DISAIN DAN IMPLEMENTASI PERANGKAT LUNAK KLASIFIKASI CITRA INDERAJA MULTISPEKTRAL SECARA *UNSUPERVISED*

Agus Zainal Arifin* dan Aniati Murni**
*Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember, Surabaya
Gedung Teknik Informatika, Kampus ITS, Keputih, Sukolilo, Surabaya,
email : agusza@its-sby.edu
**Fakultas Ilmu Komputer, Universitas Indonesia, Depok
Gedung Fasilkom UI Depok,
email : aniati@caplin.cs.ui.ac.id

ABSTRAK

Klasifikasi citra penginderaan jauh (inderaja) bertujuan untuk menghasilkan peta tematik, dimana tiap warna mewakili sebuah objek, misalkan hutan, laut, sungai, sawah, dan lain-lain. Makalah ini mempresentasikan disain dan implementasi perangkat lunak untuk mengklasifikasi citra inderaja multispektral. Metode berbasis *unsupervised* yang diusulkan ini adalah integrasi dari metode *feature extraction*, *hierarchical* (hirarki) *clustering*, dan *partitional* (partisi) *clustering*. *Feature extraction* dimaksudkan untuk mendapatkan komponen utama citra multispektral tersebut, sekaligus mengeliminir komponen yang redundan, sehingga akan mengurangi kompleksitas komputasi. Histogram komponen utama ini dianalisa untuk melihat lokasi terkonsentasinya pixel dalam *feature space*, sehingga proses split dapat menghasilkan cluster dengan cepat. Beberapa cluster yang sangat mirip akan digabungkan oleh proses merge. Pada tahap akhir, proses partisi akan mendeteksi prototype tiap cluster dengan *Fuzzy C-Mean* (FCM). Uji coba perangkat lunak ini dilakukan pada citra Landsat TM dan GOES-8. Hasilnya diukur berdasarkan homogenitas tiap cluster, heterogenitas antar cluster, waktu eksekusi, dan nilai tabel contingency. Tabel ini akan membuktikan keberhasilan klasifikasi terhadap 800 sampel dari Jawa Timur yang sebelumnya telah dikenali. Untuk bahan pembandingan, sampel juga diuji coba dengan algoritma ISMC (Improved Split and Merge Classification), yang berdasarkan penelitian sebelumnya telah terbukti lebih baik dari pada ISODATA. Secara umum, uji coba menunjukkan keunggulannya dibandingkan ISMC.

Kata kunci : Klasifikasi *unsupervised*, citra penginderaan jauh, *split*, *merge*, *fuzzy c mean*.

1. PENDAHULUAN

Analisa *cluster* merupakan suatu bentuk pengenalan pola yang berkaitan dengan pembelajaran secara *unsupervised*, dimana jumlah pola kelas tidak diketahui [1][5]. Proses *clustering* berusaha membagi *data set* dengan mengelompokkan seluruh *pixel* pada *feature space* (ruang ciri) ke dalam sejumlah *cluster* secara alami. Hampir semua algoritma *clustering* yang populer selalu mengharuskan adanya inisialisasi jumlah *cluster* awal [8][3]. Padahal jumlah ini sangatlah sulit untuk diketahui, sebab dibutuhkan orang yang benar-benar menguasai konfigurasi objeknya.

Dengan adanya fenomena tersebut, maka para peneliti dalam bidang pengenalan pola (*pattern recognition*) berusaha menghasilkan algoritma yang mampu mendeteksi jumlah *cluster* ini secara otomatis [3][8][5]. J. J. Simpson [5] telah mengembangkan algoritma ISMC (Improved Split and Merge Classification). Algoritma *clustering* ini menggabungkan proses *split* dan *merge* yang diiterasi hingga konvergen. Prosedur *split* yang dikembangkannya berusaha membagi sebuah *cluster* menjadi 2 *sub cluster*. Pembagiannya berdasarkan pasangan *pixel* yang jaraknya terjauh (2 *pixel* kutub). Setelah masing-masing *pixel* telah memilih salah satu *pixel* kutub ini, maka terbentuklah 2 *sub cluster* baru. Tiap *cluster* yang sudah terbentuk dipecah lagi dengan proses yang sama hingga ukuran *cluster* tersebut melampaui batas *Threshold* untuk *split*. Selanjutnya dilakukan proses penggabungan (*merging*) antar *cluster* yang berdekatan. Proses selanjutnya adalah *partitional*, yakni *assignment pixel* terhadap tiap pusat *cluster* untuk menentukan pusat *cluster* baru. Proses *split*, *merge*, dan *partitional* ini diulang hingga konvergen. Nampak bahwa mekanisme *split* pada algoritma tersebut tidak mempertimbangkan lokasi tempat berkumpulnya mayoritas *pixel*. Namun hanya mempertimbangkan jarak terjauh antar *pixel*. Hal ini bisa mengakibatkan pemotongan *cluster* yang berada di antara kedua *pixel* tersebut. Penyebabnya bisa berupa perbedaan distribusi atau juga ukuran cluster yang terlalu besar. Dengan demikian dibutuhkan metode *split* yang memperhatikan distribusi *pixel* dalam *feature space*. Distribusi ini dapat digambarkan melalui *histogram*, dimana tiap kurva yang terbentuk dapat diasosiasikan sebagai sebuah cluster.

Masalah dalam pembagian secara langsung ini juga dilakukan oleh Mehmet Celenk [6]. Metodenya menggabungkan *split* dan *merge*, dengan membagi seluruh citra menjadi *non-overlapping window* 4x4. Tiap *window* di-*split* menjadi 2 *cluster* dengan *K-means clustering*, dan ini bisa dilakukan secara paralel. Seluruh *cluster* hasil *split*, digabungkan dengan metode yang sama. Oleh karena tiap 1 window memiliki 2 *cluster*,

maka jumlah *cluster* menjadi sedemikian banyak, sehingga proses *merge* akan memakan waktu lama. Dengan demikian, dibutuhkan metode yang secepat mungkin mampu membentuk sejumlah *cluster*.

