SEGMENTASI CITRA BERWARNA MENGGUNAKAN ALGORITMA JSEG

Rully Soelaiman[1)], Darlis Herumurti[2)], Dyah Wardhani Kusuma.[3)]
Fakultas Teknologi Informasi,
Institut Teknologi Sepuluh Nopember (ITS), Surabaya, 60111, Indonesia
E-mail : rully@its-sby.edu[1)], darlis@its-sby.edu[2)], dyah03@cs.its.ac.id[3)]

Abstrak

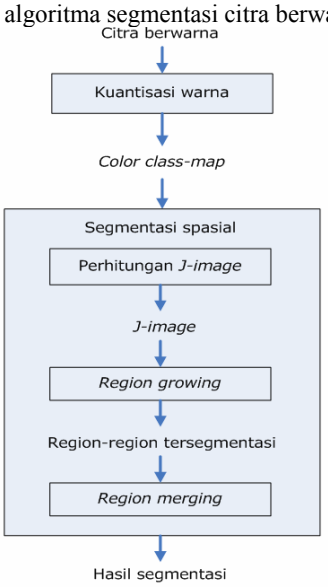
Segmentasi citra merupakan salah satu bagian penting dari pemrosesan citra, yang bertujuan untuk membagi citra menjadi beberapa region yang homogen berdasarkan kriteria kemiripan tertentu. Salah satu metode yang dapat digunakan untuk segmentasi citra berwarna adalah algoritma JSEG.
Algoritma JSEG terdiri dari dua tahap, yaitu kuantisasi warna dan segmentasi spasial. Pada tahap kuantisasi warna, warna-warna pada citra dikuantisasi menjadi kelas-kelas yang dapat digunakan untuk membedakan region-region dalam citra. Selanjutnya, warna piksel citra diganti dengan label class warna yang bersesuaian sehingga membentuk color class-map dari citra.
Pada tahap segmentasi spasial, dilakukan perhitungan ukuran segmentasi yang "baik" berdasarkan class-map yang telah terbentuk. Penerapan ukuran segmentasi ke window lokal dalam class-map menghasilkan "J-image", yang mana nilai tinggi dan nilai rendah berturut-turut bersesuaian dengan boundary region yang memungkinkan dan pusat region. Selanjutnya, metode region growing digunakan untuk mensegmentasi citra berdasarkan J-image multiskala.
Percobaan-percobaan membuktikan bahwa JSEG menghasilkan segmentasi citra yang relatif baik pada berbagai macam citra.

Kata kunci : segmentasi citra berwarna, JSEG

PENDAHULUAN

Segmentasi citra berwarna sangat bermanfaat dalam berbagai aplikasi. Dari hasil segmentasi citra, identifikasi region-region yang diinginkan dan objek-objek dalam citra dapat dilakukan, yang mana sangat bermanfaat untuk analisis citra.
Sulitnya masalah segmentasi disebabkan oleh tekstur citra. Jika citra hanya terdiri dari region-region warna yang homogen, metode *clustering* dalam ruang warna dapat diterapkan untuk segmentasi citra. Namun pada kenyataannya, citra pemandangan alam kaya akan warna dan tekstur sehingga sulit untuk mengidentifikasi region-region pada citra yang mengandung pola-pola warna-tekstur. Pendekatan yang digunakan dalam penelitian ini berdasarkan pada asumsi-asumsi sebagai berikut :

- Setiap region pada citra mengandung pola warna-tekstur yang terdistribusi *uniform*,
- Informasi warna pada setiap region citra dapat dinyatakan dengan beberapa warna-warna terkuantisasi, yang tepat untuk sebagian besar citra pemandangan alam berwarna,
- Warna-warna antara dua region tetangga dapat dibedakan. Hal ini merupakan asumsi dasar dari semua algoritma segmentasi citra berwarna.



Gambar 1. Skema algoritma JSEG

Algoritma JSEG memerlukan tiga parameter. Parameter pertama adalah threshold untuk proses kuantisasi warna. Threshold tersebut menentukan jarak minimum antara dua *cluster* warna yang akan digabungkan. Parameter lainnya adalah jumlah skala segmentasi dan threshold untuk *region merging*. Gambar 1 menunjukkan skema algoritma JSEG.

