IMPLEMENTASI PENGEMBANGAN METODE *DIFFERENTIAL EVOLUTION* UNTUK *CLUSTERING PIXEL*

Ahmad Saikhu, Hisyam Fahmi
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Surabaya
Email : saikhu@its-sby.edu, hisyam.fahm@gmail.com

ABSTRAK

Perkembangan metode komputasi telah mengalami percepatan yang luar biasa. Berbagai teknik komputasi untuk mendapatkan solusi dengan kinerja optimal terus berkembang. Sejumlah algoritma termasuk dalam rumpun Evolutionary Computation, diantaranya adalah Differential Evolution (DE) yang berhasil menyelesaikan masalah optimasi dalam berbagai bidang diantaranya masalah clustering. Keunggulan DE adalah karena implementasinya yang mudah dan kecepatan konvergensinya. Dalam clustering, DE menghadapi kendala penentuan jumlah cluster. Pada penelitian ini diimplementasikan sebuah algoritma Evolutionary Clustering (EC) yang merupakan pengembangan dari DE. EC diterapkan untuk melakukan pengelompokan pixel-pixel dari citra gray-scale atas beberapa area homogen yang berbeda satu dengan lainnya. EC tidak membutuhkan informasi awal tentang jumlah cluster yang akan terbentuk. EC menjadi salah satu solusi untuk menentukan jumlah cluster optimal dengan nilai validitas yang lebih baik. Kinerja dari EC akan dibandingkan dengan algoritma Fuzzy C-Means (FCM). Hasil dari EC dibanding FCM relatif sama dari segi nilai cluster validity index namun EC membutuhkan waktu relatif lebih singkat.

Kata Kunci: *Clustering pixel, Differential Evolution, Evolutionary Clustering, Fuzzy C Means.*

1. PENDAHULUAN

Perkembangan metode komputasi telah mengalami percepatan yang luar biasa. Berbagai teknik komputasi untuk memperoleh solusi yang optimal terus berkembang sebagai jawaban atas semakin banyaknya masalah optimasi nyata dalam kehidupan sehari-hari. Salah satu teknik komputasi diantaranya adalah *Evolutionary Computation*.

Ada beberapa algoritma yang termasuk dalam rumpun algoritma *evolutionary*, antara lain *Genetic Algorithm*, *Genetic Programming*, *Evolutionary Strategies*, *Differential Evolution*, *Evolutionary Programming*, dan *Grammatical Evolution*. Menurut Dasgupta dan Michalewicz, masih banyak lagi sistem *hybrid* yang menggabungkan berbagai fitur-fitur atau karakteristik yang dimiliki oleh algoritma-algoritma diatas, sehingga sulit untuk diklasifikasikan [1].

Salah satu algorima terbaik adalah DE yang dikenalkan oleh Storn dan Price pada tahun 1995. Beberapa alasan yang membuat algoritma ini dipilih adalah karena implementasinya yang mudah dan kecepatan konvergensinya. Algoritma ini sukses digunakan untuk menyelesaikan masalah optimasi dalam berbagai bidang, antara lain *clustering*, desain filter digital, optimasi fungsi linier, dan optimasi *multi-objective*. DE mengalami perkembangan yang cukup signifikan dengan berbagai varian sebagai usaha untuk meningkatkan performa dari algoritma ini [2].

Clustering merupakan proses membagi himpunan data yang tidak memiliki label dalam beberapa kelompok. Setiap kelompok (*cluster*) terdiri atas objek-objek yang mirip dan akan berbeda untuk anggota kelompok yang lain. Algoritma *clustering* dibagi 2, yaitu secara hirarki dan partisi [3].

Clustering secara hirarki dibedakan menjadi *agglomerative* dan *divisive*. *Agglomerative* berawal dari objek-objek yang individual dimana saat inisialisasi banyaknya *cluster* sama dengan banyaknya objek. Dua objek yang paling mirip dikelompokkan menjadi satu kelompok dan proses ini diulang sampai menghasilkan satu *cluster*.

