

Algoritma Clustering Fuzzy Hibrida untuk Klasifikasi Citra Inderaja

Agus Zainal Arifin

Teknik Informatika FTI ITS Surabaya

E-mail : agusza@its-sby.edu Telp. (031)5933928

Abstrak

Proses klasifikasi merupakan proses untuk mendapatkan gambar/peta tematik. Pada paper ini kami ingin membahas sebuah algoritma klasifikasi *unsupervised* yang berusaha mengelompokkan data citra inderaja ke dalam sejumlah *cluster*.

Algoritma ini terdiri dari 2 tahap yakni tahap pendeteksian lokasi pusat tiap *cluster* dan tahap pencarian anggotanya. Tahap I merupakan modifikasi algoritma ISMC (*Improved Split and Merge Classification*). ISMC adalah algoritma *clustering* secara *partitional* yang pada tiap iterasinya disisipi algoritma *split and merge*. Pada tahap II kami terapkan klasifikasi *fuzzy c-mean* untuk memperoleh partisi *fuzzy* citra.

Analisa kuantitatif untuk menentukan efektifitas algoritma ini adalah *between cluster scatter matrix* S_b dan *within cluster scatter matrix* S_w . Pada uji coba, berdasarkan beberapa sampel, ternyata *trace* dari S_w algoritma ini dapat mencapai 53 % lebih baik dari pada ISMC. Sedangkan *trace* dari S_b yang dihasilkan keduanya ternyata sangat mirip. Dengan demikian, algoritma ini terbukti mampu menghasilkan *cluster* yang lebih homogen dibandingkan ISMC.

Kata Index : algoritma *clustering*, citra inderaja, *fuzzy c-mean*, *split and merge*, *within cluster scatter*, *between cluster scatter*.

I. Pendahuluan

I.1. Klasifikasi citra

Klasifikasi citra merupakan proses yang berusaha mengelompokkan seluruh *pixel* pada suatu citra ke dalam sejumlah *class* (kelas), sedemikian hingga tiap *class* merepresentasikan suatu entitas dengan properti yang spesifik [1], [10]. Tujuan utama klasifikasi citra penginderaan jauh adalah untuk menghasilkan peta tematik, dimana suatu warna mewakili suatu objek tertentu. Contoh objek antara lain hutan, air,

sawah, kota, jalan, dan lain-lain. Bahkan pada citra satelit meteorologi, proses klasifikasi dapat menghasilkan peta awan yang memperlihatkan distribusi awan di atas suatu wilayah.

Algoritma klasifikasi dapat dibagi menjadi 2 kategori yakni *supervised* (terawasi) dan *unsupervised* (tak terawasi) [1][2][3][8]. Penggunaan keduanya bergantung pada ketersediaan data awal pada citra itu. Metode *supervised* patut digunakan bila telah tersedia *training set*, yakni sejumlah *pixel* yang sudah diklasifikasikan terhadap sejumlah kelas tertentu. Proses ini harus dilakukan oleh seorang pakar yang benar-benar mengetahui jenis objek yang berada pada lokasi *training set*.

Dengan cara ini tiap kelas spektrum dapat diwakili oleh suatu distribusi probabilitas, sehingga kesuksesan klasifikasi sangat tergantung pada kelengkapan sampel yang tersedia. Namun bila *training set* ini belum diketahui distribusi kelasnya, maka metode *unsupervised* akan sangat membantu. Dengan cara ini, user tidak perlu menginputkan sampel tiap kelas, sebab data akan dikelompokkan secara natural berdasarkan properti masing-masing.

I.2. Algoritma *clustering*

Analisa *cluster* merupakan suatu bentuk pengenalan pola yang berkaitan dengan pembelajaran secara *unsupervised*, dimana jumlah pola kelas tidak diketahui. [4][5]. Proses *clustering* berusaha untuk mengelompokkan *pixel* dalam *feature space* (ruang ciri) secara natural ke dalam sejumlah *cluster*. *Cluster* merupakan suatu kelompok yang homogen, dimana tiap unit di dalamnya memiliki kemiripan satu sama lain [5].

Clustering bisa berbentuk *hierarchical* atau *partitional* [4]. Metode *hierarchical* membuat konstruksi berupa urutan partisi secara *nested* dan *non iterative*. Outputnya adalah

sebuah *dendrogram*, yakni *tree* yang menunjukkan pengelompokan sampel. Prosedur ini dibagi menjadi 2 kategori, yakni *agglomerative (bottom-up)* dan *divisive (top-down)*.

Agglomerative dimulai dari n *cluster* beranggota tunggal yang digabung (*merged*) dengan *cluster* lainnya secara berurutan, hingga menjadi 1 *cluster*. Sedangkan *divisive* dimulai dari 1 partisi beranggotakan semua sampel, yang dipecah (*split*) secara berurutan, hingga menjadi sejumlah *cluster* yang masing-masing beranggotakan 1 sampel. Adapun metode *partitional*, berusaha menghasilkan partisi secara *iterative*.

