

## VARIASI SVD PADA KOMPRESI CITRA

Andi Nur Hafsah M

T. Basaruddin

Danang Jaya

Fakultas Ilmu Komputer - Universitas Indonesia

Kampus Baru UI Depok, Jawa Barat 16424

e-mail: [anh51@ui.edu](mailto:anh51@ui.edu) [chan@cs.ui.ac.id](mailto:chan@cs.ui.ac.id) [daja50@ui.edu](mailto:daja50@ui.edu)

### ABSTRAK

Laporan penelitian ini memberikan metode baru untuk mengkompres citra, yaitu menggunakan teknik *shuffle* untuk citra berwarna. Aplikasi *shuffle* yang digunakan menggunakan dua metode, yaitu matriks representasi diproses secara global dan dipilah-pilah menjadi beberapa blok (sub matriks). Kami juga melakukan percobaan terhadap pengaruh running time, ukuran blok, dan ukuran *rank* terhadap *error* dengan satuan ukuran *error*, yaitu *Mean Square Error* (MSE). Hasil penelitian menunjukkan bahwa melalui pemrosesan secara global, SSVD lebih unggul daripada SVD, tetapi melalui pemrosesan terhadap blok yang dipilah-pilah, SSVD tidak terbukti lebih unggul daripada SVD.

**Kata kunci:** Variasi SVD, *shuffle*-SVD, kompresi citra

Makalah diterima [13 desember 2006]. Revisi akhir [10 Januari 2007].

### 1. PENDAHULUAN

Kompresi citra adalah meminimalan ukuran pada *bytes* dari sebuah grafis *file* tanpa menurunkan kualitas citra. Pengurangan pada ukuran *file* berdampak pada penghematan ruang memori dan waktu untuk pengiriman gambar ke internet serta waktu untuk mengunduhnya dari *website*[3].

Ada dua macam teknik untuk mengkompres citra, yaitu *Lossless* dan *Lossy*. Teknik *lossless* memberikan jaminan bahwa citra yang telah di-*decompress* benar-benar identik dengan citra sebelum dikompres. Contoh, *Run Length encoding*, *Huffman encoding*, *Entropy coding*(Lempel/Ziv) dan *area coding*. Pada beberapa aplikasi tidak dibutuhkan pengembalian data keseluruhan dari citra aslinya, oleh karena itu digunakan teknik *lossy*. Contoh teknik ini adalah *transform coding* (SVD/KLT/DCT/Waveletets/Gabor), *vector quantisation*, metode segmentasi dan aproksimasi, metode *spline* aproksimasi (Interpolasi *Bilinear*/Regularisasi), *Fractal Coding* (*texture synthesis*, *iterated function system* [IFS], *recursives IFS* [RIFS]), dan Efisiensi & kualitas teknik kompresi *lossy* yang berbeda.[1,4].

Salah satu metode yang digunakan pada kompresi citra adalah SVD yang dapat dibentuk dari sembarang matriks dengan ukuran  $M \times N$ , dimana SVD membagi matriks menjadi tiga, yaitu  $U$ ,  $\Sigma$ ,  $V$ , sehingga  $A = U\Sigma V^T$ .  $\Sigma$  merupakan matriks diagonal dan  $U$ ,  $V$  adalah matriks *orthogonal* [2]. Kemudian aproksimasi *rank*  $r$  terhadap  $A$  adalah matriks  $A = U_r \Sigma_r V_r^T$ , dimana  $\Sigma_r$  adalah *top-left*  $r \times r$  submatriks dari  $\Sigma$ ,  $U_r$  terdiri atas kolom  $r$  pertama dari  $U$ , dan  $V_r^T$  merupakan baris  $r$  pertama dari  $V^T$ . Dengan menggunakan *rank*  $r$  aproksimasi diperoleh kompresi citra yang optimum.