METODE

Kuantisasi Warna

Pertama, warna-warna pada citra dikuantisasi secara kasar tanpa menurunkan kualitas warna secara signifikan. Tujuannya adalah untuk mengekstrak beberapa perwakilan warna yang dapat digunakan untuk membedakan region-region dalam citra. Secara khusus, diperlukan 10 sampai 20 warna dalam citra pemandangan alam. Kuantisasi warna yang baik berpengaruh terhadap proses segmentasi. Dalam implementasi penelitian ini digunakan algoritma kuantisasi warna perseptual [2].

Algoritma kuantisasi warna perseptual bekerja berdasarkan persepsi penglihatan manusia yang lebih sensitif terhadap perubahan pada *smooth region* daripada perubahan pada *detailed region (textured region)*. *Smooth region* adalah *boundary* region-region yang memungkinkan, sedangkan *detailed region* adalah region-region homogen pada citra. Karena itu, warna-warna dapat dikuantisasi secara lebih kasar pada *detailed region* tanpa mempengaruhi kualitas perseptual secara signifikan. Berdasarkan fakta tersebut, setiap piksel ditandai dengan bobot yang berdasarkan pada *variance* dalam window lokal sedemikian rupa sehingga piksel-piksel pada *smooth region* lebih penting daripada piksel-piksel pada *detailed region*.

Algoritma ini menggunakan statistik lokal yang diperoleh setelah *peer group filtering*, yaitu bobot dalam proses kuantisasi vektor. Prosedur kuantisasi warna adalah sebagai berikut :

1. Konversi ruang warna RGB ke LUV untuk menjaga kualitas warna.
2. Pertama, *peer group filtering* diterapkan untuk menghaluskan citra dan menghilangkan *impulse noise*. Hasilnya berupa :
   - *x* : vektor piksel citra yang telah dihaluskan oleh anggota *peer group*-nya
   - *N* : jumlah *cluster* awal
   - *v* : bobot perseptual untuk setiap piksel
3. *Clustering* dengan *Generalized Lloyd Algorithm* (GLA). Hasilnya berupa centroid untuk setiap *cluster* warna.
4. Penggabungan *cluster-cluster* yang jarak centroidnya kurang dari threshold kuantisasi warna.
5. Klasifikasi piksel ke dalam *cluster* warna yang centroidnya terdekat dengan intensitas piksel tersebut.
6. Konversi ruang warna LUV ke RGB untuk menampilkan citra hasil kuantisasi warna.

Peer Group Filtering (PGF)

Misalkan $x_0(n)$ menyatakan vektor piksel citra yang memberikan ciri informasi warna pada posisi *n* yang berpusat pada window *w x w*. Urutkan semua piksel pada window tersebut berdasarkan jaraknya terhadap $x_0(n)$ dalam urutan *ascending* dan nyatakan sebagai $x_i(n)$, $i = 0, ..., k = w^2 - 1$. Ukuran jarak Euclidean yang digunakan adalah :

$$d_i(n) = \left\| x_0(n) - x_i(n) \right\|, i = 0,...,k \quad (1)$$

$$d_0(n) \le d_1(n) \le ... \le d_k(n) \quad (2)$$

Peer group P(m,w) untuk $x_0(n)$ terdiri dari *m* piksel yang intensitasnya terdekat dengan $x_0(n)$ dalam window *w x w* yang berpusat pada $x_0(n)$. Penentuan *m* menggunakan estimasi diskriminan Fisher. *Criterion* yang dimaksimalkan adalah :

$$J(i) = \frac{|a_1(i) - a_2(i)|^2}{{s_1}^2(i) + {s_2}^2(i)}, i = 1,...,k \quad (3)$$

di mana :

$$a_1(i) = \frac{1}{i}\sum_{j=0}^{i-1} d_j(n),\ a_2(i) = \frac{1}{k+1-i}\sum_{j=i}^{k} d_j(n) \quad (4)$$

$${s_1}^2(i) = \sum_{j=0}^{i-1} \left| d_j(n) - a_1(i) \right|^2,\ {s_2}^2(i) = \sum_{j=i}^{k} \left| d_j(n) - a_2(i) \right|^2 \quad (5)$$

m adalah indeks di mana *J(i)* bernilai maksimum. Selanjutnya $x_0(n)$ diganti dengan rata-rata anggota *peer group*-nya.

