithm realization is performed.

Ключевые слова: *сегментация изображений, выделение однородных областей на изображениях, анализ качества сегментации.*

Вступ

Відомо, що сегментація зображень знаходить широке застосування в медицині, системах спостереження, робототехніці, при контролі якості продукції на виробництві, проведенні аналізу зображень отриманих від супутників та під час аерофотозйомки тощо. Це робить її однією з найбільш актуальних і водночас складних задач комп'ютерного зору та, як наслідок, викликає великий інтерес у дослідників [1–10]. Як наслідок, на сьогодні запропоновано і реалізовано велику кількість різноманітних алгоритмів сегментації, що в свою чергу створює проблему вибору серед них найбільш оптимального для розв'язання конкретної задачі.

Вирішенню зазначеного завдання і присвячена дана робота, метою якої є порівняльний аналіз сучасних алгоритмів сегментації зображень на основі критеріїв якості та швидкодії.

На жаль, охопити всі наявні алгоритми сегментації в межах однієї публікації практично неможливо, тому увагу, головним чином, зосереджено на автоматизованих алгоритмах, що мають високу швидкодію. У подальшому під автоматизованими розуміються такі алгоритми сегментації, які не потребують з боку користувача позначення початкового розбиття (seed points). Швидкодіючими вважаються алгоритми, для яких час обробки зображення розміром 512×512 пікселів на тестовому комп'ютері не перевищує однієї хвилини (див. розд. 3).

В роботі проведено порівняльний аналіз реалізацій дев'яти алгоритмів, а саме: сегментації по *k*-середніх, максимізації-очікування (EM – Expectation-Maximization), зсуву середнього, сегментації на основі нормалізованого перетину графів, зваженого (SWA – Segmentation by Weighted Aggregation) та статистичного об'єднання областей (SRM – Statistical Region Merging), алгоритму JSEG, сегментації кольорових текстур HGS та алгоритму ROI-SEG. Критеріями порівняння, як уже зазначалося, виступають якість сегментації та швидкодія. Оцінювання якості здійснюється за показниками ефективності, прийнятими в роботі [10].

Також здійснено короткий огляд кожного з досліджуваних алгоритмів, розглянуто критерії оцінювання якості сегментації згідно з представленими в роботі [10], наведено приклади обробки текстурного, супутникового та природного зображень за допомогою програмних реалізацій алгоритмів та виконано їх оцінювання. Наприкінці публікації зроблено висновки щодо якості сегментації та швидкодії розглянутих алгоритмів.

1. Аналіз алгоритмів сегментації зображень

1.1. Сегментація по k-середніх

Сегментація зображення цим алгоритмом здійснюється за допомогою кластеризації по *k*-середніх. Її суть полягає в оптимізації цільової функції, яка відображає якість розбиття на клас-

тери (сегменти), заданого набору даних (зображення). Цільову функцію отримують із припущення, що існує *k* кластерів, де *k* є відомим числом, вважаючи, що конкретний елемент даних розташовується поблизу центру відповідного йому кластера. Таким чином, у ролі цільової функції виступає деяка міра відстані, наприклад, евклідова:

$$\Phi = \sum_{i=1}^{k}\left[\sum_{j\in \mathbf{I}_i}(\mathbf{x}_j - \mathbf{c}_i)^T(\mathbf{x}_j - \mathbf{c}_i)\right], \qquad (1)$$

де *k* – кількість кластерів; $\mathbf{I}_i$ – множина індексів елементів, що належать до *i*-го кластеру. Процедура кластеризації може бути описана наступною послідовністю кроків:

1. Обрати *k* інформаційних точок як центри кластерів.

2. Віднести кожну інформаційну точку до того кластера, відстань до якого від даної точки є найменшою.

3. Впевнитися, що кожен кластер містить хоча б одну точку. Для цього кожен пустий кластер може бути доповнений довільною інформаційною точкою, розташованою далеко від його центру.

4. Центр кожного кластера замінити середнім значенням елементів, що йому відповідають.

5. Повторювати кроки 2-4, доки не припиниться зміна центрів кластерів.

Вибір центрів кластерів на першому кроці зазвичай здійснюється випадковим чином. Із часом описаний вище процес збігається до локального мінімуму цільової функції (1), водночас збіжність до глобального мінімуму не гарантується.

Алгоритм є простим у реалізації, має високу швидкодію. Його особливості полягають у необхідності заздалегідь знати кількість сегментів, а через випадковий вибір центрів кластерів при ініціалізації результат сегментації може виявитися різним для одного й того ж зображення. Слід також відзначити, що кластеризація по *k*-середніх використовується більш складними алгоритмами сегментації для отримання початкового розбиття на сегменти. Детальніше алгоритм сегментації по *k*-середніх описано в роботі [1].

1.2. Сегментація за допомогою максимізації-очікування

Якщо сегменти розглядати як ділянки з певними унікальними розподілами ознак, а все зображення – як лінійну комбінацію (суміш) цих розподілів, то можна сегментувати за допомогою алгоритму максимізації-очікування (EM). Цей алгоритм дозволяє оптимальним чином розбивати суміш розподілів імовірності на *k* складових. Для цього вводяться допоміжні приховані змінні *z*, які разом зі відомими змінними x, дозволяють оцінити очікуване значення логарифмічної функції правдоподібності суміші, складеної із розподілів із параметрами θ та вагами w. Далі проводиться максимізація отриманого очікування. Процедури пошуку і максимізації повторюються доти, доки не буде досягнуто заданої точності функції правдоподібності або певної кількості ітерацій. Відповідно алгоритм складається з двох кроків:

1. E-крок (expectation): пошук очікуваного значення функції правдоподібності.

2. M-крок (maximization): максимізація значення очікуваної правдоподібності, отриманої на E-кроці.

Сегментація виконується за оптимальними значеннями параметрів розподілень θ, отриманими при досягненні збіжності алгоритму.