Pada kenyataannya, proses pembentukan cluster dengan pencarian kurva pada *feature space* citra multispektral sangatlah sulit. Sebab dibutuhkan teknik *scanning* kurva yang sangat rumit. Cara yang termudah adalah mentransformasikannya menjadi satu dimensi, namun mampu mewakili seluruh spektrum. Proses ini biasa disebut dengan PCT (*Principal Component Transformation*).

Bila proses *split* di atas menghasilkan *cluster* yang cukup banyak, maka dibutuhkan metode pengabungan yang lebih ketat. Penggabungan ini tidak hanya sekedar mencari *cluster* yang terdekat kemudian digabungkan, namun juga perlu dilihat apakah efek penggabungan ini menyebabkan *chain effect*. Efek ini sangat mungkin terjadi, bila pada citra tersebut terdapat *noise*. *Noise* yang terletak diantara 2 *cluster* yang berjauhan dapat bertindak sebagai perantara. Akibatnya, kedua *cluster* yang seharusnya tidak layak digabung ini, akhirnya akan tergabung, bila salah satu *cluster* tersebut menarik *cluster noise* untuk menjadi anggotanya.

Masalah lain yang harus dihadapi oleh algoritma *clustering* adalah adanya *uncertainty* baik yang berupa *noise* maupun *outlier*. Salah satu metode yang dapat mengatasi kedua problema ini adalah *Fuzzy C-Means* (FCM) [4]. Algoritma ini selanjutnya dikembangkan oleh para peneliti [7][3][8] untuk meningkatkan kinerjanya. Posisi *noise* dan *outlier*, pada umumnya berada di antara sejumlah *cluster*, dimana jarak terhadap tiap pusat *cluster* tersebut hampir sama. Dengan metode ini, *membership* keduanya terhadap semua pusat *cluster* tidak akan terlalu besar. Sehingga tidak akan terlalu menentukan lokasi pusat *cluster* yang diikutinya pada tiap iterasi.

Penelitian ini bertujuan untuk membangun sebuah perangkat lunak yang mampu melakukan *unsupervised classification* (klasifikasi tak terawasi) terhadap citra multispektral dengan lebih akurat, dalam artian kondisi *cluster* yang lebih kompak dan perbedaan antar *cluster* yang lebih meningkat, serta ketepatan pengenalan kelas yang lebih tinggi.

2. CLUSTERING ADAPTIF

Bab ini membahas tentang algoritma *clustering* yang diusulkan yakni algoritma *Clustering* Adaptif (CA). Penamaan adaptif disebabkan pembagian pada tahap *split* dilakukan sesuai dengan distribusi *gray level* citra dalam *histogram*. Dengan demikian, jumlah hasil pembagian itu diadaptasikan sesuai dengan karakteristik citra.

2.1. Ukuran Kuantitatif

Representasi *cluster* dalam *feature space* dapat diukur melalui 2 *scatter matrix*. Tingkat kedekatan dalam satu *cluster* diukur dengan *within-cluster scatter matrix* S_W, sedangkan jarak antar *cluster* diukur dengan *between-cluster scatter matrix* S_B [12][5]. Idealnya, keanggotaan setiap *cluster* haruslah sehomogen mungkin, sehingga S_W haruslah sekecil mungkin. Di samping itu perbedaan antar *cluster* seharusnya sebesar mungkin, sehingga S_B haruslah setinggi mungkin. Misalkan C_k adalah *cluster* ke-k dari K buah *cluster* yang dihasilkan. *Cluster* C_k terdiri dari n_k buah *vector*, yakni $\{y_1^k, y_2^k, \ldots\}$. Bila pusat *cluster* C_k adalah m_k, dengan rata-rata m, maka untuk N vektor, S_B dan S_W didefinisikan sebagai berikut:

$$m_k = \frac{1}{n_k}\sum_{i=1}^{n_k} y_i^k \qquad \text{dimana} \qquad m = \frac{1}{N}\sum_{k=1}^{K} n_k m_k$$

$$S_k = \sum_{i=1}^{n_k} (y_i^k - m_k)(y_i^k - m_k)^T \quad \text{dimana } S_W = \sum_{k=1}^{K} S_k$$

$$S_B = \sum_{k=1}^{K} n_k (m_k - m)(m_k - m)^T$$

Bila terdapat 2 algoritma yang melakukan *clustering* terhadap *data set* yang sama, maka hasil *clustering* keduanya dapat diukur berdasarkan kedua variabel ini. Pembandingan dapat dilakukan dengan mengkonversikan matriks tersebut menjadi bilangan skalar. Metode yang dapat digunakan antara lain *trace* dan *determinant* [12]. Berdasarkan pertimbangan kemudahan penghitungan dan kepopulerannya [13] [2] [1] [5], maka dalam penelitian ini, metode *trace* yang digunakan dengan rumus :

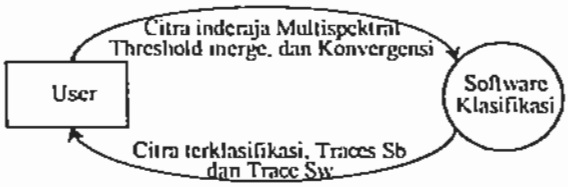
$$tr(S_w) = \sum_{k=1}^{K} tr(S_k) = \sum_{k=1}^{K} \sum_{i=1}^{n_k} \| y_i^k - m \|^2$$

