Penerapan Kohonen Self Organized Map Dalam Kuantisasi Vektor Pada Kompresi Citra Bitmap 24 Bit

Gadis Fransiska Yulianti Tae, Sri Suwarno, Widi Hapsari
Fakultas Teknologi Informasi, Program Studi Teknik Informatika
Universitas Kristen Duta Wacana Yogyakarta
Email: 22064169@students.ukdw.ac.id, sswn@ukdw.ac.id, widi@ukdw.ac.id

Abstrak :

Dalam penyimpanannya, citra bitmap merupakan salah satu format citra yang membutuhkan ruang penyimpanan yang besar. Untuk mengatasi hal tersebut, dilakukan kompresi dengan berbagai metode. Diantaranya adalah metode kuantisasi vektor. Dalam metode kuantisasi vektor sendiri, terdapat berbagai algoritma yang dapat diterapkan. Algoritma dalam jaringan saraf tiruan yaitu Kohonen self-organized , merupakan salah satu algoritma yang dapat diterapkan dalam metode tersebut. Algoritma ini bekerja dengan cara mencari kedekatan nilai warna pada citra dengan nilai-nilai pada *codebook*, dan memperbaharui nilai-nilai pada *codebook* tersebut sehingga lebih mendekati nilai warna pada citra. Dari penelitian yang dilakukan, diperoleh kesimpulan keterhubungan antara variasi warna dan dimensi citra serta ukuran blok dengan nilai rasio kompresi yang dicapai, lama waktu kompresi dan dekompresi, serta nilai *PSNR* yang diperoleh. Disarankan pula pada penelitian selanjutnya, *codebook* yang diinisialisasikan lebih mendekati warna pada masing-masing citra sehingga dapat memperoleh kualitas hasil dekompresi yang lebih baik, yaitu lebih mirip dengan citra aslinya.

Kata Kunci : *Kohonen Self-Organized Map, Kuantisasi Vektor, Kompresi Citra*

1. Pendahuluan

Kompresi citra merupakan salah satu bidang dalam pengolahan citra digital yang bertujuan mengurangi penggunaan memori dalam penyimpanan dan pengiriman citra digital sehingga menjadi lebih singkat dibandingkan citra yang tidak terkompresi.

Terdapat dua tipe kompresi, yaitu kompresi tipe *lossless* dan tipe *lossy*. Kompresi tipe *lossless* adalah kompresi dimana kualitas citra hasil kompresi tidak menurun setelah proses kompresi terjadi karena pada saat kompresi, tidak satu pun informasi citra dihilangkan. Sedangkan kompresi tipe *lossy* akan menghasilkan kualitas citra yang dihasilkan jauh lebih

rendah daripada kualitas citra aslinya. Pada kompresi tipe ini, dilakukan proses kuantisasi vektor untuk mengurangi jumlah intensitas warna sehingga dapat mengurangi jumlah bit yang digunakan untuk merepresentasi citra, dimana pengurangan intensitas warna ini dilakukan dengan melakukan pengelompokkan warna pada citra yang dikompresi.

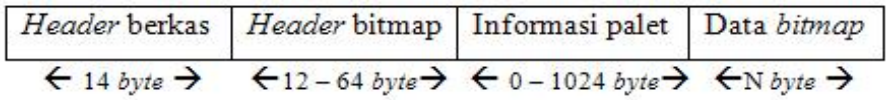
Salah satu metode dalam jaringan saraf tiruan yang dapat digunakan untuk pengelompokkan data adalah Kohonen *self-organized map*. Metode ini dapat pula diterapkan untuk mengelompokkan setiap warna pada citra dalam proses kuantisasi vektor guna mengurangi intensitas warna. Pada penelitian ini, dilakukan kuantisasi vektor pada citra digital dengan metode Kohonen *self-organized map*.