Untuk menghilangkan efek *impulse noise*, turunan pertama dari jarak $d_i(n)$, $f_i(n)$, dihitung sebelum klasifikasi *peer group* :

$$f_i(n) = d_{i+1}(n) - d_i(n) \quad (6)$$

Pengujian dilakukan terhadap *M* titik pertama dan terakhir dari $x_i(n)$ untuk memeriksa apakah titik-titik tersebut termasuk *impulse noise* :

$$f_i(n) \le \alpha \quad (7)$$

di mana $M = w / 2$, separuh dari ukuran window, dan α diset bernilai tinggi untuk citra yang sangat rusak dan diset bernilai rendah untuk citra yang sedikit rusak. Jika $f_i(n)$ tidak memenuhi kondisi tersebut, maka titik-titik terakhir $x_j(n)$ untuk $j \le i$ atau $j > i$ dianggap sebagai *impulse noise* dan dihilangkan. Kemudian $d_j(n)$ sisanya digunakan untuk mengestimasi *peer group* yang sebenarnya.

Selanjutnya dilakukan perhitungan jarak maksimum setiap *peer group T(n)*. Nilai *T(n)* mengindikasikan kehalusan region lokal. Bobot perseptual untuk setiap piksel *v(n)* dihitung dengan :

$$v(n) = exp(-T(n)) \quad (8)$$

sehingga piksel-piksel pada *detailed region* memiliki bobot yang lebih rendah daripada piksel-piksel pada *smooth region*.

Rata-rata $T(n)$, T_{avg}, mengindikasikan kehalusan keseluruhan citra. Secara umum, semakin besar nilai T_{avg}, semakin berkurang kehalusan citra dan semakin banyak jumlah *cluster* yang diperlukan untuk kuantisasi warna. Jumlah awal *cluster* N diestimasi dengan :

$$N = \beta T_{avg} \quad (9)$$

di mana β diset bernilai 2 pada percobaan.

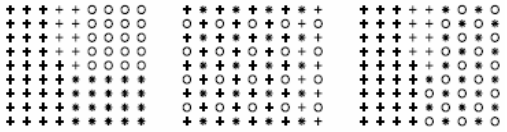
Generalized Lloyd Algorithm (GLA)

GLA digunakan untuk menandai piksel-piksel yang berbobot rendah dengan *cluster* yang lebih sedikit untuk mengurangi jumlah *cluster* warna pada *detailed region*. Centroid untuk *cluster* warna C_i dihitung dengan :

$$c_i = \frac{\sum v(n)x(n)}{\sum v(n)}, \; x(n) \in C_i \quad (10)$$

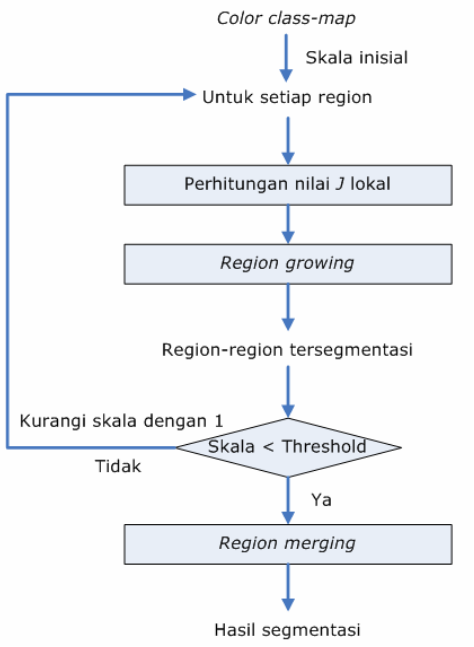
Segmentasi Spasial

Setelah tahap kuantisasi warna, diperoleh *color class-map*, yaitu label yang ditandai pada piksel-piksel dalam citra yang menyatakan klasifikasi *cluster* warna untuk piksel tersebut. Gambar 2 menunjukkan contoh *color class-map*. Nilai label diwakili oleh tiga simbol, '*', '+', dan 'o'.



