Optimasi Kinerja Algoritma Klasterisasi K-Means untuk Kuantisasi Warna Citra

Irwanto, Yudhi Purwananto dan Rully Soelaiman
Jurusan Teknik Informatika, Fakultas Teknologi Informasi, Institut Teknologi Sepuluh Nopember (ITS)
Jl. Arief Rahman Hakim, Surabaya 60111
E-mail: yudhi@if.its.ac.id

***Abstrak*—Kuantisasi warna citra merupakan operasi penting pada banyak aplikasi grafik dan pengolahan citra. Metode kuantisasi warna banyak dilakukan dengan menggunakan algoritma klasterisasi data. Kepopuleran k-means sebagai algoritma klasterisasi data yang telah umum, ternyata belum mendapat cukup perhatian pada literatur kuantisasi warna. Hal ini disebabkan karena mahalnya biaya komputasi dan sensitivitasnya terhadap pengaruh pemilihan pusat klaster. Penelitian ini memberikan metode percepatan algoritma k-means untuk kuantisasi warna. Metode yang diajukan melibatkan beberapa modifikasi pada k-means konvensional, seperti pengurangan data, pembobotan data, dan penggunaan prinsip pertidaksamaan segitiga untuk mempercepat pencarian ketetanggaan terdekat. Ujicoba dilakukan dengan beragam citra dan menunjukkan bahwa modifikasi yang telah dilakukan mampu memperlihatkan bahwa k-means juga sangat kompetitif sebagai algoritma kuantisasi warna citra, baik dalam segi efektivitas maupun efisiensinya.**

***Kata Kunci*—klasterisasi, k-means, kuantisasi warna, reduksi warna.**

I. PENDAHULUAN

CITRA dengan warna asli umumnya mengandung banyak warna hingga ribuan sehingga tak jarang menjadi masalah tersendiri saat proses penyimpanan, penampilan, pengiriman, maupun pengolahannya [1]. Di masa lalu, kuantisasi warna citra diperlukan karena keterbatasan perangkat keras yang belum mampu menampilkan lebih dari 16 juta warna pada citra 24 bita. Walaupun sekarang perangkat keras telah berkembang dengan pesat, kuantisasi warna masih menjadi perhatian karena banyak aplikasi grafik modern dan citra yang berkaitan dengan kuantisasi citra, seperti: kompresi, segmentasi, deteksi dan lokalisasi teks, analisis tekstur warna, pemberian tanda air, dan sistem temu kembali berbasis isi.

Metode kuantisasi warna banyak dilakukan dengan menggunakan algoritma pengelompokan data. Kepopuleran *k-means* sebagai algoritma pengelompokan data yang telah umum, ternyata tidak banyak digunakan untuk kuantisasi warna citra. Hal ini disebabkan mahalnya biaya komputasi dan sensitivitasnya terhadap pengaruh inisialisasi.

Penelitian ini mengusulkan metode percepatan algoritma *k-means* untuk kuantisasi warna. Penelitian ini menginvestigasi kinerja *k-means* sebagai algoritma kuantisasi warna citra, dan selanjutnya memodifikasi algoritma *k-means* konvensional. Metode yang diajukan melibatkan beberapa modifikasi pada algoritma *k-means* konvensional, seperti pengurangan data [2], pembobotan sampel data, dan penggunaan pertidaksamaan segitiga untuk mempercepat pencarian ketetanggaan terdekat [3].

II. KUANTISASI WARNA

Proses kuantisasi warna citra secara garis besar dibagi menjadi dua fase utama, yaitu:

a) Desain palet: adalah fase untuk memilih sekumpulan warna citra yang dianggap mampu mewakili citra asli. Setiap warna terpilih mengandung 3 dimensi untuk warna RGB yang bisa diananalogikan sebagai kode dalam sebuah buku kode, dimana palet adalah buku kodenya.
b) Pemetaan piksel: adalah fase untuk menandai setiap piksel sesuai desain palet. Setiap piksel pada citra asli dipetakan ke warna terdekat dalam palet. Caranya adalah dengan menemukan warna yang sesuai atau yang paling dekat dari palet.

Tujuan dari kedua fase ini adalah untuk mengurangi jumlah warna unik N' dari citra menjadi K sehingga $N' \ll K$ dengan distorsi sekecil mungkin. Pada banyak aplikasi, citra asli dengan piksel 24 bita akan dikurangi menjadi 8 bita atau lebih kecil.