Kedua metode di atas dapat digabungkan dengan melakukan *partitional clustering* yang pada tiap iterasinya disisipkan *hierarchical clustering (split and merge)*. Penggabungan ini sangat menguntungkan [4], sebab metode hierarikal murni mengharuskan adanya struktur taksonomi data yang tentunya kurang sesuai untuk citra inderaja. Selain itu juga tidak sesuai untuk data yang kompleks. Metode ini sangat tergantung pada urutan *split dan mergenya*, sehingga urutan yang berbeda akan menghasilkan struktur *cluster* yang berbeda pula. Sedangkan metode *partitional* murni, seringkali konvergen pada *local minimum* dari *clustering criterion function*.

Bila kedua metode ini dikombinasikan, maka tidak perlu terlalu tergantung pada inisialisasi distribusi *cluster*. Tentunya jumlah *cluster* pada inisialisasi tidak harus sama dengan kondisi data yang sebenarnya. Pada awal tahun 2000, J. J. Simpson dan rekan-rekannya berhasil mengembangkan sebuah algoritma hibrida yang dinamakan *Improved Split and Merge Classification (ISMC)*. Algoritma ISMC adalah pengembangan dari algoritma SMC yang telah berhasil diaplikasikan untuk mengukur suhu permukaan laut dengan mendeteksi kondisi awan di atasnya. Dalam uji cobanya, ISMC terbukti menghasilkan struktur *cluster* yang lebih homogen dan konvergen lebih cepat dari pada algoritma SMC dan ISODATA. Selain itu, parameter inputnya lebih sedikit dan lebih mudah diisi, sebab nilai *thresholdnya* disesuaikan berdasarkan karakter citra.

Paper ini membahas algoritma ISMC sekaligus mengusulkan sebuah algoritma yang berusaha meningkatkan homogenitas *cluster*

dengan cara memodifikasi ISMC tersebut dan menggabungkannya dengan algoritma *fuzzy c mean*. Nama algoritma usulan ini adalah *Fuzzy Split and Merge Clustering (FSMC)*. Di samping itu, FSMC juga berusaha meningkatkan heterogenitas antar *cluster*, sehingga perbedaan antar *cluster* diusahakan sebesar mungkin.

II. Tinjauan Algoritma ISMC

ISMC adalah algoritma *clustering* secara *partitional* yang pada tiap iterasinya disisipi dengan algoritma *split and merge*. Algoritma ini terbukti jauh lebih baik dari pada algoritma ISODATA.

II. 1. Bentuk algoritma asli ISMC

Berikut ini langkah-langkah algoritma ISMC :

1. Penentuan nilai *threshold*

Terdapat 3 parameter *threshold* yang dibutuhkan oleh ISMC, yakni τ_m (*merge*), τ_s (*split*), dan Toleransi. Penetapan nilai *threshold* yang optimal akan mampu mengurangi redundansi (langkah-langkah yang tidak diperlukan). Dalam hal ini *threshold* untuk *split dan merge* disesuaikan dengan karakteristik citra. Penghitungan τ_m dan τ_s adalah sebagai berikut :

$$\tau_s = \text{SPLIT} \times s$$

$$\tau_m = \text{MERGE} \times s$$

Variabel s dihitung berdasarkan :

$$s = \beta \sum_{i=1}^n (\max_i - \min_i)^2$$

Dimana $\beta = 1.0 \times 10^{-4}$, n adalah jumlah komponen pada *vector (pixel)*, \max_i dan \min_i adalah nilai maximum dan minimum pada elemen ke- i dari seluruh *vector*. Sedangkan nilai *default* SPLIT = 750 dan MERGE = 50 ditetapkan berdasarkan eksperimen. Adapun Toleransi yang digunakan untuk menentukan konvergensi menuju akhir proses *clustering* diberi nilai *default* 0.05.

2. *Split cluster*

Urutan langkahnya sebagai berikut :

- Berdasarkan *component-wise*, dilakukan pencarian nilai maximum dan minimum untuk tiap dimensi. Bila telah ditemukan, maka dilakukan *bounding*, sehingga

semua *vector* berada dalam batasan tersebut. Bila jumlah dimensi adalah n , maka jumlah titik sudut yang dihasilkan adalah 2^n .