Gambar 2. Contoh color class-map

Gambar 3 menunjukkan *flow chart* tahap segmentasi spasial. Mula-mula, citra input dianggap sebagai satu region inisial. Algoritma ini kemudian mensegmentasi semua region dalam citra pada skala inisial yang besar. Proses tersebut diulang pada region-region tersegmentasi baru pada skala berikutnya yang lebih kecil hingga mencapai skala minimum yang telah ditentukan.



Gambar 3. Flow chart tahap segmentasi spasial

Tabel 1 berisi himpunan skala dan ukuran region yang sesuai untuk skala tersebut. Misalnya, jika ukuran citra lebih besar daripada 256 x 256, tetapi lebih kecil daripada 512 x 512, skala awalnya adalah 3.

Perhitungan J-Image

J-image adalah citra *grayscale* yang nilai piksel-pikselnya adalah nilai J yang dihitung terhadap window lokal yang berpusat pada piksel tersebut.

$$J = \frac{S_B}{S_W} = \frac{(S_T - S_W)}{S_W} \quad (11)$$

di mana :

$$S_T = \sum_{z \in Z} \|z - m\|^2 \quad (12)$$

dan

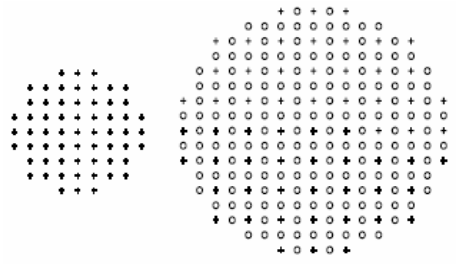
$$S_W = \sum_{i=1}^{C} S_i = \sum_{i=1}^{C} \sum_{z \in Z_i} \|z - m_i\|^2 \quad (13)$$

Keterangan :
S_T : jarak antar kelas-kelas warna yang berbeda (*between class scatter matrix*)
S_W : jarak antar anggota dalam setiap kelas warna (*within class scatter matrix*)
m : rata-rata lokasi spasial piksel dalam window
m_i : rata-rata lokasi spasial piksel dalam kelas warna i

Tabel 1. Ukuran window pada skala yang berbeda

Skala	Window (piksel)	Sampling (1/piksel)	Ukuran Region (piksel)	Valley Minimum (piksel)
1	9 x 9	1 / (1 x 1)	64 x 64	32
2	17 x 17	1 / (2 x 2)	128 x 128	128
3	33 x 33	1 / (4 x 4)	256 x 256	512
4	65 x 65	1 / (8 x 8)	512 x 512	2048

Semakin besar nilai *J*, maka piksel tersebut semakin mendekati *boundary* region. *J-image* dapat dilihat sebagai peta yang mengandung *valley* dan *mountain* yang berturut-turut mewakili pusat region dan *boundary* region.



(a) (b)

Gambar 4. Window untuk perhitungan nilai J lokal (a) Window dasar pada skala 1. (b) Ilustrasi window pada skala 2. Hanya titik-titik bertanda '+' yang digunakan untuk perhitungan nilai J lokal, sehingga membentuk window dasar yang sama dengan (a).

Window yang digunakan dalam perhitungan *J-image* berbeda-beda dalam setiap skala segmen-tasi, seperti pada Gambar 4. Tepian window dihilangkan untuk membuat window lebih sirkuler sehingga tidak menimbulkan bias terhadap objek segi empat.

Region Growing

Region growing digunakan untuk mengelom-pokkan piksel-piksel ke dalam region-region. Hasilnya berupa *region-map*, yaitu label yang ditandai pada piksel-piksel dalam citra yang merupakan klasifikasi region untuk piksel tersebut. *Region growing* terdiri dari dua tahap, yaitu :

- Valley determination

Valley determination digunakan untuk menentukan himpunan *valley* terbaik. Piksel-piksel dengan nilai *J* kurang dari threshold dihubungkan untuk membentuk kandidat *valley*. Kandidat *valley* yang ukurannya melebihi ukuran *valley* minimum pada skala segmentasi yang bersesuaian (seperti pada Tabel 1) akan menjadi *valley*.

- Valley growing

Valley growing merupakan proses pembentukan region-region dari *valley-valley*.