Metode hirarki *divisive* merupakan kebalikan dari metode hirarki *agglomerative*. Hasil pembagian kelompok dengan metode hirarki ditunjukkan dalam bentuk diagram yang disebut *dendogram* [4]. Sedangkan *clustering* secara partisi, pemisahan obyek atas sekumpulan data dilakukan secara langsung menjadi beberapa kelompok yang berbeda. Data sebagai obyek dalam proses *clustering* dapat berupa teks maupun citra [4].

Pada penelitian ini diimplementasikan sebuah algoritma EC untuk melakukan pengelompokan *pixel* secara otomatis dari sebuah citra menjadi beberapa area homogen yang berbeda satu dengan lainnya. Pada algoritma ini tidak dibutuhkan informasi awal tentang jumlah *cluster* yang akan terbentuk.

Sebuah varian penyempurnaan dari algoritma DE akan digunakan untuk menentukan jumlah *cluster* yang terjadi secara alami dalam citra dan untuk memperbaiki pusat *cluster* [3]. Selain itu, akan dibandingkan kinerja ekstensif dengan metode lain,

yaitu algoritma *fuzzy c-means* klasik (FCM) melalui sejumlah citra *grayscale*, yaitu citra yang memiliki satu nilai intensitas untuk setiap elemen dalam ruang RGB. Perbandingan ini akan mememperlihatkan keunggulan algoritma EC dalam hal kecepatan (*speed*), ketepatan (*accuracy*) dan ketahanan (*robustness*).

2. METODE DIFFERENTIAL EVOLUTION

Algoritma DE klasik adalah sebuah algoritma optimasi global berbasis populasi yang menggunakan sebuah representasi *floating-point* (*real-coded*). Algoritma DE hampir sama dengan algoritma genetik. Pada DE juga dilakukan proses genetik seperti mutasi dan *crossover*.

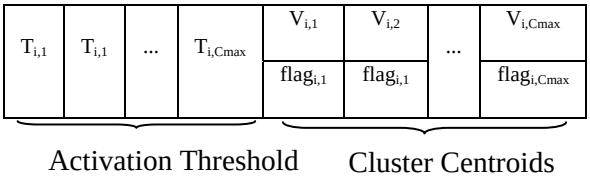
Pseudocode algoritma DE secara umum adalah sebagai berikut [5]:

```
Inisialisasi
Evaluasi
Repeat
   Mutasi
   Rekombinasi
   Evaluasi
   Seleksi
Until(kriteria berhenti
tercapai)
```

Di dalam DE, individu–individu adalah nilai riil yang merupakan nilai sebenarnya dari solusi yang dicari. Nilai riil ini selanjutnya disebut dengan istilah vektor.

2.1 Tahap Inisialisasi Kromosom

Inisialisasi kromosom merupakan tahap awal dari algoritma DE. Sebuah kromosom dinotasikan dengan matriks $\vec{Z}(t)$, yang berukuran 3×*C_max*. Matriks $\vec{Z}(t)$ berisikan variabel T_{ij} yang merupakan *activation threshold*, pusat *cluster* v_i, dan $flag_{ij}$ yang menyimpan kondisi pusat *cluster* aktif atau tidak. Inisialisasi dilakukan pada 3 komponen, yaitu inisialisasi *activation threshold*, inisialisasi pusat *cluster*, dan penentuan pusat *cluster* aktif. Representasi dari sebuah kromosom dapat dilihat pada Gambar 1.



Gambar 1. Representasi kromosom tunggal

Activation threshold merupakan variabel yang digunakan untuk menentukan sebuah pusat *cluster* aktif atau tidak. *Activation threshold* disimbolkan dengan $T_{i,j}$ dengan *i* adalah indeks dari kromosom dan *j* adalah indeks dari pusat *cluster*. Inisialisasi nilai dari *activation threshold* dilakukan dengan membangkitkan bilangan acak antara 0 sampai 1 sebanyak *c cluster* pada tiap kromosom.