Metode kuantiasasi warna secara umum dapat dibedakan menjadi dua kategori: metode bebas yang menggunakan palet yang telah ditentukan, bukan diambil dari suatu citra tertentu. Metode tak bebas, palet ditentukan berdasar distribusi warna citra. Walaupun sangat cepat, metode bebas memberikan hasil yang kurang baik karena mengabaikan isi citra. Metode tak bebas dapat dikategorikan menjadi dua keluarga, yaitu: *preclustering* (tak-seragam) dan *postclustering* (seragam). Metode *preclustering* didasarkan pada analisis statistik distribusi warna citra. Metode ini diinisialisasi dengan menentukan klaster tunggal yang mengandung semua piksel citra N. Klaster ini kemudian dibagi secara rekursif sampai K klaster diperoleh. Metode *postclustering* diinisialisasi dengan klaster tunggal N dimana tiap klaster mengandung satu piksel citra. Klaster-klaster ini kemudian digabung sampai K klaster diperoleh. Beberapa yang termasuk dalam keluarga *postclustering* antara lain*: k-means* [4], *minmax*, *competitive learning*, *fuzzy c-means*, *BIRCH*, dan *self-organized map*.

III. KUANTISASI BERBASIS K-MEANS

A. Algoritma Klasterisasi K-means

Algoritma *k-means* (KM) dianggap sebagai algoritma yang paling banyak dipakai pada aplikasi pengelompokan data. KM menggunakan prinsip titik jarak terdekat. Diberikan suatu data pada himpunan $X = \{x_1, x_2, \ldots, x_N\} \in R^D$, tujuan dari KM adalah untuk membagi X menjadi K klaster $S = \{s_1, s_2, \ldots, s_K\}$,

dimana $\cup_{k=1}^{K} S_K = X, S_i \cap S_j = \emptyset$ untuk $1 \leq i \neq j \leq K$ dengan meminimalkan *sum of square error* (SSE):

$$SSE = \sum_{K=1}^{K} \sum_{x_i \in S_K} \|x_i - c_k\|_2^2 \quad (1)$$

dimana $\|.\|_2$ menyatakan norma *euclid* (L_2) dan c_k adalah pusat klaster s_K yang dihitung berdasar rata-rata jarak titik-titik klaster ke pusat klaster. Kompleksitas dari Persamaan 1 ini diketahui *NP-hard* bahkan hanya untuk nilai $K = 2$. Metode heuristik KM memberikan solusi yang lebih sederhana. Algoritma KM memulai dengan pusat K arbitreri yang diplih secara acak dari titik-titik data. Setiap titik kemudian dipilih berdasar titik terdekat dengan titik pusat, dan pusat baru dihitung dari rata-rata titik-titik yang berdekatan dengannya. Dua langkah ini diulang sampai kriteria terminasi terpenuhi. *Pseudocode* untuk langkah-langkah ini ditunjukkan pada Gambar 1 dimana $m[i]$ menyatakan keanggotaan titik x_i.

Kompleksitas dari KM adalah $O(NK)$ per iterasi untuk setiap nilai konstanta D. Untuk aplikasi kuantisasi citra berwarna, $D = 3$ karena pengelompokan hanya melibatkan tiga dimensi warna yaitu RGB. Dari perspektif klasterisasi data, KM memiliki kelebihan sebagai berikut:

a) Konsep KM sederhana, serbaguna, dan mudah dalam implementasi.
b) Kompleksitasnya linier pada N dan K. Dan beberapa teknik percepatan banyak dibahas di literatur.
c) KM dijamin terminasi dengan tingkat konvergensi kuadratik.

Sementara itu, kelemahan utama KM adalah fakta bahwa KM seringkali terminasi pada lokal minimum dan hasilnya sangat sensitif oleh pengaruh pemilihan pusat klaster. Dari perspektif kuantisasi warna citra, KM mimiliki dua kekurangan yaitu: walaupun kompleksitas waktunya linier, sifat iteratif dari fase desain palet menyebabkan biaya komputasinya menjadi mahal. Fase pemetaan piksel tidak efisien, karena untuk setiap input piksel akan dicari satu per satu dari keseluruhan palet untuk menentukan warna terdekat.

input: $X = \{x_1, x_2, \ldots, x_N\} \in R^D$ *(N × D input data set)*
output: $C = \{c_1, c_2, \ldots, c_K\} \in R^D$ *(K klaster pusat)*
Pilih acak subhimpunan C dari X sebagai inisial klaster pusat
while *kriteria terminasi belum terpenuhi* **do**
  for $(i = 1; i \leq N; i = i + 1)$ **do**
  Masukkan x_i *pada pusat klaster terdekat*
  $m[i] = \arg\min_{K \in \{1,2,\ldots,K\}} \|x_i - c_k\|^2$;
  end
  Hitung kembali pusat klaster;
  for $(k = 1; k \leq K; k = k + 1)$ **do**
    Klaster S_k *mengandung himpunan titik-titik* x_i *yang paling dekat dengan pusat* c_k;
    $S_k = \{x_i | m[i] = k\}$;
    Hitung pusat baru c_k *berdasar nilai rata-rata titik-titik yang termasuk dalam* S_k;
    $c_k = \frac{1}{\|S_k\|} \sum_{x_i \in S_k} x_i$
  end
end

Gambar 1. Algoritma *k-means* konvensional.