Beberapa hal yang menjadi focus pada penelitian in, antara lain: Bagaimana penerapan metode Kohonen self-organized dalam kuantisasi vektor pada kompresi citra digital?; Seberapa besar rasio kompresi yang dapat diperoleh dengan metode Kohonen *self-organized map*?; Dan seberapa besar kemiripan antara citra asli dengan citra hasil dekompresi yang dapat diperoleh dengan metode Kohonen *self-organized map*?

2. Landasan Teori

2.1 Format Citra Bitmap (.BMP)

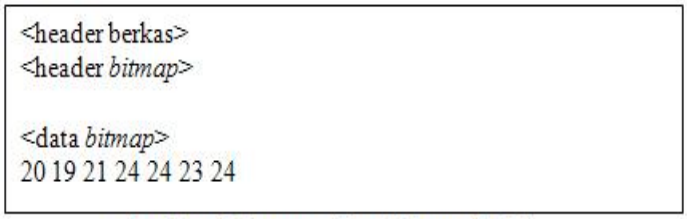
Format citra ini mendukung citra dengan jumlah bit per piksel sebanyak 1, 4, 8,16, 24, dan 32. Struktur citra *bitmap* sangat sederhana seperti tampak pada Gambar 1.



Gambar 1 Struktur File *Bitmap*

Dikutip dari: Munir, R. (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Bandung: Penerbit Informatika, hlm. 39.

"Pada file citra 24-bit tidak mempunyai palet RGB, karena nilai RGB langsung diuraikan dalam data *bitmap*." (Munir, 2004 : 42). Gambar 2 adalah contoh format citra *bitmap* 24-bit.



Gambar 2 Format Citra *Bitmap* 24-bit

Dikutip dari: Munir, R. (2004). *Pengolahan Citra Digital dengan Pendekatan Algoritmik*. Bandung: Penerbit Informatika, hlm. 42

2.2 Kompresi Citra

"Pada dasarnya, semakin besar ukuran citra maka semakin besar pula memori yang dibutuhkan untuk penyimpanan. Pada sisi lain, kebanyakan citra mengandung duplikasi data. Duplikasi data pada citra dapat berarti bahwa besar kemungkinan suatu piksel dengan piksel tetangganya memiliki intensitas yang sama, sehingga penyimpanan setiap piksel memboroskan tempat. Kompresi citra sendiri bertujuan meminimalkan kebutuhan memori untuk merepresentasikan citra digital. Prinsip umum yang digunakan pada proses kompresi citra adalah mengurangi duplikasi data di dalam citra sehingga memori yang dibutuhkan untuk merepresentasikan citra menjadi lebih sedikit daripada representasi citra semula."(Munir, 2004 : 165)

Terdapat beberapa pendekatan dalam kompresi citra, salah satunya adalah pendekatan kuantisasi dimana kompresi dilakukan dengan mengurangi jumlah derajat keabuan yang tersedia pendekatan ini dibagi lagi menjadi 2, yaitu kuantisasi skalar dan kuantisasi vektor.

2.3 Kuantisasi Vektor

Dalam bukunya, David Salomon (2000) mengatakan bahwa pada kuantisasi vektor, *encoder* akan membuat sebuah daftar (dikenal dengan nama *codebook*) vektor dan mengkompres setiap blok dengan menuliskan pointer ke blok dalam *codebook*. Sedangkan *decoder*, akan membaca pointer-pointer, mengikuti setiap pointer ke blok dalam *codebook*, dan menggabungkan blok ke citra. Kuantisasi vektor merupakan sebuah metode kompresi asimetri yaitu waktu yang diperlukan untuk dekompresi lebih lama dari pada waktu untuk mengkompresi. Pada proses *encoding*, pembuatan *codebook* dilakukan dengan membaca blok demi blok. Jika blok sudah berada dalam *codebook*, maka *encoder* menghasilkan sebuah pointer ke blok dalam codebook. Jika blok tidak ada dalam codebook, maka blok tersebut ditambahkan ke dalam *codebook* dan sebuah pointer dihasilkan. Masalah dalam metode sederhana ini adalah metode ini tidak efektif karena *codebook* akan bertambah selama proses kompresi. Setiap data dalam *codebook* adalah sebuah blok. Untuk mengkompres sebuah citra, baca blok demi blok dan untuk setiap blok temukan data *codebook* yang cocok dengannya dan berikan pointer ke data tersebut.