Inisialisasi pusat *cluster* pada setiap kromosom dilakukan dengan cara memilih nilai intensitas secara acak antara 0 sampai 255. Pusat *cluster* akan diinisialisasi dengan sejumlah *c cluster* (*Cmax*) dalam satu kromosom. Setiap melakukan inisialisasi nilai pusat *cluster* yang ke-*j* akan dicek apakah nilai pusat *cluster* sudah pernah digunakan. Jika sudah ada, akan diinisialisasi kembali secara acak sehingga dalam satu kromosom tidak ada pusat *cluster* yang sama [3].

Setelah dilakukan inisialisasi, dilanjutkan dengan memilih pusat *cluster* yang aktif berdasarkan nilai *activation threshold*. Pemilihan pusat *cluster* yang aktif ini berdasarkan Persamaan (1).

$$IF \;\; T_{i,j} > 0.5 \;\; THEN \; flag_{i,j} = 1 \;\; ELSE \; flag_{i,j} = 0 \quad (1)$$

Jika nilai *activation threshold* bernilai lebih dari 0,5 maka pusat *cluster* akan aktif dan jika kurang dari 0,5 maka pusat *cluster* tidak aktif. Variabel $flag_{i,j}$ digunakan menyimpan nilai aktif dari pusat *cluster*, di mana $flag_{i,j}$ = 1 jika pusat *cluster* aktif dan $flag_{i,j}$ = 0 jika sebaliknya.

Jumlah pusat *cluster* yang aktif akan diperiksa apakah jumlahnya kurang dari dua atau tidak. Jika kurang dari dua maka dilakukan inisialisasi ulang *activation threshold*.

2.2 Penentuan Keanggotaan Tiap Pixel

Setelah diperoleh pusat *cluster* yang aktif, selanjutnya dihitung nilai jarak tiap *pixel* dari citra masukan terhadap masing-masing pusat *cluster* yang aktif. Nilai kesamaan antara satu *pixel* dengan *pixel* yang lain diambil dari nilai jarak ke pusat *cluster*. Perhitungan jarak menggunakan jarak *Euclidean*, sehingga tiap *pixel* akan dihitung jaraknya terhadap semua pusat *cluster* yang aktif.

Matriks jarak yang diperoleh, akan digunakan untuk menentukan keanggotaan dari *pixel*. Setiap *pixel* akan diberi label berdasarkan jarak *level* intensitasnya ke pusat *cluster* yang paling dekat. Misalnya, suatu *level* intensitas paling dekat dengan pusat *cluster* pertama dibandingkan dengan pusat *cluster* yang lain, maka semua *pixel* dengan *level* intensitas tersebut akan dimasukkan menjadi anggota *cluster* satu.

Hasil *clustering pixel* akan diperiksa jumlah anggota dari tiap *cluster*-nya. Jika ada *cluster* yang beranggotakan ≤ 2, maka akan dilakukan pembaruan terhadap label dari *pixel* pada *cluster* tersebut. Pembaruan label *cluster* dari *pixel* dilakukan dengan metode tetangga terdekat (*nearest neighbor*), yaitu

dengan memilih pusat *cluster* terdekat dengan *cluster* yang akan diperbarui [9].

2.3 Mutasi dan *Crossover*

Operasi mutasi genetik dan *crossover* merupakan proses yang terpenting dalam metode DE. Pada proses ini dihasilkan individu baru yang akan diseleksi apakah akan bertahan pada generasi selanjutnya.

Proses mutasi dilakukan untuk mendapatkan vektor donor $\vec{Y}(t)$. Pada proses mutasi, DE membangkitkan suatu vektor (kromosom) baru dengan melibatkan tiga gen dari kromosom induk. Pembangkitan vektor baru dilakukan dengan menambahkan selisih antara dua gen (gen ke-1 dan ke-2) kepada gen lainnya (gen ke-3). Ada dua skema mutasi yang diusulkan oleh *Kenneth* dan *Price* [1]. Untuk setiap vektor $Z_{i,j}(t), i = 0, 1, 2, \dots, Cmax - 1$, suatu vektor baru $\vec{Y}(t)$ dibangkitkan berdasarkan *Yi,jt=Zr3, jt+F.Zr1,jt−Zr2,jt* (2).