$$tr(S_B) = \sum_{k=1}^{K} n_k \| m_k - m \|^2$$

2.2. Disain Algoritma

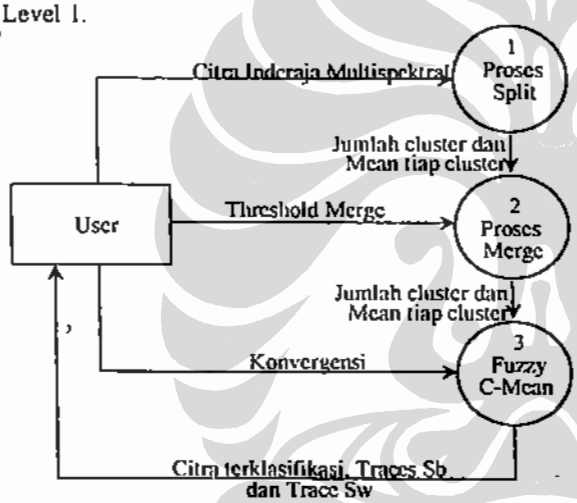
Disain dari algoritma ini direpresentasikan melalui *Data Flow Diagram* atau Diagram Alir Data (DAD). DAD merupakan metode yang sangat sesuai untuk menggambarkan disain perangkat lunak, sebab aliran data sejak awal diinputkan hingga akhir proses dapat diketahui dengan jelas.

Gambar 1 menampilkan DAD level 0 atau *context diagram*, yakni diagram yang menampilkan input, proses, dan output secara global. Inputnya adalah citra penginderaan jauh multispektral, *threshold merge*, dan *threshold* konvergensi. Sedangkan outputnya adalah citra hasil klasifikasi yang disertai dengan analisa kuantitatifnya.



Gambar 1. Context Diagram

Threshold merge adalah batas jarak minimal antar *cluster*. Dengan demikian, bila terdapat 2 *cluster* yang jaraknya kurang dari *threshold*, maka keduanya diasumsikan mewakili satu kelas yang sama, oleh karena itu keduanya harus digabung. Sedangkan konvergensi adl dimaksudkan untuk memberi batasan iterasi pada *fuzzy c mean*. Pada tiap iterasi, nilai *trace* dari S_n diukur, bila tingkat perubahan yang terjadi pada iterasi berikutnya kurang dari *threshold* ini, maka diasumsikan sudah tidak terdapat perubahan. Algoritma ini terdiri dari 3 langkah utama, yakni proses *split*, *merge*, dan *fuzzy c mean* sebagaimana nampak pada Gambar 4.2, yakni Diagram 0 Level 1.



Gambar 2. Diagram 0 Level 1

2.2.1. Proses Split

Proses *split* berisi pembagian citra multispektral yang diinputkan, menjadi sejumlah *cluster*. Tiap cluster direpresentasikan sebagai vektor *mean* yang dihitung berdasarkan *histogram* 1 dimensi. Diagram 1 Level 2 untuk proses *Split* dapat dilihat pada Gambar 3.

Proses *split* dimulai dengan langkah *Feature Extraction* untuk mengeliminir informasi yang redundan. Metode yang digunakan adalah *Principle Component Transformation* (PCT). Transformasi ini dilakukan dengan mengalikan vektor pixel dengan eigen vector dari komponen yang memiliki eigen value terbesar. Bila Σ_x adalah matriks kovarian dan I adalah matriks identitas, maka eigen value λ dapat dihitung berdasarkan rumus :

$$| \Sigma_x - \lambda I | = 0$$

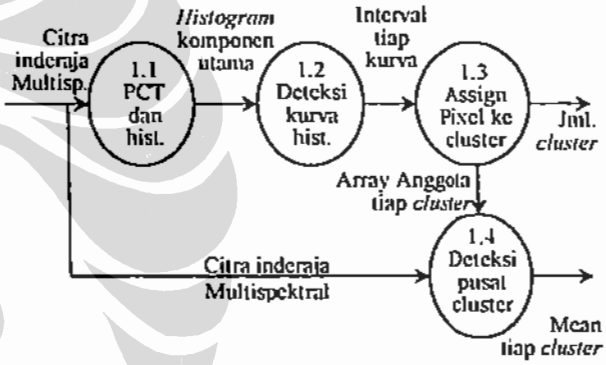
Bila terdapat K dimensi, maka akan dihasilkan K eigen value. Eigen vector ke-k E_k dapat dicari dari eigen value ke-k λ_k dengan rumus :

$$| \Sigma_x - \lambda_k I | E_k = 0$$

Dalam hal ini eigen vector yang digunakan adalah hasil dari eigen value yang terbesar diantara K eigen value.

Hasil transformasi yang merupakan komponen utama citra ini diakumulasi menjadi *histogram*. Tiap kurva dalam *histogram* tersebut dapat diasumsikan sebagai sebuah *cluster* yang terpisah [9] [11] [10].

Bila seluruh *pixel* telah diasosiasikan terhadap suatu *cluster*, maka terbentuklah daftar *cluster* yang direpresentasikan melalui struktur data *linked list*. Tiap *list* menyatakan sebuah *cluster* yang meliputi jumlah anggota beserta daftar nomor *pixel* tiap anggota tersebut. Langkah pembentukan suatu *list* sekaligus diikuti dengan penghitungan vektor *mean cluster* dalam *list* tersebut. Vektor *mean* sebuah *cluster* adalah nilai rata-rata dari seluruh vektor *pixel* anggota *cluster* tersebut.



Gambar 3. Diagram 1 Level 2 Proses Split

2.2.2. Proses Merge

Pada umumnya, diantara sejumlah *cluster* yang dihasilkan, terdapat beberapa *cluster* yang mirip satu sama lain. *Cluster* demikian sangat layak digabungkan, sebab masing-masing mencerminkan satu kelas yang mirip atau bahkan sama. Terdapat beberapa metode yang dapat diaplikasikan, diantaranya adalah *nearest-neighbor*, *furthest-neighbor*, dan *compromise* [12] [13].