2.4 Kohonen Self-Organized Map

Laurene menjelaskan proses Kohonen self-organizing dilakukan dengan algoritma berikut :

step 0. Inisialisasi bobot w_{ij}, radius tetangga, dan learning rate (α).

step 1. Selama kondisi stop bernilai false, lakukan step 2 s/d 8

step 2. Untuk setiap input vektor x, lakukan step 3 s/d 5

$$D(j) = \sum_i (w_{ij} - x_i)^2$$

step 3. Untuk setiap j hitung :

step 4. Temukan index j yang nilai D(j)-nya terkecil

step 5. Update semua bobot yang menuju indeks j dengan rumus:

$$w_{ij}(baru) = w_{ij}(lama) + \alpha[x_i - w_{ij}(lama)]$$

step 6. Update learning rate (α).

step 7. Kurangi radius tetangga

step 8. Cek kondisi stop.

2.5 Pengukuran Hasil Kompresi dan Dekompresi

"Adapun pengukuran kompresi dapat dilakukan dengan menghitung rasio kompresi citra tersebut dengan menggunakan rumus" (R. Munir, 2004 : 169) :

$$rasio\_kompresi = \left(1 - \frac{hasil\_kompresi}{data\_asli}\right) x100$$

"Kualitas hasil dekompresi citra pun dapat diukur secara kuantitatif dengan menggunakan besaran *PSNR* (*peak signal to noise ratio*) dan memiliki satuan *decibel* (dB)." (Munir, 2004 : 167)

$$PSNR = 20 * \log_{10}\left(\frac{b}{RMS}\right)$$

dimana,

$$RMS = \sqrt{\frac{1}{Lebar * Tinggi} \sum_{i=1}^{N} \sum_{j=1}^{M} \left(f_{ij} - f'_{ij}\right)^2}$$

3. Implementasi Sistem

Berikut ini adalah langkah-langkah dalam algoritma Kohonen Self Organized Map untuk mengkuantisasi setiap blok :

0. Inisialisasi kelompok warna ( ), radius tetangga, dan learning rate (α).
1. Selama perulangan lebih kecil dari 100 atau perubahan lebih kecil dari 10, lakukan step 2 s/d 8
2. Untuk setiap blok citra, lakukan step 3 s/d 5

3. Untuk setiap kelompok, hitung jarak (*distance euclidean*) terhadap setiap blok dengan rumus:

$$D = \sum_i (w - x_i)^2$$

4. Temukan index j yang nilai D(j)-nya terkecil
5. Update kelompok warna yang memiliki indeks kolom j dengan rumus:

$$w_{ij}(baru) = w_{ij}(lama) + \alpha[x_i - w_{ij}(lama)]$$

6. Update learning rate (α) dengan menggunakan rumus :

$$learning\_rate = 0.5 * learning\_rate$$

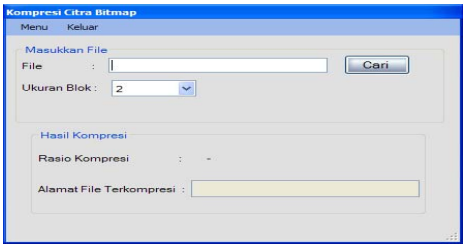
7. Kurangi radius tetangga dengan menggunakan rumus : $radius = radius - 1$
8. Tambahkan perulangan dengan rumus: $perulangan = perulangan + 1$

Dan hitung besar perubahan *codebook* yang terjadi.