Dalam laporan penelitian yang menjadi rujukan kami telah ditemukan suatu metode baru untuk mendapatkan kompresi citra yang lebih baik dibandingkan dengan metode SVD. Metode tersebut adalah *shuffle-SVD* (SSVD), dimana langkah *preprocessing*-nya pada citra  $A$  dipermutasi dengan sebuah data permutasi independen  $S$  yang disebut *shuffle* (persamaan 1)[6]. Hasil matriks  $X = S(A)$  yang kemudian didekomposisi dengan SVD. Misalkan  $X_r = U_r \Sigma_r V_r^T$ , merupakan aproksimasi *rank*  $r$  terhadap  $X$ , akan menghasilkan aproksimasi yang lebih baik untuk  $A$  daripada  $A_r$ , sehingga hasil matriks yang di-*shuffle* tadi lebih baik daripada matriks yang tidak di-*shuffle*. Baik SVD maupun SSVD dapat diterapkan ke dalam aplikasi KLT (*Karhunen-Loeve Transform*) untuk mengkompres citra tunggal. SVD dan SSVD memiliki hubungan dengan KLT pada saat memilah-milah citra tunggal ke dalam blok-blok dengan ukuran tertentu[6].

Laporan penelitian tersebut menggunakan input citra *grayscale*, sedangkan percobaan yang kami lakukan menggunakan citra berwarna. Pada penelitian kami meneliti apakah citra berwarna menghasilkan pengaruh yang sama dengan citra *grayscale* terhadap *shuffle*. Ini merupakan kontribusi kami yang pertama. Kontribusi kami yang kedua adalah menambahkan teknik *blocking* yang berbeda. Hal ini akan dijelaskan secara lengkap pada bagian 2. Kontribusi kami yang ketiga adalah teknik optimasi *rank* yang kami lakukan per blok, sehingga *rank* optimasi tiap blok tidak sama.

### 2. TEKNIK KOMPRESI CITRA BERBASIS SVD DAN SHUFFLE-SVD

Seperti disebutkan di atas bahwa teknik kompresi citra berbasis SVD dan *shuffle-SVD* menggunakan teknik *lossy*

untuk mengkompres citra. Berikut ini disajikan teknik mengkompres citra dengan menggunakan algoritma *SVD* dan *shuffle-SVD*.

## 2.1 Perbandingan terhadap *KLT*

Menurut Jain[5], pertama-tama kita mendefinisikan transformasi *KL* untuk vektor-vektor random. Misalkan  $x$  menunjukkan sebuah  $p \times 1$  vektor random, dimana  $x[i]$  adalah variable random pada suatu ruang probabilitas, untuk  $0 \leq i < p$ . Untuk lebih mudahnya kita akan memberikan index pada matriks dan vektor dimulai dari 0. Diberikan  $R = E[xx^T]$  menunjukkan matriks auto korelasi dari  $x$ .  $R$  didiagonalkan sebagai  $R = \Phi \Lambda \Phi^T$ . Kemudian transformasi *KL* dari  $x$  adalah  $y = \Phi^T x$ . Sebagai catatan  $\Phi$  bergantung pada  $x$ , dan selanjutnya baik  $\Phi$  maupun  $y$  dibutuhkan untuk merepresentasikan  $x$ .

Kemudian kita mendefinisikan[6] *KLT* untuk himpunan citra  $A_0, \dots, A_{p-1}$ . Untuk itu kita menyusun secara linier setiap pixel dalam himpunan citra menjadi sebuah vektor. Misalkan  $x_i$  menunjukkan versi linier dari  $A_i$ . Kemudian kita menggunakan kumpulan dari vektor-vektor sebagai ruang probabilitas untuk vektor random  $x$  dan adaptasi definisi sebelumnya. Misalkan  $X$  menunjukkan matriks yang dibentuk dengan menggunakan  $x_i$  sebagai vektor-vektor kolomnya. Matriks autokorelasi untuk  $x$  adalah  $R = (1/p)XX^T$ . Sekarang  $\Phi$  didefinisikan dengan mendiagonalisasikan  $R = \Phi \Lambda \Phi^T$ . Kemudian *KLT* dari  $X$  adalah  $Y = \Phi^T X$ .

Misalkan  $X = U \Sigma V^T$  adalah faktorisasi *SVD* dari  $X$ , sehingga kita mendapatkan  $\Phi \Lambda \Phi^T = R = (1/p)XX^T = (1/p)U \Sigma^2 U^T$ , jadi  $\Phi = U$  dan  $\Lambda = (1/p)\Sigma^2$ . Selanjutnya  $Y = \Phi^T X = U^T X = \Sigma V^T$ . Sehingga representasi *KLT* dari sekumpulan citra direpresentasikan oleh  $X$  adalah sama dengan *SVD* dari  $X$ .