$$Y_{i,j}(t) = Z_{r3,j}(t) + F.\left(Z_{r1,j}(t) - Z_{r2,j}(t)\right) \quad (2)$$

di mana $r_1, r_2, r_3 \in [0, Cmax - 1]$ adalah bilangan integer yang berbeda satu sama lain dan *F>0*. Ketiga bilangan r_1, r_2, r_3 dipilih secara acak dalam interval $[0, Cmax - 1]$. Sedangkan *F* disebut dengan faktor skala yang berupa bilangan real dan merupakan konstanta yang mengontrol penguatan variasi diferensial $\left(Z_{r1,j}(t) - Z_{r2,j}(t)\right)$ [2]. Faktor skala ini lebih berkaitan dengan kecepatan konvergensi [5].

Setelah diperoleh vektor hasil proses mutasi, proses selanjutnya adalah *crossover* untuk mendapatkan vektor trial $\vec{R}(t)$. Saat ini dikenal dua metode *crossover* dalam DE yaitu binomial dan eksponensial. *Crossover* eksponensial adalah metode yang dikenalkan oleh *Kenneth* dan *Price* [1]. Namun *crossover* binomial justru lebih banyak digunakan dalam aplikasinya saat ini [6].

Dalam *crossover*, dikenal parameter *Cr* yang mempunyai peranan penting dalam algoritma DE. *Cr* lebih sensitif kepada kompleksitas masalah yang diselesaikan. Penentuan nilai *Cr* yang tepat akan menghasilkan performa DE yang bagus, namun sebaliknya pemilihan nilai *Cr* yang salah akan membawa DE ke dalam performa yang buruk [6]. Persamaan untuk melakukan binomial crossover ditunjukkan pada Persamaan (3).

$$R_{i,j}(t) = \begin{cases} Y_{i,j}(t), & if (rand(0..1) \le Cr) or\ j = rand(1..D) \\ Z_{i,j}(t), & if (rand(0..1) > Cr) and\ j \ne rand(1..D) \end{cases} \quad (3)$$

2.4 Seleksi Kromosom

Pada proses selanjutnya adalah seleksi kromosom. Seleksi kromosom bertujuan untuk memilih kromosom mana yang bertahan pada generasi selanjutnya dilihat dari nilai *fitness*-nya. Nilai *fitness* dari sebuah kromosom ditentukan dengan *fitness function*. *Fitness function* didefinisikan pada Persamaan (4).

$$f = \frac{1}{PS_i(c) + eps}, \quad (4)$$

dengan $PS_i(c)$ adalah nilai *cluster validity* PS *index* yang akan dijelaskan pada Persamaan (5).

Jika nilai *fitness* dari kromosom hasil *crossover* lebih besar atau sama dengan nilai *fitness* kromosom awal, maka kromosom baru hasil *crossover* akan menggantikan kromosom awal pada generasi berikutnya. Namun jika sebaliknya, kromosom awal akan bertahan pada generasi selanjutnya.

2.5 Tahap Perhitungan *Cluster Validity*

Cluster validity index merupakan fungsi statistik yang digunakan untuk mengevaluasi hasil *clustering* secara kuantitatif. Secara umum, *cluster validity index* bertujuan untuk menentukan apakah jumlah kelas yang terbentuk telah optimal [8]. Pada penelitian ini digunakan tiga macam *cluster validity index*, yaitu PS *index*, *Davies-Bouldin index*, dan *Dunn index*.

Formula PS *index* ditunjukkan pada Persamaan (5).

$$PS(c) = \frac{1}{n_c}\sum_{i=1}^{n_c}\left[\frac{1}{\|c_i\|}\sum_{j\in c_i}\frac{d_s(x_j, v_i)\times\ d(x_j, v_i)}{\min\limits_{\substack{m,n=1,2,\dots,n_c \\ and\ m\ne n}}\{d(v_m - v_n)\}}\right],$$

dimana

$$d_s(x_j, v_i) = \min_{\substack{k=1,2,\dots,\|c_i\| \\ and\ k\ne j}}\left\{\frac{\|(x_j - v_i) + (x_k - v_i)\|}{(\|x_j - v_i\| + \|x_k - v_i\|)}\right\}, \quad (5)$$

Semakin kecil nilai PS(c) mengindikasikan sebuah partisi optimal yang valid dengan jumlah *cluster* sebanyak *c*.