- Untuk tiap sudut *boundary*, dilakukan pencarian *vector* yang terdekat dan terjauh dari tiap titik sudut tersebut. Jumlah *vector* yang dihasilkan adalah 2×2^n .
- Tiap *vector* tersebut selanjutnya diukur jaraknya satu sama lain. Pasangan *vector* yang jaraknya paling jauh, akan dijadikan sebagai pedoman untuk membagi semua *vector* yang ada ke dalam 2 *cluster*. Pasangan *vector* ini dapat dinamakan sebagai y_{\max} dan y_{\min} . Proses *split* baru boleh dilakukan, bila d_m melebihi *splitting threshold* τ_s . Bila tidak, maka proses langsung menuju langkah ke-3. Adapun nilai d_m ditentukan oleh rumus :

$$d_m = (y_{\max} - y_{\min})^T (y_{\max} - y_{\min})$$

- Proses *split* diharapkan akan menghasilkan 2 *cluster* dengan cara mengukur jarak seluruh *vector* pada *cluster* ini terhadap kedua *vector* kutub. Kutub yang terdekat itulah yang akan diikutinya sebagai *cluster* yang baru.

Langkah ke-2 ini diulang terus menerus hingga tidak terjadi *split* lagi.

3. Penghapusan seluruh *cluster* yang tidak memiliki elemen.

4. Penggabungan *cluster*

Tiap pasang *cluster* dihitung jarak antar *mean vector*nya dengan rumus :

$$d_m = (m_i - m_j)^T (m_i - m_j)$$

Bila d_m kurang dari *merging threshold* τ_m , maka lakukan penggabungan semua *vector* pada kedua *cluster* tersebut, sehingga menjadi 1 *cluster*.

Ulangi langkah ke-4 ini, hingga tidak ada lagi *cluster* yang bisa digabung. Hasil akhirnya adalah K buah *cluster*.

5. Penentuan keanggotaan tiap *vector*

Dengan berdasarkan K buah *cluster* tersebut, tiap *vector* selanjutnya diperiksa dan diputuskan untuk menjadi anggota salah satu *cluster* yang terdekat.

Pengukurannya menggunakan *Euclidean distance* dengan rumus :

$$d_k = (y - m_k)^T (y - m_k), \text{ dimana } k = 1, \dots, K$$

K adalah jumlah seluruh *cluster*. Tiap *vector* diputuskan sebagai anggota *cluster* yang menghasilkan nilai d_k terkecil.

6. Penghitungan statistik *cluster*

Penghitungan angka statistik ini meliputi *trace* dari matriks *between-cluster scatter* S_b , *mean* m , dan konvergensi.

- ✓ Rumus yang digunakan pada penghitungan S_b adalah :

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

Dimana n_k adalah jumlah *vector* dalam *cluster* ke- k , sedangkan m adalah *mean vector* dari seluruh data set.

- ✓ Sedangkan rumus penghitungan *mean vector* untuk seluruh data set adalah :

$$m = \frac{1}{N} \sum_{k=1}^K n_k m_k$$

- ✓ Pemeriksaan konvergensi

Pemeriksaan ini menggunakan *trace* matriks *between-cluster scatter* $T_r(S_b)$. Idealnya, $T_r(S_b)$ ini harus sebesar mungkin untuk menyatakan bahwa tiap *cluster* berbeda satu sama lain. Konvergensi diperoleh, bila pada iterasi berikutnya, S_b tidak mengalami perubahan yang mencolok. Rumus yang digunakan adalah :

$$\frac{T_r(S_b) - T_r(S'_b)}{T_r(S_b)} < tol$$

S'_b adalah matriks *between-cluster scatter* dari iterasi sebelumnya, sedangkan *tol* adalah nilai toleransi yang defaultnya diset 0.05. Oleh karena jumlah *cluster* awal pada ISMC ini hanya 1, maka nilai awal S_b adalah 0.

7. Bila rumus diatas terpenuhi, maka algoritma dihentikan, dan bila tidak terpenuhi, maka ulangi langkah ke-2.

II. 2. Langkah *hierarchical*

Metode *hierarchical* ditunjukkan oleh adanya *split* (langkah ke-2) dan *merge* (langkah ke-4). Langkah *split* nampak sangat efisien, sehingga bisa menghemat waktu pemrosesan. Sedangkan langkah *merge* dilakukan dengan mencari *cluster* yang berdekatan untuk digabungkan seluruh anggotanya. Bila kedua *cluster* telah digabung, maka jarak antara *cluster* gabungan dengan *cluster* lain perlu diukur ulang dengan suatu model pengukuran jarak (*distance measure*). Sangat dimungkinkan di sini, bahwa metode penggabungan secara *agglomerative* pada ISMC ini, memanfaatkan *distance measure* model *single link*.

II.3. Langkah *Partitional*

Metode *partitional* dalam algoritma ISMC ditunjukkan oleh langkah ke-5, yakni melalui penentuan keanggotaan tiap *vector*. Proses ini ternyata hanya dilakukan sekali. Sedangkan pada umumnya proses *partitional* diulang beberapa kali hingga konvergen, sebagaimana karakteristik metode ini yang memang bersifat *iterative*.