B. *Mempercepat Algoritma K-means*

Dalam rangka menjadikan algoritma KM lebih efisien untuk proses kuantisasi warna citra, penelitian ini melakukan beberapa modifikasi pada algoritma KM konvensional.

1. **Pembentukan Data Subsampel.** Langkah yang paling pasti untuk mempercepat KM adalah dengan mengurangi jumlah data, yang dapat diperoleh dengan pembentukan data subsampel dari input citra. Penelitian ini memakai dua metode pembentukan data subsampel deterministik, yaitu: Subsampel 2:1, yang melibatkan arah horisontal dan vertikal. Sehingga hanya 1/4 dari input citra yang dipakai. Cara ini terbukti efektif mengurangi waktu komputasi tanpa mengurangi kualitas hasil visual kuantisasi [2]. Metode yang kedua adalah subsampel yang hanya melibatkan piksel-piksel dengan warna unik. Piksel-piksel ini dapat ditentukan secara efisien dan mudah dengan menggunakan *hash table* dengan menggunakan fungsi *hash* $h_a(x) = (\sum_{i=1}^{3} a_i x_i) \bmod m$, dimana $x = (x_1, x_2, x_3)$ menyatakan komponen piksel merah (x_1), hijau (x_2), dan biru (x_2), dan m adalah jumlah warna primer, dan elemen dari urutan $a = (a_1, a_2, a_3)$ dipilih secara acak dari himpunan $\{0, 1, \ldots, m-1\}$. Metode subsampel ini digunakan untuk mengurangi data citra dimana kebanyakan citra mengandung warna-warna yang sama dalam jumlah yang sangat besar.
2. **Pembobotan Data Subsampel.** Kelemahan dari metode subsampel adalah bahwa metode ini mengabaikan distribusi warna citra asli. Untuk mengatasi hal ini, setiap titik diberi bobot secara proposional sesuai frekuensinya. Bobot dinormalisasi dengan jumlah piksel pada citra asli untuk menghindari ketidakstabilan numerik saat perhitungan.
3. **Penggunaan Pertidaksamaan Segitiga.** Pada algoritma KM, untuk setiap titik, jarak ke setiap titik dari pusat klaster K dihitung. Misal ada sebuah titik x_i, dan dua pusat klaster c_a dan c_b dan matrik jarak d, menggunakan pertidaksamaan segitiga, didapatkan $d(c_a, c_b) \leq d(x_i, c_a) + d(x_i, c_b)$. Jadi jika diketahui bahwa $2d(x_i, c_a) \leq d(c_a, c_b)$, dapat disimpulkan bahwa $d(x_i, c_a) \leq d(x_i, c_b)$ tanpa harus menghitung $d(x_i, c_b)$. Untuk menghindari perhitungan jarak terdekat dalam jumlah yang sangat banyak, prinsip pertidaksamaan segitiga ini dapat digunakan untuk memodifikasi algoritma KM. Bentuk modifikasinya adalah dengan melakukan perhitungan jarak berpasangan antarpusat klaster pada setiap permulaan iterasi. Untuk lebih mengurangi jumlah perhitungan jarak terdekat, modifikasi juga dilakukan dengan cara mengurutkan nilai jarak yang berasosiasi dengan setiap pusat klaster. Pada setiap iterasi, jarak titik x_i dari pusat c_k dibandingkan dengan pusat-pusat klaster lainnya dengan urutan menaik dimana x_i adalah titik yang berada pada klaster di itrasi sebelumnya. Jika pusat yang berada cukup jauh dari c_k telah dicapai, semua pusat-pusat lainnya dapat diabaikan dan langkah-langkah klasterisasi dapat dilanjutkan untuk titik-titik lainnya.