В більшості задач сегментації, в яких використовується EM-алгоритм вважається, що компоненти суміші описуються нормальними розподілами. В дійсності це виконується не завжди і може бути причиною неякісної сегментації. Іншими особливостями алгоритму є низька чутливість до шуму та неповноти даних, можливість отримання заданої кількості сегментів, доволі швидка збіжність. Слід також зазначити, що описаний вище алгоритм сегментації по k-середніх є окремим випадком EM-алгоритму.

Докладніше алгоритм максимізації очікування для сегментації зображень розглянуто в роботі [1].

1.3. Сегментація зсувом середнього

Алгоритм зсуву середнього (mean shift) є надійним алгоритмом для пошуку локальних екстремумів щільності розподілу певної множини даних. Пошук максимуму щільності розподілу, що містить *n* точок даних $\mathbf{x}_1,\ldots,\mathbf{x}_n$ у *d*-мірному просторі $\mathbf{R}^d$ здійснюється за допомогою наступного рівняння :

$$\mathbf{y} = \frac{\sum_{i=1}^{n}\mathbf{x}_i g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}{\sum_{i=1}^{n} g\left(\left\|\frac{\mathbf{x}-\mathbf{x}_i}{h}\right\|^2\right)}, \qquad (2)$$

де $\mathbf{x}_i$ – поточне положення центру ядра; $g(\cdot)$ – похідна профілю ядра взята зі знаком мінус, визначає форму вікна, в якому здійснюється пошук екстремуму щільності розподілу; *h* – так звана смуга пропускання, визначає розмір ядра (вікна пошуку); **y** – положення максимуму щільності розподілу у вікні; $\|\cdot\|$ – евклідова норма.

Сегментація зображень на основі алгоритму зсуву середнього виконується за такими кроками [2]:

1. Застосувати фільтрацію зсуву середнього до зображення та зберегти всю інформацію про *d*-мірні точки збіжності у $\mathbf{z}_i$:

1.1. виконати ініціалізацію $j = 1$ та $\mathbf{y}_{i,1} = \mathbf{x}_i$;

1.2. обчислювати $\mathbf{y}_{i,j+1}$ за допомогою (2), доки вікно не зупиниться, тобто не буде досягнуто збіжності $\mathbf{y} = \mathbf{y}_{i,c}$;

1.3. присвоїти $\mathbf{z}_i = \{\mathbf{x}_i^s, \mathbf{y}_{i,c}^r\}$. Верхні індекси *s* та *r* позначають просторову компоненту та компоненту значень векторів відповідно.

2. Виділити в об'єднаній ділянці кластери $\{\mathbf{C}_p\}_{p=1,\ldots,m}$, шляхом групування разом усіх $\mathbf{z}_i$, які ближче, ніж h_s у просторовій області та h_r у просторі ознак.

3. Для кожного $i = 1,\ldots,n$ присвоїти $L_i = \{p \mid \mathbf{z}_i \in \mathbf{C}_p\}$.

4. За необхідності виконати процедуру усунення просторових областей, що містять менше, ніж *M* пікселів.

У наведеному вище алгоритмі використано наступні позначення: $\mathbf{x}_i$ та $\mathbf{z}_i$, $i = 1,\ldots, n$ – *d*-мірне вхідне зображення та відфільтроване зображення в об'єднаній області простору-ознак відповідно, а L_i – мітка (індекс сегмента) *i*-го пікселя на сегментованому зображенні.

Особливості алгоритму зсуву середнього полягають у доволі високій його швидкодії, відносній простоті реалізації та автоматичному визначенні кількості кластерів. Зсув середнього, однак, потребує знання емпіричних параметрів h, h_s, h_r, які можуть чинити значний вплив на якість сегментації.

Детальний опис алгоритму міститься в роботі [2].

1.4. Алгоритм нормалізованого перетину графів

Зображення в цьому алгоритмі розглядається як зважений ненаправлений граф $G = (V, E)$, в якому вершини V є точками в просторі ознак, а гілки E представлено ваговими коефіцієнтами $w(i, j)$, рівними значенню функції подібності між відповідними цим гілкам вузлами – *i* та *j*.

Нормалізований перетин передбачає розділення графу на дві частин, і визначається виразом:

$$Ncut(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}, \qquad (3)$$

де $cut(A, B)$ – вага всіх гілок, по яким здійснюється перетин графу на підмножини, $assoc(A, V) = assoc(A, A) + cut(A, B)$ – сума вагових коефіцієнтів гілок, асоційованих із підмножиною *A*, у тому числі й тих, за якими здійснюється перетин; $assoc(B, V) = assoc(B, B) + cut(A, B)$ – сума вагових коефіцієнтів гілок, асоційованих із підмножиною *B*.

До того ж нормалізація $cut(A, B)$ за допомогою знаменників $assoc(A, V)$ та $assoc(B, V)$ має запобігати утворенню занадто малих перетинів, тобто надмірної сегментації.

У роботі [3] показано, що перетин, для якого значення (3) мінімальне, дає підмножини *A* та *B* такі, що зв'язок між ними є мінімальним, а в межах підмножин – максимальним.