III. Algoritma Usulan (FSMC)

Algoritma yang diusulkan ini merupakan modifikasi dari ISMC, sehingga pembahasannya tidak dapat terlepas sama sekali dari ISMC. Usulan di bawah ini berdasarkan penjelasan pada bab II.

III. 1. Ukuran jarak

Pengukuran jarak antar cluster (*distance measure*) pada *agglomerative hierarchical clustering*, pada umumnya menggunakan salah satu dari 3 algoritma berikut, yakni *nearest-neighbor*, *furthest-neighbor*, dan *compromise* [1].

Algoritma *nearest-neighbor* (*single link*) berusaha membentuk *minimal spanning tree* dengan mencari *cluster* lain yang jarak minimalnya di bawah *threshold*. Resiko yang mungkin terjadi adalah adanya *chain effect*, sehingga cara ini cukup sensitif terhadap *noise*. Sebab *cluster* baru, bisa ditambahkan bila jaraknya terhadap salah satu anggota *cluster* lama di bawah *threshold*. Bila terdapat *noise* di antara 2 *cluster* yang semestinya tidak layak untuk

digabung, maka *cluster noise* ini akan menjadi penyebab tergabungnya kedua *cluster* tersebut.

Sedangkan *furthest-neighbor* (*complete link*) berusaha membentuk *complete sub graph* pada tiap *cluster*. Jarak antara 2 *cluster* ditentukan oleh jarak *node* terjauh di antara keduanya. Bila jarak terjauh ini dianggap sebagai diameter *cluster*, maka diameter *cluster* pada tiap iterasi tentu akan makin meningkat. Dan bila diameter ini melampaui *threshold*, maka penggabungan tidak akan dilakukan. Dengan demikian, dapat dijamin bahwa, semua *cluster* yang digabungkan memang benar-benar mirip. Resikonya adalah akan terbentuk *cluster* yang cukup banyak.

Adapun *compromise*, berusaha menentukan jarak antara 2 *cluster* dengan *average* (berdasarkan rata-rata seluruh jarak antar node) atau *mean* (berdasarkan rata-rata kedua *cluster center*). Algoritma ini menjembatani kedua algoritma sebelumnya.

Berdasarkan pertimbangan di atas, maka algoritma *complete link* lebih sesuai untuk digunakan sebagai algoritma *merging*. Alasannya adalah untuk mengantisipasi kemungkinan terdapatnya *noise* dalam citra dan untuk membentuk *cluster* yang anggotanya sehomogen mungkin. Sedangkan resiko akan terbentuknya *cluster* yang sangat banyak, akibat ketatnya syarat penggabungan ini, diharapkan tidak akan terjadi. Sebab jumlah *cluster* hasil proses *split* yang diinputkan ke algoritma ini memang tidak terlalu banyak. Sedikitnya jumlah *cluster* ini diakibatkan oleh penentuan *threshold split* yang berusaha disesuaikan dengan kondisi citra aslinya, sehingga jumlahnya relatif stabil. Dengan demikian, langkah ke-4 menggabungkan *cluster-cluster* yang berdekatan dengan metode *complete link*.

III. 2. Langkah *partitional*

Pada algoritma ISMC, bagian *partitional* yang dilibatkan tersebut biasa dinamakan *K-means clustering method*. Namun demikian, proses ini ternyata hanya dilakukan satu kali iterasi, dan tanpa mengulanginya hingga konvergen. Metode *K-means clustering* memang sangat dikenal dan banyak digunakan di berbagai penelitian. Namun demikian, terdapat 2 buah kekurangan yang cukup menonjol, yakni

harus tersedianya nilai K yang cukup sensitif dan hasil klasifikasinya yang digolongkan sebagai *hard clustering*.

Pada prinsipnya, nilai K seharusnya adalah nilai yang senatural mungkin. Nilai ini biasanya diberikan oleh peneliti berdasarkan pengetahuannya yang cukup mendalam pada citra yang sedang dianalisa. Namun untuk citra yang tidak menyediakan jumlah informasi kelas yang riil, maka hal ini akan cukup mengganggu. Di samping itu perlu juga inisialisasi lokasi awal untuk tiap *cluster* tersebut, dengan menentukan pusatnya masing-masing. Dalam hal inisialisasi ini, prosedur *split and merge* di atas telah berusaha untuk menyediakan nilai K yang seoptimal mungkin dengan disertai *vector* pusat *clusternya*.