Langkah-langkah modifikasi ini disusun menjadi sebuah algoritma *weighted sort-means* (WSM). *Pseudeocode* algoritma ini ditunjukan oleh Gambar 2. Kompleksitas algoritma WSM adalah $O(K^2 + K^2 \log K + N'\gamma)$ per iterasi untuk setiap nilai D, dimana $O(K^2)$ adalah biaya perhitungan jarak berpasangan antarpusat klaster, $O(K^2 \log K)$ biaya mengurutkan pusat, dan $O(N'\gamma)$ biaya membandingkan. Biaya membandingkan dianggap sebagai waktu komputasi sebab pada aplikasi kuantisasi citra K adalah jumlah yang kecil sehingga $K << N'$. Sehingga dapat dikatakan bahwa WSM linier pada N', yang merupakan jumlah warna unik dari citra asli. Dalam hal inisialisasi, algoritma WSM memiliki skema yang sama dengan skema inisialisasi algoritma KM.

input: $X = \{x_1, x_2, \ldots, x_{N'}\} \in R^D$ ($N' \times D$ *input data set*)
$W = \{w_1, w_2, \ldots, w_{N'}\} \in [0,1]$ (N' *bobot*)
output: $C = \{c_1, c_2, \ldots, c_K\} \in R^D$ (K *klaster pusat*)
Pilih acak subhimpunan C dari X sebagai inisial klaster pusat
while *kriteria terminasi belum terpenuhi* **do**
  Hitung jarak berpasangan antara pusat klaster;
  for $(i = 1; i \leq K; i = i + 1)$ **do**
    for $(j = 1 + 1; j \leq K; j = j + 1)$ **do**
      $d[i][j] = d_{ji} = \|c_i - c_j\|^2$
    end
    Buat matrik M ukuran K × K dimana setiap baris i adalah permutasi dari 1, 2, …, K yang mewakili klaster dengan urutan jarak menaik dari pusat dari c_i;
    for $(i = 1; i \leq N'; i = i + 1)$ **do**
      Nyatakan S_p *sebagai klaster dimana* x_i *dimasukkan pada iterasi sebelumnya;*
      $p = m[i]$;
      $min\_dist = prev\_dist = \|x_i - c_p\|$
      Hitung ulang pusat terdekat jika diperlukan;
      for $(j = 2; j \leq K; j = j + 1)$ **do**
        $t = m[p][j]$;
        if $d[p][t] \geq 4\, prev\_dist$ **then**
          Tidak ada pusat terdekat lagi. Hentikan pengecekan;
          break;
        end
        $dist = \|x_i - c_t\|^2$
        if $dist \leq min\_dist$ **then**
          c_t lebih dekat x_i dari pada c_p;
          $min\_dist = dist$;
          $m[i] = t$;
        end
      end
    end
    Hitung ulang pusat klaster;
    for $(k = 1; k \leq K; k = k + 1)$ **do**
      Hitung pusat baru c_t *sebagai rata-rata bobot dari titik yang dekat dengan pusat baru;*

$$c_k = \left(\sum_{m[i]=k} w_i x_i \right) \Big/ \sum_{m[i]=k} w_i$$

    end
end

Gambar 2. Algoritma *weighted sort-means*.

C. *Model Skema Inisialisasi K-means*

Tahap inisialisasi merupakan tahap yang paling penting karena sangat mempengaruhi hasil klasterisasi. Penelitian ini menggunakan beberapa model skema inisialisasi, antara lain:

1. Forgy (FGY) [5]: Pusat klaster dipilih secara acak dari kumpulan data. Kompleksitas dari skema ini adalah $O(K)$.
2. Minmax (MMX) [6]: Pusat pertama c_1 dipilih secara acak. Pada pusat ke-c_i $(i \in \{2,3,\ldots,K\})$, c_i dipilih menjadi titik yang memiliki jarak minimum terbesar ke pusat sebelumnya yang dipilih. Kompleksitas skema ini adalah $O(NK)$.
3. Subset Farthest First (SFF) [7]: Salah satu kelemahan MMX adalah skema tersebut cenderung menemukan *outlier* (titik data yang letaknya jauh dari kumpulan titik data lainnya). Dengan menggunakan *subset* ukuran terkecil dari $2K\ln K$, jumlah total *outlier* yang dapat ditemukan oleh MMX dapat dikurangi sehingga proporsi titik *non-outlier* yang didapatkan sebagai pusat menjadi lebih banyak. Kompleksitas skema ini adalah $O(K2\ln K)$.
4. K-Means++ (KPP) [8]: Pusat pertama c_1 dipilih secara acak dan pada pusat ke c_i $(i \in \{2,3,\ldots,K\})$, c_i dipilih menjadi $x' \in X$, dengan probabilitas $\frac{D(x')^2}{\sum_{i=1}^{N} D(x_i)^2}$ dimana $D(x)$ menotasikan jarak minimal dari titik x ke pusat sebelumnya yang dipilih.

IV. UJI COBA DAN EVALUASI

A. *Data Citra Tes dan Kriteria Ujicoba*

Kumpulan data yang digunakan adalah citra tes yang biasa dipakai dalam literatur kuantisasi, yaitu citra *airplane* (29% warna unik), *baboon* (58% warna unik), *boats* (31% warna unik), *lenna* (57% warna unik), *parrots* (13% warna unik), *peppers* (42% warna unik), *fish* (47% warna unik), dan *poolballs* (7% warna unik).