Водночас пошук нормалізованого перетину безпосередньо за формулою (3) є NP-повною задачею, тому на практиці його здійснюють шляхом розв'язання системи власних векторів.

Алгоритм сегментації на основі нормалізованого перетину можна узагальнити наступним чином:

1. За множиною ознак побудувати зважений граф *G*, обчислити для його гілок вагові коефіцієнти та зберегти інформацію у матрицях **W** та **D**: у матриці **W** – вагові коефіцієнти подібності між кожною із вершин, а у діагональній матриці **D** – на головній діагоналі суму рядків матриці **W**.

2. Вирішити систему власних векторів $(\mathbf{D} - \mathbf{W})\mathbf{x} = \lambda\mathbf{D}\mathbf{x}$, де λ - власні числа.

3. Використати другий найменший власний вектор для розділення графу на дві частини.

4. Вирішити, чи необхідно розбивати отримані підграфи на менші частини, для чого перевірити стабільність отриманого перетину.

5. За потреби, повторити алгоритм для отриманих сегментів.

Оскільки матриця **W** містить коефіцієнти подібності між *кожною* із вершин, її розмірність рівна $(m \times n)^2$, де *m* та *n* – ширина та висота зображення у пікселях. З огляду на це алгоритм потребує значного об'єму пам'яті під час роботи із великими зображеннями. Також через необхідність розв'язання системи власних векторів нормалізований перетин є доволі обчислювально-складним. До особливостей алгоритму слід віднести можливість автоматичного визначення кількості сегментів.

Детальніше ознайомитись із алгоритмом сегментації на основі нормалізованого перетину графів можна в роботі [3].

1.5. Алгоритм зваженого об'єднання областей

Алгоритм сегментації на основі зваженого об'єднання областей (SWA) дуже подібний до алгоритму нормалізованого перетину. Зображення тут розглядається як зважений граф, а для його розбиття також використовується міра нормалізованого перетину. Відмінність між описаним вище алгоритмом нормалізованого пере-

тину графів та зваженим об'єднанням полягає в тому, що в останньому нормалізований перетин знаходиться дещо в інший спосіб і на багатьох масштабах. При цьому на кожному кроці на основі зібраної по сегментам статистики будується нове, більш грубе, в порівнянні з попереднім, розбиття. Загрублення (coarsing) здійснюється власне за допомогою процедури зваженого об'єднання (weighted aggregation):

$$\mathbf{W}_i \approx \mathbf{P}^T \mathbf{W}_{i-1} \mathbf{P} \qquad (4)$$

де **P** – розріджена інтерполяційна матриця, отримана за допомогою багатосіткового методу (algebraic multigrid); $\mathbf{W}_i$ та $\mathbf{W}_{i-1}$ – вагові матриці, що відображають відмінність між вершинами графу на поточному та попередньому кроках алгоритму відповідно.

Матриця **W** є симетричною, з нулями на головній діагоналі і складається з елементів, які обчислюються у вигляді $\exp[-\alpha D(a) - \beta D(b) - - \gamma D(c) - \ldots]$, де $\alpha, \beta, \gamma, \ldots$ – додатні вагові коефіцієнти, $D(\cdot)$ – міра подібності між вершинами в деякому просторі ознак.

Спрощено алгоритм зваженого об'єднання областей узагальнемо наступним чином:

1. За заданим зображенням побудувати зважений граф G_0. Знайти початкові значення матриці подібності $\mathbf{W}_0$ на основі яскравості.

2. Із G_{i-1} для $i = 1, 2, \ldots$ будувати нові графи G_i, для чого:

2.1. обрати множину вершин V_i, таку, що $V_{i-1} \setminus V_i$ має сильний зв'язок із V_i;

2.2. визначити інтерполяційну матрицю $\mathbf{P} = \mathbf{P}_{i-1}$;

2.3. обчислити зважене об'єднання (агрегацію), знайшовши $\mathbf{W}_i$ за допомогою рівняння (4);

2.4. для кожної вершини графа, її околу та сегменту, до якого вона належить, обчислити ознаки;

2.5. уточнити матрицю подібності $\mathbf{W}_i$ відповідно до знайдених на кроці 2.4 ознак.

На кроці 2.4 у якості ознак авторами запропоновано використовувати різні характеристики зображення: ширину, висоту та орієнтацію сегментів на заданому масштабі (обчислюються за допомогою перших та других просторових моментів), середню яскравість, міжмасштабну дисперсію, а також відгуки фільтрів виділення границь. При цьому на різних масштабах допускається використання різних ознак. В подробицях алгоритм описано в роботі [4].

1.6. Алгоритм статистичного об'єднання областей

Сегменти в алгоритмі статистичного об'єднання (SRM) формуються шляхом об'єднання областей із подібними статистичними властивостями. Подібними тут вважаються ділянки, пікселі яких мають однакові математичні сподівання на заданому каналі кольору. Для несхожих областей математичне сподівання має відрізнятися принаймні по одному з каналів кольору. При цьому кожен канал замінюється множиною Q незалежних випадкових змінних, що приймають додатні значення в межах g / Q, де g – максимальна кількість рівнів яскравості даного каналу (зазвичай $g = 256$). Величина Q контролює ступінь «грубості» сегментації [5]. Об'єднання областей здійснюється за виконання спеціальної умови – предикату об'єднання (merging predicate), що приймає логічне значення та має вигляд :

$$P(R_1, R_2) = \begin{cases} 1, \text{ якщо } |\overline{R_2} - \overline{R_1}| \le \sqrt{b^2(R_1) + b^2(R_2)}; \\ 0, \text{ в іншому випадку.} \end{cases} \qquad (5)$$

де R_1, R_2 – множини пікселів двох областей зображення, $\overline{R}$ – середнє значення пікселів в області R по одному з каналів кольору, $b(R) = g\sqrt{1/(2Q|R|)\ln(|R_{|R|}|/\delta)}$, де $|R|$ – кількість пікселів у області R, $|R_{|R|}|$ – загальна кількість пікселів у всіх областях, що містять $|R|$ пікселів, $\delta = 1/(6|I|^2)$, $|I|$ – кількість пікселів на зображенні.