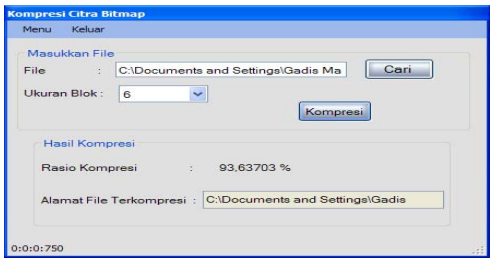
4. Hasil dan Analisis

4.1 Kompresi

Gambar 3 adalah tampilan utama *form* Kompresi



Gambar 3. Tampilan *Form* Kompresi



Gambar 4. Tampilan Hasil Kompresi

Untuk uji coba, citra bitmap 24 bit yang digunakan adalah seperti pada Gambar 5

Gambar 5. Citra Uji Coba

Hasilnya :

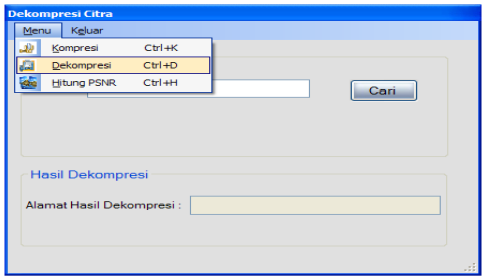
Tabel 1. Hasil Kompresi

Ukuran file (*byte*)	**Ukuran blok**	**Lama Kompresi (menit)**	**Ukuran Hasil (*byte*)**	**Rasio Kompresi (%)**
90.054	2x2	0:13:078	56.337	37,44087
	4x4	0:2:859	13.430	85,08673
	6x6	0:1:625	5.651	93,72488
	8x8	0:0:844	3.142	96,51098
	16x16	0:3:890	848	99,05834

Hasil dari proses ini adalah sebuah *file* .gds yang berisi format *file*, ukuran blok yang digunakan, dimensi citra asli, dan *codebook* yang dihasilkan dari proses kuantisasi vektor serta rasio kompresi yang dilakukan. Berdasarkan pada uji coba yang dilakukan, didapat kesimpulan bahwa semakin besar ukuran blok yang digunakan, maka rasio kompresi akan semakin besar, semakin variasi warna yang dimiliki citra, maka rasio kompresinya akan semakin kecil, dan pada kasus citra dengan 1 variasi warna, semakin besar dimensi citra, maka rasio kompresinya akan semakin kecil.

4.2 Dekompresi

Gambar 6 adalah tampilan awal *form* dekompresi



Gambar 6 Tampilan Awal Dekompresi

Hasil yang diberikan oleh proses ini adalah citra dengan format .bmp 24 bit dan lama waktu dekompresi. Hasilnya proses dekompresi seperti pada Tabel 2.

Tabel 2. Hasil Dekompresi

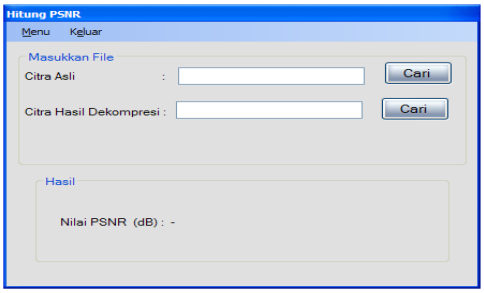
Nama *File*	Lama Kompresi (menit)	Lama Dekompresi (menit)	Ukuran File Hasil Dekompresi (*byte*)
2.gds	0:13:078	1:20:234	90.054
4.gds	0:2:859	0:5:875	90.054
6.gds	0:1:625	0:1:375	90.054
8.gds	0:0:844	0:0:563	90.054
16.gds	0:3:890	0:0:156	90.054

Berdasarkan uji coba yang dilakukan, diperoleh bahwa semakin besar ukuran blok, maka waktu kompresi maupun dekompresi akan semakin cepat, semakin besar dimensi citra, waktu kompresi dan dekompresi semakin lama, begitu pula jika semakin banyak variasi warna yang dimiliki citra, maka waktu kompresi dan dekompresi pun akan semakin lama. Gambar 7 adalah citra hasil dekompresi dengan menggunakan ukuran blok 2x2.