Citra dikuantisasi menjadi 32 warna (nilai K = 32) dengan menggunakan skema inisialisasi *forgy* untuk semua citra. Kondisi terminasi dilakukan setelah 20 kali iterasi dengan *threshold* (ambang konvergensi) ditentukan sebesar $\varepsilon = 1e-6$. Citra *error* didapatkan dari pengurangan absolut antara citra asli dengan citra terkuantisasi. Untuk mendapatkan bentuk visualisasi agar mudah dibaca, maka nilai piksel dari citra *error* dikalikan 4 kemudian dijadikan citra negatifnya.

Efektivitas metode kuantisasi dinyatakan dengan menggunakan pengukuran nilai *mean square rrror (*MSE):

$$MSE(X, \tilde{X}) = \frac{1}{HW} \sum_{h=1}^{H} \sum_{w=1}^{W} \|x(h,w) - \tilde{x}(h,w)\|_2^2 \qquad (2)$$

dimana X menyatakan $H \times W$ citra asli dan $\tilde{x}$ menyatakan $H \times W$ citra terkuantisasi dan ruang warna RGB. MSE menyatakan rata-rata distorsi dan umum digunakan untuk mengevaluasi citra pada literatur kuantisasi [1]. Efisiensi dari metode kuantisasi diukur dengan waktu komputasi CPU dalam satuan detik. Semua program diimplementasikan menggunakan bahasa pemrograman ANSI C dengan

perangkat pengembang Code::Block, dan dieksekusi pada Intel(R) Core(TM)2 Duo CPU T6500 @ 2.10GHz.

Tabel 1. Perbandingan waktu komputasi CPU antara algoritma KM dan WSM.

K	*Algoritma*	*Airplane*	*Baboon*	*Boats*	*Lenna*	*Parrot*	*Peppers*	*Fish*	*Poolballs*	*Mean*
32	KM	5.59	3.36	7.27	4.63	30.9	3.33	0.96	4.01	5.59
	WSM	0.99	0.94	1.74	1.12	4.88	1.06	0.24	0.72	**0.99**
64	KM	6.6	6.62	10.45	7.68	35.4	6.28	1.69	4.52	9.90
	WSM	1.77	1.29	2.42	1.49	6.68	1.29	0.39	0.83	**2.02**
128	KM	11.6	13.36	21.11	12.76	70.7	11.6	2.99	9.1	19.1
	WSM	2.44	1.79	2.68	2.63	8.70	1.93	0.62	1.23	**2.75**
256	KM	23.0	23.2	40.15	22.11	134.9	22.8	6.19	18	36.2
	WSM	4.05	3.39	5.13	3.90	14.35	3.74	1.81	2.17	**4.81**

B. Perbandingan Efisiensi antara KM dan WSM

Pada bagian ini dilakukan kuantisasi warna citra dengan menggunakan algoritma KM dan WSM. Kedua algoritma diinisialisasi dengan skema yang sama, yaitu *forgy*. Nilai K yang digunakan adalah 32, 64, 128, dan 256. Program melakukan terminasi setelah 20 kali iterasi dengan nilai ambang konvergensi ditentukan sebesar $\varepsilon = 1e-6$. Tabel 1 menunjukkan perbandingan waktu komputasi antara algoritma KM dan algoritma WSM. Dapat dilihat bahwa algoritma WSM lebih cepat sekitar 12 sampai 20 kali dibandingkan dengan algoritma KM. Hal ini dikarenakan algoritma WSM menghindari perhitungan warna yang redundan pada citra asli dan dengan mengurangi jumlah data sebelum melakukan fase klasterisasi.