Алгоритм сегментації є досить простим і складається лише з трьох кроків:

1. Знайти множину пар суміжних пікселів S_l та застосувати до неї функцію $f(p_1, p_2)$.

2. Відсортувати множину S_l в порядку зростання $f(p_1, p_2)$.

3. Обійти множину S_l і прийняти рішення щодо об'єднання пар пікселів $(p_1, p_2) \in S_l$, для яких $R(p_1) \neq R(p_2)$ ($R(p)$ позначає область, до якої належить піксель p), на основі предикату об'єднання $P(R(p_1), R(p_2))$ згідно з формулою (5).

Функція $f(p_1, p_2)$ визначає відмінність між пікселями p_1 та p_2, і може визначатись як модуль різниці: $f(p_1, p_2) = |p_2 - p_1|$ або бути відгуками градієнтних фільтрів виділення границь Собела [1 2 1] чи Превіта [–1 0 0 1]. Алгоритм також може бути удосконалений для роботи за наявності перешкод та шумових спотворень. До особливостей алгоритму сегментації за допомогою статистичного об'єднання областей належать його відносна простота реалізації та невисока обчислювальна складність [5].

1.7. Алгоритм JSEG

Сегментація цим алгоритмом здійснюється завдяки пошуку однорідних областей на кольоровому зображенні. Для цього здійснюється квантування кольорів, під час якого визначається первинна кількість класів та формується їх карта

(class-map). Карта класів – це масив, елементами якого є двовимірні вектори координат точок (x, y). Кожна точка карти відповідає одному з C класів кольору. Далі для здійснення сегментування вводиться спеціальний критерій:

$$\overline{J} = \frac{1}{N}\sum_{k} M_k J_k \qquad (6)$$

де k – кількість областей; M_k – кількість точок в області k; N – загальна кількість точок в карті класів; J_k – значення J, обчислене для області k.

$$J = \frac{S_T - S_W}{S_W},$$

де:
$$S_T = \sum_{z \in Z} \| z - \mu \|^2,$$

$$S_W = \sum_{i=1}^{C} \sum_{z \in Z_i} \| z - \mu \|^2,$$

Z – множина елементів карти класів, μ – середнє значення всіх елементів карти класів, μ_i – середнє значення елементів, що належать до i-го класу.

Значення $\overline{J}$ відображає, наскільки неоднорідним є задане розбиття зображення на сегменти, чим воно більше – тим неоднорідність вища. Таким чином, задача отримання оптимальної сегментації зводиться до пошуку розбиття з мінімальним $\overline{J}$. У спрощеному вигляді алгоритм сегментації JSEG складається з наступних кроків:

1. Виконати квантування зображення по кольору та отримати карту класів.

2. Знайти значення J в округлих областях діаметром d та центром у кожному пікселі зображення на різних масштабах (початковий діаметр вікна в оригінальному алгоритмі складає $d_0 = 9$ і збільшується на кожному масштабі як: $d_i = 2d_{i-1} - 1$).

3. На основі знайдених значень J знайти вихідні точки сегментації (seed points).

4. Згідно з алгоритмом вирощування областей (region growing) отримати сегментацію зображення.

5. Провести заключну обробку, що передбачає об'єднання схожих між собою сегментів.

Квантування зображення по кольору на кроці 1 здійснюється на основі алгоритму Лойда [11] з використанням кольорової моделі CIE $L^*u^*v^*$. Процедура отримання вихідних точок сегментації на кроці 3 є емпіричною, для якої потрібна спеціальна таблиця. Алгоритм JSEG розглянуто в роботі [6].

1.8. Алгоритм сегментації кольорових текстур HGS

Даний алгоритм головним чином призначений для сегментації кольорових текстур, на яких і дає найкращі результати. Він передбачає опис зображення в об'єднаному просторі кольору та локальних характеристик просторових частот, який автори алгоритму називають областю «Фур'є – довжини хвилі» (wavelength-Fourier domain). У роботі [7] показано, що ефективними дескрипторами текстури в зазначеній області є фільтри Габора, розраховані за зображенням із гаусовою моделлю кольору, яка базується на функції переходу до кольорового простору CIE XYZ.

Узагальнений алгоритм сегментації HGS полягає в наступному:

1. Перетворити зображення відповідно до гаусової моделлі кольору.

2. Знайти відгуки фільтрів Габора на зображенні із гаусовою моделлю кольору, отриманої на попередньому кроці. В оригінальній версії алгоритму обчислюється 20 фільтрів: на п'яти різних масштабах та чотирьох орієнтаціях [7].

3. Обчислити абсолютні значення фільтрів Габора та провести їх згладжування за допомогою гаусіана.

4. Зменшити розмірність простору ознак, користуючись методом аналізу головних компонент (PCA – Principal Component Analysis). В оригінальному алгоритмі вихідний простір ознак редукується до 4-мірного.