Berdasarkan eksperimen yang tidak kami sertakan di makalah ini dan berdasarkan pertimbangan kelebihan dan kekurangan yang dimiliki tiap metode, maka penelitian ini menggunakan metode *furthest-neighbor* (*complete link*). Metode ini mampu mengatasi resiko *chain effect*, yang dikhawatirkan terjadi bila terdapat *noise*. Di dalam *feature space* keberadaan *noise* dapat menyebabkan tergabungnya sejumlah *cluster* yang tidak mirip. Bila objek *noise* ini mirip dengan masing-masing *cluster* tersebut, maka akan

menjadi jembatan penghubung antar *cluster* tersebut, sehingga terjadilah *chain effect*.

Terdapat 3 proses utama yang dilakukan, yakni pengukuran jarak antar *mean*, penggabungan dengan *complete link*, dan penghitungan vektor *mean* dari *cluster-cluster* yang baru terbentuk.

Pengukuran jarak antar mean menghasilkan tabel yang mampu menyimpan jarak *euclidean* antar vektor *mean* tersebut. Bila jumlah *cluster* yang diinputkan adalah C_1, maka *memory* yang dibutuhkan adalah $O(C_1^2)$. Tabel digunakan oleh *complete link* yang algoritmanya adalah :

1. Cari pasangan *cluster* dengan jarak *euclidean* terkecil.
2. Bila kurang dari *threshold merge*, maka selesai.
3. Gabungkan kedua *cluster* tersebut.
4. Hitung ulang jarak *euclidean* antara semua *cluster* dengan *cluster* baru dengan mengambil jarak terjauh dengan anggota *cluster* baru. Kembali ke langkah 1.

Sedangkan penghitungan vektor *mean* adalah dengan menghitung rata-rata dari hasil kali *mean sub cluster* dengan jumlah anggotanya masing-masing.

2.2.3. Proses *Fuzzy C Mean*

Proses *Fuzzy C Mean* berusaha mencari prototype tiap *cluster*, sehingga dapat digunakan untuk memilih vektor *pixel* yang paling mirip untuk dijadikan anggotanya. Pada setiap iterasi dilakukan penghitungan U_{ik} (membership *pixel* k terhadap *cluster* i) dan v_i (vektor mean dari *cluster* i), dengan rumus sebagai berikut :

$$U_{ik} = \left(\sum_{j=1}^{c} \left(\frac{D_{ik}}{D_{jk}} \right)^{\frac{2}{(m-1)}} \right)^{-1} \forall i,k \quad \text{dan} \quad v_i = \frac{\sum_{k=1}^{n} U_{ik}^{m} X_k}{\sum_{k=1}^{n} U_{ik}^{m}} \forall i$$

Proses di atas diiterasi hingga tercapai *stopping criteria*, yakni berdasarkan perubahan pada Trace S_B yang tidak melebihi nilai *konvergen*.

2.3. Implementasi Algoritma

Algoritma di atas diimplementasikan pada PC Pentium MMX 233 MHz dengan memori 64 MB, VGA *memory* 4 MB, dan Sistem Operasi Windows 98. Perangkat lunak Pengembangnya adalah Borland Turbo C++ versi 4.5 berbasis *OOP (Object Oriented Programming)*. Program ini harus mendukung konsep *Multiple Document Interface* melalui penampilan *child window* untuk tiap band.

Dengan pertimbangan kemudahan pembacaannya, maka implementasi *function* yang ditulis di sini dimodifikasi menjadi pseudocode yang mirip bahasa C. Sebagaimana penjelasan pada sub bab 2.2.1. di atas, fungsi *Split* diimplementasikan sebagai berikut :

```
void Split(ClusterClass *liste)
{
    BYTE HUGE **pattern = new BYTE HUGE *[Band];
    Pattern=AmbilAddressAwalPatternUntukTiapBand();
    Histogram=PCT(pattern, Band);
    JumlahKurva=AnalisaCluster(Histogram);
    if (JumlahKurva < 2) return;
    ClusterClass **SubList =
                new ClusterClass* [JumlahKurva];
    for (cluster := 0 ; cluster < JumlahKurva ; cluster + +)
        SubList[cluster]=CreateBufferAnggota(cluster);
    CreateLinkedList(SubList, JumlahKurva);
    HapusLinkedListLama(liste);
}
```

Proses merge didahului dengan pembuatan tabel jarak antar cluster. Ukuran tabel ini adalah ½ n (n-1) untuk n cluster. Implementasi proses *merge* yang dalam hal ini menggunakan metode *complete link* adalah sebagai berikut :

```
void CompleteLink()
{   Tabel=TabelJarakAntarCluster(JumlahCluster);
    do {
        JarakMinimum=ClusterTerdekat(Tabel, C1, C2);
        if (JarakMinimum < ThresholdMerge)
                Gabungkan(C1, C2);
        else BisaMerge=0;
    } while (BisaMerge);
}
```

Adapun implementasi proses FCM adalah :

```
void FuzzyCMean()
{
    do {
        HitungMembership();
        ModifikasiVektorMean();
        BelumKonvergen= PeriksaStoppingCriteria();
    } while (BelumKonvergen);
}
```