Permasalahan *hard clustering* adalah adanya keharusan dalam tiap iterasi itu untuk mengalokasikan tiap *vector* ke suatu *cluster* tertentu. *Hard* yang dimaksud di sini adalah suatu *data point (vector)* hanya dapat dimiliki oleh 1 *cluster* [7]. Logikanya, setiap *vector* dimungkinkan pula memiliki kemiripan dengan setiap *cluster* yang ada, walaupun nantinya memang ada diantara *cluster* yang memiliki tingkat kemiripan tertinggi terhadap *vector* ini. *Cluster* inilah yang dianggap memiliki *vector* tersebut. Namun demikian tingkat kemiripan dengan *cluster* lainnya tidak dapat pula diabaikan. Sebab dalam iterasi berikutnya, mungkin saja tingkat kemiripan ini akan semakin meningkat, sehingga menyebabkan diputuskannya *vector* tersebut untuk berpindah *cluster*.

Metode *partitional* yang hard tersebut sebenarnya sudah mulai diatasi oleh para peneliti dengan menerapkan metode *fuzzy clustering*. Metode ini biasa disebut dengan *fuzzy c-mean* [6][7][9]. Notasi yang digunakan antara lain x_{js} yakni nilai ke-s dari *vector* x_j (*vector* x yang ke-j), dengan $1 \leq s \leq p$ dan p adalah jumlah band, u_{ik} adalah membership x_k terhadap *cluster* i , dimana :

$$u_{ik} = u_i(x_k), \quad 1 \leq i \leq c \text{ dan } 1 \leq k \leq n, \\ \text{dengan } 0 \leq u_{ik} \leq 1 \text{ untuk setiap } i, k \\ \sum_{i=1}^c u_{ik} = 1 \text{ untuk setiap } k \\ 0 < \sum_{i=1}^c u_{ik} < n \text{ untuk setiap } i$$

Algoritma *fuzzy c-mean* berusaha mengelompokkan *vector pixel* tersebut,

sedemikian hingga dicapai local minima dari fungsi berikut :

$$J_m(U, v ; X) = \sum_{i=1}^c \sum_{k=1}^n (u_{ik})^m D_{ik}$$

U adalah *fuzzy c segmentation*, v adalah himpunan pusat tiap *cluster*, dan X adalah input data yakni seluruh *vector pixel*. Sedangkan D_{ik} diukur berdasarkan kemiripan antara v_i dan x_k . Minimisasi J_m berdasarkan penghitungan U dan v secara *iterative* melalui persamaan :

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

Bila m diset 1, maka :

$$u_{ik} = 1, D_{ik} = \min(D_{sk}) \text{ untuk } 1 \leq s \leq c \\ u_{ik} = 0, \text{ selain di atas untuk } 1 \leq i \leq c ; 1 \leq k \leq n$$

Prosedur *fuzzy c-mean* ini dilakukan pada langkah ke-8, dimana langkah ke-1 hingga langkah ke-7 dari algoritma ISMC hasil modifikasi telah dijalankan hingga konvergensi $Tr(S_b)$ dicapai. Algoritma inilah yang selanjutnya disebut sebagai *Fuzzy Split and Merge Classification (FSMC)*.

IV. Uji Coba dan Evaluasi

IV.1. Citra input

Citra input dalam uji coba algoritma ini diperoleh dari satelit GOES-8 dengan alamat <ftp://rsd.gsfc.nasa.gov/pub/goes8>. *Geostationary Operational Environmental Satellite (GOES)* merupakan satelit meteorologi yang dioperasikan oleh *National Oceanic and Atmospheric Administration (NOAA)*. Satelit ini berdiam relatif terhadap bumi, di atas suatu tempat di atas garis khattulistiwa. Saat ini terdapat 3 jenis GOES yang beroperasi, yakni GOES-8 (pada posisi 75° BB), GOES-9 (pada posisi 135° BB), dan GOES-10 yang masih dalam taraf uji coba.

Satelit ini menyediakan 5 band citra yakni *visible, shortwave infrared window, upper level water vapor, longwave infrared window, dan infrared window more sensitive to water vapor*. Berdasarkan alasan kemudahan dalam perbandingan dengan algoritma ISMC, maka uji coba algoritma FSMC menggunakan 3 band yakni band 2, 4, dan 5. Pertimbangannya adalah bahwa uji coba algoritma ISMC menggunakan AVHRR band 2, 3, dan 4. Sementara itu Band 1, 2, 4, dan 5 dari GOES dinyatakan mirip dengan AVHRR band 1, 3, 4, dan 5. Namun demikian satelite GOES tidak memiliki near-infrared band (AVHRR Band 2), sedangkan AVHRR tidak memiliki water vapor band (GOES Band 3). Adapun perincian sampel yang digunakan dapat dilihat pada tabel IV-1.