5. Провести сегментацію, для чого:

5.1. здійснити кластеризацію редукованого простору ознак алгоритмом k-середніх із великим параметром k;

5.2. провести об'єднання областей, для яких виконується умова подібності $(\mu_i - \mu_j)^T [\Sigma_i - \Sigma_j]^{-1} (\mu_i - \mu_j) < t$, де μ_i, μ_j – середні значення, а Σ_i, Σ_j – коваріаційні матриці, обчислені по векторам ознак областей i та j відповідно, $t \in [6, 9]$;

Слід зазначити, що запропоновані дескриптори текстур можуть забезпечувати за необхідністю інваріантність до тіней та зміни відтінку. Крім того, будучи орієнтованим на роботу з кольором, алгоритм може застосовуватись і для сегментації сірих зображень. Повний опис алгоритму HGS наведено у роботі [7].

1.9. Алгоритм ROI-SEG

Даний алгоритм оснований на концепції об'єднання областей кольорового зображення, отриманих із попередньо знайдених областей інтересу (regions-of-interest – ROI). Алгоритм можна спрощено поділити на чотири етапи:

1. Перевести вхідне зображення в кольоровий простір CIE $L^*u^*v^*$.

2. Розділити отримане зображення на однакові за розміром області інтересу, які потім моделювати сумішшю гаусіанів. Для цього:

2.1. Алгоритмом сегментації по *k*-середніх здійснюється розбиття зображення на області інтересу в п'ятимірному просторі ознак (три значення *L*u*v** та координати пікселів).

2.2. Кожну з отриманих областей представити у вигляді набору розподілів Гауса, які вилучити за допомогою максимізації-очікування (EM). Ініціалізацію EM-алгориму виконати за допомогою зсуву середнього (mean-shift).

3. Кожен піксель зображення впорядковується шляхом розрахунку для нього відстані Бхаттачарії [12] (робиться з метою інтеграції в процес сегментації кольорової інформації).

4. Застосувати до отриманих на попередньому кроці відстаней Бхаттачарії детектор максимально стабільних екстремальних областей (MSER) [8], потім з'ясувати, які області слід поєднувати та отримати остаточну сегментацію.

Особливістю ROI-SEG із зрозумілих причин є його низка ефективність при роботі з сірими зображеннями. Алгоритм має широкий спектр налаштувань, які головним чином стосуються допустимих розмірів сегментів та підалгоритмів, що в ньому застосовуються.

Більш детально ознайомитись з алгоритмом ROI-SEG та його модифікацією можна у роботі [9].

2. Критерії оцінювання якості сегментації

Якість сегментації оцінюється на основі показників ефективності, запропонованих у роботі [10]. Всього їх п'ять: коректна сегментація (CS – Correct Segmentation), надмірна сегментація (OS – Over Segmentation), неповна сегментація (US – Under Segmentation), пропуск (ME – Miss Error) та шум (NE – Noise Error). Розрахунок кожного з цих критеріїв здійснюється шляхом порівняння машинної та еталонної (ground truth) сегментацій. При цьому визначається відсоток пікселів від їх загальної кількості на зображенні, що задовольняє умовам кожного з показників. Самі умови формулюються наступним чином: нехай сегментація, що підлягає оцінюванню, ділить зображення на M областей, а еталонна сегментація – на N, $O_{mn} = |R_m \cap R_n|$ – позначає кількість пікселів, що належать перетину між областями R_m, отриманої з оцінюваної сегментації та R_n, отриманої з еталонної сегментації, $P_m = |R_m|$ та $P_n = |R_n|$ – кількості пікселів у відповідних сегментах, t – поріг класифікації $t \in (0{,}5; 1)$; тоді для всіх пар сегментів оцінюваної та еталонної сегментації:

- коректна сегментація (correct segmentation) – характеризує відсоток пікселів, що були сегментовані вірно, має місце при виконанні умови:

$$\begin{cases} O_{mn} \geq t \cdot P_m; \\ O_{mn} \geq t \cdot P_n; \end{cases}$$

- надмірна сегментація (over-segmentation) – характеризує кількість точок, що потрапили в декілька сегментів, хоча насправді повинні належати одному сегменту:

$$\forall\ i \in x,\ O_{m_i n} \geq t \cdot P_{m_i},\ 2 \leq x \leq M,$$

$$\sum_{i=1}^{x} O_{m_i n} \geq t \cdot P_n;$$

- неповна сегментація (under-segmentation) – елементи зображення, які інтерпретовані як один сегмент, а в дійсності мають представляти декілька окремих сегментів:

$$\sum_{i=1}^{x} O_{mn_i} \geq t \cdot P_m,\ 2 \leq \mathrm{x} \leq \mathrm{N},$$

$$\forall\ i \in x,\ O_{mn_i} \geq t \cdot P_{n_i};$$

- пропуск (missed) – пікселі, які б мали бути окремим сегментом, хоча таким не є та які не задовольняють умові неповної сегментації:

$$R_n \notin \text{correct detection},$$
$$R_n \notin \text{over-segmentation},$$
$$R_n \notin \text{under-segmentation};$$

- шум (noise) – пікселі, що належать окремим сегментам, яких взагалі не повинно бути та які не задовольняють умові надмірної сегментації:

$$R_m \notin \text{correct detection},$$
$$R_m \notin \text{over-segmentation},$$
$$R_m \notin \text{under-segmentation};$$

У випадку, коли умови задовольняються одночасно для двох показників серед CS, OS і US, то обирається той із них (вважається ненульовим), середнє значення якого є більшим.