2.4. Analisa Kompleksitas

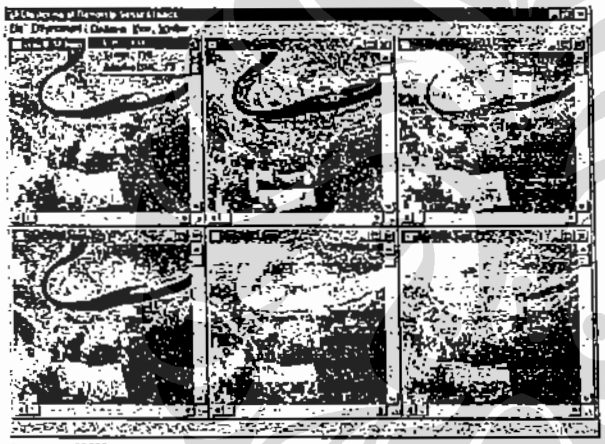
Kompleksitas komputasi ini dihitung untuk 2 faktor, yakni pembandingan (Cmp) dan perkalian atau pembagian (Ar). Pada bagian inisialisasi, pencarian vektor max_i dan min_i, untuk *pixel* sejumlah n dan *band* sejumlah b, membutuhkan Cmp = n.b serta Ar = 2 + b (tanpa *threshold split*). Tahap *split* diawali Pembuatan matriks kovariansi dengan $Ar = (b + n.b^2 + b^2)$, kemudian deteksi nilai *eigen* terbesar dan vektor *eigen*, serta *transpose*-nya membutuhkan Ar = O (b^2), selanjutnya pada transformasi Ar = n.b. Sedangkan pada pencarian *peak histogram*, Cmp = O ($256.b^{½}$). Bila terdapat v kurva, maka penentuan keanggotaan *pixel* membutuhkan Cmp = v.n, diakhiri dengan deteksi v vektor *mean* dengan Ar = b.v.

Proses *merge* berlangsung sama dengan *merge* pada algoritma ISMC. Pada pembuatan matriks jarak antar *mean*, Ar = ½.v.(v-1).b. Pencarian jarak terdekat membutuhkan Cmp = O (v^3). *Update* jarak juga membutuhkan Cmp = O (v^2). Dan diakhiri penghitungan k vektor *mean* baru, dengan Ar = (v.b + k.b).

Pada proses *Fuzzy C Mean*, penghitungan *membership* membutuhkan Ar = n.k.(b + 3). Selanjutnya untuk penghitungan vektor *mean* yang baru berdasarkan *membership* tersebut dibutuhkan Ar = k.(n.2.b + b). Penghitungan S_B, harus didahului dengan penentuan keanggotaan tiap vektor, dengan Cmp = k.n. Dilanjutkan penghitungan vektor *TotalMean* seluruh *cluster*, dengan Ar = k.b + b. Berdasarkan vektor *mean* tiap *cluster* dan *TotalMean* tersebut, maka Matriks S_B di sini membutuhkan Ar = $2.b^2.k$, ditambah pemeriksaan konvergensi dengan Ar = 1 dan Cmp = 1. Seluruh proses *Fuzzy C Mean* di atas diulang hingga konvergen.

3. UJI COBA DAN EVALUASI

Sebagaimana telah dibahas, bahwa pembandingan hasil klasifikasi ini nantinya akan dilakukan terhadap algoritma ISMC. Oleh karena itu, implementasi juga dilakukan terhadap algoritma ISMC. Adapun bentuk antar muka perangkat lunak untuk eksekusi kedua algoritma ini



dapat dilihat pada Gambar 4.

Gambar 4. Antar muka grafis perangkat lunak

Urutan uji coba ini adalah sebagai berikut :

1. Uji coba terhadap algoritma CA dan ISMC dengan input threshold merge dan konvergensi sama. Hasil yang dihitung adalah jumlah cluster, waktu eksekusi, $tr(S_B)$ dan $tr(S_W)$.
2. Uji coba terhadap algoritma CA dengan input jumlah cluster yang dihasilkan oleh ISMC di atas. Hasil yang dihitung adalah waktu eksekusi, $tr(S_B)$ dan $tr(S_W)$.
3. Uji coba secara visual terhadap algoritma CA dan ISMC dengan input jumlah cluster yang sama untuk sampel daerah Jawa Timur. Sampel lokasi yang diambil sebanyak 800 pixel untuk lokasi perairan dan perkotaan.

3.1. Data Sampel

Karakteristik data sampel dapat dilihat pada Tabel 1. Sampel dari Landsat TM sebenarnya adalah 7 band. oleh karena perbedaan resolusi, maka band 6 tidak digunakan. sehingga jumlahnya menjadi 6 band.

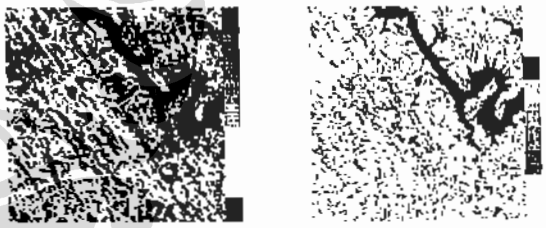
Tabel 1. Sampel citra uji coba

Kode	Lokasi	Satelit	Ukuran
A	Jawa Tengah	Landsat TM	256^2
B	Rinu	Landsat TM	300^2
C	California	GOES-8	256^2
D	Galapagos	GOES-8	256^2
E	Panama	GOES-8	250^2
F	Texas	GOES-8	250^2
G	Nicaragua	GOES-8	250^2

Sedangkan pada GOES-8, dari 5 band tersedia, hanya band 2, 4, dan 5 yang digunakan, agar lebih relevan dengan uji coba ISMC [5] yang menggunakan AVHRR band 2, 3, dan 4.

3.2. Eksekusi dengan Kesamaan Parameter

Data sampel tersebut di atas dieksekusi dengan algoritma ISMC dengan input 3 parameter, yakni SPLIT = 750, MERGE = 50, dan Tol = 0,05. Sedangkan CA hanya membutuhkan 2 parameter di atas MERGE = 50. dan Konvergensi = 0,05. Gambar 5 menampilkan hasil klasifikasi ISMC (a) dan CA (b) untuk data sampel A. Sedangkan untuk sampel B, output ISMC dan CA ditunjukkan oleh Gambar 6. (a) dan (b).



(a) (b)

Gambar 5. Output ISMC dan CA untuk sampel A

(a) (b)

Gambar 6. Output ISMC dan CA untuk sampel B

Parameter jumlah cluster dan waktu eksekusi ditampilkan pada tabel 2 beserta peningkatan kecepatan pada CA dibandingkan ISMC. Adapun parameter $tr(S_B)$ dan $tr(S_W)$ ditampilkan pada Tabel 3 dan 3.4 dengan disertai peningkatan yang terjadi. Perlu diingat bahwa kinerja yang lebih baik, ditentukan oleh lebih tingginya $Tr(S_B)$ dan lebih rendahnya $Tr(S_W)$ [5].