Tabel IV-1
Daftar sampel citra uji coba

Kode	Lokasi	Tanggal	# <i>pixel</i>
A	Argentina	23-02-2001	100 ²
B	Brazil	23-02-2001	100 ²
C	California	23-02-2001	100 ²
D	California	18-03-2001	256 ²
E	Colorado	15-03-2001	100 ²
F	Florida	18-03-2001	100 ²
G	Galapagos	18-03-2001	256 ²
H	Nicaragua	18-03-2001	256 ²
I	Panama	18-03-2001	256 ²
J	Texas	18-03-2001	256 ²

Dengan demikian *vector* input sebagai representasi nilai *pixel* terdiri dari 3 komponen.

IV.2. Analisa kuantitatif

Pengukuran secara kuantitatif untuk menentukan efektifitas algoritma ini, menggunakan 2 variabel. Variable tersebut adalah *between cluster scatter matrix Sb* dan *within cluster scatter matrix Sw*.

Idealnya, setiap *cluster* yang dihasilkan oleh suatu algoritma *clustering* haruslah sehomogen mungkin. Oleh karena itu *Sw* harus diusahakan sekecil mungkin. Di samping itu, perbedaan antara suatu *cluster* dengan *cluster* yang lain haruslah sebesar mungkin. Oleh karena itu *Sb* harus setinggi mungkin.

Rumus untuk menghasilkan *Sb* dapat dilihat pada langkah ke-6 ISMC. Sedangkan rumus *Sw* akan dibahas lebih lanjut.

Misalkan C_k adalah *cluster* ke- k dari K buah *cluster* yang dihasilkan. *Cluster* ini terdiri dari n_k *vector*, yakni $\{y_1^k, y_2^k, \dots, y_{n_k}^k\}$. *Sw* untuk C_k adalah :

$$S_k = \sum_{i=1}^{n_k} (y_i^k - m_k)(y_i^k - m_k)^T$$

Dimana m_k adalah *mean vector* untuk C_k . Sedangkan *Sw* untuk seluruh data set adalah :

$$S_w = \sum_{i=1}^K S_i$$

IV.3. Uji Coba dan evaluasi

Setiap citra pada tabel IV-1 dikenakan 2 algoritma yakni ISMC dan FSMC. Pada setiap akhir eksekusi, citra output hasil klasifikasi disimpan dalam format BMP. Adapun hasil penghitungan statistiknya yang meliputi jumlah iterasi yang dilalui, Trace dari *Sb* dan *Sw*, serta perkembangan jumlah *cluster* hingga akhir eksekusi disimpan dalam bentuk text.

Hasil eksekusi dapat dilihat pada tabel IV-2. Semua citra sampel konvergen setelah 2 kali iterasi kecuali citra I yang membutuhkan 3 kali iterasi. Nampak pada tabel IV-2, bahwa hasil trace *Sb* untuk algoritma ISMC dan FSMC dicatat menjadi satu. Hal ini disebabkan oleh karena hasilnya hampir selalu sama. Sekalipun pada sebageian sampel nilai *Sb* ini meningkat yang berarti mengalami perbaikan, namun pengingkatannya kurang significant, sebab perbedaannya sangat kecil.

Nampak pula bahwa FSMC menghasilkan nilai *Sw* yang lebih kecil dari pada nilai yang dihasilkan oleh ISMC untuk semua sampel. Hal ini menunjukkan bahwa *cluster* yang dihasilkan oleh FSMC secara umum lebih homogen daripada *cluster* yang dihasilkan oleh ISMC. Peningkatan homogenitas *cluster* berkisar antara 2% hingga 53%.

Jumlah *cluster* yang sama antara ISMC dan FSMC menunjukkan bahwa proses *split* yang dihasilkan oleh langkah ke-2 ternyata sudah optimal, sehingga tidak lagi membutuhkan

Tabel IV-2
Hasil Eksekusi ISMC dan FSMC

Kode	Jumlah Cluster	Trace Sb ISMC/FSMC	Trace Sw ISMC	Trace Sw FSMC	Perbaikan Trace Sw
A	5	5,005681e+15	1,489051e+06	6,999090e+05	53%
B	6	3,122723e+15	9,097564e+05	7,043362e+05	23%
C	6	1,101994e+15	1,405739e+06	9,124760e+05	35%
D	7	1,660603e+17	2,461730e+07	1,530510e+07	38%
E	8	7,580751e+14	1,352846e+06	1,325475e+06	2%
F	6	1,046466e+15	1,000971e+06	8,750798e+05	13%
G	5	2,369541e+17	1,725970e+07	1,408119e+07	18%
H	5	1,490510e+17	1,486727e+07	1,100553e+07	26%
I	6	6,342382e+16	4,543842e+06	3,340266e+06	26%
J	9	7,789833e+16	8,211367e+06	7,753856e+06	6%

adanya *merge*. Hal ini nampak pada jumlah *cluster* hasil langkah ke-2 yang tidak berubah setelah melalui langkah ke-4. Bahkan kondisi iterasi pertama tidak berubah pada iterasi kedua, kecuali pada citra I yang membutuhkan 3 iterasi. Namun demikian, iterasi kedua dan ketiga citra I juga mengalami hal yang sama, dan jumlah *cluster* pada kedua iterasi tersebut lebih besar dari pada jumlah *cluster* pada iterasi pertama. Tentunya ini juga membuktikan bahwa tidak ada *merge* yang terjadi.