Davies-Bouldin index dihitung berdasarkan ukuran kesamaan dari *cluster* (R_{ij}) yang berbasis ukuran penyebaran anggota *cluster* pada sebuah *cluster* (s_i) dan ukuran ketidaksamaan *cluster* (d_{ij}). R_{ij} didefinisikan pada Persamaan (6).

$$R_{ij} = \frac{s_i + s_j}{d_{ij}}, \text{ dimana}$$
$$d_{ij} = d(v_i, v_j),\ s_i = \frac{1}{\|c_i\|}\sum_{x\in c_i} d(x, v_i). \quad (6)$$

Sehingga *Davies-Bouldin index* dapat diformulasikan sesuai persamaan (7).

$$DB = \frac{1}{n_c}\sum_{i=1}^{n_c} R_i, \text{dimana}$$
$$R_i = \max_{j=1\dots n_c, j\ne i}(R_{ij}),\ i = 1 \dots n_c. \quad (7)$$

Davies-Bouldin index mengukur rata-rata kemiripan antar *cluster* dan salah satu yang paling

mirip. Semakin rendah nilai *Davies-Bouldin index*, maka *cluster* tersebut semakin baik.

Sedangkan *Dunn index*, formulasinya persamaan (8).

$$D = \min_{i=1\ldots n_c}\left\{\min_{j=i+1\ldots n_c}\left(\frac{d(c_i,c_j)}{\max_{k=1\ldots n_c}(diam(c_k))}\right)\right\},$$

dengan

$$d(c_i,c_j) = \min_{x\in c_i, y\in c_j}\{d(x,y)\}$$

dan $diam(c_i) = \max_{x,y\in c_j}\{d(x,y)\}$. (8)

Jika *cluster* yang dihasilkan terpisah dengan baik, maka jarak antar *cluster* akan menjadi besar dan ukuran diameter dari *cluster* akan menjadi kecil. Nilai *Dunn index* yang besar berarti *cluster* yang dihasilkan adalah baik (9).

3. UJI COBA

Data uji coba pada penelitian ini adalah citra *grayscale*. Data masukan yang digunakan adalah *brain.gif*, *cameraman.gif*, *coins.gif*, dan *pepper.gif*. Keempat citra masukan tersebut berukuran 256×256 *pixel*.

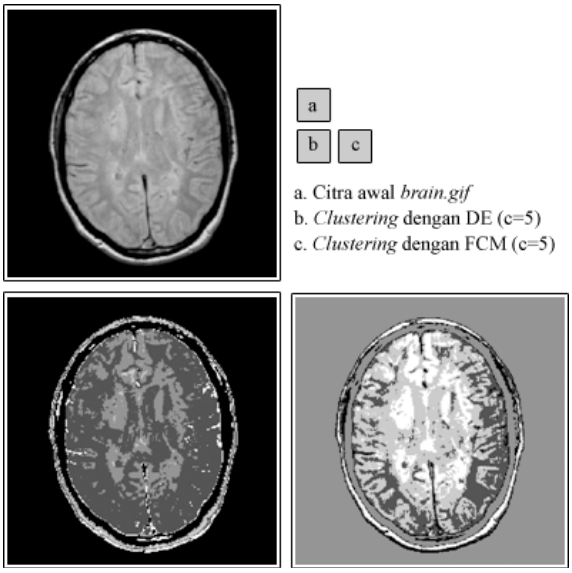
Uji coba dilakukan dengan 3 skenario, yaitu mengubah nilai parameter masukan *Cmax*, *Pop_size*, dan *t_max* pada setiap kali percobaan. Dari hasil uji coba akan dilakukan perbandingan nilai *cluster validity* dengan metode FCM. Perbandingan akan dilakukan pada empat data citra masukan.