Для формалізації пошуку наведених вище умов в роботі [10] пропонується будувати матриці $O = (O_{mn})_{M,\,N}$, $O_m = (O_{mn} / P_m)_{M,\,N}$ та $O_n = (O_{mn} / P_m)_{M,\,N}$.

3. Результати досліджень

Аналіз якості та швидкодії сегментації здійснено для алгоритмів, що були розглянуто вище, у розд. 1. При цьому програмні реалізації кожного з них є або вільно доступними в мережі Інтернет, або частинами стандартного набору функції середовища Matlab. Результати роботи алгоритмів з огляду на обмежений обсяг публікації наводяться лише для трьох зображень: текстурного (складається виключно з текстур), супутникового (знімок поверхні Землі) та природного (фотографія природного середовища). Зображення першого типу має роздільну здатність 512×512 і утворене з декількох приро-

дних та рукотворних текстур за допомогою Інтернет ресурсу [13] – рис. 1., *а* наведене зліва. Зображення другого типу має розмір 481×321 пікселів і взято із бази [14] – рис. 1., *а* – по центру. Зображення третього типу має роздільну здатність 256×256 та є фрагментом супутникової карти (широта: 50°26′51.99″ та довгота: 30°33′58.1″), вільно доступної в мережі Інтернет [15] – рис. 1., *а*, справа. Всі типи зображень доповнені результатами еталонної сегментації: для першого типу еталон генерований автоматично, а для другого і третього – створений вручну (рис. 1, *б*).

Запуск програм сегментації здійснювався на персональному комп'ютері з процесором Intel Pentium Core Duo 1,83 ГГц, кеш першого рівня L1 32 Кб, другого рівня – L2 2 Мб, ємність ОЗУ 1024 Мб. Використовувалось наступне програмне забезпечення: операційна система Windows XP, Matlab 7.9.0.529 (R2009b). Під час тестування використано наступні налаштування алгоритмів:

- Сегментація по *k*-середніх здійснювалась з урахуванням кольору та координат пікселів, текстурне та супутникове зображення розбивалось на $k = 5$ сегментів, а природне – на $k = 3$.
- Сегментація на основі максимізації-очікування виконана без врахування кольору та просторового розташування пікселів. Розбиття здійснено на $k = 5$, $k = 3$ та $k = 5$ сегментів для текстурного природного та супутникового зображень відповідно.
- Сегментація зсувом середнього проведена із параметрами просторової пропускної здатності $h_s = 7$, та пропускної здатності в області ознак $h_r = 12{,}5$. Мінімальний розмір сегменту встановлено у 3 % (тут і надалі розуміється відсоток від загальної кількості пікселів на зображенні) для текстурного та 7 % для природного і супутникового зображень.
- Параметри алгоритму сегментації на основі нормального перетину графів залишені по замовчуванню.
- Результатом сегментації алгоритму SWA вважається розбиття, отримане на другому з кінця масштабі. Під час сегментації текстурного та природного зображень враховувались текстурні ознаки. Всі інші параметри залишені по замовчуванню.
- Результат сегментації на основі алгоритму SRM отримано на $Q = 8$ ступені «грубості», починаючи зі значення $Q = 128$. Мінімальний розмір сегменту – 1 %.
- Для алгоритму JSEG використовувались поріг квантування $q = 255$, кількість масштабів $l = 10$ та поріг об'єднання $m = 0{,}7$.
- Алгоритм HGS запускався із параметром, що забезпечує низькою інваріантністю до тіней та відтінку, а також порогом об'єднання $t = 6{,}0$. Мінімальний розмір сегменту було встановлений у 1 %.
- Параметри алгоритму ROI-SEG залишені по замовчуванню.

Результати роботи алгоритмів наведено на рис. 1. Якість сегментації кожного з них оцінена відповідно до показників, наведених у розділі 2 при значенні порогу класифікації $t = 0{,}9$. Оцінювання проведено за допомогою ресурсу «The Prague Texture Segmentation Datagenerator and Benchmark» [13] та наведено у табл. 1. Стрілки біля назв показників вказують: «↑» – чим вище параметр, тим сегментація виконана якісніше, «↓» – чим нижче параметр, тим якість сегментації гірша.

Сегментація за допомогою нормалізованого перетину виконувалась на зображеннях, зменшених вдвічі, що обумовлено недостатністю пам'яті.

З метою мінімізації впливу операційного середовища на точність оцінювання швидкодії програмних реалізацій алгоритмів їх запуск здійснювався тричі для кожного зображення, при цьому часом обробки вважається середнє значення тривалості обчислень. Результати оцінювання швидкодії проілюстровані гістограмою, наведеною на рис. 2.

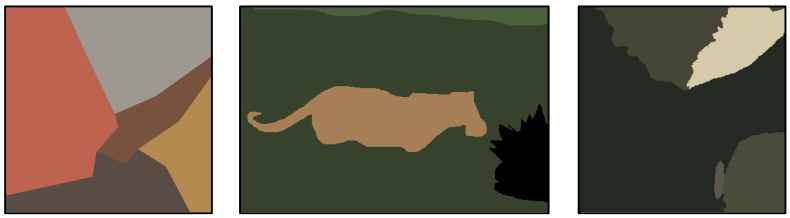
Таблиця 1. Оцінювання якості сегментації на основі показників коректності (CS), надмірності (OS), неповноти (US), пропуску (ME) та шуму (NE) (див. розд. 2)

Алгоритм		Показники				
		CS↑	OS↓	US↓	ME↓	NE↓
Природне	*k*-середніх	66,70	0,00	0,00	25,81	31,62
	EM	0,00	0,00	0,00	100,00	100,00
	Зсув середнього	44,78	0,00	53,90	0,00	0,00
	Норм. перетин	73,50	53,26	0,00	9,79	1,92
	SWA	49,06	16,04	34,82	0,00	0,07
	SRM	74,08	0,00	23,53	0,00	0,00
	JSEG	90,02	6,47	0,00	9,77	6,49
	HGS	78,14	0,00	0,00	20,97	16,63
	ROI-SEG	48,52	0,00	40,92	9,77	2,59

Продовження табл. 1.