Gambar 7. Citra Hasil Dekompresi

4.3 Hitung PSNR

Gambar 8 adalah tampilan awal *form* Hitung PSNR



Gambar 8. Tampilan Hitung PSNR

Hasil perhitungan PSNR dapat dilihat seperti pada Tabel 3.

Tabel 3. Hasil Perhitungan PSNR

Nama Citra Hasil Dekompresi	PSNR (dB)
2.bmp	22,8811
4.bmp	19,9128
6.bmp	18,2306
8.bmp	17,2855
16.bmp	14,5698

Berdasarkan uji coba yang dilakukan, diperoleh bahwa semakin besar ukuran blok, maka nilai PSNR yang dicapai akan semakin kecil, namun variasi warna, tidak mempengaruhi PSNR dan pada citra dengan variasi warna satu, dimensi citra tidak akan mempengaruhi PSNR juga.

5. Kesimpulan

Dari beberapa percobaan yang telah dilakukan, maka ada beberapa hal yang dapat penuils simpulkan. Semakin banyak variasi warna yang dimiliki citra, maka rasio kompresi yang dicapai akan semakin kecil, waktu kompresi dan dekompresi akan semakin lama. Namun variasi warna tidak mempengaruhi nilai *PSNR* yang dicapai. Di sisi lainnya, semakin besar ukuran blok yang digunakan untuk proses kompresi, maka ukuran *file* hasil kompresi akan semakin kecil sehingga rasio kompresinya akan semakin besar. Selain itu, waktu dekompresi pun akan semakin cepat namun waktu kompresi belum tentu semakin cepat, tergantung pada perulangan kuantisasi yang dialaminya. Selain itu, pada citra dengan warna bervariasi, semakin besar ukuran blok, maka nilai *PSNR* yang dicapai akan semakin kecil (kualitas kompresi semakin rendah).

Pada citra yang hanya memiliki satu warna, semakin besar dimensi citra, maka rasio kompresi yang dicapai akan semakin kecil, waktu kompresi dan dekompresi akan semakin lama, namun tidak mempengaruhi nilai *PSNR* yang dicapai. Kedekatan warna blok dengan *codebook* yang diinisialisasikan mempengaruhi *PSNR* yang dicapai dan lamanya waktu kompresi. Semakin dekat nilainya dengan *codebook*, maka waktu kompresinya akan semakin cepat dan nilai *PSNR* akan semakin besar yang menunjukkan kualitas kompresi yang tinggi. Secara umum, pada kasus citra yang memiliki variasi warna banyak, metode *Kohonen Self-Organized Map* memberikan hasil dekompresi yang terlihat jelas berbeda dengan citra aslinya.

Daftar Pustaka

[1] Cristophe, A. Michel, V. Et. (1998). Image compression by Self-organized kohonen map. IEEE Transactions on Neural Networks. Vol. 9 (3), 503-507.

[2] Hecht-Nielsen, R. (1991). Neurocomputing. Adisson-Wesley Publishing Company, California.

[3] Miano, J. (2000). Compressed image file formats JPEG, PNG, GIF, XBM, BMP. Addison-Wesley.

[4] Munir, R. (2004). Pengolahan citra digital dengan pendekatan algoritmik. PT Informatika, Bandung.

[5] Salomon, D. (2000). Data compression: The complete reference 2nd Editon. Springer, New York.

[6] Sayood, K. (2005). Introduction to data compression 3rd edition. Morgan Kaufmann Publisher, San Fransisco.