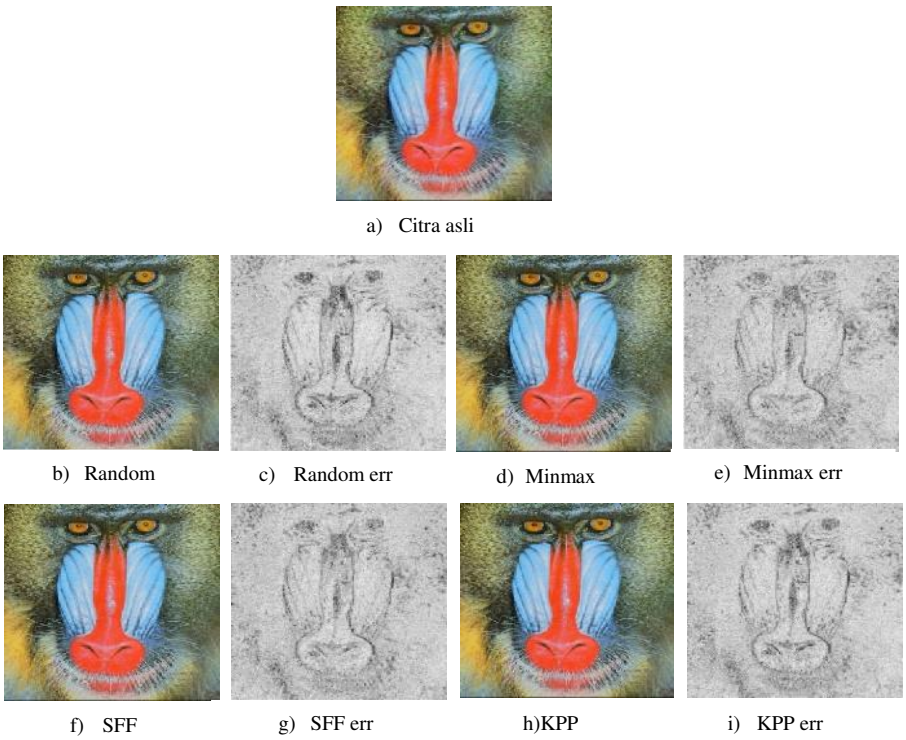
C. Perbandingan Model Skema Inisialisasi Algoritma WSM

Pada bagian ini dilakukan kuantisasi citra menggunakan algoritma WSM dengan menggunakan empat skema inisialisasi, yaitu *forgy* (FGY), *minmax* (MMX), *subset farthest first* (SFF), dan *k-means++* (KPP). Nilai K yang digunakan adalah 32, 64, 128, dan 256 dengan nilai ambang konvergensi ditentukan sebesar $\varepsilon = 1e-6$ untuk semua skema inisialisasi. Ujicoba ini dilakukan untuk mengetahui skema inisialisasi yang paling efisien digunakan pada algoritma WSM untuk kuantisasi warna citra. Tabel 2 menunjukkan perbandingan waktu komputasi CPU terhadap skema inisialiasai yang diterapkan pada algoritma WSM. Dari Tabel 2 didapatkan bahwa skema inisialisasi SFF secara konsisten menghasilkan waktu komputasi CPU paling cepat dibandingkan dengan skema inisialisasi lainnya. Untuk mengetahui skema inisialisasi yang memberikan kualitas visualisasi yang paling baik, dilakukan pengukuran nilai MSE pada masing-masing citra terkuantisasi. Seperti yang dijelaskan sebelumnya bahwa semakin kecil nilai MSE, citra terkuantisasi semakin menyerupai citra asli, artinya hasil visual yang diperoleh lebih baik. Tabel 5 menunjukkan perbandingan nilai MSE yang dihasilkan dari skema-skema inisialisasi yang diterapkan pada algoritma WSM. Dari Tabel 5 dapat diketahui bahwa algoritma WSM dengan skema KPP secara konsisten menghasilkan kualitas yang lebih baik (nilai MSE terkecil) jika dibandingkan dengan skema inisialisasi lainnya. Gambar 3 menunjukkan contoh hasil citra terkuantisasi dari masing-masing skema inisialisasi.

Pengaruh K pada kompleksitas algoritma WSM dapat ditunjukkan oleh Gambar 4. Berbeda dengan algoritma KM, kompleksitas algoritma WSM sublinier terhadap K. Sebagai contoh, pada citra *Parrots*, kenaikan K dari 32 ke 256, menghasilkan kenaikan waktu komputasi CPU dengan kelipatan sebesar 2.9 (dari 4.48 sekon ke 14.46 sekon).

Tabel 2. Perbandingan waktu komputasi CPU algoritma WSM dengan beberapa model skema inisialisasi.

K	*Skema*	*Airplane*	*Baboon*	*Boats*	*Lenna*	*Parrot*	*Peppers*	*Fish*	*Poolballs*	*Mean*
32	FGY	0.99	**0.94**	1.74	1.12	4.88	1.06	0.24	0.72	1.46
	MMX	0.87	1.00	2.2	**1.00**	4.94	1.14	0.3	**0.61**	1.50
	SFF	**0.68**	1.05	**1.38**	1.11	**4.55**	**0.89**	**0.22**	0.62	**1.31**
	KPP	2.34	1.95	2.98	2.27	9.72	2.42	0.54	1.25	2.93
64	FGY	1.32	**1.28**	2.05	1.61	**6.48**	**1.41**	0.37	0.92	1.93
	MMX	1.81	1.76	2.65	1.65	7.96	1.83	0.42	0.99	2.38
	SFF	**0.92**	1.4	**1.72**	**1.23**	7.30	1.55	**0.35**	**0.74**	**1.90**
	KPP	2.95	3.3	5.24	3.39	17.69	4.6	1.05	2.83	5.13
128	FGY	2.25	2.26	**2.76**	2.65	9.93	2.31	**0.65**	**1.34**	3.01
	MMX	2.04	2.99	4.04	3.32	13.18	2.71	0.73	1.61	3.82
	SFF	**1.28**	**2.24**	3.49	**2.28**	**7.73**	**2.12**	0.72	1.55	**2.67**
	KPP	5.41	6.06	9.98	7.27	35.48	6.32	1.72	3.51	9.46
256	FGY	3.38	4.04	5.41	4.83	**14.46**	**3.31**	**1.88**	2.68	4.99
	MMX	4.29	6.17	7.8	5.81	22.15	4.74	2.24	3.23	7.05
	SFF	**3.21**	**3.87**	**4.98**	**3.32**	15.00	3.51	2.01	**2.34**	**4.78**
	KPP	10.89	11.62	18.96	11.68	64.41	12.8	3.87	7.91	17.7