Алгоритм		Показники				
		CS↑	OS↓	US↓	ME↓	NE↓
Текстурне	*k*-середніх	0,00	74,80	0,00	24,77	10,11
	EM	0,00	74,20	0,00	24,77	10,95
	Зсув середнього	0,00	75,15	0,00	24,77	8,33
	Норм. перетин	0,00	59,64	0,00	24,71	31,13
	SWA	0,00	57,81	0,00	24,77	35,69
	SRM	0,00	75,17	0,00	24,77	8,92
	JSEG	16,23	84,63	0,00	4,13	3,22
	HGS	0,00	57,81	0,00	24,77	35,69
	ROI-SEG	11,46	73,05	0,00	10,41	3,04
Супутникове	*k*-середніх	31,65	62,96	0,00	0,68	0,00
	EM	71,13	0,00	0,00	26,16	24,42
	Зсув середнього	73,37	13,84	0,00	10,99	5,51
	Норм. перетин	10,06	49,13	0,00	26,16	39,74
	SWA	97,33	14,35	0,00	0,68	0,00
	SRM	96,88	0,00	0,00	0,68	0,00
	JSEG	97,94	0,00	10,66	0,00	0,00
	HGS	55,19	0,00	0,00	36,36	44,38
	ROI-SEG	75,49	0,00	10,96	15,17	21,24

а



б

в

г

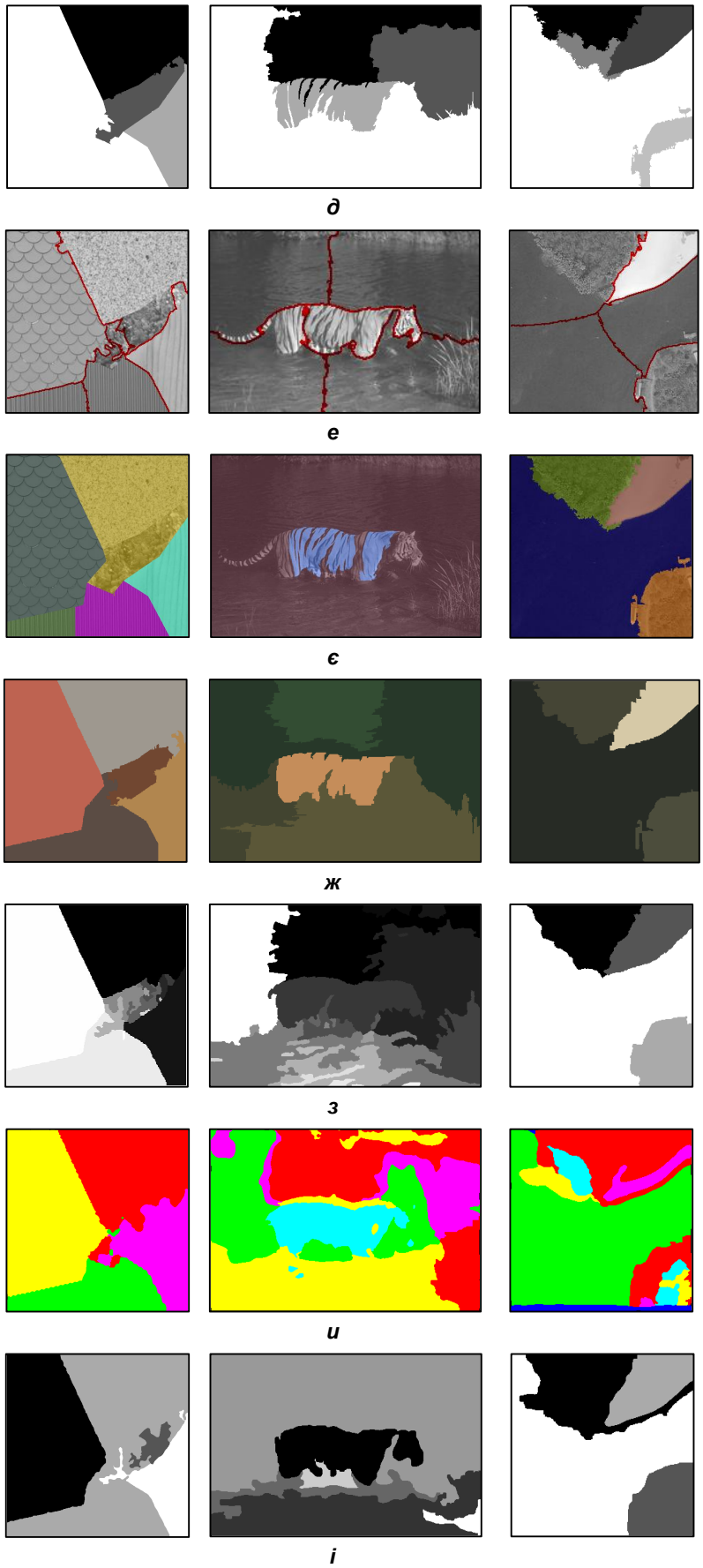


Рис. 1. Результати сегментації: *а* – вихідні зображення; *б* – еталонний результат; *в* – по *k*-середніх; *г* – на основі максимізації-очікування (EM); *д* – зсувом середнього; *е* – нормалізованим перетином графів; *є* – зваженим об'єднанням областей (SWA); *ж* – статистичним об'єднанням областей (SRM); *з* – JSEG; *и* –HGS; *і* – ROI-SEG