Contoh hasil uji coba pada citra *brain.gif* dapat dilihat pada Gambar 2. Hasil uji coba *clustering* menggunakan EC akan dibandingkan dengan FCM. Nilai yang dibandingkan adalah *cluster validity index*. Sesuai persamaan 5, 7 dan 8, hasil clustering adalah optimal jika memiliki nilai *PS index* dan *Davies-Bouldin index* yang kecil dan nilai *Dunn index* yang besar.

Tabel 1. Perbandingan Nilai *Cluster Validity*

Citra	*Validity Index*	Rata-rata Nilai *Cluster Validity*	
		EC	FCM
brain.gif	PS-Index	0,0236*	0,0363
	Davies-Bouldin	0,2818*	0,5434
	Dunn	0,0113*	0,0102
cameraman.gif	PS-Index	0,0196*	0,0232
	Davies-Bouldin	0,2622*	0,5577
	Dunn	0,0131*	0,0112
coins.gif	PS-Index	0,0177*	0,0322
	Davies-Bouldin	0,7193	0,1838*
	Dunn	0,0127	0,0147*
pepper.gif	PS-Index	0,0187	0,0105*
	Davies-Bouldin	0,4987	0,4678*
	Dunn	0,0117	0,0157*

EC=Evolutionary Clustering, FCM=Fuzzy C-Means



a. Citra awal *brain.gif*
b. *Clustering* dengan DE (c=5)
c. *Clustering* dengan FCM (c=5)

Gambar 2. Contoh hasil *clustering* pada citra *brain.gif*

Pada Gambar 2. Contoh hasil *clustering* pada citra *brain.gif*

ditunjukkan perbandingan nilai *cluster validity* antara hasil *clustering* dengan EC dibanding FCM dengan 20 kali percobaan. Dapat dilihat bahwa *clustering* menggunakan EC adaah lebih baik pada citra *brain.gif* dan *cameraman.gif* dengan nilai *PS-Index* dan *Davies-Bouldin Index* yang lebih kecil dan *Dunn-Index* yang lebih besar untuk EC, yang ditandai dengan (*). Sedangkan metode FCM lebih unggul pada citra *coins.gif* dan *pepper.gif*,

Tabel 2. Perbandingan Jumlah Optimal *Cluster*

Citra	Jumlah Cluster Optimal	Rata-rata Jumlah *Cluster*	
		DE	FCM
1	**2**	**3**	**4**
brain.gif	3	4 22	8,11
cameraman.gif	3	4 92	8,96
coins.gif	3	4 33	9,18
pepper.gif	3	4 18	8,77

Pada Tabel 2 ditunjukkan bahwa percobaan dengan 3 skenario dan perulangan sebanyak 20 kali, didapatkan rata-rata jumlah *cluster* yang mendekati nilai optimal adalah hasil clustering menggunakan EC, sesuai kolom (3). Jumlah *cluster* optimal pada kolom (2) merupakan hasil *clustering* dengan nilai *PS-index* terkecil.

Pada Tabel 3 ditunjukkan waktu eksekusi yang dibutuhkan oleh masing-masing metode untuk melakukan *clustering*. Dapat dilihat bahwa EC membutuhkan waktu yang relatif lebih cepat untuk mendapatkan hasil *cluster* yang optimal.

Tabel 3. Perbandingan Waktu Proses Eksekusi

Citra	Rata-rata Waktu yang Dibutuhkan (detik)	
	DE	FCM
brain.gif	49,01	101,41
cameraman.gif	58,97	102,21
coins.gif	57,03	109,94
pepper.gif	58,19	111,27

4. EVALUASI HASIL UJI COBA

Evaluasi hasil uji coba dilakukan dengan menguji apakah nilai *cluster validity* dipengaruhi oleh citra masukan. Pengujian ini menggunakan *One-Way Multivariate Analysis of Variance* (MANOVA) karena melibatkan satu variabel tetap yaitu citra masukan dan tiga variabel terikat yaitu nilai PS *index Davies-Bouldin* dan *Dunn index*.