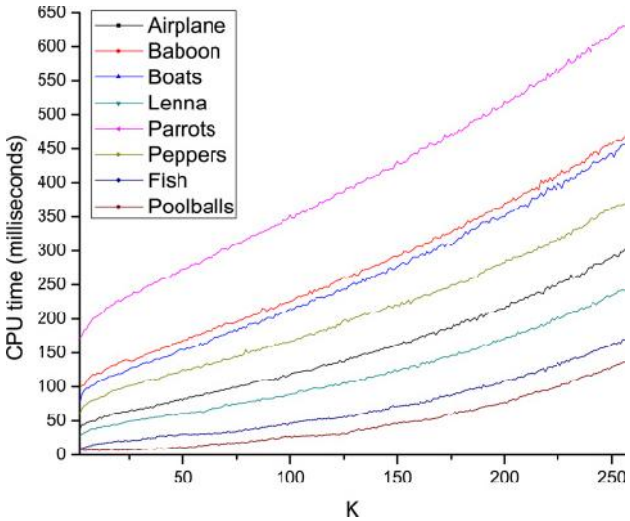
a) Citra asli

b) Random c) Random err d) Minmax e) Minmax err

f) SFF g) SFF err h)KPP i) KPP err

Gambar 3. Hasil kuantisasi citra *Baboon* dengan beberapa skema inisialisasi pada algoritma WSM (K = 256).

Tabel 3 memberikan peringkat nilai MSE dan Tabel 4 menunjukkan waktu komputasi CPU dari algoritma WSM dengan empat model skema inisialisasi. Nilai MSE terbaik dihasilkan oleh skema inisialisasi KPP, sementara waktu komputasi CPU terbaik dihasilkan oleh skema SFF. Walaupun skema inisialiasai KPP memberikan kualitas hasil visualisasi paling baik, skema inisialiasiasi ini ternyata memakan biaya komputasi paling besar. Skema inisialisasi *forgy* diujicobakan karena skema ini adalah skema yang paling sederhana dan umum digunakan. Skema inisialisasi *minmax* diujicobakan karena skema ini termasuk skema sederhana untuk diimplementasikan pada data multidimensi. Sementara itu, skema inisialisasi *subset farthest first* dipakai untuk memperbaiki skema *minmax* yang cenderung menghasilkan data *outlier* (titik data yang letaknya jauh dari kumpulan titik data lainnya.



Gambar 4. Grafik pengaruh K terhadap waktu komputasi CPU.

Tabel 3. Peringkat nilai MSE algoritma WSM dengan beberapa skema inisialisasi.

Skema	***Nilai MSE***				
	32	**64**	**128**	**256**	**Mean**
FGY	189.7	113.6	66.95	40.57	102.70
MMX	173.5	104.7	62.98	39.38	95.14
SFF	160	93.7	57.34	35.87	86.72
KPP	157.7	91.28	55.16	34.26	**84.6**

Tabel 4. Peringkat waktu komputasi CPU algoritma WSM dengan beberapa skema inisialisasi.

Skema	***Waktu Komputasi CPU***				
	32	**64**	**128**	**256**	**Mean**
FGY	1.46	1.93	3.01	4.99	2.84
MMX	1.50	2.38	3.82	7.05	3.68
SFF	1.31	1.90	2.67	4.78	**2.66**
KPP	2.93	5.13	9.46	17.7	8.80

Tabel 5.
Perbandingan nilai MSE algoritma WSM dengan beberapa skema inisialisasi.