Untuk menggunakan hal tersebut untuk meng-encode citra tunggal  $A$ , kita menspesifikasikan cara untuk memecah citra  $A$  menjadi citra  $A_0, \dots, A_{p-1}$  yang dilinierkan dan kemudian membuatnya menjadi matriks  $X$ . Sehingga, kita membutuhkan untuk menambahkan operator  $P$  yang bisa digunakan untuk mengkonstruksi  $X = P(A)$ . Lebih jelasnya, kolom dari  $P(A)$  akan menjadi citra pada himpunan citra (setelah dilinierkan). Secara umum,  $P$  bisa menjadi operator permutasi[6].

## 2.2 *SVD*

Diberikan matriks  $A$  dengan ukuran  $N \times N$ . Misalkan  $P$  menjadi matriks identitas, maka  $X = A$ . Citra pada himpunan citra adalah kolom-kolom dari  $X = A$ . Jadi *KLT* dari  $X$  menjadi *SVD* matriks  $A$ [6].

## 2.3 *Shuffle-SVD*

Diberikan sebuah matriks  $A$  ukuran  $N \times N$ . Dianggap bahwa  $N = n^2$  dengan  $n$  adalah integer. Dimana  $P$  akan dipilih menjadi operator *shuffle*  $S$  yang telah disebutkan sebelumnya. Ranade dkk membentuk  $X = S(A)$  dengan (i) memilah-milah  $A$  menjadi blok dengan ukuran  $n \times n$ , (ii) mengambil blok ke- $i$  pada urutan mayor baris dan mengatur pikselnya pada mayor baris untuk menghasilkan baris ke- $i$  dari  $X$ . Lebih jelasnya, diberikan rumus sebagai berikut untuk metode *shuffle*-nya[6]:

$$X[\lfloor i/n \rfloor n + \lfloor j/n \rfloor, (i \bmod n)n + j \bmod n] = A[i,j] \quad (1)$$

## 3. PERCOBAAN NUMERIK DAN ANALISIS

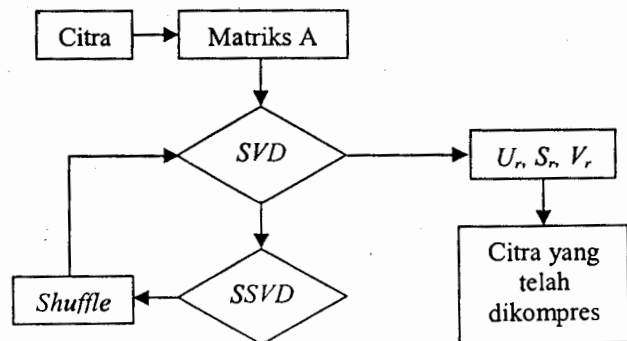
Input citra yang digunakan adalah input citra standard, yaitu Lena, Peppers, dan Baboon yang berukuran  $256 \times 256$  dengan format gambar berupa png dengan ukuran file 355 KB, 358 KB, dan 413 KB. Penulis akan melaporkan hasil percobaan dari pemrosesan citra Lena secara rinci.

### 3.1 Pemrosesan secara Global

Pada tahap ini penulis akan membandingkan *SVD* dan *SSVD* terhadap citra yang telah direpresentasikan ke dalam matriks yang berukuran *double*. Dimana satu citra tersebut dianggap sebagai satu blok utuh.

Satuan ukuran *error* yang akan digunakan adalah *MSE*, dimana *MSE* ini merupakan *MSE* rata-rata dari *layer* RGB masing-masing citra.

Tahap pengerjaan *SVD* dan *SSVD* dapat dilihat dari *flowchart* berikut:



Gambar 1. *Flowchart* tahap *SVD* dan *SSVD* pada citra

Berikut, hasil dari pemrosesan citra Lena secara global:



(a) Citra asli



(b) SVD rank 10



(c) SSVD rank 10

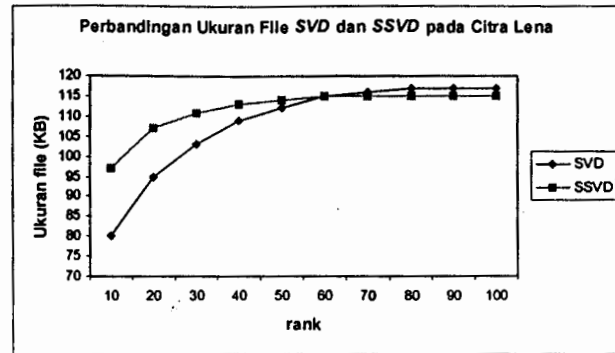


(d) SVD rank 20



(e) SSVD rank 20

Gambar 2. Perbandingan citra SVD dan SSVD



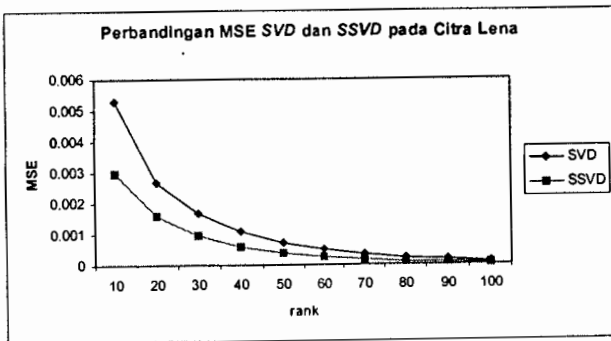
Gambar 3. Grafik ukuran file SVD dan SSVD

Dari Grafik perbandingan MSE bisa dilihat bahwa SSVD lebih sedikit *error*-nya dibandingkan dengan SVD. Hal ini menunjukkan bahwa metode SSVD lebih baik dibandingkan dengan SVD. Tapi berbeda dengan MSE, untuk ukuran file terlihat pada grafik bahwa untuk rank 70-100 ukuran file untuk SSVD lebih unggul dibandingkan dengan SVD, tapi hal ini tidak berlaku untuk citra Peppers dan Baboon. Pada citra Peppers SSVD akan lebih unggul untuk ukuran file-nya pada saat rank 90 dan 100, sedangkan citra baboon ukuran file pada SVD lebih unggul daripada SSVD.

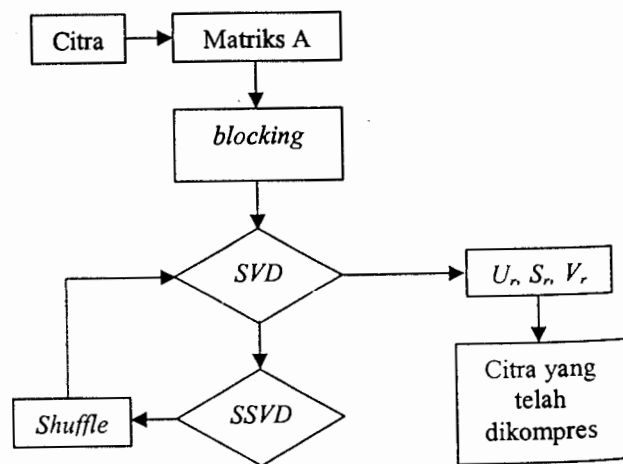
### 3.2. Pemrosesan secara Blocking

Yang dimaksud dengan *blocking* adalah membagi citra ke dalam blok-blok dengan ukuran tertentu, dimana tiap blok akan diproses dengan metode SVD maupun SSVD dengan menggunakan optimasi rank dari RGB-nya.

Tahap pengerjaannya bisa dilihat pada flowchart sebagai berikut:

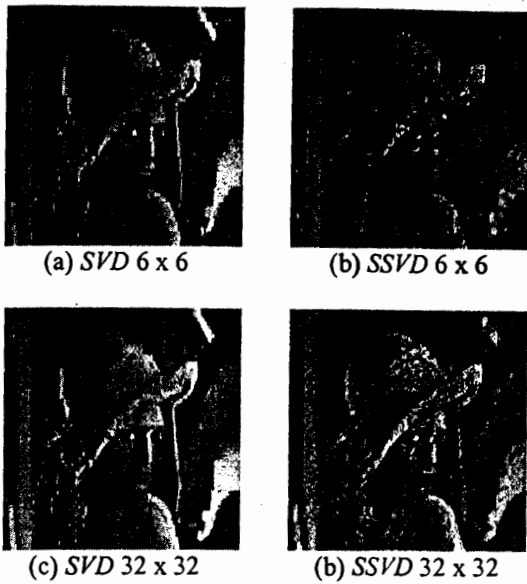


Gambar 3. Grafik MSE SVD dan SSVD

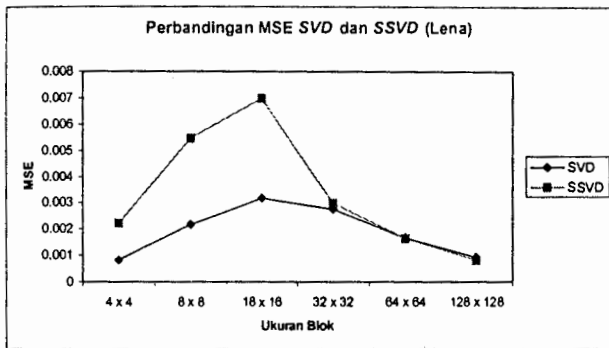


Gambar 4. Flowchart tahap SVD dan SSVD pada citra

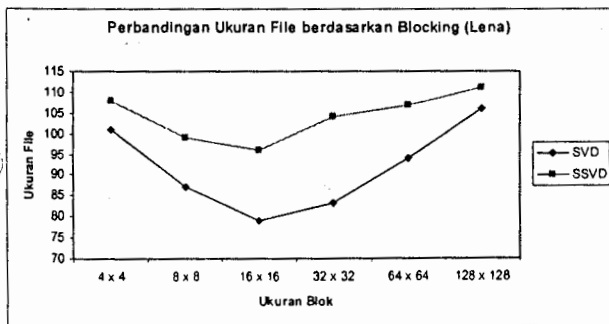
Berikut ini adalah hasil dari pemrosesan citra Lena dengan teknik *blocking* dimana ukuran bloknya adalah 4 x 4, 8 x 8, 16 x 16, 32 x 32, 64 x 64, 128 x 128:



Gambar 5. Perbandingan citra SVD dan SSVD



Gambar 6. Grafik MSE SVD dan SSVD



Gambar 7. Grafik ukuran file SVD dan SSVD

Berbeda dengan pemrosesan secara global, hasil yang didapatkan untuk pemrosesan secara *blocking* sangat berbeda. Pada pengukuran *MSE*, untuk *SVD* lebih bagus dibandingkan dengan *SSVD*, begitu juga dengan ukuran *file*-nya.

#### 4. KESIMPULAN

Dari hasil penelitian, dapat disimpulkan bahwa melalui pemrosesan secara global, *SSVD* lebih unggul daripada *SVD*, tetapi melalui pemrosesan terhadap blok yang dipilah-pilah (*blocking*), *SSVD* tidak terbukti lebih unggul daripada *SVD*. Hal ini disebabkan karena representasi matriks pada citra untuk *SSVD* telah di-*shuffle*, sehingga menyebabkan perubahan pada matriksnya lebih banyak daripada yang tanpa di-*shuffle*.

#### 5. POSSIBLE WORK (PENELITIAN LANJUTAN)

Untuk penelitian selanjutnya penulis menyarankan melakukan percobaan dengan blok *non-square* pada pemrosesan citra dan untuk teknik *blocking*-nya bisa digunakan teknik *blocking* dimana ukuran blok-bloknya tidak seragam (*non-uniform*).

#### REFERENSI

- [1] Castleman, R.K., "Digital Image Processing", Prentice Hall International, Inc, 1996.
- [2] Golub, G.H. and Van Loan, C.F. "Matrix Computation", John Hopkin University Press, 1989.
- [3] [http://whatis.techtarget.com/definition/0,,sid9\\_gci212327,0.html](http://whatis.techtarget.com/definition/0,,sid9_gci212327,0.html), (diakses tanggal 8-11-2006).
- [4] <http://www.ee.bgu.ac.il/~greg/graphics/compress.html>, (diakses tanggal 2-12-2006)
- [5] Jain, A. K., "Fundamental of Digital Image Processing", Prentice Hall of India, October 2006.
- [6] Ranade, A., Srikanth, S. M., Satyen, K., "A Variation on SVD Based Image Compression", <http://www.cse.iitb.ac.in/~ranade/SSVD.pdf>, (diakses tanggal 27-9-2006).