Dari **Error! Reference source not found.** diketahui bahwa nilai *cluster validity index* dipengaruhi jenis citra masukan di mana nilai pengujian *Wilks' Lambda* menghasilkan nilai signifikansi di bawah 0,05 ($p < 0.05$).

Melalui MANOVA, diperoleh bahwa *cluster validity* yang dipengaruhi oleh citra masukan adalah *Dunn index*, karena nilai signifikan < 0,05, sesuai hasil yang ditampilkan pada Tabel 5. Kemudian dilakukan uji *t* untuk mengetahui jenis citra apa saja yang berbeda nilai *cluster validity*-nya. Pada uji *t* dilakukan pengujian secara berpasangan, sehingga ada enam pasang pengujian pada masing-masing nilai *cluster validity*.

Tabel 4. Pengujian Multivariat

Efek	Nilai	F	df	Error df	Sig.
Pillai's	0.36	3.414	9	225	0.00
Wilks'	**0.661**	**3.662**	**9**	**177.81**	**0.00**
Hotelling's	0.48	3.822	9	215	0.00
Roy's	0.399	9.963[a]	3	75	0

Pada Tabel 6, ditunjukkan hasil uji *t* berpasangan berdasarkan nilai *Dunn index*. Dapat dilihat bahwa citra yang berpasangan dengan citra *pepper.gif* nilai selang kepercayaan (min-maks) tidak memuat nilai nol, sehingga disimpulkan bahwa citra *pepper.gif* berbeda nilai *Dunn index*-nya dibanding citra yang lain.

Selain dilakukan pengujian terhadap pengaruh citra masukan, juga dilakukan pengujian apakah nilai *cluster validity* dipengaruhi oleh parameter yang digunakan, yaitu *Cmax Pop_size* dan *t_max*. Pengujian dilakukan dengan analisis *Two-Way* MANOVA. Hasil uji MANOVA untuk pengaruh dari *Cmax* dan *Pop_size* ditunjukkan pada Tabel 7 sedangkan untuk parameter *t_max* ditunjukkan pada Tabel 8.

Tabel 5. Pengaruh Jenis Citra terhadap *Validity Index*

Sumber	Variabel Terikat	Sum of Squares	df	Mean Square	F	Sig.
Citra	PS	0.002	3	0.001	2.027	0.117
	DAVIES	0.95	3	0.317	1.402	0.249
	DUNN	**4.70E-05**	**3**	**1.60E-05**	**5.74**	**0**
Error	PS	0.022	75	0		
	DAVIES	16.943	75	0.226		
	DUNN	0	75	2.76E-06		
Total	PS	0.044	79			
	DAVIES	43.872	79			
	DUNN	0.008	79			

a. R Squared = ,075 (Adjusted R Squared = ,038)
b. R Squared = ,053 (Adjusted R Squared = ,015)
c. R Squared = ,187 (Adjusted R Squared = ,154)

Tabel 6. Uji *t* Pada Tiap Pasang Citra Masukan

Pasangan Citra		Selang Kepercayaan		
Citra 1	Citra 2	Mean	Min	Maks
Brain	*Cameraman*	0,0000	-0,0013	0,0014
Brain	*Coins*	-0,0005	-0,0018	0,0009
Brain	*Pepper*	-0,0019	-0,0033	-0,0005
Cameraman	*Coins*	-0,0005	-0,0019	0,0009
Cameraman	*Pepper*	-0,0020	-0,0033	-0,0006
coins	*pepper*	-0,0015	-0,0028	-0,0001

Dari Tabel 7 dan 8 dapat ditarik kesimpulan bahwa parameter yang berpengaruh terhadap nilai *cluster validity* adalah *Cmax*. Hal ini dapat dilihat pada nilai pengujian *Wilks' Lambda* pada efek *Cmax* bernilai 0 452 yang menghasilkan nilai signifikansi 0 036 (p<0 05). Sedangkan pada efek parameter yang lain niai pengujian *Wilks' Lambda* tidak signifikan.