Tabel 2. Jumlah cluster dan waktu eksekusi

	Jumlah cluster		Waktu eksekusi (detik)		
	ISMC	CA	ISMC	CA	Peningkatan
A	9	21	166,53	39,11	76,5%
B	9	17	193,55	65,91	65,9%
C	7	19	11,87	11,54	2,8%
D	5	17	11,09	21,20	-91,2%
E	6	15	10,54	17,96	-70,4%
F	9	18	14,61	21,04	-44,0%
G	5	17	9,88	20,15	-103,9%

Tabel 2 menunjukkan kecenderungan CA untuk menghasilkan jumlah cluster yang lebih banyak daripada ISMC, walaupun kedua input parameternya sama. Sedangkan dalam hal waktu eksekusi, ternyata tidak semua sampel dieksekusi dalam waktu yang lebih cepat. Hal ini merupakan *uncertainty* yang keberadaannya sangat tergantung kondisi sampel. Namun demikian berdasarkan uji t (*student-t test*), kedua sebaran nilai di atas tidak dianggap mengalami perbedaan yang signifikan. Sebab hasil uji statistik, nilai t adalah 1,857. Sedangkan pada nilai t tabel untuk jumlah sampel 7 (df = 6) dan tingkat signifikansi 0,01 memperlihatkan nilai kritis (*critical value*) sebesar 3,143. Hipotesa null menyatakan bahwa perbedaan rata-rata kedua waktu tersebut adalah 0. Oleh karena 1,857 tersebut lebih rendah dari pada nilai kritis, maka hipotesa null dapat diterima. Namun demikian terdapat sebuah pola yang cukup menarik, yakni peningkatan kecepatan yang sangat tajam makin terlihat pada sampel dengan jumlah *band* yang lebih banyak.

Tabel 3. Nilai $Tr(S_B)$ sampel

Kode	$Tr(S_B)$		
	ISMC	CA	Peningkatan
A	4,478022e+07	5,382784e+07	16,81%
B	4,401811e+07	8,321390e+07	47,10%
C	2,773709e+08	2,898602e+08	4,31%
D	3,611318e+08	3,740104e+08	3,44%
E	6,924016e+07	7,156918e+07	3,25%
F	1,732320e+08	1,747408e+08	0,86%
G	2,060136e+08	2,174889e+08	5,28%

Idealnya, $Tr(S_B)$ haruslah setinggi mungkin, sebab perbedaan antar *cluster* harus sebesar mungkin. Tabel 3 menunjukkan peningkatan $Tr(S_B)$ untuk seluruh data sampel. Hal ini menunjukkan keunggulan CA dalam hal heterogenitas antar *cluster*.

Tabel 4. Nilai $Tr(S_W)$ sampel

Kode	$Tr(S_W)$		Peningkatan homogenitas
	ISMC	CA	
A	2,052004e+07	8,617945e+06	58,00%
B	3,950288e+07	8,126886e+06	79,43%
C	2,461730e+07	9,319228e+06	62,14%
D	1,725970e+07	4,020906e+06	76,70%
E	4,741370e+06	1,922373e+06	59,46%
F	8,211367e+06	5,502580e+06	32,99%
G	1,486727e+07	3,354379e+06	77,44%

Sedangkan kondisi ideal $Tr(S_W)$ adalah harus serendah mungkin atau dengan kata lain keadaan tiap data dalam setiap *cluster* harus dibuat sehomogen mungkin. Ternyata Tabel 4 menunjukkan penurunan $Tr(S_W)$. Hal ini membuktikan keunggulan CA dalam hal heterogenitas antar *cluster*.

3.3. Eksekusi dengan Jumlah Cluster Sama

Setelah data sampel diuji coba dengan ISMC, maka akan diketahui jumlah cluster yang dihasilkannya. Bila algoritma CA dipaksa untuk menghasilkan jumlah cluster yang sama, maka waktu eksekusi, homogenitas, dan heterogenitas akan dapat dihitung dengan lebih obyektif.

Pemaksaan ini dilakukan dengan mengganti *stopping criteria* pada *complete link* yang semula berdasarkan *threshold merge*, sekarang digantikan dengan jumlah *cluster*. Tabel 5, 3.6, dan 3.7 memperlihatkan waktu eksekusi, $tr(S_B)$, dan $tr(S_W)$ uji coba CA ini, sekaligus membandingkannya dengan hasil ISMC sebelumnya.

Tabel 5. Jumlah cluster dan waktu eksekusi

	Jumlah cluster		Waktu eksekusi (detik)		
	ISMC	CA	ISMC	CA	Peningkatan
A	9	9	166,53	27,19	83,7%
B	9	9	193,55	26,20	86,5%
C	7	7	11,87	26,92	-126,8%
D	5	5	11,09	8,13	26,7%
E	6	6	10,54	8,40	20,3%
F	9	9	14,61	11,64	20,3%
G	5	5	9,88	7,69	22,2%

Adapun perbandingan waktu eksekusi seperti nampak pada Tabel 5, ternyata juga mengalami hal yang sama dengan perbandingan waktu pada uji coba tahap pertama. Tidak semua sampel dapat dieksekusi lebih cepat oleh algoritma ini dibandingkan algoritma ISMC.

Namun demikian berdasarkan uji t (*student-t test*), kedua sebaran nilai di atas juga tidak dianggap mengalami perbedaan yang signifikan. Sebab hasil uji statistik, nilai t adalah 1,720. Sedangkan pada tabel distribusi t, nilai kritis (*critical value*) adalah 3,143. Oleh karena 1,720 tersebut

lebih rendah dari pada nilai kritis, maka hitpotesa null juga dapat diterima.