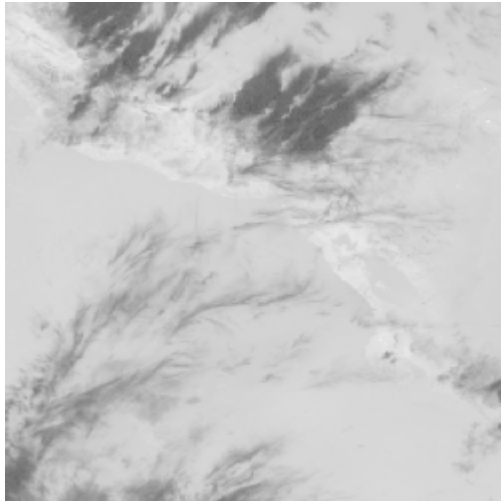
Pada gambar IV-1 ditampilkan 3 contoh citra GOES-8 untuk daerah Nicaragua yang diambil pada tanggal 18 Maret 2001. Gambar (a) berasal dari band 2, (b) band 4, dan (c) band 5. Perlu diketahui, bahwa ketiga citra ini, sebelum ditampilkan, dilakukan proses *invers* lebih dahulu. Tujuannya agar dapat dengan mudah dilihat secara visual. Namun demikian, proses klasifikasi tetap menggunakan citra asli.

Adapun hasil klasifikasinya dapat dilihat pada gambar IV-2. Pada gambar tersebut terdapat 2 citra hasil klasifikasi. Citra (a) adalah hasil klasifikasi dengan ISMC, dan citra (b) adalah hasil klasifikasi dengan FSMC. Sepintas memang agak sulit membedakan hasil dari kedua algoritma tersebut, namun terdapat beberapa perbedaan pada keanggotaan *pixel-pixel*nya terhadap 5 *cluster* yang terbentuk.

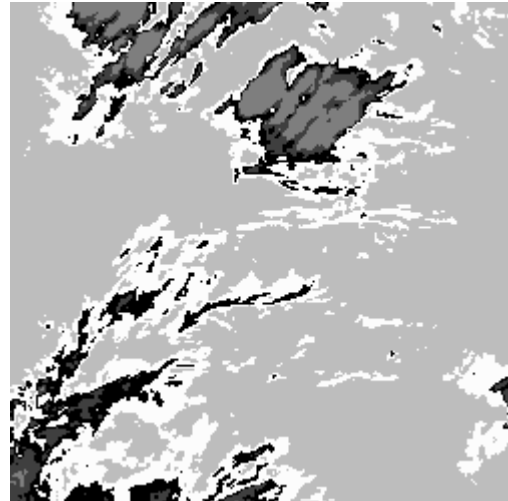
V. Kesimpulan

Berdasarkan uji coba pada sejumlah sampel tersebut, dapat ditarik beberapa kesimpulan, yakni antara lain:

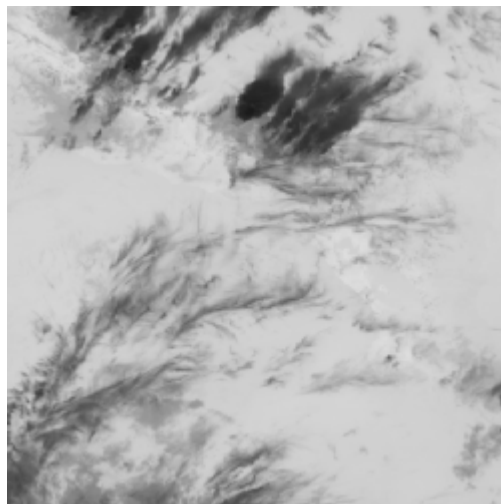
1. Penambahan algoritma *partitional clustering* berupa *fuzzy c-mean* ternyata cukup efektif untuk meningkatkan homogenitas tiap *cluster* yang dihasilkan. Hal ini dibandingkan dengan hanya mengelompokkan seluruh *pixel* ke pusat *cluster* yang terdekat.
2. Algoritma *partitional* sangat tergantung terhadap inisialisasi *cluster*, baik jumlah *cluster* awal, maupun posisi awal pusat *cluster*. Hal ini dapat diantisipasi dengan baik, bila sebelumnya diterapkan algoritma *split and merge* untuk mencari jumlah *cluster* seklaigus posisi center-nya yang paling optimal.
3. Penetapan *threshold* yang tepat untuk *split dan merge* berakibat tidak bergunanya proses *merge*, sebab bila *cluster* sudah berhasil *displit* hingga menjadi sejumlah pusat *cluster*, maka jarak antar pusat *cluster* tersebut tidak memungkinkan lagi untuk digabung. Kedua angka *threshold* ini hanya bisa diperoleh melalui uji coba dengan berbagai kombinasi.