Відзначимо, що виконане оцінювання швидкодії носить якісний характер і наводиться виключно для відносного порівняння алгоритмів між собою. При цьому можна вважати, що умови апробації всіх алгоритмів є однаковими, незважаючи на реалізацію деяких із них в середовищі Matlab, а інших – на мові C/C++, оскільки алгоритми, реалізовані в Matlab, мають найбільш обчислювально-ємні частини, написані на мові C/C++.

Рис. 2. Час обробки текстурного та природного зображень кожним із алгоритмів

Висновки

Із проведеного вище порівняльного аналізу сучасних алгоритмів сегментації випливає, що застосування більшості з них дає задовільний результат на природному зображенні, що обумовлено його достатньою складністю. Встановлено, що найкращим одночасно по критеріям якості сегментації і швидкодії є алгоритм ROI-SEG. Найкращу ж сегментацію забезпечує алгоритм JSEG, однак його швидкодія є посередньою.

Інші алгоритми не забезпечують коректність сегментації (CS) вищу нуля на природному зображенні, однак показують результат не набагато гірший, аніж той, що отриманий при алгоритмами ROI-SEG та JSEG.

Визначено, що найгіршу якість сегментації демонструє алгоритм на основі максимізації очікування (окрім супутникового зображення). Причиною цього є недостатність інформації, яка ним використовується під час обробки (колір та просторове розташування пікселів не було враховано). Досить непогані результати одночасно по якості та швидкодії продемонстровано алгоритмами на основі зсуву середнього та статистичного об'єднання областей (SRM).

Встановлено, що найбільш повільним є алгоритм HGS, хоча на його місці мав бути алгоритм на основі нормалізованого перетину, оскільки йому через брак пам'яті довелося працювати із вдвічі меншими зображеннями. В той же час даний алгоритм забезпечує досить непогану якість сегментації, хоча і не враховує кольорову інформацію.

Сегментацію супутникового зображенням всіма алгоритмами виконано коректно, хоча алгоритми *k*-середніх та нормалізований перетин не забезпечили параметр CS вищий 50 %, що головним чином зумовлено надмірним розбиттям.

Література

1. *Форсайт Д.*, Понс. Ж. Компьютерное зрение. Современный подход: Пер. с англ. – М.: Издательский дом «Вильямс». – 2004. – 928 с.
2. *Comaniciu D.*, Meer P. Mean shift: A robust approach toward feature space analysis // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2002. – Vol. 24. – P. 603–619.
3. *Shi J.*, Malik J. Normalized Cuts and Image Segmentation // IEEE Transactions on Pattern

Analysis and Machine Intelligence. – 2000. – Vol. 22, № 8. – P. 888–905.
4. *Galun M.*, Sharon E., Basri R., Brandt A. Texture Segmentation by Multiscale Aggregation of Filter Responses and Shape Elements // Proceedings of the Ninth IEEE International Conference on Computer Vision. – 2003. – Vol. 1. – P. 716–723.
5. *Nock R.*, Nielsen F. Statistical Region Merging // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2004. – Vol., 26 № 11. – P. 1452–1458.
6. *Deng Y.*, Manjunath B.S. Unsupervised Segmentation of Color-Texture Regions in Images and Video // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 2001. – Vol. 23, № 8. – P. 800–810.
7. *Hoang M.A.*, Geusebroek J.-M., Smeulders A.W.M. Color Texture Measurement and Segmentation // Signal Processing. – 2005. – Vol. 85, № 2. – P. 265–275.
8. *Matas J.*, Chum O., Urban M., Pajdla T. Robust Wide Baseline Stereo from Maximally Stable Extremal Regions // Proc. of British Machine Vision Conf. – 2002. – P. 384–393.
9. *Donoser M.*, Bischof H. ROI-SEG: Unsupervised Color Segmentation by Combining Differently Focused Sub Results // 2007 IEEE Conference on Computer Vision and Pattern Recognition. – 2007. – P. 1–8.
10. *Hoover A.*, Jean-Baptiste G., Jiang X., Flynn P.J., Bunke H., Goldgof D., Bowyer K., Eggert D, Fitzgibbon A., Fisher R.. An Experimental Comparison of Range Image Segmentation Algorithms // IEEE Transactions on Pattern Analysis and Machine Intelligence. – 1996. – Vol. 18, № 7. – P. 673–689.
11. *Lloyd S.* Least squares quantization in PCM // IEEE Transactions on Information Theory. – 1982. – Vol. 28, № 2. – P. 129–137.
12. *Bhattacharyya A.* On a measure of divergence between two statistical populations defined by their probability distributions // Bulletin of the Calcutta Mathematical Society. – 1943. – № 35. – P. 99–109.
13. *The Prague* Texture Segmentation Datagenerator and Benchmark. http://mosaic.utia.cas.cz/index.php – 19.05.2011.
14. *The Berkeley* Segmentation Dataset and Benchmark. http://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/ – 19.05.2011.
15. Яндекс. Карти. http://maps.yandex.ru/ – 19.05.2011.

Национальный технический университет Украины «Киевский политехнический институт»

Поступила в редакцию 15 сентября 2011 г.