Tabel 6. Nilai $Tr(S_B)$ sampel

Kode	$Tr(S_B)$		
	ISMC	CA	Peningkatan
A	4,478022e+07	4,757978e+07	5,88%
B	4,401811e+07	7,608887e+07	42,15%
C	2,773709e+08	2,878832e+08	3,65%
D	3,611318e+08	3,657468e+08	1,26%
E	6,924016e+07	7,062125e+07	1,96%
F	1,732320e+08	1,712113e+08	-1,18%
G	2,060136e+08	2,135345e+08	3,52%

Berdasarkan jumlah *cluster* yang sama, nilai $Tr(S_B)$ ini juga menunjukkan peningkatan, sebagaimana Tabel 6. Namun, terdapat 1 sampel (F) yang justru menurun. Jadi algoritma CA ini memang menunjukkan peningkatan heterogenitas antar *cluster*, namun dimungkinkan pula terjadinya anomali pada suatu kondisi tertentu yang *uncertainty*.

Tabel 7. Nilai $Tr(S_W)$ sampel

Kode	$Tr(S_W)$		
	ISMC	CA	Peningkatan
A	2,052004e+07	1,444018e+07	29,63%
B	3,950288e+07	1,523252e+07	61,44%
C	2,461730e+07	1,631752e+07	33,72%
D	1,725970e+07	1,966575e+07	-13,94%
E	4,741370e+06	3,917660e+06	17,37%
F	8,211367e+06	8,948068e+06	-8,97%
G	1,486727e+07	1,179007e+07	20,70%

Tabel 7 memperlihatkan lebih rendahnya nilai $Tr(S_W)$ pada algoritma CA. Namun, sebagaimana Tabel 6, di sini terjadi pula anomali, yakni pada sampel D dan F. Algoritma CA ini memang menunjukkan homogenitas keanggotaan *cluster* yang lebih meningkat, namun dimungkinkan pula terjadi sebaliknya pada suatu kondisi cluster tertentu.

3.4. Pengujian Secara Visual

Sampel untuk uji coba ini adalah citra Landsat TM dari daerah Jawa Timur, tepatnya daerah Surabaya dan sekitarnya. Salah satu *band* yang digunakan, yakni *band* 5 dapat dilihat pada Gambar 7. Citra ini telah dikenai proses *histogram equalisasi*, agar dapat dilihat dengan jelas. Nampak terdapat 8 lokasi berupa kotak yang diberi warna hitam atau putih yang masing-masing berukuran 10 x 10 *pixel*. Nantinya masing-masing lokasi tersebut dijadikan sebagai sampel kelas.

Adapun koordinat sudut kiri atas kedelapan blok tersebut adalah (18, 40), (233.85), (18.0), dan (234.120) untuk perairan, selanjutnya koordinat awal (183.219), (129.187), (140.221), dan (127.167) untuk perkotaan.

Gambar 7. Pengambilan 800 sampel daerah Jawa Timur

Eksekusi dengan ISMC ternyata menghasilkan 14 *cluster*. Oleh karena diinginkan agar hasil pemetaan itu bisa dilihat secara visual, maka algoritma CA juga harus menghasilkan jumlah *cluster* yang sama, yakni 14 *cluster*. Gambar 8 (a) menampakkan hasil ISMC dan Gambar 8 (b) menampakkan hasil CA. Sekalipun terkesan mirip, namun, ternyata CA mampu menampakkan keadaan kota secara lebih detail dibandingkan ISMC.

Gambar 8. Hasil klasifikasi daerah Jawa Timur

Pengujian secara visual melibatkan 2 kelas, yakni perairan dan perkotaan. Perairan meliputi laut, rawa, dan tambak. Sedangkan perkotaan meliputi jalan, perindustrian, dan perumahan. Jumlah sampel tiap kelas adalah 4 blok, sedangkan tiap blok berukuran 10x10 *pixel*. Jadi sampel untuk perairan dan perkotaan masing-masing berjumlah 400 pixel.

Tiap koordinat blok hasil eksekusi kedua algoritma dideteksi nomor kode warnanya, yang sekaligus menunjukkan kode cluster daerah tersebut. Selanjutnya tiap nomor diakumulasi, sehingga diketahui jumlah keseluruhannya. Deteksi ini sekaligus membandingkan tingkat ketelitian kedua algoritma dalam proses *clustering*. Matriks *confusion* pada Tabel 8 (a) menunjukkan bahwa ISMC mengklasifikasikan 800 *pixel* sampel tersebut ke dalam 9 kelas. Sedangkan CA menjadikannya 11 kelas, sebagaimana nampak pada Tabel 8 (b).

Pada Tabel 8 (a) nampak terjadi dualisme pada *cluster* nomor 2, dimana 6 *pixel* diantaranya dikenali sebagai perairan, namun 8 *pixel* diantaranya dikenali sebagai perkotaan. Keenam *pixel* tersebut terdapat pada blok dengan koordinat awal (234,130), yakni yang berlokasi di daerah tambak.

Kelas lahan perairan dan perkotaan pada Tabel 8 (b) ternyata berhasil dikenali dengan baik oleh 800 sampel *pixel* tersebut. Hal ini membuktikan bahwa algoritma CA memiliki ketepatan pengelompokan *pixel* yang lebih tinggi terhadap suatu kelas lahan.