Region Merging

Region merging digunakan untuk mengga-bungkan region-region yang jaraknya kurang dari threshold *region merging*. Mula-mula, dilakukan perhitungan jarak antara dua region tetangga dan hasilnya disimpan dalam tabel jarak. Kemudian pasangan region dengan jarak minimum digabungkan. Vektor fitur warna untuk region tersebut dihitung dan tabel jarak di*update*. Proses tersebut berlanjut hingga mencapai threshold maksimum untuk jarak. Setelah *region merging*, diperoleh hasil segmentasi.

HASIL UJI COBA

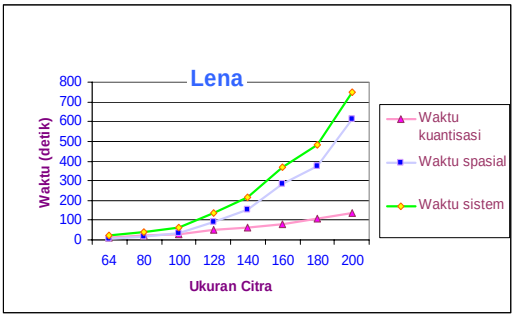
Uji coba ini meliputi uji coba kinerja. Semua eksperiman dilakukan pada sebuah PC dengan prosesor Intel Pentium IV 2,44 GHz dengan memori sebesar 512 MB. Sistem operasi yang digunakan adalah Windows XP Professional.

Citra yang digunakan untuk uji coba berformat jpg, bmp, ppm, dan tif. Untuk efisiensi komputasi ukuran citra dibatasi minimal 64 x 64 piksel dan maksimal 200 x 200 piksel. Implementasi program menggunakan Matlab 7.0.

Hasil segmentasi pada Lena.tif ditunjukkan pada Gambar 9. Parameter threshold kuantisasi warna diset bernilai 255, jumlah skala 2, dan threshold *region merging* bernilai 0,4. Sedangkan hasil segmentasi untuk citra-citra lain dengan jumlah skala 1 dapat dilihat pada Gambar 10.

Uji Coba dengan Variasi Ukuran Citra

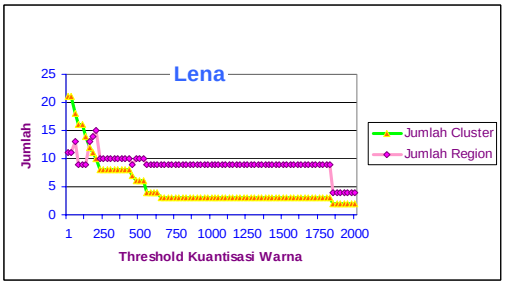
Grafik perbandingan waktu komputasi dengan variasi ukuran citra pada Lena.tif ditunjukkan pada Gambar 5. Dari pengamatan pada gambar, semakin besar ukuran citra, maka waktu komputasi juga semakin besar karena jumlah piksel dalam citra semakin banyak.



Gambar 5. Grafik perbandingan waktu komputasi dengan variasi ukuran citra pada Lena.tif

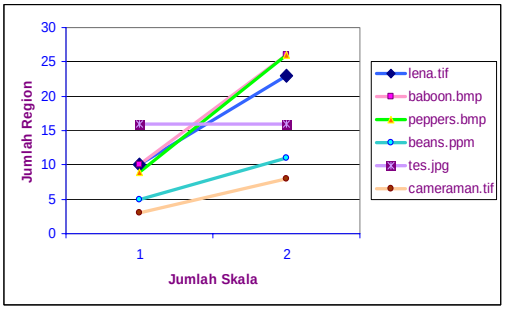
Uji Coba dengan Variasi Threshold Kuantisasi Warna

Grafik perbandingan jumlah *cluster* warna dan jumlah region dengan variasi threshold kuantisasi warna pada Lena.tif ditunjukkan pada Gambar 6.



Gambar 6. Grafik perbandingan jumlah cluster dan jumlah region dengan variasi threshold kuantisasi warna pada Lena.tif

Dari pengamatan pada gambar, semakin besar threshold kuantisasi warna, maka jumlah *cluster* warna semakin berkurang karena semakin banyak *cluster* warna yang digabungkan. Namun jumlah region yang terbentuk tidak selalu berkurang mengikuti jumlah *cluster*.