5. KESIMPULAN

Dari uji coba dan analisis hasil, dapat ditarik sejumlah kesimpulan bahwa:

a. *Clustering pixel* secara otomatis pada citra *grayscale* dapat dilakukan menggunakan EC tanpa harus menentukan jumlah *cluster* yang akan terbentuk.
b. Hasil *clustering* menggunakan EC dibanding FCM relatif sama dari segi nilai *cluster validity index.*
c. Metode EC membutuhkan waktu relatif lebih singkat untuk menentukan jumlah *cluster* optimal secara otomatis dibanding FCM.

d. Parameter masukan dari metode EC yang paling berpengaruh terhadap nilai *cluster validity index* adalah parameter *Cmax*, yaitu jumlah maksimum dari *cluster*.

Tabel 7. Pengujian Efek *Cmax* dan *Pop_size*

Efek		Nilai	F	Hip df	Error df	Sig.
Cmax	Pillai's	0.57	2.257	6	34	0.061
	Wilks'	**0.45**	**2.6**	**6**	**32**	**0.04**
	Hotelling's	1.165	2.912	6	30	0.023
	Roy's	1.122	6.358[a]	3	17	0.004
PopSize	Pillai's	0.162	0.498	6	34	0.805
	Wilks'	0.84	0.486	6	32	0.813
	Hotelling's	0.189	0.473	6	30	0.823
	Roy's	0.179	1.016[a]	3	17	0.41
Cmax * PopSize	Pillai's	0.509	0.92	12	54	0.534
	Wilks'	0.544	0.919	12	42.624	0.536
	Hotelling's	0.744	0.909	12	44	0.546
	Roy's	0.598	2.691[a]	4	18	0.064

Tabel 8. Pengujian Multivariat Efek t_max

Efek		Nilai	F	Hip. Df	Error df	Sig.
t_max	Pillai's	0.21	0.904	6	46	0.5
	Wilks'	0.79	0.904	6	44	0.5
	Hotelling's	0.26	0.9	6	42	0.5
	Roy's	0.24	1.828[a]	3	23	0.17

6. DAFTAR PUSTAKA

[1] Babu B.V. dan Angira R. 2003 "Optimization of Water Pumping System Using Differential Evolution Strategies" Proceedings of the The Second International Conference on Computational Intelligence Robotics and Autonomous Systems (CIRAS-2003).Singapore 15-18 Desember 2003.

[2] Prince KV Storm RM Lampinen JA 2005. Differential Evolution - A Practical Approach to Global Optimization Natural Computing Science. Berlin : Springer.

[3] Das S. Abraham A. Konar A 2008 " Automatic Clustering Using an Improved Differential Evolution Algorithm" IEEE Transaction on Systems Man and Cybernetics Part A 38(1) 218–237.

[4] Hair Joseph F. Jr William C. Black Barry J. Babin Rolph E. Anderson 2010 Multivariate Data Analysis 7/e Prentice Hall.

[5] Qin A.K. Suganthan 2005 "Self-adaptive Differential Evolution Algorithm for Numerical Optimization" . Skotlandia : u.n. 2005. Proceedings of IEEE Congress on Evolutionary Computation (CEC2005) pp. 1785–1791 Scotland

[6] Zaharie D. 2007 "A Comparative Analysis of Crossover Variants in Differential Evolution" Proceedings of International Multi conference on Computer Science and Information Technology. pp. 171-181.

[7] Karaboga D. Okdem S. 2004 "A Simple and Global Optimization Algorithm for ngineering Problems: Differential Evolution Algorithm" Turkish Journal Electrical Engineering & Computer Science Vol. 12. 2004 12(1) 53–60

[8] *Kovács* F. Legány C. and Babos A. 2006 "Cluster Validity Measurement Techniques" Proceedings of 5th WSE AS International Conf. Artificial Intell. Know Engineering. Hungary: Budapest.

[9] Pramusinto Isnani 2010 "Implementasi Self-Adaptive Differential Evolution with Neighborhood Search (SANSDE) untuk Optomasi Sistem Pompa Air" Prosiding Seminar Nasional Aplikasi Teknologi Informasi 2010 (SNATI 2010) Yogyakarta 19 Mei 2010.