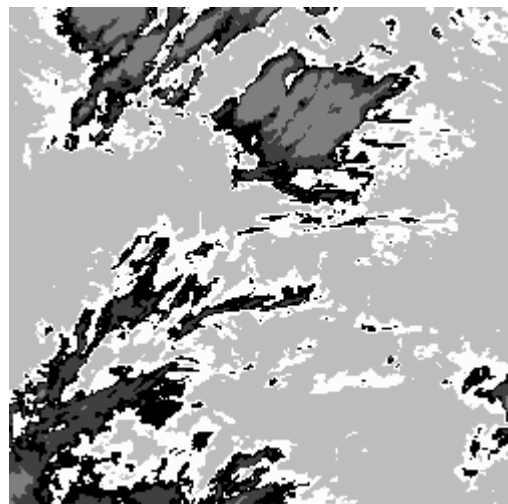
(a) Band 2



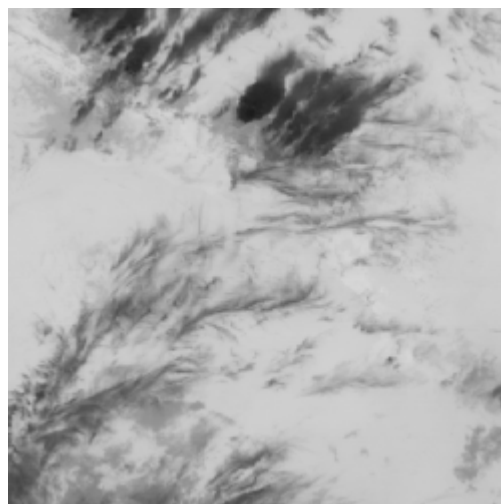
(a) Hasil klasifikasi ISMC



(b) Band 4



(b) Hasil klasifikasi FSMC



(c) Band 5

Gambar IV-1 Citra H (Nicaragua)

Gambar IV-2

Citra hasil klasifikasi ISMC dan FSMC

Ucapan terima kasih

Syukur Alhamdulillah, penulis ucapkan ke hadirat Allah SWT atas segala taufik dan hidayahNya, sehingga penelitian ini berhasil diselesaikan. Selanjutnya penulis menghaturkan banyak terima kasih kepada para peneliti yang nama-namanya tercantum dalam daftar referensi paper ini, dan juga kepada Ibu Dr. Aniasi Murni yang berkenan memberi saran dalam berdiskusi. Akhirnya, ucapan terima kasih juga penulis sampaikan kepada semua pihak yang membantu terlaksananya penelitian ini.

Referensi

1. R. O. Duda dan P. E. Hart, *Pattern Classification and Scene Analysis*, New York: Wiley, 1973.
2. J. A. Richards, *Remote Sensing Digital Image Analysis, An Introduction*, Springer-Verlag Berlin Heidelberg, 1986.
3. J. G. Moik, *Digital Processing for Remotely Sensed Images*, Washington, 1960.
4. J. J. Simpson, T. J. McIntire, M. Sienko, "An Improved Hybrid Clustering Algorithm for Natural Scenes", *IEEE Trans. on Geoscience and Remote Sensing*, vol. 38, no. 2, Maret 2000.
5. R. M. Haralick dan L. G. Saphiro, *Computer and Robot Vision*, Addison-Wesley, 1993, Vol. I dan II.
6. J. C. Bezdek, *Pattern Recognition with Fuzzy Objective Function Algorithms*, New York: Plenum, 1981.
7. Patrik B. G. Dammert, Jan I. H. Askne, dan S. Kuhlmann, "Unsupervised Segmentation of Multitemporal Interferometric SAR Images", *IEEE Trans. on Geoscience and Remote Sensing*, vol. 37, no. 5, September 1999.
8. T. M. Lillesand, R. W. Kiefer, *Remote Sensing and Image Interpretation*, John Wiley & Sons, 1994.
9. N. A. Mohamed, M. N. Ahmed and A. A. Farag, "Modified Fuzzy C-Mean in Medical Image Segmentation," *Proc. of IEEE-EMBS*, vol 20, part 3, halaman 1377-1380, 1998.
10. C. I. Chang, H. Ren, "An Experiment-Based Quantitative and Comparative Analysis of Target Detection and Image Classification Algorithms for Hyperspectral Imagery", *IEEE Trans. on Geoscience and Remote Sensing*, vol. 38, no. 2, Maret 2000.