Gambar 7. Grafik perbandingan jumlah region dengan variasi jumlah skala segmentasi

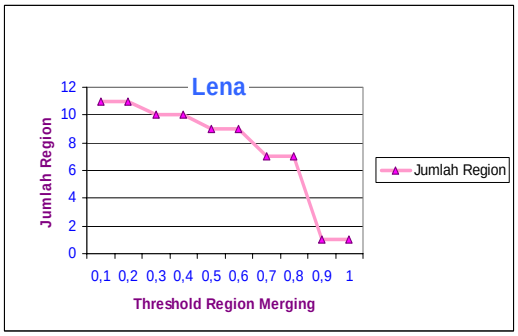
Uji Coba dengan Variasi Jumlah Skala

Grafik perbandingan jumlah region pada berbagai citra dengan variasi jumlah skala ditunjukkan pada Gambar 7.

Data pada grafik menunjukkan bahwa semakin besar jumlah skala, maka hasil segmentasi semakin detail. Hal ini ditunjukkan dengan semakin banyaknya jumlah region yang terbentuk.

Uji Coba dengan Variasi Threshold Region Merging

Grafik perbandingan jumlah region dengan variasi threshold *region merging* pada Lena.tif ditunjukkan pada Gambar 8.



Gambar 8. Grafik perbandingan jumlah region dengan variasi threshold region merging pada Lena.tif

Data pada grafik menunjukkan bahwa semakin besar threshold *region merging*, maka jumlah region yang terbentuk semakin sedikit karena semakin banyak region yang digabungkan.

SIMPULAN

Dari hasil uji coba yang dilakukan, didapatkan beberapa simpulan sebagai berikut:

1. Algoritma JSEG berhasil diimplementasikan untuk segmentasi citra berwarna.
2. Semakin besar ukuran citra input yang akan diproses, maka waktu komputasi sistem yang diperlukan juga semakin besar.
3. Nilai parameter threshold kuantisasi warna mempengaruhi hasil segmentasi dan waktu komputasi sistem. Semakin besar threshold kuantisasi warna, maka jumlah *cluster* warna yang terbentuk semakin sedikit karena semakin banyak *cluster* warna yang digabungkan. Waktu komputasi sistem juga semakin kecil karena semakin sedikit *cluster* warna yang diproses.
4. Nilai parameter jumlah skala mempengaruhi hasil segmentasi dan waktu komputasi sistem. Semakin besar jumlah skala, hasil segmentasi semakin detail karena semakin banyak region yang terbentuk. Waktu komputasi sistem juga semakin besar karena proses semakin spasial diulangi sebanyak jumlah skala yang dipilih oleh pengguna.
5. Nilai parameter threshold *region merging* mempengaruhi hasil segmentasi dan waktu komputasi sistem. Semakin besar threshold *region merging*, maka semakin sedikit jumlah region yang terbentuk. Waktu komputasi sistem juga semakin kecil karena semakin sedikit region yang diproses.
6. Pengembangan yang mungkin dilakukan adalah pencarian algoritma optimasi untuk mengurangi waktu komputasi algoritma JSEG. Selain itu, juga dilakukan peningkatan kegunaan dari algoritma JSEG dengan membangun perangkat lunak untuk keperluan *image retrieval* atau *object recognition* yang memanfaatkan hasil segmentasi dari algoritma JSEG.

DAFTAR PUSTAKA

[1] Y. Deng, B. S. Manjunath, dan H. Shin. "Color Image Segmentation", In : *Proceedings of IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR '99,* Fort Collins, CO, vol. 2, pp. 446-51, Juni 1999.

[2] Y. Deng, C. Kenney, M. S. Moore, dan B. S. Manjunath, "Peer Group Filtering and Perceptual Color Image Quantization", *to appear in Proc. of ISCAS*, 1999.

[3] Y. Deng, B. S. Manjunath, "Unsupervised Segmentation of Color-Texture Regions in Images and Video", *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI '01)*, vol. 23, no. 8, pp. 800-810, Agustus 2001.

[4] C. Kenney, Y. Deng, B. S. Manjunath, dan G. Hewer, "Peer Group Image Enhancement", *IEEE Transactions on Image Processing*, vol. 10, no. 2, Februari 2001.

[5] R. O. Duda dan P. E. Hart, *Pattern Classification and Scene Analysis*, John Wiley & Sons, 1970.

[6] Linear discriminant analysis – Wikipedia, the free encyclopedia, <URL: http://en.wikipedia.org/wiki/Fisher_linear_discriminant>