Tabel 8. Matriks *confusion* Klasifikasi Jawa Timur

(a)

Hasil Klasifikasi ISMC		
No Cluster	Kelas Lahan #1	#2
1	110	
2	6	8
3	284	
4		7
5		156
6		1
7		96
8		24
9		108
Total	400	400

(b)

Hasil Klasifikasi CA		
No Cluster	Kelas Lahan #1	#2
1	107	
2	11	
3		6
4		54
5		36
6		130
7		35
8		8
9		122
10	282	
11		9
Total	400	400

4. KESIMPULAN

Berdasarkan penelitian yang diawali dengan mengkaji berbagai perkembangan algoritma *clustering* hingga diusulkannya algoritma CA ini, maka dapat ditarik beberapa kesimpulan berikut ini :

1. Berdasarkan uji coba dengan parameter merge dan threshold yang diset sama dengan rekomendasi dari algoritma ISMC, terbukti bahwa algoritma CA ini mampu menghasilkan *cluster* yang lebih kompak dan lebih menampakkan heterogenitas antar *cluster*, serta lebih tepat dalam mengidentifikasi kelas.
2. Peningkatan kecepatan eksekusi algoritma CA dibandingkan algoritma ISMC, akan lebih tajam untuk citra dengan jumlah *band* yang lebih banyak.
3. Proses pembagian (*split*) suatu *cluster* akan dapat dilakukan dengan lebih cepat, bila mempertimbangkan juga faktor distribusi warna *pixel* yang dalam penelitian ini direpresentasikan melalui *histogram*. Sebab dengan cara ini, lokasi terkonsentrasinya *pixel*, yang diasumsikan sebagai lokasi pusat suatu *cluster*, dapat dideteksi secara langsung.
4. Penggabungan (*merge*) sejumlah *cluster* yang serupa memang dapat dilakukan dengan berbagai metode, namun perlu diperhatikan karakteristik data tersebut. Keberadaan *noise* juga akan berpengaruh terhadap komposisi *cluster*. Dengan menggabungkan *cluster-cluster* yang diasumsikan terlalu berdekatan, akan menyebabkan jarak antar *cluster* makin jauh. Hal ini akan semakin meningkatkan perbedaan *prototype* suatu *cluster* dengan *cluster* lainnya.
5. *Fuzzy C Mean* yang memang berjenis *partitional* ini, tetap saja mengharuskan tersedianya inisialisasi lokasi pusat tiap *cluster*. Oleh karena itu, diperlukan teknik yang tepat untuk menghasilkan perkiraan awal lokasi pusat *cluster* tersebut, agar keadaan konvergen lekas tercapai. Dalam penelitian ini, *Fuzzy C Mean* didahului dengan proses *split* dan *merge* yang menghasilkan jumlah *cluster* yang optimal dan lokasi pusat *cluster* yang cukup mewakilinya.
6. Penggunaan PCT untuk proses pembuatan *histogram* sangat dipengaruhi oleh korelasi antar spektrum, oleh karenanya algoritma CA ini lebih sesuai untuk citra multispektral yang korelasi antar spektrumnya tinggi.
7. Kualitas hasil klasifikasi algoritma CA ini sangat tergantung pada distribusi pixel pada feature space yang sulit diprediksi. Oleh karena itu, sulit untuk menghindari terjadinya anomali sebagaimana pada citra sampel C dan F.

5. SARAN

Berikut ini adalah beberapa saran untuk penyempurnaan dan pengembangan lebih lanjut :

1. Metode yang diusulkan ini masih menggunakan jarak *euclidean* sebagai parameter penentu keanggotaan suatu *cluster*.
2. Pemakaian jarak ini berakibat ketidakmampuannya untuk membedakan *cluster* yang berlainan ukuran. Oleh karena itu, perlu juga mempertimbangkan penggunaan variansi.
3. Penentuan nilai *threshold* yang tepat, juga akan sangat membantu proses ini. Oleh karena itu dibutuhkan eksperimen yang lebih banyak untuk menggeneralisir *threshold* tersebut.
4. Mengingat penambahan jumlah *band* membuat peningkatan waktu yang cukup drastis, maka diperlukan proses pemilihan ciri untuk mereduksi jumlah dimensi yang terlibat proses *clustering*.

REFERENSI

[1] Aniati Murni dan S. Setiawan, *Pengantar Pengolahan Citra*, Jakarta; Elex Media Komputindo, 1992.

[2] Anita M. Seddon dan Garry E. Hunt, "Segmentation of Cloud Cluster Analysis", *International Journal of Remote Sensing*, (6), 1985, 717-731.

[3] H. Frigui dan R.Krishnapuram, "A Robust *Clustering* Algorithm Based on Competitive Agglomeration and Soft Rejection of Outliers", *Proceedings of the 1996 Conference on Computer Vision and Pattern Recognition*, 1996.

[4] J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York: Plenum, 1981.

[5] James J. Simpson, Timothy J. McIntire, dan Matthew Sienko, "An Improved Hybrid *Clustering* Algorithm for Natural Scenes", *IEEE Trans. on Geoscience and Remote Sensing*, 38(2), 2000.

[6] Mehmet Celenk, "Hierarchical color clustering for segmentation of textured images", *Proceedings of the 29th Southeastern Symposium on System Theory (SSST '97)*, 1997.

[7] Nevin A. Mohamed, M. N. Ahmed dan A. A. Farag, "Modified Fuzzy C-Mean in Medical Image Segmentation," *Proc. of IEEE-EMBS*, 20(3), 1998, 1377-1380.

[8] Nozha B., "On Competitive Unsupervised *Clustering*", *Proceedings of the International Conference on Pattern Recognition (ICPR'00)*, 2000.

[9] John. A. Richards, *Remote Sensing Digital Image Analysis, An Introduction*, Berlin: Springer-Verlag Berlin Heidelberg, 1986.

[10] Ph. Schmid dan S. Fischer, "Color segmentation for the analysis of Pigmented Skin Lesions", *Proceedings of the Sixth International Conference on Image Processing and Its Applications*, 2, 1997, 688-692.

[11] R. M. Haralick dan L. G. Saphiro, *Computer and Robot Vision*, Vol. I dan II, Addison-Wesley, 1993.

[12] Richard O. Duda dan Peter E. Hart, *Pattern Classification and Scene Analysis*, New York; John Wiley & Sons, 1973.

[13] William R. Dillon dan Matthew Goldstein, *Multivariate Analysis Methods and Applications*, New York: John Wiley & Sons, 1984.