K	*Skema*	*Airplane*	*Baboon*	*Boats*	*Lenna*	*Parrot*	*Peppers*	*Fish*	*Poolballs*	*Mean*
32	FGY	119.9	370.6	123.4	123.5	264.5	228.9	148.8	138	189.7
	MMX	85.83	332.3	120.3	127.7	234.9	224.1	157	106.1	173.5
	SFF	58.57	326.4	115.7	119.6	237.6	220.4	140.3	62.2	160.0
	KPP	58.66	326.1	117.1	118.8	227.4	222.5	141.3	50.36	157.7
64	FGY	60.3	204.9	72.79	73.23	148.6	139.3	99.4	110.7	113.6
	MMX	46.05	201.2	68.21	76.46	133.7	137	95.73	79.75	104.7
	SFF	36.55	197.7	66.51	72.86	127.6	131.4	88.31	28.79	93.7
	KPP	33.97	195.2	63.22	72.43	127.4	132.2	84.38	21.48	91.28
128	FGY	32.6	124.4	42.95	48	82.09	82.55	57.37	65.64	66.95
	MMX	31.02	125.7	42.05	48.76	82.79	84.2	60.12	29.27	62.98
	SFF	23.03	124	39.31	46.29	75.45	80.89	56.44	13.37	57.34
	KPP	21.49	123.2	37.13	45.95	73	79.81	50.72	10.05	55.16
256	FGY	21.3	79.47	28.43	31.3	49.37	53.33	38.57	22.81	40.57
	MMX	18.93	82.34	26.4	31.81	48.96	53.3	37.21	16.14	39.38
	SFF	15.13	80.42	25	30.66	45.07	51	33.91	5.82	35.87
	KPP	14.05	78.48	23.09	30.03	42.64	49.6	31.27	4.97	34.26

V. SIMPULAN DAN SARAN

Dari ujicoba yang telah dilakukan dan setelah menganalisis hasil pengujian terhadap implementasi algoritma WSM, dapat diambil beberapa kesimpulan antara lain:

a. Pada permasalahan kuantisasi wana citra, algoritma WSM menghasilkan waktu komputasi 12 hingga 20 kali lebih cepat daripada algoritma KM konvensional.
b. Kompleksitas WSM sublinier terhadap K.
c. Dilihat dari perspektif biaya waktu komputasi CPU, skema inisialisasi *subset farthest first* secara konsisten menghasilkan waktu komputasi yang lebih cepat dibanding dengan skema inisialisasi lainnya.
d. Dilihat dari perspektif kualitas hasil visual citra terkuantisasi, skema inisialisasi *k-means++* secara konsisten menghasilkan kualitas yang lebih baik jika dibandingkan dengan skema inisialisasi lainnya.
e. Skema inisialisasi *forgy* menghasilkan nilai MSE dan waktu komputasi CPU yang tidak konsisten. Hal ini disebabkan karena setiap kali iterasi, selalu dilakukan pemilihan pusat klaster secara acak tanpa mekanisme cerdas untuk menghindari pemilihan pusat klaster yang kurang representatif.

Untuk pengembangan lebih lanjut, algoritma WSM dapat diuji dengan menggunakan skema inisialisasi yang lain antara lain: *splitting*, *density-based* dan skema *maximum variance,* karena selain dapat diterapkan pada data multidimensi seperti pada data citra, kompleksitas skema-skema ini tidak kuadratik atau lebih besar. Algoritma WSM dikaji dan dibandingkan dengan algoritma kuantisasi citra yang telah populer seperti algoritma *median-cut*, metode *otto*, *octree*, *variance-based*, *greedy*, *binary splitting*, dan *modified minmax*.

DAFTAR PUSTAKA

[1] L. Brun, A. Trémeau, Digital Color Imaging Handbook, CRC Press, Ch. Color Quantization, (2002) 589–6388.
[2] Y.-C. Hu, M.-G. Lee, K-means based color palette design scheme with the use of stable flags, Journal of Electronic Imaging 16 (3) (2007) 033003.
[3] S. Phillips, Acceleration of k-means and related clustering algorithms, Proc. of the 4th Int. Workshop on Algorithm Engineering and Experiments, (2002) 166–177.
[4] H. Kasuga, H. Yamamoto, M. Okamoto, Color quantization using the fast k-means algorithm, Systems and Computers in Japan 31 (8) (2000) 33–40.
[5] E. Forgy, Cluster analysis of multivariate data: efficiency vs. interpretability of classification, Biometrics 21 (1965) 768.
[6] D. Hochbaum, D. Shmoys, A best possible heuristic for the k-center problem, Mathematics of Operations Research 10 (2) (1985) 180–184.
[7] M. Al-Daoud, S. Roberts, New methods for the initialisation of clusters, Pattern Recognition Letters 17 (5) (1996) 451–455.
[8] D. Arthur, S. Vassilvitskii, k-means++: the advantages of careful seeding, Proc. Of the 18th Annual ACM-SIAM Symposium on Discret Algorithms, (2007) 1027–10.